Commentarii
informaticae didacticae | 2

Peer Stechert

Fachdidaktische Diskussion von Informatiksystemen und der Kompetenzentwicklung im Informatikunterricht

Universitätsverlag Potsdam

Commentarii informaticae didacticae (CID)

Commentarii informaticae didacticae (CID) | 2

Peer Stechert

Fachdidaktische Diskussion von Informatiksystemen und der Kompetenzentwicklung im Informatikunterricht

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Universitätsverlag Potsdam 2009
http://info.ub.uni-potsdam.de/verlag.htm

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 4623 / Fax: 3474
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe Commentarii informaticae didacticae (CID) wird herausgegeben von:
Johannes Magenheim, Universität Paderborn
Sigrid Schubert, Universität Siegen
Andreas Schwill, Universität Potsdam

zugl. Siegen, Univ., Diss., 2009
Die Dissertationsschrift ist bereits online veröffentlicht auf dem Publikations- und Dokumentenserver der Universitätsbibliothek Siegen
URN urn:nbn:de:hbz:467-3859
http://nbn-resolving.de/urn:nbn:de:hbz:467-3859

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URL http://pub.ub.uni-potsdam.de/volltexte/2009/3795/
URN urn:nbn:de:kobv:517-opus-37959
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-37959

Zugleich gedruckt erschienen im Universitätsverlag Potsdam
ISBN 978-3-86956-024-3
ISSN 1868-0844

Fachdidaktische Diskussion von Informatiksystemen und der Kompetenzentwicklung im Informatikunterricht

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 Peer Stechert

1. Gutachterin: Prof. Dr. Sigrid Schubert, Universität Siegen
2. Gutachter: Prof. Dr. Peter Hubwieser, TU München

Tag der mündlichen Prüfung: 4. Juni 2009

Kurzfassung

In der vorliegenden Arbeit wird ein Unterrichtsmodell zur Kompetenzentwicklung mit Informatiksystemen für die Sekundarstufe II vorgestellt. Der Bedarf wird u. a. damit begründet, dass Informatiksysteme zu Beginn des 21. Jahrhunderts allgegenwärtig sind (Kapitel 1). Für Kompetenzentwicklung mit Informatiksystemen sind diese in ihrer Einheit aus Hardware, Software und Vernetzung anhand ihres nach außen sichtbaren Verhaltens, der inneren Struktur und Implementierungsaspekten zu analysieren. Ausgehend vom Kompetenzbegriff (Kapitel 2) und dem Informatiksystembegriff (Kapitel 3) erfolgt eine Analyse des fachdidaktischen Forschungsstandes zur Kompetenzentwicklung mit Informatiksystemen. Die Ergebnisse lassen sich in die Bereiche (1) Bildungsziele, (2) Unterrichtsinhalte, (3) Lehr-Lernmethodik und (4) Lehr-Lernmedien aufteilen (Kapitel 4).

In Kapitel 5 wird die Unterrichtsmodellentwicklung beschrieben. Den Zugang zu Informatiksystemen bildet in der vorliegenden Dissertationsschrift das nach außen sichtbare Verhalten. Es erfolgt eine Fokussierung auf vernetzte fundamentale Ideen der Informatik und Strukturmodelle von Informatiksystemen als Unterrichtsinhalte. Es wird begründet, dass ausgewählte objektorientierte Entwurfsmuster vernetzte fundamentale Ideen repräsentieren. In Abschnitt 5.4 werden dementsprechend Entwurfsmuster als Wissensrepräsentation für vernetzte fundamentale Ideen klassifiziert. Das systematische Erkunden des Verhaltens von Informatiksystemen wird im Informatikunterricht bisher kaum thematisiert. Es werden Schülertätigkeiten in Anlehnung an Unterrichtsexperimente angegeben, die Schüler unterstützen, Informatiksysteme bewusst und gezielt anzuwenden (Abschnitt 5.5). Bei dieser Lehr-Lernmethodik werden das nach außen sichtbare Verhalten von Informatiksystemen, im Sinne einer Black-Box, und das Wechselspiel von Verhalten und Struktur bei vorliegender Implementierung des Systems als White-Box analysiert. Die Adressierung schrittweise höherer kognitiver Niveaustufen wird in die Entwicklung einbezogen. Unterstützend wird für das Unterrichtsmodell lernförderliche Software gestaltet, die vernetzte fundamentale Ideen in Entwurfsmustern und das Experimentieren aufgreift (Abschnitt 5.6). Schwerpunkte bilden im Unterrichtsmodell zwei Arten von lernförderlicher Software: (1) Die Lernsoftware Pattern Park wurde von einer studentischen Projektgruppe entwickelt. In ihr können in Entwurfsmustern enthaltene fundamentale Ideen der Informatik über ihren Lebensweltbezug im Szenario eines Freizeitparks analysiert werden. (2) Als weitere Art Lernsoftware werden kleine Programme eingesetzt, deren innere Struktur durch ausgewählte Entwurfsmuster gebildet und deren Verhalten direkt durch die darin enthaltenen fundamentalen Ideen bestimmt wird. Diese Programme können durch die Experimente im Unterricht systematisch untersucht werden.

Mit dem Ziel, die normative Perspektive um Rückkopplung mit der Praxis zu ergänzen, werden zwei Erprobungen im Informatikunterricht vorgenommen. Diese liefern Erkenntnisse zur Machbarkeit des Unterrichtsmodells und dessen Akzeptanz durch die Schüler (Kapitel 6 und 8). Exemplarisch umgesetzt werden die Themen Zugriffskontrolle mit dem Proxymuster, Iteration mit dem Iteratormuster und Systemzustände mit dem Zustandsmuster. Der intensive Austausch mit Informatiklehrpersonen in der Kooperationsschule über Informatiksysteme und Kompetenzentwicklung sowie die Durchführung von zwei Lehrerfortbildungen ergänzen die Beobachtungen im unterrichtlichen Geschehen. Die erste Unterrichtserprobung resultiert in einer Weiterentwicklung des Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung (Kapitel 7). Darin erfolgt eine Fokussierung auf das nach außen sichtbare Verhalten von Informatiksystemen und eine Verfeinerung der Perspektiven auf innere Struktur und ausgewählte Implementierungsaspekte. Anschließend wird die zweite Unterrichtserprobung durchgeführt und evaluiert (Kapitel 8). Am Schluss der Forschungsarbeit steht ein in empirischen Phasen erprobtes Unterrichtsmodell.

Abstract

In the 21st century, informatics systems are ubiquitous. Therefore, the author presents an educational model for competencies with respect to informatics systems (Chapter 1). To achieve such competencies at upper secondary level, observable behaviour, internal structure and implementation aspects of informatics systems have to be analysed by students. Based on a definition of the terms competency (Chapter 2) and informatics system (Chapter 3), the state of the art in Didactics of Informatics is investigated. In the national and international scientific work, (1) educational objectives, (2) themes and subject matters, (3) teaching and learning methods, as well as (4) educational means and educational media are identified (Chapter 4).

In Chapter 5 the development of the educational model is described. The approach to competencies with respect to informatics systems concentrates on the observable behaviour of the systems. We focus on networked fundamental ideas of informatics as a quality factor and structural models of informatics systems. Selected object-oriented design patterns represent networked fundamental ideas. In Section 5.4 design patterns as knowledge representations of fundamental ideas are classified. Systematic exploration of informatics systems is uncommon in informatics education at upper secondary level. Therefore, students' activities are developed according to educational experiments to enable students to use systems consciously (Section 5.5). Systematic exploration puts students in a position to analyse the observable behaviour as a black box. Given the source code and documentation of a system, experimenting with such a system relates behaviour to its internal structure. Succeeding cognitive processes are also considered in this approach. To support learning, software was developed, which emphasises fundamental ideas in design patterns and enables experimenting (Section 5.6). There are two kinds of learning software: (1) The learning software Pattern Park was developed by a student project group. In the software fundamental ideas within design patterns can be understood through a real-life analogy in the context of a theme park. (2) As a second kind of learning software we use small programs, whose internal structure is built by selected design patterns. Their observable behaviour depends on networked fundamental ideas of informatics. These programs can be analysed systematically by students through experiments.

Aiming at complementing the normative perspective with concrete learning processes, two classroom practice projects were conducted. These offered results with respect to feasibility of the educational model and acceptance by the students (Chapter 6 and 8). Exemplarily, access control by Proxy design pattern, iterati-

on by Iterator design pattern, and states of systems by State design pattern were chosen. Cooperation with teachers and conduction of teacher training workshops complement observations within the classroom projects. The first classroom project resulted in a refinement of theory to foster competencies with respect to informatics systems (Chapter 7). In particular, perspectives on informatics systems were elaborated. Afterwards, a second classroom project was conducted and evaluated (Chapter 8). In conclusion of the research project, there is an empirically tested educational model to foster competencies with respect to informatics systems.

Vorwort

Die vorliegende Dissertationsschrift entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl „Didaktik der Informatik und E-Learning" an der Universität Siegen. Ich möchte mich bei Frau Prof. Dr. Sigrid Schubert bedanken für die intensive Betreuung meines Promotionsvorhabens und kontinuierliche Förderung meiner wissenschaftlichen Arbeit. Herrn Prof. Dr. Peter Hubwieser danke ich für die Übernahme des Koreferats sowie für die zahlreichen guten Kommentare und Anregungen, die zur Präzisierung meiner Vorstellungen beigetragen haben.

Des Weiteren bedanke ich mich bei meinen Kollegen Christian Eibl, Stefan Freischlad und Christian Kollee für die hilfreichen Diskussionen und kritischen Nachfragen rund um das Unterrichtsmodell zur Kompetenzentwicklung mit Informatiksystemen. Den Projektpartnern im DFG-Projekt „Entwicklung von qualitativen und quantitativen Messverfahren zu Lehr-Lern-Prozessen für Modellierung und Systemverständnis in der Informatik" von der Universität Paderborn möchte ich danken für die gewinnbringenden Diskussionen zur Entwicklung eines Kompetenzmodells. Namentlich genannt seien Prof. Dr. Johannes Magenheim, Wolfgang Nelles, Thomas Rhode und Prof. Dr. Niclas Schaper. Den Informatiklehrern Herrn Werner Eling, Frau Milena Ganea und Herrn Hartmud Koch danke ich für die intensiven Diskussionen zur Kompetenzentwicklung mit Informatiksystemen sowie für die Möglichkeit, das Unterrichtsmodell in ihren Informatikkursen zu erproben. Den Schülerinnen und Schülern danke ich für die konstruktive Mitarbeit.

Außerdem danke ich den aktiven Teilnehmern des „Internationalen Doktorandenkolloquiums zur Didaktik der Informatik – IDDI", die mir auf unseren Treffen in Berlin, Zürich, Siegen und Münster viele Anregungen gaben. Auch an viele Studierende, die in ihren Hauptseminararbeiten, fachdidaktischen Praktika, Diplomarbeiten und der Projektgruppe Pattern Park zur Konkretisierung meiner Ideen beitrugen, geht ein großes Dankeschön. Genannt seien an dieser Stelle Carsten Dittich, Demian Franke, Lars Friedrich, Thomas Gerding, Daniel Graf, Florian Haug, Benjamin Klein, Rudolf Koslowski, Pamina Stupperich, Daniel Sülz, Jan Tenhumberg, Jonathan Ufer, Swetlana Warkentin und Michelle Weyer.

Ganz besonderer Dank gilt meiner Familie: meiner Freundin Kirstin Schwidrowski, meinen Großeltern, Eltern, und Geschwistern sowie meiner Tante und meinem Onkel für die liebevolle Unterstützung während der Promotion.

Peer Stechert Siegen, im Februar 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACM	Association for Computing Machinery
ADT	Abstrakter Datentyp
API	Application Programming Interface
ATEE	Association for Teacher Education in Europe
CLT	Cognitive Load Theory
CPU	Central Processing Unit
CRC	Class-Responsibility-Collaborator
CSCL	Computer Supported Collaborative Learning
CSER	Computer Science Education Research
DeSeCo	Definition and Selection of Key Competencies
DFG	Deutsche Forschungsgemeinschaft
DOS	Disk Operating System
ECDL	European Computer Driving Licence
ECIS	Entwicklung von Curriculumelementen für den Informatikunterricht in der Sekundarstufe I
EPA	Einheitliche Prüfungsanforderungen in der Abiturprüfung
EQR	Europäischen Qualifikationsrahmen für lebenslanges Lernen
EVA	Eingabe-Verarbeitung-Ausgabe
GI	Gesellschaft für Informatik e. V.
HTML	Hypertext Markup Language
ICER	International Computing Education Research Workshop
ICT	Information and Communication Technology
IFIP	International Federation for Information Processing
INFOS	GI-Fachtagung „Informatik und Schule"
ISO	International Organization for Standardization
ISSEP	Informatics in Secondary Schools – Evolution and Perspectives
IT	meist: Information Technology; nur UNESCO-Curricula: Informatics Technology
ITiCSE	Innovation and Technology in Computer Science Education
MVC	Model-View-Controller

NCTM	National Council of Teachers of Mathematics
OECD	Organisation for Economic Co-operation and Development
OOM	Objektorientiertes Modellieren
OSI	Open Systems Interconnection
PISA	Programme for International Student Assessment
RAM	Random Access Machine, Registermaschine
SIGCSE	Special Interest Group on Computer Science Education: Technical Symposium on Computer Science Education
UML	Unified Modeling Language
UNESCO	United Nations Educational, Scientific and Cultural Organization
WCCE	World Conference on Computers in Education
WWW	World Wide Web

1. Einleitung

1.1 Problemlage und Motivation

Informatiksysteme sind in der Wissensgesellschaft zu Beginn des 21. Jahrhunderts allgegenwärtig (UNESCO 2005). Dies äußert sich in dem Symptom der Informationsflut sowie in ihrer Bedeutung für Wirtschaft und Arbeitsmarkt (vgl. Hubwieser 2007a, S. 58f). Bereits seit Anfang der neunziger Jahre wird die Vernetzung dieser Systeme in der Informatik besonders hervorgehoben:

> „Als neues Paradigma der Informatik ergibt sich also: Eine Gruppe von gleichrangigen, selbstständigen, einigermaßen intelligenten Akteuren, die bestimmte Aufgaben erledigen und dazu miteinander und mit der Umgebung interagieren" (Brauer und Brauer 1992, S. 15).

Die Vernetzung von Rechnern in lokalen Netzen und mit dem Internet wird sowohl in der nationalen als auch der internationalen fachdidaktischen Diskussion herangezogen, um Unterricht zu Wirkprinzipien von Informatiksystemen zu fordern. Aus der Veränderung der Lebenswelt folgen neue Anforderungen für den Informatikunterricht und fachdidaktische Gestaltungsvorschläge (vgl. (Baumann 1993), (Baumann 1996), (Hubwieser und Broy 1997b), (Hubwieser 2007a), (Hampel et al. 1999), (Magenheim und Schulte 2006), (Brinda und Schubert 2001), (Schubert und Schwill 2004)). Dies äußert sich auch durch Bildungs- und Lernzielbeschreibungen in nationalen und internationalen Bildungsempfehlungen (vgl. (GI 2000), (ACM 2006), (UNESCO 2002), (GI 2008)). So veröffentlichte die Gesellschaft für Informatik e. V. (GI) schon im Jahr 2000 Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen, um Schülern[1] durch informatische Denk- und Arbeitsweisen die bewusste Anwendung von Informations-

[1] Alle Personenbezeichnungen gelten gleichermaßen für die männliche und weibliche Form.

und Kommunikationstechnologien zu ermöglichen (GI 2000, S. 1). Für die informatische Bildung werden darin vier Leitlinien festgelegt, die sich mittlerweile auch in Rahmenlehrplänen mehrerer Bundesländer wieder finden, und in denen Informatiksysteme zentral sind:

- Interaktion mit Informatiksystemen,
- Wirkprinzipien von Informatiksystemen,
- Informatische Modellierung,
- Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft.

In den aktuellen Bildungsstandards der GI für die Sekundarstufe I heißt es:

> „Zur Nutzung von Informatiksystemen ist ein grundlegendes Verständnis ihres Aufbaus und ihrer Funktionsweise notwendig. [...] Kompetenzen, die das Verständnis von Informatiksystemen fördern, sind vor allem deshalb von Bedeutung, weil die Schülerinnen und Schüler auch in der Lage sein sollten, sich weitere, ihnen bislang möglicherweise unbekannte Systeme zu erschließen" (GI 2008, S. 37).

Da zurzeit nur in wenigen Bundesländern ein verpflichtender Informatikunterricht in der Sekundarstufe I existiert, können diese Kompetenzen der Schüler für die Sekundarstufe II nicht vorausgesetzt werden; darüber hinaus sind deren Erweiterung und Vertiefung zu diskutieren. Die United Nations Educational, Scientific and Cultural Organization (UNESCO) und der Informatikweltverband International Federation for Information Processing (IFIP) erstellten gemeinsam ein Curriculum zum Einsatz von Informations- und Kommunikationstechnologie (Information and Communication Technology – ICT) in Schulen. Darin werden exemplarisch drei sehr weit gefasste Bildungsziele für Schüler zu Informatiksystemen angegeben:

> - „Students should be able to differentiate between the basic components of a computer system [...].
> - Students should know what system software is and how the use of this software relates to the operating systems software.
> - They should be aware of the connectivity of computers in a local and an external network and be familiar with the appropriate functions of the networks" (UNESCO 2002, S. 68).

Die Wichtigkeit von Informatiksystemen für die informatische Bildung ist in den Bildungsempfehlungen somit unbestritten. Es fehlt jedoch an einer Verfeinerung dieser Zielebene mit Beschreibung konkreten, empirisch erprobten Unterrichts zu Informatiksystemen und Kompetenzentwicklung, der die Feinziele erreichbar macht. Humbert führt empirische Studien durch, die zeigen, dass Schüler sich von Rechnern bedroht fühlen (Humbert 2003). Damit hat das Ziel der Entmystifizierung von Informatiksystemen nicht an Aktualität eingebüßt (vgl. Schubert und Schwill 2004, S. 253). Der in Forschungsarbeiten und Bildungsempfehlungen geforderte Unterricht zu Wirkprinzipien von Informatiksystemen hat die Schulpraxis erst punktuell

erreicht. In der unterrichtlichen Praxis investieren Schüler viel Zeit in die Entwicklung kleiner Programme. Der Förderung von Basiskompetenzen, z. B. die Erkundung und Analyse von Informatiksystemen aus dem Alltag, wird kaum Lernzeit eingeräumt (vgl. Schubert und Schwill 2004, S. 216). Als Ursachen für die bislang kaum vorhandene Umsetzung in ein Unterrichtsmodell lassen sich u. a. folgende Forschungslücken und Problembereiche benennen:

Konkretisierung der Bildungsergebnisse und -inhalte Welche Kompetenzen benötigt ein Schüler, um Informatiksysteme den eigenen Zielen und Bedürfnissen entsprechend effektiv zu nutzen? Schüler sind angewiesen auf ein kognitives Modell von Informatiksystemen und von deren Möglichkeiten und Grenzen zur Teilnahme in der Gesellschaft als mündige Bürger. Die allgemeinen Bildungsanforderungen zu Wirkprinzipien von Informatiksystemen sind nur realisierbar, wenn eine Konkretisierung der Bildungsergebnisse in Form von Kompetenzbeschreibungen und des Bildungszugangs in Form von Lernzielen und -inhalten erfolgt. Mit der Veröffentlichung von Bildungsstandards wurde 2008 für die Sekundarstufe I ein Anfang gemacht (GI 2008). Der Autor begann ab 2005 mit Publikationen zu Informatiksystemen und Kompetenzentwicklung für die Sekundarstufe II.

Lehr-Lernmethodik Wie können Schüler Informatiksysteme erkunden und Kompetenz handlungsorientiert entwickeln? Welche Aufgabenklassen lassen sich identifizieren? Notwendig ist eine nachvollziehbare Strukturierung von Unterrichtsinhalten für ein Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung. Es fehlt an empirisch überprüften Strategien, mit denen Schüler das Verhalten von Informatiksystemen entdecken und auf vernetzte informatische Konzepte zurückführen. Im Informatikunterricht fehlt das systematische Beobachten von Informatiksystemen bisher fast vollständig (vgl. Kapitel 4). Es existieren keine Unterrichtsmodelle, die das systematische Vorgehen, das zum Bewerten und Beobachten von Informatiksystemen notwendig ist, thematisieren und mit ausgewählten Informatikkompetenzen verknüpfen.

Lehr-Lernpsychologie Wie müssen Inhalte zu Wirkprinzipien von Informatiksystemen repräsentiert werden, damit sie ein korrektes Bild von Informatiksystemen vermitteln und von Schülern kognitiv erfasst werden können? Es fehlt an lernpsychologisch fundierten Wissensrepräsentationen, um die in Informatiksystemen vernetzten informatischen Inhalte zu vermitteln. Bei der Publikation von Unterrichtskonzepten spielen vernetzte fundamentale Ideen (Schwill 1993a) bisher oft keine Rolle. Dies ist auch darauf zurückzuführen, dass das Problem der Vernetzung der Inhalte von Lehrpersonen selbst gelöst werden muss. So ist die Vernetzung im Unterricht oft durch langjährige Lehr-Lernerfahrungen zu Algorithmenklassen, Datentypen und Datenstruk-

turen, also durch die Vorgehensweise beim Programmieren im Kleinen und die eingesetzten Programmierstile (z. B. prädikativ, funktional) begründet.

Lehr-Lernmedien Welche medialen Repräsentationen von Informatiksystemen und Wirkprinzipien in Informatiksystemen unterstützen den Erkenntnisprozess? Es fehlt an geeigneten Unterrichtsmitteln inklusive Lernsoftware und fachdidaktisch vorbereiteter Software für den Lehr-Lernprozess (Magenheim 2001). Formulierungen von Gestaltungsanforderungen an Lernsoftware, die von Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik ausgehen, gibt es nicht. So werden in den vorherrschenden Ansätzen zur Konstruktion von Informatiksystemen oft professionelle Entwicklungswerkzeuge eingesetzt, die Schüler in der Sekundarstufe II überfordern.

Die hier nur knapp dargestellte Wichtigkeit von Informatiksystemen sowie die Problembereiche und Forschungslücken sind für den Autor Motivation, diese Arbeit zu schreiben. Ziel der vorliegenden Arbeit ist es, ein Unterrichtsmodell zu Basiskompetenzen zu Informatiksystemen zu entwickeln und in der Unterrichtspraxis zu erproben. Damit soll auch ein Beitrag zu Bildungsstandards für die Sekundarstufe II geleistet werden. Vom Autor wird erforscht, welche Bildungsziele für Schüler notwendig sind und wie der Lehr-Lernprozess diese fachlich korrekt erreichbar macht, ohne Abbild der Hochschulinformatik zu sein. Eine Kompetenzstrukturierung zu Informatiksystemen bietet eine neue Perspektive auf Lehr-Lernprozesse im Informatikunterricht der Sekundarstufe II. Dabei wird vorausgesetzt, dass Kompetenzentwicklung mit Informatiksystemen zum Bildungsauftrag des allgemein bildenden Informatikunterrichts gehört und in Alltag sowie Berufsleben von Nutzen ist (Abschnitt 4.2; (Hubwieser 2007a)).

1.2 Forschungsmethodik und Forschungsverlauf

1.2.1 Intervenierende Fachdidaktik

Die Forschungsmethodik der vorliegenden Arbeit basiert auf fachdidaktischer Theoriebildung und wird fundiert durch empirisch explorative Fallstudien. Sie ist inspiriert von intervenierender Fachdidaktikforschung wie sie Hubwieser und Broy (1999) sowie Dagiene (1999) auf der IFIP Working Group 3.1 and 3.5 Open Conference „Communications and Networking in Education: Learning in a Networked Society" vorstellten. Für intervenierende Fachdidaktik ist kennzeichnend, dass Forschende neue Konzepte der Fachdidaktik entwickeln und in Feldstudien, d. h. Interventionen, umsetzen. Die Interventionsforschung befasst sich mit der Entwicklung von Maßnahmen, während die Evaluationsforschung deren Bewertung zum Gegenstand hat (Bortz und Döring 2002, S. 107). Die Trennung von Interventions- und Evaluationsforschung ist in der Praxis jedoch selten strikt. Im Gegenteil wird in der

Tabelle 1.1: Forschungsverlauf

Aktivitäten und Meilensteine	**Veröffentlichungen**
Analyse des Bildungsbedarfs	
Informatikbildungsstandards im digitalen Medienumbruch	Schubert et al. (2005a)
Vernetzung fundamentaler Ideen (Doktorandenforum der INFOS 2005)	Schubert et al. (2005b)
Theoriebildung	
Entwurfsmustern als Wissensrepräsentation vernetzter fundamentaler Ideen	Stechert (2006a) Stechert (2006b)
Entwicklung des Unterrichtmodells	Stechert (2006c)
Verfeinerung des Unterrichtsmodells	Stechert (2007b)
Erprobung und Evaluation des Unterrichtsmodells	
Entwicklung der Lernsoftware Pattern Park (Projektgruppe)	Franke et al. (2007), Schubert et al. (2009)
Beschreiben von Schülertätigkeiten (Seminararbeit)	Weyer (2007a)
1. Erprobung des Unterrichtsmodells	
Auswertung der 1. Unterrichtserprobung	Stechert (2007c)
Strukturierung von Lerngegenstand und Basiskompetenzen	Stechert und Schubert (2007)
Ausgestaltung von Unterrichtsbeispielen (Seminararbeit)	Sülz (2007)
Verfeinerung von Aufgaben mit Laut-Denken (Seminararbeit)	Stupperich und Warkentin (2007)
Klassifikation von Architekturmustern (Diplomarbeit)	Ufer (2007)
Workshop zu Unterrichtsinterventionen	Schubert et al. (2007)
Gestaltung von Unterricht zu Datenbanken (Seminararbeit)	Dittich (2008)
Gestaltung von Gruppenarbeitsbeispielen (Seminararbeit)	Gerding (2008)
2. Erprobung des Unterrichtsmodells	
Klassifikation von Entwurfsmustern (Diplomarbeit)	Weyer (2007b)
Auswertung der 2. Unterrichtserprobung	Stechert (2008a), Stechert (2008b), Stechert (2008c)
Abschlussdiskussion	
Internationaler Workshop zu lernförderlicher Software	Freischlad und Stechert (2008)
Weiterführende Forschung zu eingebetteten Mikrosystemen	Schubert und Stechert (2008), Schwidrowski et al. (2009)
Zusammenfassung der Ergebnisse	Dissertationsschrift

Verbindung von Theorie und Praxis bei der Entwicklung und Evaluation von Unterrichtskonzepten Potential gesehen (Tulodziecki und Herzig 1998), um neben der Erforschung der pädagogischen Praxis theoretische Ansätze weiter zu entwickeln:

> „Ein solches Verfahren stellt nicht die Formulierung allgemeiner Hypothesen und ihre experimentelle Prüfung in den Mittelpunkt, sondern die Entwicklung und Erprobung praxisrelevanter Unterrichtskonzepte auf der Basis von Voraussetzungs-Ziel-Mittel-Aussagen, die ihrerseits auf explizit formulierten lern- und lehrtheoretischen Annahmen beruhen" (Tulodziecki und Herzig 1998, S. 29).

Normative Implikationen gehen dabei in die Zielvorstellungen ein. Die Rückkopplung mit der Praxis erlaubt eine kritische Reflexion und die damit verbundene

Weiterentwicklung der theoretischen Ergebnisse. Dazu gehört eine Forschungsmethodik mit folgenden Phasen:

1. Analyse des Bildungsbedarfs,
2. Theoriebildung zum Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung,
3. Entwicklung eines Unterrichtsmodells und Beschreibung von Schülertätigkeiten einschließlich Lehr-Lernmaterial zur Vorbereitung von Unterrichtsprojekten,
4. Unterrichtserprobungen,
5. Evaluation und Zwischendiskussionen des Lehr-Lernprozesses und der Lehr-Lernmaterialien mit Auswirkung auf Theoriebildung und Unterrichtsmaterialentwicklung,
6. Abschlussdiskussion der empirisch erprobten Ergebnisse.

Zu den Phasen der Forschungsmethodik und einzelnen Aktivitäten siehe auch Tabelle 1.1. Durch mehrere Unterrichtserprobungen entsteht ein Entwurfs-, Interventions- und Evaluationszyklus der Unterrichtsmodellentwicklung.

Phase 1: Analyse des Bildungsbedarfs

Der erste Schritt der Forschungsmethodik ist die Analyse des Bildungsbedarfs. Zentral ist dabei die Fragestellung, welche Kompetenzen zu Informatiksystemen Schüler in der Lebenswelt benötigen und wie der Lehr-Lernprozess diese fachlich korrekt erreichbar macht, ohne Abbild der Hochschulinformatik zu sein. Es ist notwendig, den aktuellen Informatiksystembegriff in den unterschiedlichen Bereichen der Kerninformatik zu untersuchen, um Kompetenzen zu strukturieren und Lernziele beschreiben zu können. Unter Kerninformatik werden in dieser Arbeit Theoretische Informatik, Praktische Informatik und Technische Informatik zusammengefasst. Kerninformatik ist in Abgrenzung zu gesellschaftlichen Bezügen der Informatik, zur Angewandten Informatik und zur Didaktik der Informatik zu sehen (vgl. Schubert und Schwill 2004, S. 6ff). Außerdem kann anhand des Informatiksystembegriffs eine Analyse des fachdidaktischen Stands der Forschung zur Kompetenzentwicklung mit Informatiksystemen vorgenommen werden. Dazu werden nationale und internationale Informatikcurricula und Bildungsempfehlungen ebenso wie Konferenzpublikationen aus der Didaktik der Informatik hinsichtlich begründeter Bildungsziele, -inhalte, Lehr-Lernmethoden und Lernmedien für Informatiksysteme und Kompetenzentwicklung analysiert. Aus der Analyse gewonnene Ergebnisse können anhand der fachlichen Merkmale von Informatiksystemen geclustert werden. Daraus lässt sich eine Strukturierung von Basiskompetenzen ableiten (Kapitel 3 und Kapitel 4).

Phase 2: Theoriebildung für das Unterrichtsmodell zu Informatiksysteme und Kompetenzentwicklung

Auf der Strukturierung von Basiskompetenzen aufbauend, wird in Phase 2 eine Unterrichtsmodellentwicklung theoretisch begründet und am Beispiel Informatiksysteme und Kompetenzentwicklung vollzogen. Die Entwicklung eines fundierten Unterrichtsmodells besteht in der Integration von Erkenntnissen aus der Fachwissenschaft, der Erziehungswissenschaft und anderen Fachdidaktiken in die Didaktik der Informatik. Als theoretische Säule der Forschungsmethodik und Voraussetzung der Intervention werden Erkenntnisse aus der Forschung zum didaktischen System nach Brinda und Schubert genutzt (vgl. Brinda und Schubert 2001). Es ist eine Formalisierung des Lehr-Lernprozesses und besteht aus Wissensstrukturen, Aufgabenklassen und Lernsoftware. Traditionelle Komponenten des Lehr-Lernprozesses werden abhängig von Szenario und Zielgruppe mit fachdidaktischen Empfehlungen verknüpft, wodurch ein Beitrag zu Bildungsstandards geleistet wird (vgl. Schubert und Schwill 2004, S. 134). Stärke dieses Ansatzes ist, dass seit 2001 das didaktische System in der Dortmunder (bis 2002) und Siegener Forschergruppe auf die Bereiche objektorientiertes Modellieren, Internetworking und Kompetenzentwicklung mit Informatiksystemen angewendet und kontinuierlich weiterentwickelt wird (vgl. (Freischlad und Schubert 2006), (Schwidrowski 2007)). Eine Begründung für die Entwicklung von Lernsoftware innerhalb des Forschungsprozesses liefert Reinmann (2006). Ihrer Ansicht nach liefert gerade die Entwicklung von Lernsoftware als Teil des Forschungsprozesses relevante Forschungsergebnisse und nicht nur die quantitativ empirische Überprüfung der Lernwirksamkeit ausgereifter Lernsoftware und Lernkonzepte (Kapitel 5).

Phase 3: Entwicklung eines Unterrichtsmodells und Beschreibung von Schülertätigkeiten einschließlich Lehr-Lernmaterial zur Vorbereitung von Unterrichtsprojekten

Die nächste Phase in der Forschungsvorgehensweise bildet die Entwicklung von Unterrichtsprojekten. Aus der Handlungs- und Anwendungsorientierung ergeben sich unverzichtbare Schülertätigkeiten, für deren Ausgestaltung Lehr-Lern-Materialien erstellt werden. Auch die Formulierung von Lernzielen nach der überarbeiteten Bloom'schen Lernzieltaxonomie wird vorgenommen.
Auf Basis der normativen Analyse und theoretischen Erkenntnissen aus Fachwissenschaft, Erziehungswissenschaft, Lehr-Lerntheorie und Fachdidaktik entwickelt eine studentische Projektgruppe mit sechs Studierenden in insgesamt 3600 Arbeitsstunden die Lernsoftware. Entwicklungsbegleitend werden prototypische Module im Rahmen eines ersten Unterrichtsprojekts in der Sekundarstufe II eingesetzt. Durch die Rückmeldungen der Schüler kann die Lernsoftware weiter verbessert werden (Arnold und Hartmann 2007). Projektgruppen nehmen durch ihre Nähe zur Praxis und zum Berufsalltag in der universitären Informatikausbildung eine Sonderstellung

ein, denn hier arbeitet ein Team von Informatikstudierenden über mehrere Semester hinweg an einer komplexen Aufgabenstellung, die einschlägig für ihren Studiengang ist. Die Kombination von Theorie und Praxis ermöglicht eine forschende Lehre im Rahmen der Lehramtsausbildung (Tulodziecki und Herzig 1998). So können studentische Arbeiten zur Erstellung von Unterrichtsmaterialien, Beschreibung von Unterrichtsmethoden und Untersuchungen von auftretenden Fehlvorstellungen und kognitiven Barrieren, beispielsweise mittels Laut-Denken, einbezogen werden (Kapitel 5 und Kapitel 7).

Phase 4: Unterrichtserprobungen

Merkmal quasi-experimenteller Felduntersuchung ist, dass sie in natürlichen, d. h. nicht randomisierten und im Zuge des Forschungsprozesses kaum veränderten Umgebungen stattfinden. Die intervenierende Unterrichtsforschung erfordert eine kritische Reflexion. Atteslander beschreibt dazu vier Problemkreise, die zur qualitativen teilnehmenden Forschung zu klären sind:

> „[...] zum einen müssen die Teilnehmerrollen so offen und flexibel zu handhaben sein, dass der Forscher im Feld agieren und reagieren kann, zum Zweiten müssen die Rollen dem Feld entsprechen bzw. in diesem bereits angelegt sein, damit das Feld durch die Forschung nicht verändert wird, drittens muss überlegt werden, ob die Forscherrolle offen gelegt wird oder teilweise bzw. ganz verdeckt bleibt und viertens muss das Verhältnis zwischen Forscher- und Teilnehmerrolle (Distanz und Teilnahme) geklärt werden" (Atteslander et al. 2006, S. 92).

Durchgeführt werden zwei Unterrichtserprobungen, so dass die erste als Pilotstudie dienen kann (Bortz und Döring 2002, S. 359f). Eine Größenordnung von etwa 12 Unterrichtsstunden à 45 Minuten je Unterrichtserprobung entspricht einerseits einem gewissen Konsens über die Länge der Unterrichtsinterventionen in der Informatikdidaktik (vgl. Wiesner und Brinda (2007), Antonitsch (2007), Voß (2006), Hubwieser (2005)) und ist andererseits der Bindung der Schulen an den Lehrplan und die Anforderungen eines Zentralabiturs geschuldet. Wegen der anstehenden oder bereits umgesetzten Verkürzung der Schulzeit auf 12 Schuljahre werden die Jahrgangsstufen 11 und 12 für die Unterrichtsinterventionen anvisiert. Sowohl zur Unterrichtsvorbereitung wie auch zur Nachbereitung werden Gespräche mit Praktikern und Fachdidaktikern durchgeführt. Abschließend werden eine Lernerfolgskontrolle, eine Akzeptanzbefragung der Schüler und ein Interview mit dem stets anwesenden Kurslehrer durchgeführt (Kapitel 6 und Kapitel 8).

Phase 5: Evaluation und Zwischendiskussionen des Lehr-Lernprozesses und der Lehr-Lernmaterialien mit Auswirkung auf Theoriebildung und Unterrichtsmaterialentwicklung

Die fünfte Phase bildet die Evaluation. Dazu werden die theoretischen Überlegungen zum Unterrichtsmodell mit den Ergebnissen der Feldstudie verglichen:

> „[...] previously developed theory is used as a template with which to compare the empirical results of the case study" (Bassey 1999, S. 31).

Die qualitativ-formativen Evaluationsaktivitäten und -Ergebnisse, d. h. die begleitend gewonnenen und durch den Einsatz von Interview und Akzeptanzbefragung zur Überprüfung des Lehr-Lern-Szenarios angereicherten Ergebnisse, führen zu einem in empirischen, explorativen Phasen erprobten Unterrichtsmodell (vgl. Atteslander et al. 2006, S. 31). In Kapitel 7 werden die Weiterentwicklung und Verfeinerung des Unterrichtsmodells beschrieben.

Phase 6: Abschlussdiskussion der empirisch erprobten Ergebnisse

In der sechsten Phase werden die in den Unterrichtsinterventionen gewonnenen Erkenntnisse zur Weiterentwicklung und Verfeinerung des in der dritten Phase erarbeiteten Unterrichtsmodells vorgenommen. Es findet eine Abschlussdiskussion des im Informatikunterricht erprobten Unterrichtsmodells statt (Kapitel 9).

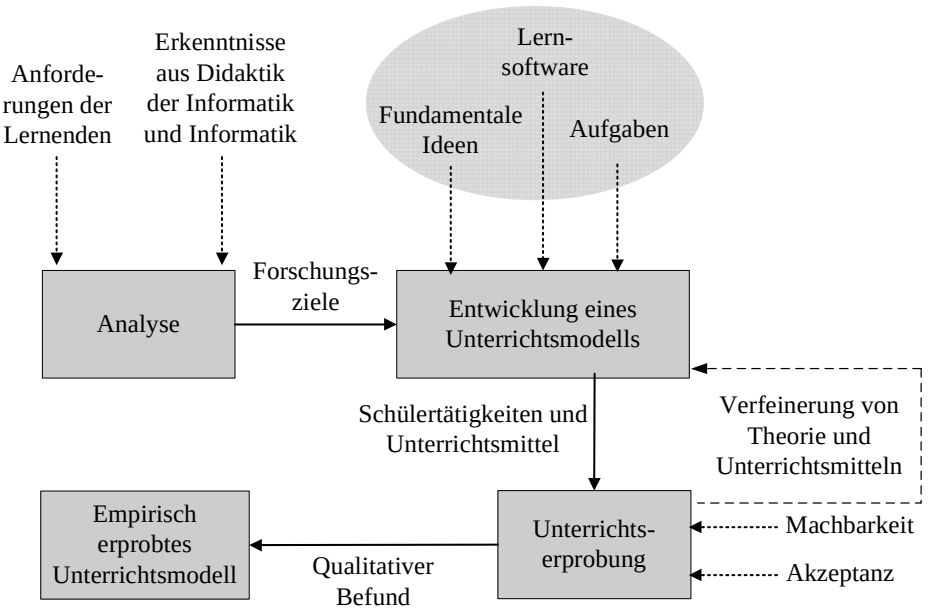


Abbildung 1.1: Entwurfs-, Interventions- und Evaluationszyklus der Unterrichtsmodellentwicklung

Zusammenfassend sind die Phasen der intervenierenden Fachdidaktik in Abbildung 1.1 dargestellt.

1.2.2 Methodenkritik zur intervenierenden Fachdidaktik

Analyse des Bildungsbedarfs und Theoriebildung zum Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung

Bei der intervenierenden Fachdidaktik sind im Gegensatz zur hypothesenprüfenden experimentellen Forschung normative Einflüsse in Form der Zielvorstellungen transparent (Tulodziecki und Herzig 1998). Sie wirken in das Unterrichtsmodell bezüglich der Voraussetzungen, Ziele und Unterrichtsmittel ein. Tulodziecki und Herzig (1998) beschreiben darüber hinaus, dass eine Forschungsmethodik, die zur Entwicklung und Evaluation von Unterricht Praxis und Theorie integriert, besonders geeignet für forschendes Lehren und Lernen im Rahmen der Lehrerausbildung ist, da die Arbeitsschritte transparent und in ihrem Praxis- und Theoriebezug nachvollziehbar sind (Tulodziecki und Herzig 1998, S. 29). Somit kann die Forschungsperson sowohl bei der Theoriebildung als auch in späteren Unterrichtserprobungen von Lehramtsstudierenden unterstützt werden.

Unterrichtserprobungen

Die Unterrichtserprobungen sind Stichproben bei kleiner Teilnehmerzahl. Für die Lehrperson ist es während der Intervention notwendig, das zugrunde liegende Forschungskonzept zu kennen. Nur dadurch ist es möglich, im Rahmen des Forschungskonzeptes auf die Lernsituation zu reagieren, ohne an Korrektheit einzubüßen. Für eine erste Erprobung während der Entwicklung der Konzepte ist es daher nahe liegend, dass sie von der Forschungsperson durchgeführt wird. Da diese jedoch keine Unterrichtserfahrung mit Schülern hatte, sind die Ergebnisse eingeschränkt übertragbar. Dem entgegen wirken jedoch kontinuierliche Rückmeldungen einer stets anwesenden Informatiklehrperson und die Durchführung der zweiten Unterrichtserprobung durch Informatiklehramtsstudierende, die vom Autor und der Informatiklehrperson bei den Unterrichtsstunden begleitet wurden.

Die besondere Situation, Forschungs- und Lehrperson gleichzeitig zu sein, wird in der quantitativen Forschung jedoch kritisch gesehen. Durch die geringe Stichprobengröße und den Verzicht auf Kontrollgruppenbildung sind Generalisierungen der Aussagen nicht möglich. Damit kann die Machbarkeit zwar widerlegt, aber nur bedingt bestätigt werden. Allein Plausibilitätserklärungen und unterstützende Aussagen zu ähnlichen Untersuchungen können getroffen werden. Deshalb ist die vorliegende Untersuchung qualitativ als Hinweis auf Machbarkeit, Durchführbarkeit und hinsichtlich der Akzeptanz bei den Lernenden zu interpretieren. Dafür wird davon ausgegangen,

> „[...] dass die Teilnahme im Feld Empathie und Identifikation mit den Untersuchungspersonen voraussetzt, da erst so die Interpretationsprozesse der Untersuchungspersonen erfasst und verstanden werden können" (Atteslander et al. 2006, S. 94).

Die Teilnahme des Forschers an der qualitativen Studie ist mit Vorteilen verbunden, obwohl im Gegensatz zu einer quantitativen Untersuchung Kritik an der Repräsentativität geübt wird. Die Vorteile überwiegen jedoch:

> „Eine solche Kritik verkennt aber die genuinen Vorzüge dieser Methode, denn qualitativ-teilnehmende Beobachtungen zeichnen sich gegenüber anderen Methoden ja gerade durch die Authentizität der gewonnen Daten aus" (Atteslander et al. 2006, S. 95).

Insgesamt muss die Auswertung der Unterrichtsinterventionen in dem Bewusstsein geschehen, dass quasi-experimentelle Untersuchungen

> „[...] mehr Erklärungsvarianten zu[lassen] als die Ergebnisse reiner experimenteller Untersuchungen, d. h. sie haben eine geringere interne Validität" (Bortz und Döring 2002, S. 527).

Auf Kontrollgruppenbildung wird einerseits aus Kapazitätsgründen verzichtet, andererseits würden die kleinen Fallzahlen auch bei hohen Abweichungen keinen Nachweis von Signifikanzen im Paarvergleich zulassen (Holl 2003), der wiederum bei Gruppengrößen von mehr als zehn Schülern kaum noch zu bewältigen ist (Bortz und Döring 2002, S. 162). Darüber hinaus müssten Vergleichsgruppen möglichst von der gleichen Lehrperson zu ähnlichen Tageszeiten unterrichtet werden, eine ähnliche Verteilung der Jungen und Mädchen aufweisen sowie ein ähnliches Leistungspotenzial besitzen (Randomisierung). Letzteres ließe sich nur durch künstliche Zusammenstellung der Lerngruppen, z. B. durch Abfrage der Durchschnittsnoten in relevanten Fächern, gewährleisten. Auf eine Abfrage der Computernutzungsstrategien und Einstellungen zur Rechnernutzung wird verzichtet. Damit geht eine Minderung der internen Validität einher. Deshalb wird exemplarisch die Methode des Laut-Denkens außerhalb der Unterrichtserprobungen eingesetzt, um Problemlösestrategien der Schüler bei umfangreichen Anforderungssituationen anhand von Gedankenstichproben transparent zu machen (Bortz und Döring 2002, S. 324). Wie alle Laborstudien weist Laut-Denken im Gegensatz zur Feldstudie eine höhere interne, aber niedrigere externe Validität auf (Bortz und Döring 2002, S. 117).

Die Daten der Lernerfolgskontrolle werden dokumentiert. Aufgrund ihrer hohen (Auswertungs-) Objektivität (Holl 2003) werden die in der modernen Testkonstruktion vorherrschenden Multiple-Choice-Aufgaben (Bortz und Döring 2002, S. 214) für Lernerfolgskontrollen genutzt. Die Kompetenzmessung mit solchen Lernerfolgskontrolle ist jedoch umstritten, da Wiedererkennung von Anforderungen eine große Rolle spielt (Bortz und Döring 2002, S. 214). Somit sind entweder auch offene Anforderungssituationen in der Lernerfolgskontrolle oder unter Beobachtung durch die Forschungsperson im Unterricht zu stellen, um Einblick in die Schülervorgehensweisen zu erhalten. Auf einen Eingangstest zu Beginn der Unterrichtserprobung wird verzichtet, da bereits dieser Test Lernen und Sensibilisierung auslösen würde. Dieser so genannte Pretest-Effekt schränkt insbesondere die externe Validität ein,

also die Generalisierbarkeit auf vergleichbare Situationen (Bortz und Döring 2002, S. 505). Außerdem ist darauf hinzuweisen, dass standardisierte Leistungstests und Messinstrumente fehlen, so dass Leistungszuwächse hinsichtlich Informatiksysteme und Kompetenzentwicklung zurzeit empirisch nicht valide zu evaluieren sind. Da aus Lösungen der schriftlichen Tests und von Aufgaben meist nicht mehr erkennbar ist, welche Lösungsstrategien eingesetzt wurden, ist in den Unterrichtsinterventionen besonderer Wert auf Beobachtungen der hospitierenden Forscher hinsichtlich der Schülerarbeitsweisen gelegt worden.

Im Anschluss an Unterrichtserprobungen wird ein Leitfaden-Interview mit dem Fachlehrer durchgeführt und eine Tonaufzeichnung vorgenommen. Damit ist nachträglich der Einfluss des Interviewers, ggf. als Fehlerquelle, feststellbar. Interviewer ist die Forschungsperson, da die befragende Person ausreichend über das Thema informiert sein muss, um flexibel auf den Befragten eingehen zu können (Bortz und Döring 2002, S. 308). Zu beachten ist, dass der Interviewer gleichzeitig „Erhebungsinstrument" ist, so dass seine Gedanken, Reaktionen und Analysen berücksichtigt werden können, aber auch Beeinflussung der Fragestellungen und der an ihn gerichteten Antworten zu erwarten sind, d. h. die Interaktion von Interviewer und Befragtem basiert auf wechselseitigen Erwartungshaltungen. Beeinflussung der Ergebnisse sowie Verfälschungen der Erhebungssituationen durch Tonaufzeichnungen sind in der Regel nicht zu erwarten. Im Gegenteil kann davon ausgegangen werden, dass sich die Mitarbeit des Interviewten erhöht. Transkriptionen werden wegen des geringeren Aufwands in einer „geglätteten" Variante angefertigt, sofern es das theoretische Interesse erlaubt (Bortz und Döring 2002, S. 312).

Die schriftliche Befragung zu Akzeptanz und Selbsteinschätzung der Schüler findet einmalig nach der Lernerfolgskontrolle statt. Schriftliche Befragungen erfordern hohe Strukturierbarkeit der Inhalte und eignen sich besonders zur Befragung homogener Gruppen (Bortz und Döring 2002, S. 253). Die Ergebnisse der Lernerfolgskontrolle sind den Schülern vorher nicht bekannt, um übermäßig positive und negative Rückmeldungen zu vermeiden. Dennoch ist die subjektive Einschätzung des Schwierigkeitsgrads der Lernerfolgskontrolle ein Einflussfaktor auf die Akzeptanzbefragung und Selbsteinschätzung. Aus Gründen des Persönlichkeitsschutzes werden die Daten anonymisiert erhoben und wegen der geringen Stichprobengröße nicht vollständig veröffentlicht. Denn bei der geringen Gruppengröße können personenbezogene Daten, wie Geschlecht und Alter Rückschlüsse auf einzelne Schüler ermöglichen.

Die intervenierende Fachdidaktik ist auf subjektive Aussagen der Lerngruppenmitglieder angewiesen, um Motivation, Einstellungen und Bereitschaften als wichtige Kompetenzfacetten einschätzen zu können. Es wird erfasst, wie die Schüler den Lebensweltbezug des Unterrichts und des Lerngegenstands einschätzen, da diese Kenngrößen nicht in der Lernerfolgskontrolle erfassbar sind (Holl 2003). Durch die

Anwesenheit des Kurslehrers bei der Beantwortung des Akzeptanzfragebogens werden eine hohe Rücklaufquote gesichert und Stichprobenfehler fast ausgeschlossen (Bortz und Döring 2002, S. 253). Die Abwesenheit der Forschungsperson bei der Beantwortung minimiert die Beeinflussung während der Befragungssituation. Die Selbsteinschätzung der Schüler erreicht jedoch nicht die Objektivität, wie sie durch Fremdevaluation in einer Lernerfolgskontrolle möglich ist. So wird bei der schriftlichen Befragung vorausgesetzt, dass die Schüler ihre Leistung erfassen und auf eigene Lernprobleme aufmerksam werden. Aussagen lassen sich auch aus relativen Veränderungen der Einschätzungen zwischen den unterschiedlichen Unterrichtserprobungen treffen.

Die Evaluation des Konzeptes durch intervenierende Fachdidaktik impliziert, möglichst viele Personen in die theoretische und praktische Entwicklung des Unterrichtsmodells und die Diskussion von Wertentscheidungen zu den normativen Voraussetzungs-Ziel-Mittel-Aussagen einzubeziehen (Tulodziecki und Herzig 1998, S. 29). Dies kann geschehen durch Präsentation der Ergebnisse auf nationalen und internationalen Fachtagungen, durch Einbeziehen von Hinweisen hospitierender Fachlehrer und Lehramtsstudierender während der Unterrichtserprobungen oder durch Lehrerfortbildungen und Diskussion der Ergebnisse mit Studierenden in einer forschenden Lehre. Die Stärke der intervenierenden Fachdidaktik wird in der Ausarbeitung und Erkundung von komplexen Unterrichtssituationen gesehen, die den Weg für hypothesenprüfende experimentelle Lehr-Lernforschung weist (Abschnitt 9.3).

1.3 Gliederung der Arbeit

Kapitel 2: Einordnung in die Kompetenzdiskussion

Die Forschungsmethodik wird auf die Gliederung der Dissertationsschrift abgebildet (vgl. Abbildung 1.1). Abbildung 1.2 zeigt die schematische Struktur der Arbeit. In Kapitel 2 wird die vorliegende Arbeit in die aktuelle Diskussion um Kompetenz eingeordnet.

Kapitel 3: Der Informatiksystembegriff

In Kapitel 3 wird der fachwissenschaftliche Begriff des Informatiksystems beschrieben (Abschnitt 3.2). Dafür ist es notwendig, den aktuellen Informatiksystembegriff in den unterschiedlichen Bereichen der Kerninformatik zu untersuchen. Zusammen mit dem Kompetenzbegriff können Analysekriterien formuliert werden (Abschnitt 2.1), anhand derer der nationale und internationale Stand der Forschung untersucht werden kann. Ziel des Kapitels ist es, durch eine Konkretisierung des Informatiksystembegriffs und einer Strukturierung von Kompetenzen aus der Fachwissenschaft

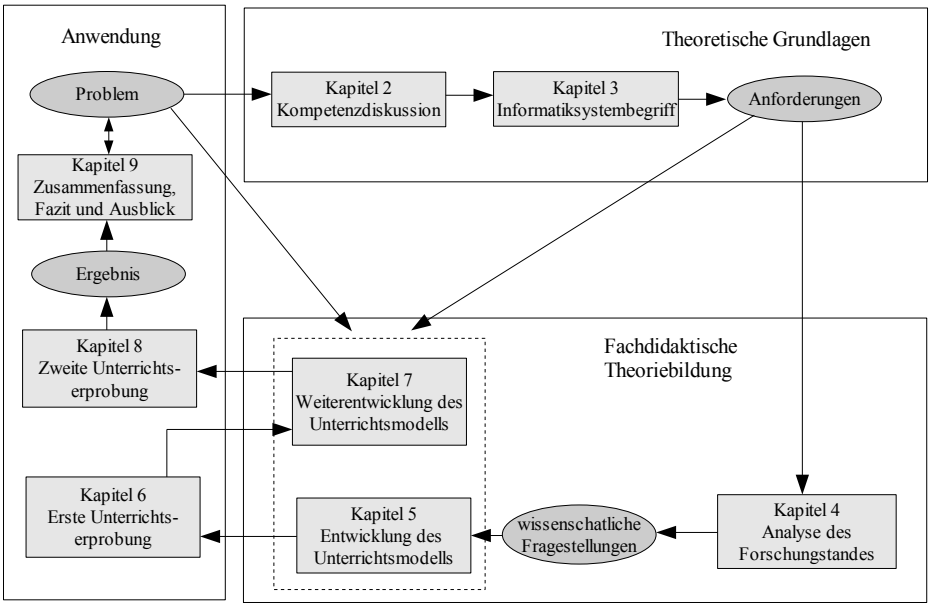


Abbildung 1.2: Schematische Struktur der Arbeit

heraus die Analyse des fachdidaktischen Forschungsstandes vorzubereiten. Es werden erste Schlussfolgerungen und wissenschaftliche Fragestellungen zur Erstellung eines Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung abgeleitet.

Kapitel 4: Stand der Forschung zu Informatiksystemen und Kompetenzentwicklung

In Kapitel 4 werden anhand der in Kapitel 3 ausgearbeiteten Kriterien der nationale und internationale Stand der Forschung zu Informatiksystemen und Kompetenzentwicklung in der Schulinformatik analysiert (Abschnitte 4.2 und 4.3). Ziel des Kapitels ist es, eine Clusterung der Bildungsziele, -inhalte, Lehr-Lernmethoden und Lernmedien aus der fachdidaktischen Diskussion für Kompetenzentwicklung mit Informatiksystemen vorzunehmen. Dabei wird einerseits die Forschungslücke offen gelegt und andererseits werden weitere Schlussfolgerungen und wissenschaftliche Fragestellungen zur Erstellung eines Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung abgeleitet.

Kapitel 5: Vorgehensweise zur Entwicklung eines Unterrichtsmodells am Beispiel Informatiksysteme und Kompetenzentwicklung

In Kapitel 5 wird die Entwicklung eines Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung vorgestellt, um die in Kapitel 4 herausgearbeiteten Problemstellen zu bewältigen. Zuerst werden die Ziele des Unterrichtsmodells beschrieben (Abschnitt 5.2). Anschließend werden die Ergebnisse aus der Analyse des Forschungsstandes fokussiert, indem eine Strategie zur Strukturierung im Rahmen des Unterrichtsmodells vorgestellt wird (Abschnitt 5.3). Der Autor begründet mit Erkenntnissen der Lernpsychologie, Informatik und Informatikdidaktik, dass Entwurfsmuster informatikspezifische, extern darstellbare Wissensrepräsentationen von vernetzten fundamentalen Ideen der Informatik sind (Abschnitt 5.4). Einige nach fachdidaktischen Kriterien ausgewählte Entwurfsmuster sind gleichzeitig Lernmittel und Lerngegenstand. Damit können sowohl spezifische Schülervorgehensweisen zur Erkundung von Informatiksystemen vorgestellt (Abschnitt 5.5) als auch die Erstellung eines Konzepts für die Nutzung von Lernmedien, speziell Lernsoftware, anhand von Wissensrepräsentationen skizziert werden (Abschnitt 5.6). Die Lernsoftware dient zur Überwindung von Fehlvorstellungen und kognitiven Barrieren. Neben der theoretischen Grundlage durch Entwurfsmuster als Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik sind die Erkenntnisse aus dem Einsatz von Modulprototypen in der ersten Unterrichtserprobung maßgeblich. Ziel des Kapitels ist es, Intentionen des Unterrichtsmodells sowie die Strukturierung der Unterrichtsinhalte und Unterrichtsmethoden in der Form auszuarbeiten, dass sie exemplarisch für das Unterrichtsmodell in konkreten Unterricht überführt werden können.

Kapitel 6: Erste exemplarische Erprobung des Unterrichtsmodells

In Kapitel 6 wird eine erste Unterrichtserprobung beschrieben. Neben der exemplarischen Auswahl der Unterrichtsinhalte ist eine Anpassung an Rahmenbedingungen einschließlich Vorwissen der Schüler in der Sekundarstufe II erforderlich (Abschnitt 6.3). Problemstellen des Unterrichts werden erläutert (Abschnitt 6.4). Am Ende des Kapitels stehen die formativ gewonnenen Daten aus Unterrichtsbeobachtung, Lernerfolgskontrolle, Akzeptanzbefragung der Schüler und Leitfadeninterview mit dem Klassenlehrer (Abschnitt 6.5).

Kapitel 7: Weiterentwicklung, Verfeinerung und Ergänzung des Unterrichtsmodells

Die durch die erste Erprobung gewonnenen Daten und Erfahrungen werden ausgewertet. Sie münden in eine feinere Strukturierung der Basiskompetenzen (Abschnitt 7.2). Außerdem wird wegen der Überlagerung von Fehlvorstellungen und kognitiven Barrieren der Kompetenzentwicklung mit Informatiksystemen durch typische

Fehlvorstellungen der Objektorientierung eine exemplarische Analyse der systematischen Erkundung mittels Laut-Denken vorgenommen. Dadurch ergibt sich eine Überarbeitung der Unterrichtmethodik der systematischen Erkundung (Abschnitt 7.4). Die didaktischen Klassifizierungen von Entwurfs- und Architekturmustern als Wissensrepräsentationen werden weiterentwickelt ((Ufer 2007), (Weyer 2007b)).

Kapitel 8: Zweite exemplarische Erprobung des Unterrichtsmodells

In Kapitel 8 wird abschließend eine zweite Unterrichtserprobung vorgestellt. Die ergänzenden Erfahrungen mit dem bereits erprobten und überarbeiteten Unterrichtsmodell unterstützen die Evaluation des Lehr-Lernprozesses, für die auch eine Analyse der in dem Informatikkurs gestellten Aufgaben erfolgt. Aus den Untersuchungen kann auf die erworbene Kompetenz zu Informatiksystemen der beteiligten Schüler geschlossen und damit die Qualität des überarbeiteten Unterrichtsmodells bewertet werden.

Kapitel 9: Zusammenfassung, Fazit und Ausblick

Den Abschluss bildet Kapitel 9 mit Zusammenfassung, Fazit und Ausblick auf weitere Forschung.

2. Einordnung in die Kompetenzdiskussion

2.1 Der Kompetenzbegriff

2.1.1 Kompetenzdefinition

Ziel des Forschungsprojektes des Autors ist es, Vorarbeit zur Entwicklung eines Kompetenzmodells zu leisten. Dafür wird eine Strukturierung von Basiskompetenzen vorgenommen, die ein Schüler für Handlungen mit Informatiksystemen im Alltag benötigt. Ausgelöst wurde die aktuelle Kompetenzdiskussion vor allem durch die PISA-Studie (Programme for International Student Assessment) der OECD (Organisation for Economic Co-operation and Development). Die in der empirischen Bildungsforschung zentrale Definition von Kompetenz nach Weinert (2001) wird in der vorliegenden Arbeit verwendet, da sie vielfältige Einflussfaktoren auf erfolgreiches Bewältigen von Anforderungssituationen erfasst. Nicht zuletzt durch die Klieme-Expertise (Klieme et al. 2007) hat sie großen Einfluss auf die aktuelle Diskussion um nationale Bildungsstandards und verbindliche Anforderungen an das Lehren und Lernen in der Schule. Durch die Verbindung von Kompetenzeigenschaften (Kapitel 2) mit Merkmalen von Informatiksystemen (Kapitel 3) kann die nationale und internationale fachdidaktische Diskussion auf Basiskompetenzbeschreibungen analysiert werden (Kapitel 4).

Der Kompetenzbegriff nach Weinert (2001) bzw. Klieme et al. (2007) ist:

> „In Übereinstimmung mit (Weinert 2001, S. 27f) verstehen wir unter Kompetenzen die bei Individuen verfügbaren oder von ihnen erlernbaren kognitiven Fähigkeiten und Fertigkeiten, bestimmte Probleme zu lösen, sowie die damit verbundenen

> motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können" (Klieme et al. 2007, S. 70).

Kompetenzen stützen sich auf Handlungen im Alltag, die in der vorliegenden Arbeit auf Tätigkeiten mit Informatiksystemen bezogen werden. Deshalb sind Problemsituationen zu betrachten, die entweder durch den Einsatz von Informatiksystemen bewältigt werden können oder die durch Informatiksysteme ursächlich ausgelöst wurden (Abbildung 2.1).

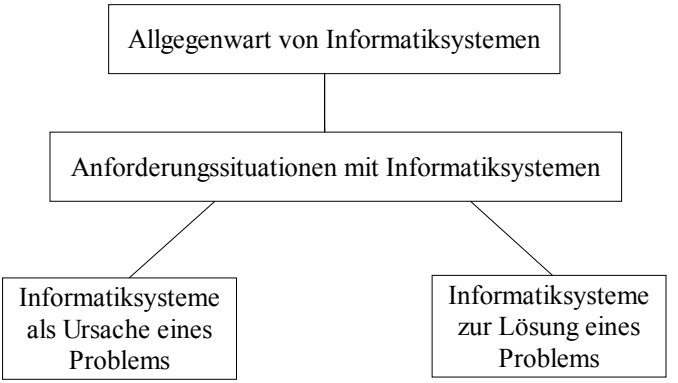


Abbildung 2.1: Anforderungssituationen zur Kompetenzentwicklung mit Informatiksystemen

Kompetenzen beziehen sich demnach auf komplexe Anforderungssituationen und bestehen nicht allein aus einzelnen Fähigkeiten und Fertigkeiten. Sie beinhalten neben kognitiven auch nicht-kognitive, motivationale, willensmäßige, personale und sozial-kommunikative Aspekte, die zum selbständigen Lösen domänenrelevanter Probleme notwendig sind. Vor allem die Bereitschaft, ein Problem zu lösen, gehört zur Kompetenz:

> „Die Kompetenzorientierung steht für den Anspruch, dass die Ergebnisse schulischen Lernens handlungsrelevant, praktisch anwendbar sowie persönlich und gesellschaftlich bedeutsam sein sollen" (Heymann 2004, S. 8).

Somit gehen die Zielvorstellungen von den Anforderungen der Lebenswelt aus. Weinert unterscheidet folgende, zusammenwirkende Kompetenzfacetten: Fähigkeit, Wissen, Verstehen, Können, Handeln, Erfahrung, Motivation (vgl. Klieme et al. 2007, S. 73).

Ausgeschlossen wird durch den Kompetenzbegriff die Fokussierung auf Zertifikatskurse bzw. Qualifikation, die nicht das Verhalten von Personen in Anforderungssituationen, sondern die erfolgreiche Bewältigung einer Prüfungssituation beurteilt.

Nicht die traditionellen Kenntnisse und Fähigkeiten stehen bei PISA der OECD im Vordergrund, sondern deren Anwendung in unterschiedlichen Kontexten:

> „Der Begriff *Grundbildung (literacy)* wurde gewählt, um zu betonen, dass mathematische Kenntnisse und Fähigkeiten, wie sie im traditionellen Curriculum der Schulmathematik definiert werden, im Rahmen von OECD/PISA nicht im Vordergrund stehen. Statt dessen liegt der Schwerpunkt auf der funktionalen Anwendung von mathematischen Kenntnissen in ganz unterschiedlichen Kontexten und auf ganz unterschiedliche, Reflexion und Einsicht erfordernde Weise" (OECD 2000, S. 47; Hervorh. im Original)

Bei PISA wurden jedoch nur Lesekompetenz sowie mathematische und naturwissenschaftliche Grundbildung definiert. Für die Informatik fehlt etwas Vergleichbares, obwohl der Bedarf formuliert wird:

> „Aber als generelle Prämisse für die Teilhabe an gesellschaftlicher Kommunikation reicht die Dimension der einfachen Kulturtechniken nicht mehr aus. Die Heranwachsenden müssen vielmehr fähig werden für den Gebrauch der Computer, für den Umgang mit Medien, für die Herausforderungen einer multikulturellen Welt, und sie müssen zugleich in der Form der Welterfahrung von den einfachen Formen des Ich-zentrierten Umgangs mit Welt auf die grundlegenden wissenschaftlichen Modi der Welterfahrung übergehen können" (Klieme et al. 2007, S. 67).

In Anlehnung an die OECD-Definition der mathematischen Grundbildung kann formuliert werden: Kompetenzentwicklung mit Informatiksystemen beinhaltet also, dass Schüler in die Lage versetzt werden, die Rolle, die Informatiksysteme in der Welt spielen, zu erkennen und zu verstehen, begründete informatische Urteile zu Informatiksystemen abzugeben und sich auf eine Weise mit Informatiksystemen zu befassen, die den Anforderungen ihres gegenwärtigen und künftigen Lebens als eines konstruktiven, engagierten und reflektierenden Bürgers entspricht (vgl. OECD 2000, S. 14).

Somit müssen Schüler in der Lage sein, Zusammenhänge und Strukturen von Informatiksystemen in realitätsnahen Situationen zu erkennen und diese beurteilen zu können. Kompetenzentwicklung mit Informatiksystemen führt demzufolge zu einer kritisch-reflektierten Auseinandersetzung mit Informatiksystemen, die gesellschaftliches Engagement fördert.

Durch den oben genannten Kompetenzbegriff ergeben sich wichtige Implikationen. So ist es erforderlich, Kompetenzen nicht ausschließlich normativ aus der Fachsystematik der Informatik abzuleiten; vielmehr ist auch die psychologische Komponente von Kompetenz durch empirische Untersuchungen zu berücksichtigen. Durch die in Kapitel 1 beschriebenen Restriktionen geschieht das in der vorliegenden Arbeit hinsichtlich Machbarkeit und Plausibilitätserklärungen. In weitergehende Arbeiten kann Kompetenzentwicklung als gradueller, gestufter Prozess zum Erwerb von Expertise zur Bewältigung komplexer Anforderungssituationen empirisch fundiert werden (Abschnitt 9.3).

Im Folgenden werden weitere Konzepte von Kompetenz aufgeführt und bezüglich Informatiksysteme diskutiert. Diese sind Schlüsselkompetenzen gemäß OECD, Literacys zu Informatiksystemen gemäß UNESCO und der Europäische Qualifikationsrahmen für lebenslanges Lernen. Wegen starker Überschneidungen mit den genannten werden die technologiebezogenen Bildungsstandards für Schüler „National Educational Technology Standards and Performance Indicators for Students" der ISTE (International Society for Technology in Education) nicht betrachtet. Diese sind: (1) Creativity and Innovation, (2) Communication and Collaboration, (3) Research and Information Fluency, (4) Critical Thinking, Problem Solving, and Decision Making (5) Digital Citizenship, and (7) Technology Operations and Concepts (ISTE 2007). Der Beitrag des Unterrichtsmodells für Kompetenzentwicklung mit Informatiksystemen bzw. der Unterrichtserprobungen wird in den Abschnitten 5.7, 6.6.2 und 8.5.2 analysiert.

2.1.2 Schlüsselkompetenzen

Es ist die Frage zu stellen, wie Kompetenzentwicklung mit Informatiksystemen zur Ausprägung von Schlüsselkompetenzen beiträgt. Die OECD hat in dem Bericht „Definition and Selection of Key Competencies (DeSeCo)" drei Bereiche für Schlüsselkompetenzen definiert:

1. Interaktive Anwendung von Medien und Mitteln (Tools),
2. Interagieren in heterogenen Gruppen,
3. Eigenständiges Handeln (OECD 2005).

Durch die steigenden Anforderungen im Alltag der Wissensgesellschaft sind Schlüsselkompetenzen von vorrangiger Bedeutung. Sie umfassen mehr als Wissen und Fertigkeiten, denn sie betonen nicht-kognitive Fähigkeiten wie Bereitschaften und Einstellungen. Schlüsselkompetenzen sind nicht nur für Spezialisten, sondern für alle Individuen notwendig, denn sie stellen eine Voraussetzung für lebensbegleitendes Lernen dar.

Die erste Kompetenzkategorie „Interaktive Anwendung von Medien und Mitteln (Tools)" beinhaltet (1a) interaktive Anwendung von Sprache, Symbolen und Texten, (1b) interaktive Nutzung von Wissen und Informationen sowie (1c) interaktive Anwendung von Technologien. Die Rollen von Informatiksystemen für die Ausprägung dieser Schlüsselkompetenzen liegen auf der Hand: Zur Übertragung von Sprache werden Informatiksysteme genutzt ebenso wie zur Verarbeitung von Symbolen und Texten (DeSeCo: 1a). Die Anwendung von Informatiksystemen als Werkzeug (DeSeCo: 1c) benötigt Wissen und Auseinandersetzung mit zeitgemäßen Möglichkeiten der Anwendung im Alltag durch Individuen, z. B. neue Eingabe- und Ausgabemöglichkeiten von mobilen Systemen. Dementsprechend können moderne

Kommunikationstechnologien wie Informatiksysteme den Arbeitsalltag und die Kooperation zwischen Individuen verändern, z. B. durch zunehmende Unabhängigkeit von einem Ort und weltweite soziale Vernetzung. Die interaktive Nutzung von Wissen und Informationen (DeSeCo: 1b) setzt Kenntnisse zu deren Speicherung und Übertragung in Informatiksystemen voraus.

Die zweite Kompetenzkategorie „Interagieren in heterogenen Gruppen" bezieht sich auf zwischenmenschliche Beziehungen. Entsprechende Schlüsselkompetenzen sind unverzichtbar für pluralistische Gesellschaften. Dazu gehören (2a) gute und tragfähige Beziehungen unterhalten, (2b) Fähigkeit zur Zusammenarbeit und (2c) Bewältigen und Lösen von Konflikten. Bezüglich Informatiksysteme ist zu beobachten, dass sie es unterstützen, Beziehungen über große Entfernungen zu unterhalten (DeSeCo: 2a). Schüler benötigen die Fähigkeit, Beziehungen mit anderen über Informatiksysteme zu initiieren und zu erhalten bei unterschiedlichen Karrieren, Berufen und Hintergründen der Individuen (DeSeCo: 2b). Schlüsselkompetenzen ermöglichen es Individuen, mit anderen angemessen zu kommunizieren und sich in andere Personen hineinzuversetzen. Gerade das Bewältigen und Lösen von Konflikten in Beziehungen, die ohne persönlichen Kontakt auskommen müssen, stellt Menschen vor Schwierigkeiten, z. B. Etikette im Netz, Bewusstsein über die Öffentlichkeit von Kommunikation in Internetforen und Aspekte von Interkulturalität (DeSeCo: 2c).

Die dritte Kompetenzkategorie „Eigenständiges Handeln" besteht aus (3a) Handeln im größeren Kontext, (3b) Realisieren von Lebensplänen und persönlichen Projekten sowie (3c) Verteidigung und Wahrnehmung von Rechten, Interessen, Grenzen und Erfordernissen. Das Handeln im größeren Kontext bedarf des Erkennens von gesellschaftlichen Mustern zur Nutzung von Informatiksystemen und der Einbettung in den jeweiligen Kontext, z. B. Vorratsdatenspeicherung (DeSeCo: 3a). Schüler müssen eigenständig Informatiksysteme einsetzen können, um an unterschiedlichen Projekten innerhalb und außerhalb der Schule aktiv teilnehmen zu können (DeSeCo: 3b). Individuen müssen die Konsequenzen ihres Handeln einschätzen können, um Entscheidungen zu fällen, Verantwortung zu übernehmen und Rechte wahrzunehmen, z. B. E-Government, Fragen des Urheberrechts und E-Learning (DeSeCo: 3c). Damit unterstützen Schlüsselkompetenzen zu Informatiksystemen die Ausprägung von Selbstvertrauen.

Fazit ist, dass Schlüsselkompetenzen insbesondere die nicht-kognitiven Elemente von Kompetenz betonen. Informatiksysteme determinieren neue Handlungsoptionen, die zur Ausprägung von Schlüsselkompetenzen kreativ genutzt werden müssen.

2.1.3 ICT Literacy und Information Literacy

In der UNESCO-Publikation „Understanding Information Literacy: A Primer" (UNESCO 2008) werden die wichtigsten „Literacys" aufgeführt, die mündige Bürger im 21. Jahrhundert benötigen:

> „The family of 21st Century 'survival literacies' includes six categories: (1) the Basic or Core functional literacy fluencies (competencies) of reading, writing, oralcy and numeracy; (2) Computer Literacy; (3) Media Literacy; (4) Distance Education and E-Learning; (5) Cultural Literacy; and (6) Information Literacy.
> The boundaries between the various members of this family overlap, but they should be seen as a closely-knit family" (UNESCO 2008, S. 3).

Hinsichtlich des Einsatzes von Informatiksystemen werden im Folgenden „Computer Literacy", „Media Literacy" und „Information Literacy" kurz vorgestellt.

Die Computer Literacy bildet zusammen mit der Media Literacy die „ICT Literacy" (Information and Communication Technology; (UNESCO 2008, S. 5)). Letztere wird in Kapitel 4 bei der Analyse internationaler Curricula auf ihren Beitrag zur Kompetenzentwicklung mit Informatiksystemen betrachtet. Computer Literacy wiederum wird weiter unterteilt in „Hardware Literacy", „Software Literacy" und „Applications Literacy". Diese Einteilung korrespondiert stark mit der Definition von Informatiksystemen gemäß Duden Informatik ((Claus und Schwill 2006, S. 314); Abschnitt 3.2.1).

Hardware Literacy beinhaltet Basisfähigkeiten, um einen Rechner oder ein mobiles Endgerät effizient anwenden zu können. Dabei bezieht sie sich auf die sichtbaren und greifbaren Komponenten und deren Nutzung, z. B. traditionelle Eingabe- und Ausgabegeräte wie Maus, Tastatur und Drucker. Hardware Literacy ist somit in vielen Fällen eine Grundlage für Mensch-Maschine-Interaktion. Neue Interaktionsformen, z. B. Gestenerkennung, die über Sensoren geschieht, zeigen jedoch die Grenzen des Begriffs der Hardware Literacy, da das Anfassen und die Sichtbarkeit der Komponenten zur Interaktion nicht mehr erforderlich sind.

Im Gegensatz dazu betrifft Software Literacy die Basisfähigkeiten zur gezielten Nutzung von Software, die ein Informatiksystem zur Lösung eines allgemeinen Anwendungsproblems benötigt. Darunter fällt in erster Linie das Betriebssystem, das auf allen Rechnern existiert, aber auch typische Anwendungssoftware wie Textverarbeitungssysteme, Tabellenkalkulationen und E-Mail-Programme. Trotz des Hinweises auf E-Mail und das Internet steht ein isolierter Einzelplatzrechner im Vordergrund, und Computer bzw. Software Literacy müssen bezüglich Informatiksysteme im Zusammenspiel mit den weiteren Kompetenzen (Literacies) gesehen werden.

In Abgrenzung zur Software Literacy wird Applications Literacy definiert als Wissen und Können, um spezielle Anwendungssoftware einzusetzen. Darunter sind beispielsweise firmenspezifische Finanz- oder Personalverwaltungssoftware zu verstehen, aber auch Software zur Unterstützung von Arbeitsabläufen und weitere domänenspezifische Software. Für Basiskompetenzen zu Informatiksystemen an allgemein bildenden Schulen wird Applications Literacy deshalb in dieser Arbeit nicht weiter betrachtet.

Media Literacy hingegen betont die kritische Auseinandersetzung mit Medien aller Art einschließlich Informatiksystemen und deren kreative Anwendung. Sie wird in drei Bereiche unterteilt:

> „[...] media literacy implies having access to the media, understanding the media and creating/expressing oneself using the media" (UNESCO 2008, S. 6).

Information Literacy meint Wissen, Fähigkeiten und Einstellungen, um zu entscheiden, wann zusätzliche Information notwendig ist, wie Information effizient gefunden wird und wie sie zu interpretieren ist. Dazu zählt die Einschätzung von Glaubwürdigkeit und Authentizität. Information Literacy ist eng verknüpft mit kritischem Denken und dem „Lernen-zu-Lernen", z. B. der Nutzung von E-Learning-Angeboten. Für den Informatikunterricht sind insbesondere der kritische Umgang mit Information und die Informationssuche in Netzen wichtige Anknüpfungspunkte.

Für die vorliegende Arbeit gilt, dass diese Kompetenzen auf Informatiksysteme bezogen und im Rahmen einer informatischen Bildung diskutiert werden.

2.1.4 Europäischer Qualifikationsrahmen für lebenslanges Lernen

Der „Europäische Qualifikationsrahmen für lebenslanges Lernen" (EQR) verknüpft die Qualifikationssysteme verschiedener Länder miteinander, indem er als Referenzrahmen „Übersetzungsinstrument" für Qualifikationen ist (EU 2008). Er soll Mobilität zwischen unterschiedlichen Ländern und Bildungssystemen unterstützen, indem er Ausbildungen vergleichbar macht. Im EQR werden dafür acht Referenzniveaus unterschieden. Als Beispiel wird für Niveau 1 ein Schulabschluss, für Niveau 8 die Promotion genannt. Durch diese weite Spanne ist der EQR geeignet, lebensbegleitendes Lernen zu erfassen. Für Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II bedeutet dies im Umkehrschluss, dass nicht alle Niveaustufen zu betrachten sind. Eine Einordnung der Kompetenzentwicklung mit Informatiksystemen im Verlauf der Sekundarstufe II ermöglicht jedoch eine Vergleichbarkeit zwischen Ausbildungsgängen. Die Niveaus beziehen sich explizit auf Lernergebnisse, nicht auf Inhalte. Die Lernergebnisse wiederum werden unterteilt in Kenntnisse, Fähigkeiten und Kompetenz. Während Kenntnisse Theorie- und Faktenwissen umfassen und Fertigkeiten sowohl kognitiv als auch praktisch sein können, ist Kompetenz als Übernahme von Verantwortung und Selbständigkeit definiert (EU 2008, S. 11).

Bei den Kenntnissen lassen sich viele Bezüge zu technischen Aspekten von Informatiksystemen und deren Anwendungsgebiet herstellen (EU 2008, S. 12): Niveau 1 entspricht einem grundlegenden Allgemeinwissen und Niveau 2 einem grundlegendes Faktenwissen in einem Arbeits- oder Lernbereich. Niveau 3 beinhaltet Kenntnisse

von Fakten, Grundsätzen, Verfahren und allgemeinen Begriffen in einem Bereich und Niveau 4 ein breites Spektrum an Theorie- und Faktenwissen. Ab Niveau 5 wird eine metakognitive Ebene adressiert, d. h. die Grenzen ihrer Kenntnisse sind den Lernenden bewusst.

Bei den Fertigkeiten kann vornehmlich auf Informatiksysteme als Werkzeug eingegangen werden (EU 2008, S. 13): Auf Niveau 1 sind grundlegende Fertigkeiten gemeint, die zur Ausführung einfacher Aufgaben erforderlich sind. Niveau 2 umfasst grundlegende kognitive und praktische Fertigkeiten, die zur Nutzung relevanter Informationen erforderlich sind, um Aufgaben auszuführen und Routineprobleme unter Verwendung einfacher Regeln und Werkzeuge zu lösen. Niveau 3 beinhaltet eine Reihe kognitiver und praktischer Fertigkeiten zur Erledigung von Aufgaben und zur Lösung von Problemen, wobei grundlegende Methoden, Werkzeuge, Materialien und Informationen ausgewählt und angewandt werden. Ab Niveau 4 werden speziellere, abstraktere und komplexere Probleme im Arbeits- oder Lernbereich adressiert, die kreative Lösungen erfordern.

Bei den Kompetenzniveaus wird die Einbettung der Kenntnisse und Fertigkeiten in einen sozialen Kontext beschrieben (EU 2008, S. 13). Handlungen mit Informatiksystemen sind somit im sozio-technischen Kontext zu diskutieren: Kompetenzen auf Niveau 1 umfassen das Arbeiten oder Lernen unter direkter Anleitung in einem vorstrukturierten Kontext, auf Niveau 2 das Arbeiten oder Lernen unter Anleitung mit einem gewissen Maß an Selbstständigkeit, z. B. Anwendung von Informatiksystemen zur Informationsbeschaffung. Niveau 3 beinhaltet die Übernahme von Verantwortung für die Erledigung von Arbeits- oder Lernaufgaben und das Anpassen des eigenen Verhaltens an die jeweiligen Umstände bei der Lösung von Problemen, z. B. Auswahl eines geeigneten Informatiksystems. Auf Niveau 4 ist selbstständiges Tätigwerden innerhalb der Handlungsparameter von Arbeits- oder Lernkontexten, die in der Regel bekannt sind, sich jedoch ändern können, ein Lernergebnis. Außerdem ist ein Lernender in der Lage, die Beaufsichtigung der Routinearbeit anderer Personen zu übernehmen, wobei eine gewisse Verantwortung für die Bewertung und Verbesserung der Arbeits- oder Lernaktivitäten übernommen wird. Das heißt, Schüler müssen in der Lage sein, Informatiksysteme und Handlungen anderer mit Informatiksystemen bei Routineaufgaben bewerten zu können. Niveau 5 beinhaltet das Leiten und Beaufsichtigen in Arbeits- oder Lernkontexten, in denen nicht vorhersehbare Änderungen auftreten, sowie die Überprüfung und Entwicklung der eigenen Leistung und der Leistung anderer Personen. Somit ist ein (selbst-) kritischer Umgang mit Informatiksystemen notwendig, und Lernende müssen Informatiksysteme zu komplexen Lernumgebungen zusammenstellen. Ab Niveau 6 steht die Übernahme von Personalverantwortung und Gestaltung komplexer Arbeits- und Lernkontexte im Vordergrund.

2.1.5 Kompetenzstufung

Im EQR erfolgt eine sehr differenzierte Unterscheidung von Niveaustufen, um Kenntnisse, Fertigkeiten und Kompetenzen des lebensbegleitenden Lernens einordnen zu können. Für die Sekundarstufe II sind insbesondere Lernzieltaxonomien geeignet, um Stufungen vorzunehmen, da mit ihnen reichhaltige Erfahrungen vorliegen. Weit verbreitet ist eine Unterteilung in die Anforderungsbereiche Wissen, Anwenden und Gestalten, wie sie beispielsweise in den Anforderungsbereichen I, II und III der Einheitlichen Prüfungsanforderungen in der Abiturprüfung (EPA) im Fach Informatik vorgenommen wird (KMK 2004). Etwas genauer ist die von Anderson und Krathwohl überarbeitete Bloom'sche Taxonomie (Anderson und Krathwohl 2001), die sechs gestufte kognitive Prozesse bzw. Erkenntnisdimensionen anspricht: Erinnern, Verstehen, Anwenden, Analysieren, Bewerten und (Er)schaffen (vgl. Schobel und Holdt 2004). Die informatikspezifische Lernzieltaxonomie nach Fuller et al. (2007) bezieht sich auf Andersons und Krathwohls Taxonomie, ordnet die kognitiven Prozesse jedoch zweidimensional an. Anwenden und Gestalten bilden darin eine eigene Dimension, da ihnen für die Informatik ein besonderes Gewicht beigemessen wird:

> „We can also distinguish between disciplines in which there is an emphasis on learning through interpreting and those in which learning is predominantly achieved through doing. [...] Computing students are expected to do a lot of learning through doing, whether it is learning about software engeneering by developing systems of increasing complexity, learning about networking by implementing protocols or learning about group dynamics by working in teams" (Fuller et al. 2007, S. 163).

Produzieren	6. (Er)schaffen				
	3. Anwenden				
	∅				
		1. Erinnern	2. Verstehen	4. Analysieren	5. Bewerten
		Interpretieren			

Abbildung 2.2: Lernzielebenen der informatikspezifischen Lernzieltaxonomie nach (Fuller et al. 2007, S. 164)

Tabelle 2.2 zeigt die informatikspezifische Lernzieltaxonomie nach Fuller et al. (2007). In der vorliegenden Arbeit werden die sechs kognitiven Prozesse nach Anderson und Krathwohl (2001) bzw. Fuller et al. (2007) verwendet.

2.2 Basiskompetenzen und Bildungsstandards

Der Begriff der Basiskompetenzen wird in der internationalen Diskussion einerseits für Minimalanforderungen in Bildungsstandards und andererseits für untere Stufen eines Kompetenzmodells verwendet (vgl. Klieme et al. 2007, S. 100). Die jeweilige Grenze zwischen Basiskompetenzen und weiteren Kompetenzen wird im Allgemeinen von Expertengremien getroffen. Sie ist meist dadurch begründet, dass Schüler einer Risikogruppe zugeordnet werden, die beispielsweise den angestrebten Schulabschluss oder eine Berufsausbildung vermutlich nicht erreichen wird, wenn sie die Minimalanforderungen nicht erfüllen.

Kompetenzmodelle konkretisieren die Kompetenzbeschreibungen (Klieme et al. 2007, S. 9) und vermitteln wissenschaftlich fundiert zwischen Bildungszielen und konkreten Lehr-Lernprozessen. Ein Kompetenzmodell umfasst normative Vorgaben und empirische Validierung. Operationalisierte Kompetenzen unterscheiden sich von operationalisierten Lernzielen im Wesentlichen nicht in der Formulierung, sondern durch ihren theoretischen Hintergrund, d. h. ein Kompetenzmodell.

Bildungsstandards beschreiben Ergebnisse von Bildung (Output-Orientierung) durch Kompetenzen, die am Ende des Lehr-Lernprozesses gemessen werden können. Klieme et al. (2007) merken an, dass die Orientierung an Grunddimensionen der Kompetenzentwicklung sogar einfacher zu erstellen ist, als ein detaillierter Katalog von Lernzielen und -inhalten (vgl. Klieme et al. 2007, S. 48). Klieme ergänzt jedoch:

> „Wie jede Lehrplanung stellen daher auch Standards einen Kompromiss dar zwischen Orientierung an fachlicher Systematik, an funktionalen Anforderungen der Lebens- und Arbeitswelt und an Lernvoraussetzungen und Entwicklungsbedürfnissen der Lernenden. Allerdings haben sich die Schwerpunkte heute eindeutig zugunsten der funktionalen Anforderungen verschoben" (Klieme 2004, S. 10).

In der bildungstheoretischen Diskussion existiert zurzeit kein Konsens über die angemessene Operationalisierung von Kompetenzen, mehr noch ist es nicht möglich, aus allgemeinen Bildungszielen eindeutig konkrete Lernziele ohne Widerspruch abzuleiten (vgl. Klieme et al. 2007, S. 64). Für die unterrichtspraktische Erprobung des angestrebten Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung werden dementsprechend Lernziele als kleinschrittige Zielfunktionen angegeben und mit den angestrebten Kompetenzen in Verbindung gesetzt bzw. in eine fachdidaktisch begründete Strukturierung der Kompetenzen eingeordnet.

Merkmale von Bildungsstandards sind Fachlichkeit, Fokussierung, Kumulativität, Verbindlichkeit für alle, Differenzierung, Verständlichkeit und Realisierbarkeit (Klieme et al. 2007, S. 24f). Um einen Beitrag zu den Bildungsstandards für die Informatik zu leisten, werden deshalb

1. die Ergebnisse auf den Lernbereich „Kompetenzentwicklung mit Informatiksystemen" bezogen und dafür die Grundprinzipien für das Unterrichtsfach klar herausgearbeitet (Fachlichkeit),
2. Lernziele und Kompetenzbeschreibungen aufeinander aufbauend und miteinander vernetzt beschrieben (Kumulativität),
3. Kompetenzstufungen vorbereitet, die als Grundlage für eine empirische Überprüfung der Stufung dienen können (Differenzierung),
4. die Lernziele und Kompetenzbeschreibungen klar, knapp und nachvollziehbar formuliert (Verständlichkeit),
5. Unterrichtserprobungen des Unterrichtsmodells vorgenommen (Realisierbarkeit).

Fokussierung als weiteres Merkmal von Bildungsstandards wird im Rahmen dieser Arbeit bereits durch die thematische Eingrenzung auf Informatiksysteme und Kompetenzentwicklung unterstützt. Die Verbindlichkeit als Merkmal fordert schulformübergreifende Mindeststandards und geht damit über das Ziel dieser Arbeit, Informatiksysteme und Kompetenzentwicklung für die Sekundarstufe II, hinaus. Die bislang in Lehrplänen und Empfehlungen vorherrschende Beschreibung von Unterrichtsinhalten und Lernzielen wird also um die neue Perspektive der Kompetenzbeschreibung ergänzt. Klieme merkt jedoch an, dass die Forschung zu Kompetenzmodellen noch am Anfang steht:

> „Vor allem fehlen Kompetenzmodelle, welche die Entwicklung über die Jahrgangsstufen hinweg beschreiben können. Was in der Grundschule eine Problemlöseaufgabe ist, könnte ja in der 9. Jahrgangsstufe als Routineaufgabe und in der Sekundarstufe II als alltagsbezogenes Denken gelten. Diese Zuordnung ist jedoch rein hypothetisch; es fehlt an entsprechenden Modellen, die mit Längsschnittdaten belegt sind. Man wird sich daher bei der Arbeit an den Bildungsstandards in vielen Fällen zunächst auf das Erfahrungswissen der Fachdidaktiken stützen" (Klieme 2004, S. 12).

Damit ist die Frage der Aneignung von Informatikwissen genauso bedeutsam wie die Frage, wie zu typischen Prozessen aus der Lebenswelt der Schüler motivierende Lehr-Lernprozesse konstruiert werden können, die Kompetenzentwicklung mit Informatiksystemen fördern. Die von Klieme et al. (2007) geforderte Analyse des Erfahrungswissens der Fachdidaktiken und in der Schulinformatik wird in Kapitel 4 vorgenommen.

2.3 Kompetenz statt „Informatiksystemverständnis"

Ziel der Arbeit ist die Entwicklung und Erprobung eines Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung. Als Arbeitstitel der vorliegenden

Dissertationsschrift wurde lange Zeit der Begriff „Informatiksystemverständnis" bzw. „Verstehen von Informatiksystemen" genutzt, der auch in vielen Publikationen des Autors Verwendung findet. Durch die aktuelle Diskussion zu Kompetenzen und Standards in der informatischen Bildung wurde dieser Begriff jedoch verworfen, da er als undifferenziertes, nicht messbares Lernziel im Widerspruch zur Intention des Kompetenzbegriffs steht. Vielmehr ist er durch den häufigen Gebrauch eine Ursache für die Kompetenzdiskussion.

Um dies zu verdeutlichen und Missinterpretationen zu vermeiden, wird im Folgenden der Verstehensbegriff geklärt. Der Verstehensbegriff wurde auf zwei Ebenen genutzt: Einerseits im Sinne einer Bildungszielbeschreibung und andererseits als kognitiver Prozess bei der Operationalisierung von Lernzielen. Für ersteres, also als grobe Bildungszielbeschreibung im Sinne des Informatiksystemverständnisses, ist der Verstehensbegriff in Anlehnung an Forneck hilfreich, der Ergebnisse der Kognitionspsychologie in die Informatikdidaktik transferiert hat:

> „Ausgangspunkt der Informatik-Grundbildung sind komplexe Situationen, die aufgrund ihrer Komplexität oder aufgrund methodischer Interventionen der Lehrperson kognitive Dissonanzen auslösen [...]. Am Ende eines gelungenen Lernprozesses, wenn Komplexität abgearbeitet ist, steht Verstehen" (Forneck 1997, S. 24f).

Forneck beruft sich auf die kognitionspsychologische Forschung und einen daraus resultierenden veränderten Lernbegriff (Forneck 1997, S. 24f). Verstehen könne dabei als kognitive Konstruktion, d. h. als sinnerzeugender Vorgang gesehen werden, bei dem neues Wissen in existierende Strukturen integriert wird und diese dadurch erweitert und verändert werden. Für den Lernprozess werde das Wissen hierarchisch geordnet und somit schrittweise überprüfbar in Form von konkreten Aufgaben inklusive Bewertung. In der Kognitionspsychologie werde Verstehen dann unterstellt, wenn ein Sachverhalt

> „in das subjektive Handlungs- und Prozeßwissen integriert werden kann" (Aebli 1980, S. 189).

Forneck fordert deshalb, dass der Informatikunterricht phasenweise in die Verantwortung der Schüler übergeht und die Lehrperson dabei auf Kohärenz, Differenziertheit und Vollständigkeit achtet (Forneck 1997, S. 25). Verstehen ist dabei ein nicht abschließbarer, mehrperspektivischer Vorgang, der als „Sehen von Zusammenhängen" bezeichnet wird. Da plötzliche Eingebungen nicht planbar sind, sollten Zusammenhänge grafisch sichtbar gemacht werden (Forneck 1997, S. 25). Abschließend betont Forneck, dass Verstehen ein kontextuell eingebetteter Vorgang ist, der kontextabhängig immer neue Wahrnehmungs-, Analyse-, Deutungs- und Handlungsmöglichkeiten eröffnet (Forneck 1997, S. 25). Verstehen ist damit aber auch stark von dem Erkenntnis-gewinnenden Subjekt abhängig, das wahrnimmt, deutet und handelt. Damit besteht ein Widerspruch zwischen dem Verstehensbegriff und

dem Kompetenzbegriff, denn durch Kompetenz wird auf die messbaren Lernergebnisse fokussiert, anstatt auf das im Subjekt stattfindende Lernen. Zusätzlich zielt Kompetenz auf die Alltagsrelevanz des Erlernten ab.

Rehm schlüsselt für die Didaktik der Naturwissenschaften die Dimensionen der Kompetenz „Verstehen von Phänomenen und Begriffen" auf (Rehm 2006). Er bezieht sich dabei auf Wagenschein (1991), der Verstehen als

> „echtes, ursprüngliches und vor allem selbst vollzogenes Verstehen" (Rehm 2006, S. 30)

beschreibt. Die Schüler verstehen Phänomene in ihrer Welt:

> „Sie stellen eine Beziehung zwischen sich, dem Phänomen in der Welt her und bauen hierdurch neue Strukturen auf" (Rehm 2006, S. 30).

Rehm beschreibt vier Kompetenzstufen des Verstehens:

1. Ein Phänomen als fragwürdig erkennen können;
2. Eine Beziehung zum Phänomen aufbauen können;
3. Sinn konstruieren können;
4. Der Schüler versteht das Phänomen (vgl. Rehm 2006, S. 34).

Auffallend ist die Ähnlichkeit der Kompetenzstufen mit dem Unterrichtsexperiment nach Meyer (2006). Somit können die Kompetenzstufen mit Schritten in einem Informatikexperimentiervorgang mit Informatiksystemen in Verbindung gesetzt werden (Abschnitt 5.5). Zur empirischen Überprüfung der Stufung nutzt Rehm den Ansatz der Phänomenographie (vgl. Fincher und Petre (2004) für den Bereich „Computer Science Education Research" (CSER)). Nach der Auswertung so genannter Phänomenprotokolle stellt er jedoch kritisch fest, dass Schüler der Sekundarstufe dann die Stufe drei und vier nicht mehr messbar erreichen, wenn sie größere fachliche Vorkenntnisse besitzen. Er vermutet, dass durch Vorkenntnisse allgemein gängigen Deutungen und Halbwahrheiten Vorschub geleistet wird (Rehm, S. 271). Fazit ist, dass eine Verknüpfung des Kompetenz- und Verstehensbegriffs möglich ist, die dreifach Bedeutung des Verstehens als Bildungsziel, Kompetenzfacette nach Weinert und Lernziel jedoch nicht aufgehoben wird.

Zur Operationalisierung von Lernzielen wird eine sehr viel feinere Bedeutung des Verstehens benötigt. Wittmann argumentiert in der Didaktik der Mathematik:

> „Was soll heißen ‚verstanden haben'? Soll es heißen ‚dem Beweis folgen können', ‚den Beweis unter Benützung eines Lehrbuches wiedergeben können', ‚den Zusammenhang mit anderen Sätzen am rechtwinkligen Dreieck aufzeigen können', ...? Ebensowenig ist klar, was ‚anwenden können' heißen soll" (Wittmann 1981, S. 120).

Zur Operationalisierung der Lernziele eignet sich Blooms überarbeitete Taxonomie nach Anderson und Krathwohl (2001). Sie unterscheidet sechs gestufte kognitive Prozesse (Abschnitt 2.1.5) und vier Wissensdimensionen: Faktenwissen (Factual Knowledge), Begriffliches Wissen (Conceptual Knowledge), Verfahrensorientiertes Wissen (Procedural Knowledge) und Metakognitives Wissen (Meta-cognitive Knowledge). Eine deutsche Übersetzung der in der überarbeiteten Taxonomie verwendeten Begriffe inklusive operationalisierter Verben liefern Schobel und Holdt (2004). Bezüglich Lernzielformulierung wird der Verstehensbegriff in der Literatur durchaus kritisch gesehen, da er implizit ist, und trotz Klassifikation durch vorgegebenen Verben und Nomen, die eine festgelegte Bedeutung im Lehr-Lernkontext besitzen, ist das Erstellen von Lernzielen schwierig:

> „There are two reasons for this difficulty. The first is that statements of objectives may contain more than verbs and nouns [...]. So, modifying phrases or clauses should be ignored in classifying the objectives; they may cause confusion when one is attempting to identify relevant parts for categorizing. The second reason for the difficulty in classifying objectives is that the verb may be ambiguous in terms of the intended cognitive process or the noun may be ambiguous in its intended knowledge" (Anderson und Krathwohl 2001, S. 33).

Deshalb ist anzumerken, dass die Zuordnung eines kognitiven Prozesses durchaus vom Vorwissen der Schüler abhängt und neben den kognitiven Prozessen implizit Stufungen in den Wissensdimensionen vorgenommen werden:

> When the task is an unfamiliar problem, however, students must determine what knowledge they will use. [...] then, to *understand conceptual knowledge* is a prerequisite to being able to *apply procedural knowledge*" (Anderson und Krathwohl 2001, S. 77; Hervorh. im Original).

Auch dadurch, dass Lernziele nach Blooms überarbeiteter Taxonomie formuliert werden, wird sichergestellt, dass das Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung nicht einseitig anhand von Fachthemen konzipiert wird, sondern das Handeln der Schüler in den Mittelpunkt rückt (Schobel und Holdt 2004, S. 49). Fazit ist, dass neben dem Widerspruch zwischen dem Verstehens- und dem Kompetenzbegriff die dreifache Verwendung des Verstehensbegriffs, als Bildungsziel, Kompetenzfacette und Lernziel, ausschlaggebend dafür ist „Informatiksystemverständnis" als Begriff nicht zu verwenden.

2.4 Zusammenfassung und Fazit für Kompetenzentwicklung mit Informatiksystemen

Für Kompetenzen mit Informatiksystemen sind im schulischen Kontext mehrere Aspekte zu berücksichtigen. Erstens ist zu bestimmen, welche Teilkompetenzen bzw. Kompetenzfacetten für eine angestrebte Kompetenz zu berücksichtigen sind.

Abbildung 2.3: Übersicht über Variationen des Kompetenzbegriffs zu Informatiksystemen

Zweitens ist für Kompetenzentwicklung die Abbildung von Entwicklungs- bzw. Niveaustufen anzustreben. Der Kompetenzbegriff impliziert, technische Grundlagen

handlungsorientiert anhand der Anwendung in Anforderungssituationen zu erlernen. Es besteht somit der Zielkonflikt, einerseits alltägliche Anforderungssituationen und andererseits technische Grundlagen von Informatiksystemen wissenschaftspropädeutisch im Informatikunterricht der Sekundarstufe II zu thematisieren. Abbildung 2.3 zeigt Variationen des Kompetenzbegriffs zu Informatiksystemen. Darin werden unterschiedliche Konzepte von Kompetenz einschließlich Schlüsselkompetenz auf Informatiksysteme bezogen. Insbesondere werden in der vorliegenden Arbeit Aspekte von Kompetenz und Grundbildung der OECD, der EU und der UNESCO aufgegriffen.

Vor der Analyse des Erfahrungswissens der Fachdidaktiken und in der Schulinformatik zur Identifikation normativer Vorgaben (Kapitel 4), werden im Folgenden die fachlichen Aspekte als Grundlage der Analyse erörtert (Kapitel 3).

3. Der Informatiksystembegriff

3.1 Überblick

Seit Anfang der neunziger Jahre wird die Vernetzung von Informatiksystemen für den Informatikunterricht betont und Wirkprinzipien von Informatiksystemen werden als eine Leitlinie des Unterrichts gefordert. In diesem Kapitel wird der fachwissenschaftliche Begriff des Informatiksystems beschrieben (Abschnitt 3.2). Dafür ist es notwendig, den aktuellen Informatiksystembegriff in den unterschiedlichen Bereichen der Kerninformatik zu untersuchen. Zusammen mit dem Kompetenzbegriff können Analysekriterien formuliert werden (Abschnitt 3.3).

Anhand dieser Kriterien werden der nationale und internationale Stand der Forschung zur Kompetenzentwicklung mit Informatiksystemen in der Schulinformatik analysiert (Kapitel 4). Ziel des Kapitels ist es damit, durch eine Konkretisierung des Informatiksystembegriffs und eine Strukturierung von Kompetenzen aus der Fachwissenschaft heraus die Analyse des fachdidaktischen Forschungsstandes vorzubereiten. Daraus werden erste Schlussfolgerungen und wissenschaftliche Fragestellungen zur Erstellung eines Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung abgeleitet. Abbildung 3.1 zeigt die Struktur der Kapitel 1 bis 3.

3.2 Der Systembegriff und fachdidaktische Schlussfolgerungen

3.2.1 Begriffsdefinition

Ziel dieses Abschnitts ist es, eine für die Informatik tragfähige Definition eines Informatiksystems aus der Kerninformatik herauszuarbeiten. Dafür wird kurz auf

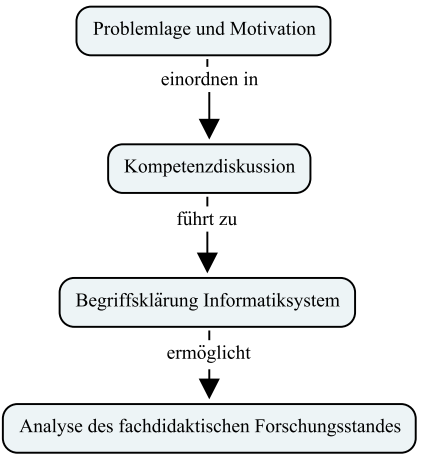


Abbildung 3.1: Struktur der Kapitel 1 bis 3

den Systembegriff der Wissenschaftstheorie Bezug genommen. Der Systembegriff leitet sich aus dem griechischen sýstēma, das Zusammengestellte, ab. In der Wissenschaftstheorie unterscheidet man zwischen gegenständlichen Systemen, welche in der Wirklichkeit vorkommende, zusammengestellte Gegenstände bezeichnen, und gedanklichen Systemen. Letztere sind Zusammenordnungen von Aussagen, Erkenntnis oder Wissen über die Wirklichkeit. Gegenständliche Systeme wiederum werden weiter unterteilt in natürliche und vom Menschen geschaffene, also zweckbehaftete Systeme (vgl. Seiffert 1989, S. 330). Gegenständliche und besonders technische Systeme werden oft als Vielfalt von Komponenten beschrieben, die untereinander in Wechselwirkung treten (Strukturdenken). Wissenschaftstheoretisch moderner ist die Betonung des Beziehungsgefüges unter Prozessen. Es besteht also ein Wandel vom Strukturdenken zum Prozessdenken (Seiffert 1989, S. 331).

Die heutige Systemtheorie hat sich vor allem aus der allgemeinen Systemtheorie und der Kybernetik entwickelt. Die allgemeine Systemtheorie (engl.: General System Theory) beschäftigte sich mit den Gemeinsamkeiten physikalischer, biologischer und gesellschaftlicher Systeme und wurde von Ludwig von Bertalanffy in den 1930er Jahren begründet und vor allem von Wirtschaftswissenschaftlern und Soziologen betrieben. Die Kybernetik entwickelte sich parallel dazu mit Bezug zu technischen, biologischen und gesellschaftlichen Systemen. Kennzeichnend für sie ist die Darstellung von Regelkreisen mit positiven und negativen Rückkopplungen. Sie wurde und wird vornehmlich zur Beschreibung technischer Systeme genutzt (vgl. Seiffert 1989, S. 332). Die heutige, so genannte „vervollständigte Systemtheo-

rie" (engl.: System Dynamics) greift die vorgestellten Strömungen auf und rückt dynamische, evolvierende Systeme in ihr Zentrum, während technische, d. h. strukturbewahrende Systeme weniger Aufmerksamkeit erfahren.

Wie ist nun der Systembegriff der Informatik? Broy und Steinbrüggen (2004) stellen die Informatik als Wissenschaft und Technik der Modellierung dar. Zentral sind für sie die informationsverarbeitende Maschine und informationsverarbeitende Vorgänge in verteilten reaktiven Systemen:

> „Information – oder genauer ihre Darstellung – wird in der Informatik durch digitale Maschinen verarbeitet, gespeichert, übertragen und dargestellt. [...] Maschinenbeschreibungen können selbst wiederum als Informationen aufgefasst werden. Daraus ergibt sich ein wesentliches Prinzip der Informatik: Neben die elementaren Informationen treten Informationen, die das Verhalten komplexer Maschinen darstellen. [...] Verteilte reaktive Systeme sind in unserem digitalisierten Alltag allgegenwärtig. Wir sprechen von *informatischen Systemen*" (Broy und Steinbrüggen 2004, S. vi; Hervorh. im Original).

Reaktive Systeme sind die im Prinzip endlos laufenden Systeme zur Prozessüberwachung sowie eingebettete und adaptive Systeme, die nicht nur ein abschließendes Ergebnis berechnen, sondern ständig auf Ereignisse und Daten von außen reagieren (Goos und Zimmermann 2006). Somit ist in der Informatik nicht nur der Informationsbegriff, sondern auch der Systembegriff zentral. Was aber bezeichnet nun ein informatisches System? Wedekind et al. (1998) nennen drei Bedingungen für informatische Systeme:

> „Als System im ‚weiteren Sinne' [...] gilt dabei
> (a) eine Menge von Elementen (Systembestandteilen), die
> (b) durch bestimmte Ordnungsbeziehungen miteinander verbunden und
> (c) durch klar definierte Grenzen von ihrer Umwelt geschieden sind" (Wedekind et al. 1998, S. 268).

Goos und Zimmermann (2006) stellen fest, dass dieser Systembegriff rekursiv (vgl. Brügge 2000, S. 9ff) und damit skalierbar (vgl. Brügge 2000, S. 11) ist, denn Komponenten sind wieder (Teil-) Systeme. Das Wissen von Systemeigenschaften im Kleinen kann auf größere übertragen werden. Wenngleich auch hier eine positive und negative Rückkopplung (vgl. Kybernetik) anzunehmen ist (Brauer 2001), wodurch die Skalierbarkeit gefährdet ist. Präzisiert wird der Systembegriff im Duden Informatik. In diese Definition ist auch der Zweck des Systems eingeschlossen:

> „**System**: In der Informatik versteht man hierunter die Zusammenfassung mehrerer Komponenten zu einer als Ganzes aufzufassenden Einheit. Die Komponenten können von gleicher Art (homogenes System, z. B. Programmsysteme) oder sehr unterschiedlich sein (heterogene Systeme, z. B. die Zusammenfassung von Hard- und Softwaresystemen zu einem Computersystem [Informatiksystem; Anm. d. V.], oder hybride Systeme, wenn gewissen Komponenten sehr unterschiedliche technologische Konzeptionen zugrunde liegen wie in der Zusammenfassung von analogen und digitalen Systemen). Ein System löst oder bearbeitet in der Regel ein wohldefiniertes Bündel von Aufgaben. Systeme können außerordentlich komplex oder vernetzt sein (wie z. B. ökologische Systeme, Telefonsysteme, Verkehrssysteme oder das Internet)" (Claus und Schwill 2006, S. 677; Hervorh. im Original).

Der Begriff bedarf jedoch der Konkretisierung:

> „Der Systembegriff ist sehr allgemein und daher wenig aussagekräftig. Als konkrete Ausgestaltung sind in diesem Duden u. a. Betriebssysteme, CAD-Systeme, Datenbanksysteme, Informationssysteme, Textverarbeitungssysteme oder allgemein Informatiksysteme beschrieben" (Claus und Schwill 2006, S. 678).

Was also zeichnet ein allgemeines Informatiksystem aus? Der Begriff ist nach Claus und Schwill (2006) folgendermaßen definiert:

> „Als Informatiksystem bezeichnet man die spezifische Zusammenstellung von Hardware, Software und Netzverbindungen zur Lösung eines Anwendungsproblems. Eingeschlossen sind alle durch die Einbettung des Systems in den Anwendungsbereich beabsichtigten oder verursachten nicht-technischen Fragestellungen und ihre Lösungen, also Fragen der Gestaltung des Systems, der Qualifizierung der Nutzer, der Sicherheit sowie der Auswirkungen und Folgen des Einsatzes.
> *Informatik* ist dann die Wissenschaft von Entwurf und Gestaltung von Informatiksystemen" (Claus und Schwill 2006, S. 314; Hervorh. im Original).

Konsistent dazu ist die Definition aus dem Positionspapier der Gesellschaft für Informatik e. V. (GI) „Was ist Informatik?", in der neben dem System aus Hard- und Software die Erfüllung von Informationsverarbeitungs- und -übertragungsaufgaben betont werden (vgl. Biundo et al. 2006, S. 10). Damit ist der Begriff Informatiksystem eine Konkretisierung sowohl eines Rechnersystems bzw. Computersystems (Claus und Schwill 2006, S. 314) als auch eines Rechnernetzes, die meist ohne Bezug zu einem Anwendungsproblem beschrieben werden und technische Fragestellungen überbetonen. Nach Appelrath und Ludewig (2000) besteht ein Rechnersystem aus Software und Hardware einschließlich peripherer Geräte zur Datenerfassung, -speicherung, -ausgabe und -übertragung. Dabei betonen sie, dass zwischen interner Datenübertragung im Rechnersystem und externer, das Rechnersystem in ein Rechnernetz verlassender Datenübertragung unterschieden wird (Appelrath und Ludewig 2000, S. 38). Die fehlende Angabe eines Zwecks ist vermutlich auf die implizit vorausgesetzte Universalität bzw. freie Programmierbarkeit der Von-Neumann-Rechner zurückzuführen.

Ein Informatiksystem kann auch als Teil eines sozio-technischen Systems aufgefasst werden, in dem ein soziales System und ein technisches System, z. B. ein Informatiksystem, ein Ganzes bilden (Ropohl 1991). Die Softwaretechnik betont besonders den sozio-technischen Kontext von Informatiksystemen. Balzert expliziert ausgehend vom Computersystem einen sehr differenzierten Systembegriff:

> „Anwendungssoftware, Systemsoftware und Hardware bilden zusammen ein **Computersystem** bzw. **DV-System**" (Balzert 2000, S. 24; Hervorh. im Original).

Diesen Terminus grenzt Balzert ab von technischen Systemen, die eine Zusammensetzung aus einem Computersystem und sonstigen technischen Komponenten sind.

Des Weiteren werden organisatorische Systeme (soziale Systeme), Informationssysteme (soziale Systeme und technische Systeme), Anwendungssysteme (Informationssysteme inklusive Computersysteme) unterschieden (vgl. Balzert 2000, S. 25). Der international beachtete Glossar des Institute of Electrical and Electronics Engineers (IEEE) definiert den Begriff „Computer System" wie folgt:

> „**computer system.** A system containing one or more computers and associated software." (IEEE Standards Department 1990, S. 15; Hervorh. im Original)

Als Zweck eines Systems wird die Erfüllung einer Menge an Funktionen genannt (IEEE Standards Department 1990). Hinzu kommt die Sichtweise auf die Komponenten eines Informatiksystems als selbständige und einigermaßen intelligenter Akteure, die miteinander und der Umgebung interagieren (vgl. Brauer und Brauer 1995). Dabei sind unter Akteuren bzw. Agenten Softwaremodule oder Maschinen wie z. B. Haushaltsgeräte oder roboterartige Geräte zu verstehen (Brauer und Münch 1996).

Um die teilweise widersprüchliche Nutzung des Systembegriffs aufzulösen, wurde in der vorliegenden Arbeit die Entscheidung getroffen, das Informatiksystem von dem kontinuierlich ablaufenden physikalischen, sozio-technischen oder chemischen System, in das es einwirkt, abzugrenzen. Dennoch fällt die Trennung dieser gekoppelten Systeme schwer, und die Theorie zu solchen hybriden Systemen ist noch in den Anfängen (Brauer 2001, S. 27ff). Es ist zu fragen, wie ein Informatiksystem nun betrachtet werden kann:

> „Wir unterscheiden zwei Fälle:
> 1. Für das Verständnis des Gesamtsystems sind nur die Zustände der Teilsysteme **als Ganzes** interessant; die Teilsysteme werden als Objekte aufgefasst;
> 2. Für das Verständnis des Gesamtsystems muß man die Beziehungen und möglichen zeitlichen Änderungen innerhalb der Teilsysteme **im einzelnen** studieren" (Goos und Zimmermann 2006, S. 19; Hervorh. im Original).

Ersteres entspricht einer Black-Box-Sicht, letzteres einer Glass-Box-Sicht. Claus und Schwill (2006) differenzieren weiter:

> „Systeme lassen sich meist durch mindestens einen der folgenden Punkte charakterisieren:
> - durch ein nach außen sichtbares Verhalten (a). Dieses kann man versuchen, durch Tests und Experimente zu ermitteln.
> - durch eine innere Struktur (b). Diese ist in der Regel nur den Entwicklern, nicht aber den Benutzern von Systemen bekannt. Sie lässt sich im Allgemeinen nicht durch Experimente, sondern nur durch eine genaue Analyse der Komponenten ermitteln.
> - durch Eigenschaften (c). Diese erfasst man mit Hilfe einer Spezifikation, die zur Entwicklung einer konkreten Realisierung dienen kann" (Claus und Schwill 2006, S. 677).

Dabei zeigt sich, dass zur Beschreibung der zuletzt genannten Eigenschaften, die in einer Spezifikation festgehalten werden, nicht auf Implementierungsdetails eingegangen wird:

> „**Spezifikation:** präzise Darstellung und Beschreibung der Eigenschaften, der Verhaltensweisen, des Zusammenwirkens und des Aufbaus von (bestehenden oder zu entwickelnden) Modellen, Programmen oder Systemen, wobei meist von Details abstrahiert wird und konkrete Implementierungsfragen keine Rolle spielen" (Claus und Schwill 2006, S. 643; Hervorh. im Original).

Dies ist ein Indiz dafür, dass Implementierungsdetails nicht relevant sind für Kompetenzentwicklung mit Informatiksystemen. Oft werden funktionale Spezifikationen genutzt, in denen Definitions- und Wertebereich sowie der funktionale Zusammenhang zwischen Ein- und Ausgabe beschrieben werden. Mit Blick auf die Zielebene der Förderung der Kompetenzentwicklung mit Informatiksystemen an allgemein bildenden Schulen wird davon ausgegangen, dass Details nicht relevant für den Lehr-Lernprozess sind. Deshalb wird im Folgenden der Informatiksystembegriff ausschließlich auf Fragestellungen zu Funktionen und Verhalten von Informatiksystemen sowie zu deren innerer Struktur betrachtet.

Durch die Begriffsklärung ist somit klar, dass Kompetenzentwicklung mit Informatiksystemen aus fachdidaktischer Perspektive nicht gleichzusetzen ist mit dem in der Fachwissenschaft Informatik verwendeten „Verstehen von (Software-) Systemen" (system comprehension) bzw. „Programmverstehen" (program comprehension). Diese Begriffe werden meist im Zusammenhang mit Reverse Engineering bzw. als notwendige Voraussetzung für die reibungslose Wartung von Softwareprodukten genutzt. Einen Überblick über Arbeiten in dem Bereich geben Hadar und Hazzan, die den Beitrag ausgewählter Diagrammarten der Unified Modeling Language (UML; (Rumbaugh et al. 2005)) auf das Softwaresystemverständnis erforschen.

3.2.2 Perspektiven auf Informatiksysteme

Funktionen von Informatiksystemen

In den „Great Principles of Computer Science" beschreibt Denning (2003) fünf Blickwinkel (windows of computing mechanics) auf die Informatik. Seine Arbeit beruht auf den Erkenntnissen einer Expertengruppe der ACM (Association for Computing Machinery) zur Erstellung der ACM-Curricula für die Hochschulinformatik. Diese erweitert er auf sieben Kategorien der Informatik: computation, communication, coordination, automation, recollection, evaluation, design (Denning 2007). Denning setzt die Kategorien mit den Hauptfunktionen von Informatiksystemen in Verbindung:

> „These categories cover the main functions of computing systems" (Denning 2007, S. 15).

Damit liefert Denning eine implizite Antwort auf die Frage, welche Sichten Lernende auf den Lerngegenstand „Informatiksysteme" einnehmen können und müssen,

Tabelle 3.1: Kategorien der Informatik nach (Denning 2003, S. 17) und (Denning 2007, S. 15)

Kategorie	**Exemplarische Inhalte**
Computation	Bedeutung von Berechnungen und Fragen der Berechenbarkeit: Automaten, formale Sprachen, Turingmaschinen, Universalität, komplexitätstheoretische Fragestellungen, Übersetzung, die technisch-physikalische Realisierung von Informatiksystemen
Communication	Zuverlässige Datenübertragung: Informatiksystem als Medium, speziell als Nachrichtensystem im Shannon'schen Sinne eines Kanalmodells mit Kodierung, Kanalkapazität, Rauschunterdrückung, Datenkompression, Kryptographie, rekonfigurierbarer Paketvermittlung und Ende-zu-Ende Fehlerbehandlung
Coordination	Kooperation zwischen vernetzten Entitäten: zwischen Menschen deren Arbeitsabläufe durch Informatiksysteme unterstützt werden (Workflows), Eingabe- und Ausgabeverhalten der Informatiksysteme sowie Antwortzeit. Synchronisation zwischen Informatiksystemen mit Race-conditions und Deadlocks, Serialisierbarkeit und atomaren Aktionen
Automation	Fragen zu Grenzen der Simulation kognitiver Prozesse: philosophische Betrachtungen zur Automatisierung, zu Expertise und Expertensystemen, Verbesserung von (künstlicher) Intelligenz, Turingtest, Bedeutung des Maschinenlernens und -erkennens mittels evolutionärer Algorithmen, Bionik
Recollection	Speicherhierarchien, Lokalität von Referenzen, Caching, Adressbereich und Abbildung, Namenskonventionen, Suche sowie Retrievaltechniken durch Name oder Inhalt
Evaluation	Leistungsvoraussagen und Kapazitätsplanung: Sättigung und Flaschenhälse in Netzen
Design	Entwicklung qualitativ hochwertiger Informatiksysteme: Ebenenmodell des Rechners, Schichtenarchitektur des Internets, Modularisierung, Geheimnisprinzip und Abstraktion

um ein konsistentes Gesamtbild von Informatiksystemen zu bekommen. Im Folgenden werden die sieben Kategorien kurz genannt und im Sinne unterschiedlicher Perspektiven auf Informatiksysteme erläutert (Tabelle 3.1).

Aspekte von „computation", „communication", „coordination", „automation" und „recollection" spielen in allen Informatiksystemen eine Rolle. Die Blickwinkel auf die Informatik nach Denning können mit dem fachdidaktischen Sichtenkonzept kombiniert werden. Brinda fordert für Lernsoftware, den zu explorierenden Lerngegenstand in unterschiedlichen, interaktiv erfahrbaren und synchronisierten Sichten darzustellen (Brinda 2004a, S. 52). So ist eine zuverlässige Datenübertragung, wie sie unter dem Blickwinkel der Kommunikation (communication) analysiert wird, beispielsweise bezüglich ihres Entwurfs (design) und hinsichtlich der Nebenläufigkeit von Prozessen im Rahmen der Koordinierung (coordination) zu betrachten. Die Sichten überschneiden sich jedoch stark. Dies wird dadurch verstärkt, dass in jeder Sicht mehrere Modelle und Darstellungsformen zur Hervorhebung eines Gesichtspunktes zum Einsatz kommen, die wiederum auch in anderen Sichten genutzt

werden. Deshalb wird eine Vielzahl an Facetten auf Informatiksysteme ermöglicht, die schwierig zu synchronisieren sind und einer weiteren Strukturierung bedürfen. Ausgehend von diesen Perspektiven auf Informatiksysteme, die im Einzelfall nicht trennscharf sind, lautet die Frage insbesondere, welche und ggf. wie viele Sichten auf Informatiksysteme im Informatikunterricht angeboten werden sollten, und wie diese lernförderlich zu verknüpfen sind. Zur Strukturierung bietet es sich an, dass Informatiksysteme anhand der drei Charakteristika „nach außen sichtbares Verhalten", „innere Struktur" und „Entwicklung einer konkreten Realisierung" untersucht werden können (vgl. Claus und Schwill 2006, S. 677). Denn die Fragestellungen bezüglich der einzelnen Kategorien können nach diesem Schema geordnet werden. Das Prinzip des Sichtenwechsels (vgl. Brinda 2004a, S. 125), in dem Schüler die Konsequenzen ihres Handelns anhand des Wechselns zwischen den Perspektiven und der Analyse der jeweiligen Änderungen verstehen, wird durch die Perspektiven „Blick auf das System" und „Blick in das System" unterstützt.

Brauer und Brauer (1992) nennen vier Arten von Unterstützungssystemen, die als neue Anforderungen von der Informatik erwartet werden. Sie haben nicht an Aktualität verloren. Als erstes sind es Systeme für die „fünf C", die den Kategorien von Denning sehr ähnlich sind: communication, cooperation, collaboration, coordination, concurrency. Zweitens werden Unterstützungssysteme des individuellen und kollektiven Gedächtnisses erwartet wie Nicht-Standard-Datenbanken und intelligente, kooperative Informationssysteme zur Bereitstellung gemeinschaftlich verwendeter Arbeitsumgebungen. Drittens sind es Informatiksysteme zur Unterstützung des Verstehens und Steuerns komplexer Systeme beispielsweise durch Simulation, Visualisierung und Prozessregulierung. Zuletzt nennen Brauer und Brauer adaptive und lernfähige Schnittstellensysteme für die Interaktion zwischen Mensch und Maschine sowie zwischen Umwelt und Maschine, die Änderungen ihrer Arbeitsbedingungen bewerten können (vgl. Brauer und Brauer 1992, S. 12). Diese vier unterschiedlichen Systemklassen strukturieren die konkreten Informatiksysteme, die im Informatikunterricht für Kompetenzentwicklung mit Informatiksystemen behandelt werden können.

Systemverhalten als Folge von Zuständen

Wie wird nun das Verhalten von Informatiksystemen erklärbar? Broy und Rumpe (2007) nutzen Zustandsmaschinen und Schnittstellenbeschreibungen zur Veranschaulichung des Verhaltens von Informatiksystemen und grenzen sie von der Architektur ab:

> „Im Gegensatz zu Zustandsmaschinen und unserem Schnittstellenmodell, das abstraktes Verhalten im Sinne einer ‚Black Box'-Sicht beschreibt, zielen Architekturen darauf ab, die Struktur also den inneren Aufbau eines Systems darzustellen" (Broy und Rumpe 2007, S. 11).

Die Schnittstelle dient zur Beschreibung der Systemgrenze (vgl. (Goos und Zimmermann 2006), (Brügge 2000)). Darüber hinaus kann ein System statisch oder

dynamisch sein. Änderungen beziehen sich auf seinen Zustand und resultieren in verändertem Systemverhalten:

> „[...] das System als solches bleibt gleich, es behält seine Identität und seinen Namen. Der Zustand besteht aus den jeweils vorhandenen Komponenten, ihren Eigenschaften und ihren Beziehungen. Die Folge der Zustände bezeichnet man als das **Verhalten des Systems**" (Goos und Zimmermann 2006, S. 18; Hervorh. im Original)

Systeme mit abzählbar vielen Zuständen werden auch als diskrete Systeme bezeichnet (vgl. Claus und Schwill 2006, S. 678). Dabei ist zu bemerken, dass sich die Informatik nicht nur mit diskreten Systemen beschäftigt. Selbst ohne Berücksichtigung des sozio-technischen Kontextes eines Informatiksystems stellt die (oft bewusste) Einschränkung auf diskrete Systeme eine Idealisierung der technischen Realisierung dar: Informatiksysteme basieren durch fortschreitende Miniaturisierung und Erhöhung der Arbeitsgeschwindigkeit auf einer physikalischen Emulation eines Bit-Objekts, die sich durch kontinuierliche, zeitbehaftete Zustandsübergänge sowie spontane Störvorgänge (z. B. durch Soft Errors, bei denen Strahlung die Speicherzellenbelegung verändert, oder Störeinflüsse bei der drahtlosen Kommunikation) auszeichnet (vgl. Hertwig und Brück 2000). Goos und Zimmermann (2006) unterscheiden darüber hinaus deterministische und indeterministische Systeme. Bei ersteren ist das zukünftige Verhalten aus dem Systemverhalten in der Vergangenheit ableitbar. Für die Förderung der Kompetenzentwicklung mit Informatiksystemen ist die Betrachtung der Systemzustände eine Möglichkeit, das Systemverhalten zu erklären. Zusätzlich kann der Einfluss von Störungen an der Hardware auf das Systemverhalten thematisiert werden.

3.2.3 Innere Struktur von Informatiksystemen

Auswahl informatischer Grundlagen

Welche informatischen Grundlagen zur Erklärung der inneren Struktur von Informatiksystemen sind notwendig für den Einsatz von Informatiksystemen zur Lösung eines Anwendungsproblems? Nievergelt (1995) beschreibt dazu ein Schichtenmodell der Informatik, den Informatikturm (Abbildung 3.2). Als oberste und größte Ebene nennt er die Anwendungsmethodik, die beschreibt, wie Informatiksysteme zur Lösung eines Anwendungsproblems eingesetzt werden. Darunter ist die Ebene der Systemrealisierung mit Entwurf und Implementierung in Hard- und Software. Eine weitere Ebene tiefer liegt die Algorithmik. Auf der untersten Ebene liegen fundamentale theoretische Erkenntnisse der Informatik. Nievergelt betont, dass er eine ausgewogene Mischung von Theorie und Anwendung befürwortet und hält das Fundament aus den drei Schichten für notwendig, damit die Schicht der Anwendung von Informatiksystemen verstanden wird. Gleichzeitig beantwortet er die Frage nach einer verpflichtenden Menge Theorie, um Informatiksysteme anwenden zu

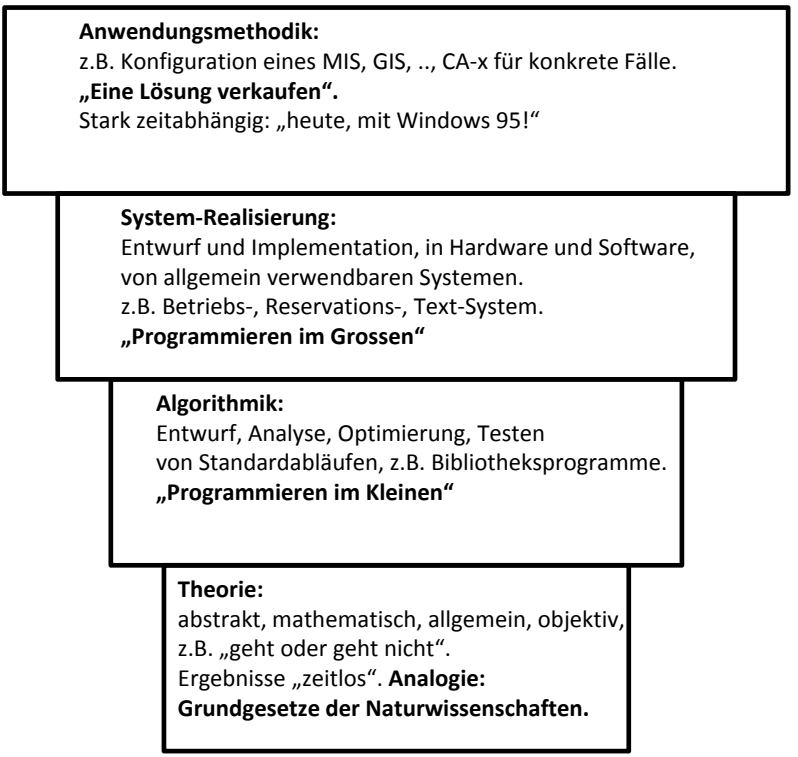


Abbildung 3.2: Informatikturm nach (Nievergelt 1995, S. 342; Hervorh. im Original)

können, nicht pauschal. Er fordert, Einzelurteile zu unterschiedlichen theoretischen Grundlagen zu fällen (Nievergelt 1995, S. 344).

Thomas (2003) identifiziert im Sprachgebrauch der Kerninformatik fünf Hauptmodelltypen: Architekturmodelle, Vorgehensmodelle, Entwurfsmodelle, Untersuchungsmodelle und mentale Modelle (siehe Tabelle 4.2 in Abschnitt 4.2.4). Diese Systematik führt bezüglich der inneren Struktur von Informatiksystemen zur Betrachtung von Architekturmodellen. Unter Architekturmodellen fasst Thomas Rechnerarchitekturen (z. B. Von-Neumann-Rechner), theoretische Maschinenmodelle (z. B. Turingmaschine, Automaten), Rechenmodelle (z. B. imperativ, logisch-deklarativ, funktional) und Referenzmodelle (z. B. Client-Server, OSI-Schichtenmodell (Open Systems Interconnection Basic Reference Model)) zusammen. Den Informatiksystemen zugrunde liegende Architekturmodelle wie Von-Neumann-Architektur und das Maschinenmodell der Turingmaschine verdeutlichen die grund-

sätzliche Arbeitsweise von Rechnern. Gleiches gilt beispielsweise für das imperative Rechenmodell, dem das mathematische Modell der Registermaschine (Random Access Machine (RAM)) zugrunde liegt (Schubert und Schwill 2004, S. 168). Wesentlich sind das Variablenkonzept und Befehlsfolgen, die Prozeduren bilden und den Zustand des Rechners verändern (Claus und Schwill 2006, S. 525). Synonym zur Architektur wird oft der Begriff „Strukturmodell" verwendet. Unter Strukturmodellen verstehen Broy und Rumpe (2007) Architekturen, die Perspektiven auf die innere Strukturierung eines Systems beschreiben (vgl. Wedekind et al. 1998). Da in den vorherigen Abschnitten vornehmlich die Perspektive der Praktischen Informatik beschrieben wurde, werden im Folgenden Sichtweisen der Technischen und Theoretischen Informatik ergänzt.

Architekturen der Technischen Informatik

In der Technischen Informatik nennen Tanenbaum und Goodman (2001) die Lücke zwischen maschinenverständlichen Befehlen und den für Informatiksysteme typischen Anwendungsproblemen als Grund für die Bildung von Schichtenarchitekturen (Tanenbaum und Goodman 2001, S. 19). Die einfachen bzw. primitiven Instruktionen von Maschinensprachen sind für Menschen schwierig zu benutzen, weshalb von den Instruktionen auf unterschiedlichen Ebenen abstrahiert wird:

> „Dies hat im Laufe der Zeit zur Strukturierung von Computern als Reihe von Abstraktionen geführt. Dabei baute jede Abstraktion auf der jeweils darunterliegenden auf. Auf diese Weise kann Komplexität gemeistert werden [...]" (Tanenbaum und Goodman 2001, S. 19).

Solche Ebenen sind Teil der Architektur:

> „Die auf jeder Ebene verfügbaren Datentypen, Operationen und Merkmale nennt man **Architektur** (Architecture). Die Architektur betrifft Aspekte, die für den Benutzer der jeweiligen Ebene sichtbar sind. Merkmale, die der Programmierer sieht, z. B. die verfügbare Speichermenge, sind Teil von ihr. Implementierungsaspekte, etwa welche Chiptechnik zur Speicherimplementierung verwendet wird, sind nicht Teil der Architektur" (Tanenbaum und Goodman 2001, S. 26; Hervorh. im Original).

Diese moderne Definition der Architektur vereint zwei Sichtweisen: erstens die Architektur als konzeptuelle Beschreibung einschließlich des Systemverhaltens wie sie bereits 1964 definiert wurde (Amdahl et al. 1964). Zweitens werden die weiteren Vorgänge in der Rechnerorganisation einbezogen (Stone, Harold S. 1975).

Eine Komponentenarchitektur kann aus mehreren Subsystemen bestehen (vgl. Broy und Rumpe 2007, S. 10). Damit gehören auch Fragestellungen zu Daten, Kontrollfluss, Speicher und Adressierung sowie Beziehung zwischen Befehlssatz und Speicherorganisation zum Gebiet der Rechnerarchitektur (vgl. IEEE Standards Department 1990, S. 4). Jede Abstraktionsebene wiederum wird als abstrakte Maschine mit entsprechender Maschinensprache betrachtet:

> „[...] bei sequentiellen Rechnern und Programmen werden häufig verschiedene Schichten abstrakter Maschinen aufeinandergebaut, wobei die Maschinen einer Schicht die der jeweils darunterliegenden Schicht verbergen" (Mühlhäuser 2002, S. 675).

Somit kann man einen Rechner in einem Schichtenmodell beschreiben.

Maschinenmodelle der Theoretischen Informatik

Von-Neumann-Rechner zeichnen sich dadurch aus, dass sie universell einsetzbar sind (Claus und Schwill 2006, S. 145). Um nun grundlegende Aussagen über solche Informatiksysteme zu machen, werden in der Theoretischen Informatik Maschinenmodelle genutzt, die die technischen Strukturelemente von Informatiksystemen abstrahieren:

> „Ein Modell [Maschinenmodell; Anm. d. V.] sollte einerseits so abstrakt sein, daß man auf relativ einfache Weise möglichst weitreichende Aussagen machen kann. Andererseits darf es sich nicht zu sehr von den in der Wirklichkeit vorhandenen Rechnerarchitekturen entfernen" (Reischuk 1999, S. XV).

Maschinenmodelle stellen also eine Abstraktion der Komponenten und Prozesse in Informatiksystemen dar, um beispielsweise Fragen der Berechenbarkeit und Komplexität anhand von Systemeigenschaften zu erklären. Damit sind Maschinenmodelle je nach Abstraktionsgrad wie Architektur- und Entwurfsmuster Abstraktionen informatischer Problemlösungen. Turingmaschine und Registermaschine sind Abstraktionen der in sequentiellen Rechnern vorherrschenden Architekturkomponenten. Wichtigste Komponenten sind – wie auch beim Von-Neumann-Rechner – Prozessor, Speicher, Programm sowie Eingabe und Ausgabe (vgl. Reischuk 1999, S. 6).

Registermaschine und Turingmaschine sind sehr flexibel, im Gegensatz dazu können Schaltkreise nur Bitstrings einer fest vorgegebenen Länge verarbeiten und dienen zur Lösung eines konkreten Problems. Um Schaltkreise als universelles Maschinenmodell nutzen zu können und beispielsweise mit Turingmaschinen zu vergleichen, betrachtet man daher Schaltkreisfamilien $G = G_1, G_2, \ldots$ von $n-$Input Schaltkreisen G_n (vgl. Reischuk 1999, S. 87). Schaltkreise arbeiten auf Bits und damit auf einem sehr elementaren Niveau. Somit sind Schaltkreise geeignet, die Arbeitsweise eines Rechners zu beschreiben und besonders das Konzept der Digitalisierung zu betonen wie auch folgende Arbeiten unter Mitwirkung des Autors zeigen (Hinkelmann et al. 2007), (Hinkelmann et al. 2008).

Des Weiteren wird zwischen deterministischem und nichtdeterministischem Maschinenmodell unterschieden. Auch wenn heutige Rechner deterministisch arbeiten (Claus und Schwill 2006, S. 456), lässt sich für manche Zuordnungs- und Optimierungsprobleme eine effiziente Lösung nur nichtdeterministisch beschreiben, so dass man die Arbeitsweisen deterministischer und entsprechender nichtdeterministischer Maschinenmodelle vergleicht. Zur Erkennung regulärer Sprachen werden endliche Automaten verwendet, die das Konzept der Zustände mit Zustandsübergängen hervorheben, so dass das Verhalten bzw. die Ausgabe des Automaten direkt auf dessen Zustandswechsel zu beziehen ist. Insbesondere für das Konzept der Rekursion gibt es Maschinenmodelle zur Veranschaulichung der Arbeitsweise. Zu nennen ist die Formularmaschine, die auch eine reale Entsprechung (vgl. Bauer und Goos 2004) hat:

> „Diese [Formularmaschine; Anm. d. V.] ist eine sehr ‚menschliche' Maschine; man kann sich gut vorstellen, wie der Gang der Berechnung für einen Einzelrechner (etwa nach dem Kellerprinzip) oder für eine Rechnergruppe organisiert wird [...]" (Bauer und Goos 2004, S. 165).

Es ist damit festzuhalten, dass Maschinenmodelle insbesondere die Arbeitsweise von Informatiksystemen erklären und veranschaulichen. Außerdem wird durch diese Betrachtungsweise ersichtlich, dass zur inneren Struktur von Informatiksystemen nicht nur ihre Komponenten, sondern insbesondere auch Abläufe bzw. Prozesse zwischen den Komponenten gehören (vgl. Abschnitt 3.2.1).

Qualitätsanforderungen an Informatiksysteme

Neben den bisher angefügten expliziten Beschreibungen von Informatiksystemen kann auch implizit nach den Anforderungen und Qualitätsmerkmalen von Informatiksystemen gefragt werden. Qualität von Informatiksystemen und das Bemerken mangelhafter Qualität sind notwendig, wenn es gilt, Vor- und Nachteile des Einsatzes von Informatiksysteme abzuschätzen (Claus und Schwill 2006, S. 97f). Für Kompetenzentwicklung mit Informatiksystemen kann eine Thematisierung solcher Qualitätsmerkmale hilfreich sein. Sehr technisch orientierte Normen sind jedoch auf ihren Bezug zur Lebenswelt der Schüler zu prüfen und müssen für Schüler nachvollziehbar gemacht werden. Normen und Spezifikationen existieren in vielen Bereichen der Informationstechnik (Wende 2002). So gibt es Normen zu Programmiersprachen und Softwareentwicklung, z. B. legt die DIN ISO/IEC 12119 Qualitätsanforderungen an Softwareprodukte fest und stellt Prüfregeln auf. DIN EN ISO 9000-3 gibt einen Leitfaden zur Entwicklung, Lieferung und Wartung von Software. Dazu gehören Korrektheit, Effizienz, Robustheit, Wiederverwendbarkeit, Kompatibilität, Modularisierung, Benutzungsfreundlichkeit und Wartbarkeit (vgl. (Appelrath et al. 2002, S. 106), (Hesse et al. 1994, S. 97), (Böszörmenyi 2001, S. 15)). Zur Kommunikation und verteilten Verarbeitung gibt es als Grundlage von Internet-Spezifikationen das OSI-Referenzmodell, das in EN ISO/IEC 7498-1 beschrieben und durch EN ISO/IEC 7498 Teil 2 bis 4 erweitert wird. Das ISO/OSI 7-Schichtenmodell wurde bereits 1979 standardisiert. Notwendige Sicherheitsdienste von Informatiksystemen zur Gewährleistung des Datenschutzes sind beispielsweise gemäß EN ISO 7498-2 Authentifikation und Identifikation, Zugriffskontrolle, Vertraulichkeit, Integrität und Verbindlichkeit (ISO 1989). Hinzugefügt wird oft die Anforderung nach Verfügbarkeit. Normen zu Bürogeräten und Ergonomie stellen DIN EN ISO 9241 dar, die ergonomische Anforderungen an Bildschirmarbeitsplätze formulieren. Teil 10 nennt beispielsweise Aufgabenangemessenheit und Individualisierbarkeit im Zuge der Dialoggestaltung als ergonomische Anforderungen (ISO 1996).

3.3 Fazit und Kriterien für die Analyse des Forschungsstandes

Ein Informatiksystem besteht aus Hardware, Software und Vernetzung zur Lösung eines Anwendungsproblems im sozio-technischen Kontext. Komponenten sind wiederum selbständige Akteure, die miteinander und mit ihrer Umgebung interagieren. Hauptfunktionen eines Informatiksystems beziehen sich auf „computation", „communication", „coordination", „automation", „recollection", „evaluation" und „design". Durch die Feststellung, dass diese Funktionen sich überschneidende Sichten auf den Lerngegenstand anbieten, bedarf es einer weiter gehenden Strukturierung. Dafür bietet es sich an, dass Informatiksysteme anhand ihres nach außen sichtbaren Verhaltens, ihrer inneren Struktur und durch Entwicklung einer konkreten Realisierung untersucht werden (vgl. Claus und Schwill 2006, S. 677). Ihr nach außen sichtbares Verhalten steht dabei in Abhängigkeit zu Systemzuständen. Ihre innere Struktur wird durch die Systemkomponenten inklusive interner Prozesse und Organisationsbeziehungen beschrieben. Diesbezüglich bilden Strukturmodelle eine besondere Grundlage, um Informatiksysteme zu beschreiben, z. B. Softwarearchitekturen, Rechnerarchitekturen und abstrakte Maschinen.

Damit ergibt sich folgende Grobstrukturierung der Kompetenzen in drei Bereiche $S_i, i \in \{A, B, C\}$ mit $A :=$ „nach außen sichtbares Verhalten", $B :=$ „innere Struktur", $C :=$ „Implementierungsaspekte", in die Lernziele eingeordnet werden können. Die Schüler sind in der Lage, eine Anforderungssituation zu bewältigen hinsichtlich ...

- S_A: des nach außen sichtbaren Verhaltens von Informatiksystemen,
- S_B: der inneren Struktur und interner Prozesse von Informatiksystemen,
- S_C: ausgewählter Aspekte der Implementierung einer konkreten Realisierung (Stechert 2006c).

In den Bereichen S_i können Lernziele durchnummeriert werden: $S_{i,j}, i \in \{A, B, C\}$, $j \in \mathbb{N}$.

Bei der Grobstrukturierung ist anzumerken, dass eine Formulierung, die sich nur auf das Bewältigen einer komplexen Anforderungssituation mittels eines Informatiksystems bezieht, eine Einschränkung darstellen würde. Basiskompetenzen zu von Informatiksystemen sind nun von generellen Informatikkompetenzen abzugrenzen. Deshalb wird für Basiskompetenzen vorausgesetzt, dass Schüler im Alltag auf Anforderungssituationen treffen, die durch das Verhalten von Informatiksystemen ausgelöst sind. Die Erstellung einer konkreten Realisierung eines Informatiksystems kann davon klar unterschieden werden. In der vorliegenden Arbeit wurde aufgrund des Kompetenzbegriffs die Entscheidung getroffen, die Systementwicklung (S_C)

nicht als Schwerpunkt zu setzen. Dementsprechend steht die Algorithmik nicht im Fokus der nachfolgenden Analyse des fachdidaktischen Forschungsstandes. Winograd und Flores (1988) betonen, dass Programmiersprachen zu stark an der Hardwarestruktur der Rechner orientiert sind und nicht an deren Verhalten:

> „Programming languages are one approach to formalizing this domain, but in general they are not well suited to the communication of intent and conceptual structure. They are too oriented to the structure of the machine, rather than to the structure of its behaviour" (Winograd und Flores 1988, S. 176).

Damit sind nur wenige ausgewählte Implementierungsaspekte anhand von Programmiersprachen für Kompetenzentwicklung mit Informatiksystemen zu betrachten. Vielmehr sind die konzeptuelle und logische Struktur eines Informatiksystems geeignet, um das nach außen sichtbare Verhalten von Systemen zu erklären. Abbildung 3.3 fasst die Grobstukturierung zusammen.

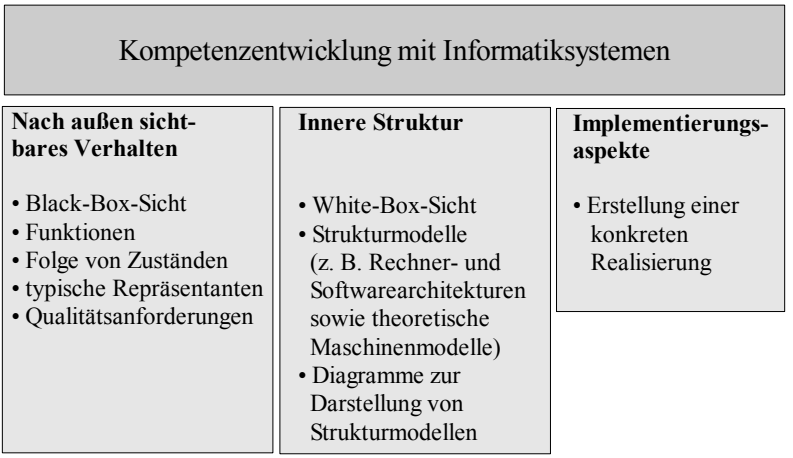


Abbildung 3.3: Schematische Darstellung der Strukturierung der Basiskompetenzen zu Informatiksystemen

Zur Herleitung von komplexen Anforderungssituationen für Schüler bietet die fachliche Strukturierung der Hauptfunktionen von Informatiksystemen nach Denning (2007) Anhaltspunkte. Diese können mit den Perspektiven auf das nach außen sichtbare Verhalten und auf die innere Struktur von Informatiksystemen in Verbindung gesetzt werden (Stechert 2008a). Tabelle 3.2 verbindet die Sichten miteinander.

Ziel ist speziell, dass zentrale Zusammenhänge zwischen der inneren Struktur von Informatiksystemen und ihrem nach außen sichtbaren Verhalten von Schülern erkannt und daraus angemessene Handlungsentscheidungen abgeleitet werden. Die

Tabelle 3.2: Strukturierung nach Hauptfunktionen und Charakteristika von Informatiksystemen

Hauptfunktionen	sichtbares Verhalten	innere Struktur	Implementierungsaspekte
Computation Communication Coordination Automation recollection Evaluation Design	Wie passt das Verhalten zu Zweck und Funktion?	Welche technischen Strukturen und Prozesse (z. B. Hard- und Softwarearchitektur) unterstützen die Hauptfunktion?	Welche Spezifikations- und Implementierungsaspekte beeinflussen die Hauptfunktion?

unterschiedlichen Perspektiven auf Informatiksysteme unterstützen die kognitive Flexibilität der Schüler (Cognitive Flexibility; (vgl. Hubwieser 2007a, S. 11)): Informatiksysteme werden unter unterschiedlichen Blickwinkeln in verschiedenen Kontexten betrachtet.

Folgende Kriterien und Fragen zur Analyse des fachdidaktischen Forschungsstands zur Kompetenzentwicklung mit Informatiksystemen lassen sich aus diesen Überlegungen ableiten:

Bildungsziele Welche Bildungsziele und Kompetenzaspekte mit Informatiksystemen werden für Schüler an allgemein bildenden Schulen skizziert? Welche Bildungsanforderungen zu Informatiksystemen in Form von Lernzielen werden beschrieben?

Typische Repräsentanten von Informatiksystemen Für welche konkreten Informatiksysteme wird der Einsatz im Informatikunterricht begründet? Wie ist die Rolle dieser Informatiksysteme im Lehr-Lernprozess?

Verhalten von Informatiksystemen Welche Hauptfunktionen von Informatiksystemen werden in der fachdidaktischen Diskussion thematisiert? Werden bestimmte Hauptfunktionen stärker betont als andere? Welches Verhalten von Informatiksystemen müssen Schüler verstehen können? Welche Vorgehensweisen zur Analyse und Erkundung von Informatiksystemen werden beschrieben?

Innere Struktur von Informatiksystemen Welche Strukturierungsmodelle und Architekturen werden für den Informatikunterricht empfohlen? Wie gelingt die Verknüpfung von Hardware, Software und Vernetzung im Lehr-Lernprozess?

Da gerade für Schüler der Sinn eines Artefakts erkennbar sein sollte, wird im Folgenden immer vorausgesetzt, dass ein Informatiksystem zweckbehaftet ist, nämlich zur Lösung eines Anwendungsproblems dienen soll. Mit Blick auf Rechner- und Softwarearchitekturen ist eine Annahme für Unterricht zu Informatiksystemen und Kompetenzentwicklung, dass durch die grobe Art der Zerlegung von Systemen in wenige Architekturelemente bzw. aufeinander aufbauende Schichten das Verstehen von Systemcharakteristika gefördert wird. Die innere Struktur umfasst Prozesse und Organisationsbeziehungen.

Wichtig ist auch eine Abgrenzung: In der vorliegenden Arbeit zu Informatiksystemen und Kompetenzentwicklung bilden Verstehen und Anwenden von Informatiksystemmodellen keinen Schwerpunkt. Informatisches Modellieren wird oft als aktives Erstellen einer konkreten Realisierung eines Informatiksystem gesehen (vgl. Brauer und Brauer 1992), z. B. das Durchlaufen des Softwareentwicklungsprozesses. Für Basiskompetenzen zu Informatiksystemen werden selbige im Alltag betrachtet. Modelle werden genutzt, um das nach außen sichtbare Verhalten von Informatiksystemen und deren innere Struktur zu beschreiben. Sie sind damit u. a. ein Mittel zur Kompetenzentwicklung. In Abgrenzung zum Systembegriff sei ein Modell also eine abstrahierte Beschreibung eines realen oder geplanten Systems (vgl. Hubwieser 2007a, S. 86).

4. Stand der Forschung zu Informatiksystemen und Kompetenzentwicklung

4.1 Überblick

In diesem Kapitel werden anhand der in Kapitel 3 ausgearbeiteten Kriterien der nationale und internationale Stand der Forschung zu Informatiksystemen und Kompetenzentwicklung in der Schulinformatik analysiert (Abschnitte 4.2 und 4.3). Ziel des Kapitels ist es, die Ergebnisse nach Bildungszielen, Unterrichtsinhalten und -gegenständen, Lehr-Lernmethoden sowie Lernmedien für Informatiksysteme und Kompetenzentwicklung zu gruppieren (Abschnitt 4.4). In der Diskussion der analysierten Publikationen wird jeweils eine Referenz zu dem Ergebnisbereich gesetzt, z. B.:

→ Ziel 1: Aufbau, Vernetzung und Funktionsweise,

→ Inhalt 1: Typische Repräsentanten,

→ Methodik 1: Universalität,

→ Medien 1: Visualisierung verborgener Prozesse.

Die Ergebnisse werden in Abschnitt 4.4 ab Seite 135 vorgestellt. Dort befindet sich auch eine Übersichtsgrafik (Abbildung 4.1). Es werden einerseits Forschungslücken offen gelegt und andererseits weitere Schlussfolgerungen und wissenschaftliche Fragestellungen abgeleitet, die in einem Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung fokussiert werden.

In Deutschland existiert das Fachgebiet Didaktik der Informatik erst seit 1996 in Form von Lehrstühlen an Hochschulen (vgl. Humbert 2003, S. 55). Davor verfassten meist Fachwissenschaftler die Studien und Empfehlungen zur Schulinformatik. Eine strikte Trennung in Fachdidaktik und Fachwissenschaft wird bei der folgenden Analyse des Forschungsstandes vermieden, allerdings mit Schwerpunktverlagerung auf die Fachdidaktikforschung ab den 1990er Jahren. Die Forschung zur Schulinformatik in der Bundesrepublik Deutschland wurde bereits von verschiedenen Autoren in Phasen und Ansätze strukturiert und kommentiert (z. B. (Forneck 1990), (Baumann 1993), (Eberle 1996), (Humbert 2003)). Zur Förderung und empirischen Analyse der Kompetenzentwicklung mit Informatiksystemen liegt jedoch bisher kein auf die Schulinformatik bezogenes, hinreichend theoretisch fundiertes, differenziertes und mit dem Erfahrungswissen der Fachdidaktik unterlegtes Unterrichtsmodell vor.

Die erste Bildungsempfehlung der GI von 1976 fordert die verpflichtende informatische Bildung, und ihr Einfluss prägte die weitere Entwicklung der Schulinformatik in der Bundesrepublik Deutschland stark. Daher werden im Folgenden nationale Bildungsempfehlungen der GI und wissenschaftliche Publikationen ab 1976 zur Schulinformatik unter normativer Perspektive auf Kompetenzentwicklung mit Informatiksystemen untersucht. Vor 1970 war ein logisch-kybernetischer und fächerübergreifender Ansatz in der Schulinformatik vorherrschend. Der darauf folgende hardwareorientierte Ansatz bis etwa 1976 befasste sich mit logischen Grundlagen und technischer Realisierung unter Zuhilfenahme von Modellrechnern und Digitalbausteinen. Hinweise auf damalige Unterrichtsinhalte finden sich bei Meißner (1971).

Die seit 1984 zweijährlich stattfindende Konferenz Informatik und Schule (INFOS) der GI hat von Beginn an großen Einfluss auf die nationale Schulinformatik. Deshalb werden neben den Empfehlungen der GI zur Schulinformatik vornehmlich Beiträge dieser Konferenz und der seit 1981 herausgegebenen Zeitschrift LOG IN analysiert.

Auf internationaler Ebene fordert Ershov durch seine Stellungnahme „Programming, the second literacy" auf der 3. „World Conference on Computers in Education (WCCE)" 1981 eine verpflichtende Schulinformatik und die entsprechende Fachdidaktikforschung. Zur Analyse des internationalen Forschungsstandes werden insbesondere die WCCE der IFIP und Bildungsempfehlungen der ACM und der UNESCO für die Schulinformatik betrachtet. Letztere wurden in Kooperation mit der IFIP entwickelt. Die Bildungsempfehlungen sind jedoch auch kritisch zu sehen, da sie Expertenempfehlungen und damit nicht zwangsläufig auf Forschungsergebnisse abgestützt sind. In der Analyse des Forschungsstandes werden Publikationen betrachtet, die bis zum Jahr 2007 erschienen sind.

4.2 Analyse des nationalen Forschungsstands zu Informatiksystemen und Kompetenzentwicklung in der Schulinformatik

4.2.1 Informatiksysteme im algorithmen-, anwendungs-, und benutzerorientierten Informatikunterricht

Informatiksysteme in der GI-Empfehlung zu Zielsetzungen und Inhalten des Informatikunterrichts

Die Empfehlung „Zielsetzungen und Inhalte des Informatikunterrichts", die von der GI 1976 veröffentlicht wurde, fordert eine verpflichtende informatische Bildung für alle Schulabgänger. Motivation ist der zunehmende Einsatz von Rechnern und damit einhergehend die wirtschaftspolitisch begründete Forderung, dass Schulabgänger einfache Problemlösungen maschinenverständlich formulieren können (Brauer et al. 1976, S. 35). Hauptzielsetzung im Unterricht ist daher das systematische Finden algorithmischer Lösungen inklusive Thematisierung von Daten- und Kontrollstrukturen. Sie beherrscht die drei erstgenannten von fünf Inhaltsbereichen: (1) Vom Problem zur Lösung (Algorithmen), (2) Realisierung von Algorithmen in einer algorithmischen Sprache, (3) Vertiefung des Gelernten durch Anwendung auf praxisorientierte Problemkreise, (4) Funktioneller Aufbau einer Rechenanlage und DV-Organisation, (5) Informatik in der Gesellschaft (Brauer et al. 1976, S. 35). Die Inhaltsebene „Funktioneller Aufbau einer Rechenanlage und DV-Organisation" ist relevant bezüglich des Informatiksystembegriffs, denn darin werden Zentraleinheit mit Speicher, Leitwerk und Rechenwerk sowie Abläufe bei der Befehlsverarbeitung in der Zentraleinheit behandelt (→ Inhalt 2: Strukturmodelle). Dennoch wird der Stellenwert der technischen Organisation des Rechners gegenüber der Algorithmik relativiert:

> „Nur sofern es zur Modellbildung beiträgt, soll man während des Unterrichts auch auf die Funktionseinheiten des verfügbaren Rechners wie Speicher, Register, Ein-Ausgabegeräte eingehen" (Brauer et al. 1976, S. 36).

Der Beitrag des Verstehens von Funktionseinheiten zur Modellbildung ist für Kompetenzentwicklung mit Informatiksystemen zu berücksichtigen (→ Methodik 2: Kognitive Modelle). Während betont wird, dass Möglichkeiten und Grenzen der Anwendung von Rechnern zu thematisieren sind (→ Ziel 3: Bewusste Anwendung), werden technische Maschinenmodelle als optional deklariert. Hinsichtlich Informatiksysteme und Kompetenzentwicklung stellt dies eine erhebliche Einschränkung dar, die im historischen Kontext jedoch als Abgrenzung der Algorithmik von den hardwareorientierten Ansätzen zu sehen ist. Mit dem Einbeziehen von Fragestellungen zur Informatik in der Gesellschaft ist die Empfehlung nicht vollständig dem

algorithmenorientierten Ansatz zuzurechnen. Dennoch werden insbesondere Methoden der Informatik in den Vordergrund gestellt, die den Programmerstellungsprozess unterstützen und ermöglichen. Wie wissenschaftspropädeutisch sinnvoll Algorithmen im Informatikunterricht auch sein mögen, so wird der Ansatz den Herausforderungen in einer von Informatiksystemen durchdrungenen Welt allein nicht gerecht. Dies gilt umso mehr, wenn man Basiskompetenzen zu Informatiksystemen in der Lebenswelt der Schüler als Bildungsziel hat.

Informatiksysteme im anwendungsorientierten Informatikunterricht

Bereits 1978 wird die Entmystifizierung von Rechnern als Ziel des Informatikunterrichts beschrieben (→ Ziel 4: Entmystifizierung):

> „Der Begriff ‚Computer' stimuliert bei vielen Menschen einerseits Gefühle mystischen Bewunderns einer götterähnlichen Vollkommenheit des Denkens und andererseits angstbesetzte Vorstellungen über die allmächtige, sich verselbständigende und den Menschen manipulierende Maschine" (Koerber 1978, S. 6f).

Des Weiteren wird auf die sich abzeichnende Bedeutung der Kommunikations- und Interaktionsmöglichkeiten mit der sich vom Werkzeug zum „Denkzeug" wandelnden Maschine hingewiesen, die auch als ADV-System (automatisierte Datenverarbeitung) bezeichnet wird.

Koerber et al. (1981) beschreiben den anwendungsorientierten Informatikunterricht. Sie betrachten Entwicklung und Anwendung von „Software bzw. von DV-Systemen" als zwei Stränge im Informatikunterricht (vgl. Koerber et al. 1981, S. 30). Sie unterscheiden zwischen problembezogener, modellbezogener und informatikmethodenbezogener Ebene. Der Softwareentwicklungsprozess verläuft von der Problemdefinition (problembezogene Ebene) über eine Modellbeschreibung (modellbezogene Ebene) zum Detailentwurf (informatikmethodenbezogene Ebene). Der Anwendungsprozess ist in diesem Modell analog zum Softwareentwicklungsprozess, allerdings ohne Einbezug der informatikmethodenbezogenen Ebene, die nur als automatisch ablaufender Prozess wahrgenommen wird, und mit stark reduziertem Anteil der modellbezogenen Ebene. Überschneidungen von Anwendung und Entwicklung gibt es im anwendungsorientierten Informatikunterricht somit hauptsächlich auf der problembezogenen Ebene.

Koerber et al. (1981) fordern, dass Anwendung von ADV-Systemen vollständig Gegenstand im Unterricht ist, d. h. von der Problemstellung bis zur Stilllegung des Systems thematisiert wird. Die Anwendungsorientierung umfasst somit die Schwerpunkte Modellbildung (→ Inhalt 2: Strukturmodelle; → Methodik 2: Kognitive Modelle) und softwaretechnisches Vorgehen, aber auch die Algorithmenorientierung. Es stellt sich die Frage, welche Informatikmethoden bzw. -vorgehensweisen wichtige Unterrichtsinhalte für Informatiksysteme und Kompetenzentwicklung sind, wenn Softwareentwicklung nicht im Vordergrund steht. Nach Forneck (1990) liegt die

Schwierigkeit des anwendungsorientierten Ansatzes in dem ganzheitlichen, vom Lebensweltbezug ausgehenden Vorgehen, in dem die Algorithmik weiterhin einen großen Anteil an Unterrichtszeit beansprucht. Zusätzlich ist nach Forneck die umfassende Thematisierung aller Konsequenzen von Informatikanwendungen für viele Lehrpersonen eine Überforderung.

1986 wird als Resultat des von 1977 bis 1983 durchgeführten Berliner Modellversuchs „Entwicklung von Curriculumelementen für den Informatikunterricht in der Sekundarstufe I" (ECIS) zur inhaltlichen Ausgestaltung eines anwendungsorientierten Informatikunterrichts gefordert, die Entmystifizierung von Rechnern durch die Darstellung ihres Werkzeugcharakters und des damit verbundenen Interessenbezugs (→ Ziel 5: Bereitschaften) zu erreichen (Bosse et al. 1986, S. 8). Dieser anwendungsorientierte Informatikunterricht integriert algorithmische und gesellschaftliche Aspekte auf Grundlage des Modellbildungsprozesses (vgl. Bosse et al. 1986, S. 9) und führt zu einem didaktischen Fünf-Phasen-Modell: Problem- und Zielformulierung, Problemanalyse und Modellansatz, Algorithmierung, Programmierung und Anwendung des Systems. Letzteres umfasst Analyse der Auswirkungen des Einsatzes des neu geschaffenen Arbeitsmittels. Zwei Dinge sind auffällig an der vorgestellten Konzeption: Zum einen wird die von Koerber et al. (1981) beschriebene Anwendungsorientierung durch die Orientierung am Softwareentwicklungsprozess im didaktischen Fünf-Phasen-Modell umgesetzt. Zum anderen ist es der Versuch der Entmystifizierung allein durch Darstellung des Werkzeugcharakters von Rechnern zur Lösung eines Anwendungsproblems. Ein direkter Bezug zwischen informatischen Konzepten, die sich in dem Artefakt widerspiegeln, wird nicht hergestellt.

Die GI fordert 1986 in der Rahmenempfehlung für die Informatik im Unterricht der Sekundarstufe I, dass ohne wissenschaftspropädeutischen Anspruch, aber dennoch fachlich vertretbar, zentrale Fragestellungen beantwortet werden müssen, z. B. was ein Rechner ist und wie er arbeitet. Das Aktivwerden des Schülers wird dabei besonders betont:

> „Ohne die Erfahrung des eigenen Tuns ist eine Grundkompetenz in Informatik nicht zu vermitteln. Hierbei brauchen Lösung und Programm nicht von Grund auf neu entwickelt zu werden, der Schüler kann auch durch Ändern und Erweitern vorgegebener Programme zu eigenen Ergebnissen kommen. Der Schüler soll einen Einblick in Prinzipien der Organisation und Verarbeitung von Information in computergestützten Systemen erhalten" (Löthe et al. 1986, S. 141).

Für Informatiksysteme und Kompetenzentwicklung in der Sekundarstufe II ist anregend, dass Datenbanksysteme und spezielle Anwendersysteme als Unterrichtsinhalte thematisiert werden sollen (→ Inhalt 1: Typische Repräsentanten) und Modifikation statt Konstruktion vorgeschlagen wird (→ Methodik 6: Modifikation statt Entwicklung).

1989 ist zur Legitimation und Aktualisierung des allgemein bildenden Informatikunterrichts das Argument der Vernetzung hinzu gekommen (Koerber et al. 1989,

S. 3). Weiterhin bilden Problemlösen und Modellbildung die Schwerpunkte des Unterrichts. Zur Abbildung auf den Rechner muss die Problemlösung sowohl in Algorithmen zur Ablaufbeschreibung der Problemlösung als auch in Datenstrukturen umgesetzt werden. Ein System wird als Repräsentation der vom Entwickler eingesetzten Problemlösungsstrategien gesehen:

> „Informationsverarbeitende technische Systeme stellen ein Werkzeug für das Lösen von Problemen dar, das stets menschliche Ideen, Wertungen und auch Irrtümer widerspiegelt" (Koerber et al. 1989, S. 9).

Für Informatiksysteme und Kompetenzentwicklung ist dies ein Ansatzpunkt dahingehend, dass eine systematische Analyse eines Informatiksystems die Verbindung zwischen dem nach außen sichtbaren Verhalten des Systems und informatischen Konzepten bilden kann (→ Methodik 3: Verbindung von Verhalten und Struktur; → Methodik 4: Analyse des Systems – Experimente). Es werden vier Handlungsperspektiven der Lernenden gesehen: als Betroffene hinsichtlich gesellschaftlicher Auswirkungen, als Benutzer in Beruf und Alltag, als Konstrukteur unter sozio-kulturellen Gegebenheiten und als Bediener, der die Funktionszusammenhänge versteht (vgl. Koerber et al. 1989, S. 13). Für Informatiksysteme und Kompetenzentwicklung implizieren diese Handlungsperspektiven einen von den Anwendungen ausgehenden Informatikunterricht (→ Inhalt 4: Sichten; → Methodik 7: Von der Anwendung zur Maschine). Darüber hinaus ist es aber auch notwendig, dass Grundlagen der Rechnerorganisation und der Vernetzung thematisiert werden, um Funktionszusammenhänge zu verstehen (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise).

1990 fordert Cyranek, Systeme als Black-Box zu betrachten und schrittweise von gesellschaftlichen Auswirkungen der Informationstechnologien, über Nutzung von Anwendungssystemen und Problemlösen mit algorithmischen Methoden zu Prinzipien der Rechnerarchitektur zu gelangen (Cyranek 1990, S. 2). Dabei beruft sich Cyranek auf die Vereinigung der Lehrerbildung in Europa Association for Teacher Education in Europe (ATEE; (Weert 1984)) :

> „Möglichkeiten, dieses Ziel zu erreichen, sind nach der ATEE: [...] Methoden der Modellbildung, die die Schüler befähigen, adäquate Vorstellungen über ihre Erfahrungen mit automatischen Systemen zu entwickeln. Die Entwicklung dieser Denkmodelle nutzt systematisch das Prinzip der ‚black boxes'" (Cyranek 1990, S. 3).

Die Forderung nach einem Vorgehen, das bei der Lebenswelt der Schüler mit Black-Boxes beginnt und schrittweise bis zu Prinzipien der Rechnerarchitektur voranschreitet, ergänzt für Informatiksysteme und Kompetenzentwicklung das Vorgehen nach Koerber et al. (1989), in dem der Schwerpunkt auf der Modellbildung mit Algorithmen und Datenstrukturen liegt (→ Methodik 7: Von der Anwendung zur Maschine).

Meier (1990) beschreibt zehn „Anforderungen an die Informatikausbildung in den neunziger Jahren aus Sicht der Wirtschaft". Eine Anforderung ist dabei besonders

interessant: Er fordert vernetztes Denken und neue Formen der Teamarbeit (vgl. Brauer und Münch 1996, S. 14), wobei er sich auf Vester (1988) beruft, der für den sinnvollen Einsatz aller Kommunikationsmedien plädiert, um neue Verständnisebenen zu erschließen:

> „Wer die Verflechtung eines Problems mit seinem Umfeld akzeptiert, vernetzt denkt, kommt folgerichtig zu neuen Formen der Teamarbeit" (Vester (1988) zitiert nach (Meier 1990, S. 57)).

Die Rolle des vernetzten Denkens ist für Kompetenzentwicklung mit Informatiksystemen zu klären (→ Methodik 9: Vernetzung der Unterrichtsinhalte). So wurde bereits von Koerber et al. (1989) betont, dass informationsverarbeitende technische Systeme Ideen widerspiegeln, die zur Problemlösung genutzt werden und demnach vernetzt sind. Dazu kommt von Vester (1988) der Aspekt des vernetzten Denkens durch Kooperation (Abschnitt 5.4.1; → Medien 3: Kooperation und Kommunikation).

Die Informatiklehrer und Fachberater für Informatik Gasper, Leiß, Spengler und Stimm aus Rheinland-Pfalz veröffentlichen 1992 eine informatikdidaktische Ausarbeitung zum Thema „Technische und theoretische Informatik" mit Lehr-Lernmaterial:

> „Es behandelt auf maschinenunabhängige Weise technische Grundlagen eines Computers, beschreibt ihn durch theoretische Automatenmodelle und zeigt prinzipielle, nicht durch die technische Entwicklung zu durchbrechende Grenzen seiner Leistungsfähigkeit auf. Schließlich werden die Grenzen des verantwortlichen Computereinsatzes im Zusammenhang mit gesellschaftlichen Aspekten der Informatik diskutiert" (Gasper et al. 1992, S. 3).

Ziel darin ist, zeitbeständige Methoden und Inhalte zu vermitteln. Inhaltlich werden dazu höhere Programmiersprachen, Assemblerprogrammierung, Rechnerarchitekturen, Schaltungen, Automatenmodellen, formalen Sprachen, Komplexität und gesellschaftliche Verantwortung thematisiert. Die Ausarbeitung beginnt mit der Denksportaufgabe des Bauern, der in einem kleinen Boot Kohlkopf, Wolf und Ziege ans andere Flussufer transportieren muss. Das Boot aber fasst nur den Bauern und eines der Tiere bzw. den Kohlkopf. Dabei würde ohne Aufsicht der Wolf die Ziege bzw. die Ziege den Kohl fressen. Die Problemstellung wird auf drei verschiedene Weisen gelöst: einmal mit einem Automatenmodell, mit logischen Schaltungen und mit einer formalen Sprache. Die enge Verzahnung von Technischer und Theoretischer Informatik verdeutlicht die Notwendigkeit, aber auch Möglichkeit eines ganzheitlichen Blicks auf Rechner (→ Inhalt 2: Strukturmodelle; → Inhalt 3: Fundamentale Ideen). Kritisch ist bei diesem Ansatz anzumerken, dass Grundbausteinen logischer Schaltungen und der Assemblerprogrammierung sehr viel Zeit eingeräumt wird.

Informatiksysteme im benutzerorientierten Informatikunterricht

Peschke prägt den Ansatz der Benutzerorientierung, der Schülern ein Instrumentarium zur selbständigen Einarbeitung in Standardsoftware vermitteln soll. Der

Ansatz vollzieht damit eine Abkopplung von der Informatik. Peschke spricht jedoch auch vom systemorientierten Ansatz, wobei er die Bezeichnung jedoch auf das sozio-technische System der Anwendung von Standardsoftware bezieht (Peschke 1991, S. 152). Dieser Ansatz ist nicht zu verwechseln mit dem systemorientierten Ansatz nach Baumann (Abschnitt 4.2.3). Die widersprüchliche Nutzung des Systembegriffs, die in Abschnitt 3.2.1 geklärt wird, ist Ursache der unterschiedlichen Interpretationen. Peschke argumentiert gegen die strikte Trennung von technischem Apparat und gesellschaftlichen Auswirkungen. Vielmehr sieht er in der Wechselwirkung zwischen Mensch und Computer ein tragendes Leitprinzip für den Informatikunterricht (Peschke 1990, S. 30). Forneck (1990) beschreibt rückblickend den benutzerorientierten Ansatz. Dieser meidet die Algorithmik, indem er auf Programmierung verzichtet und Anwendersysteme im Unterricht behandelt. Seine Grundlage ist, dass es mit modernen Anwendungssystemen möglich ist, innerhalb spezieller Anwendungsbereiche individuelle Lösungen zu entwickeln, ohne eine Programmiersprache erlernt zu haben:

> „Damit geht es beim benutzerorientierten Ansatz nicht mehr um Maschinenkunde oder um eine wissenschaftspropädeutische Einführung in die Fachdisziplin [...] Der benutzerorientierte Ansatz ist eine Allgemeinbildungskonzeption, die die technologische Entwicklung in erster Linie als epochales Kulturphänomen thematisiert" (Forneck 1990, S. 40).

Damit sind Qualifizierung zum rationalen Umgang mit Anwendungen sowie Steigerung der Beurteilungsfähigkeit Ziel der Benutzerorientierung (vgl. Hubwieser 2007a, S. 52). Für Kompetenzentwicklung mit Informatiksystemen sind diese Ziele ebenfalls relevant (→ Ziel 3: Bewusste Anwendung). Es mangelt dem Ansatz jedoch an einer ausgezeichneten Vorgehensweise:

> „Es gibt kein ausgezeichnetes methodisches Vorgehen, man durchläuft die folgenden Tätigkeitsgebiete in unterschiedlicher Reihenfolge (manche auch mehrfach): Finden, Erkennen und Analysieren eines Problems, Strukturieren des Problems und Entwickeln modellhafter Lösungsmöglichkeiten, Nutzen von Anwendersystemen und Programmierumgebungen, Beurteilen der Ergebnisse, Reflektieren und Bewerten der Nutzung der Technologien" (Hubwieser 2007a, S. 52).

Informatikunterricht für Kompetenzentwicklung mit Informatiksystemen muss jedoch die innere Struktur der Systeme, d. h. unterschiedliche Perspektiven aufgreifen (→ Inhalt 4: Sichten). Und es ist somit eine systematische Vorgehensweise zur Analyse von Informatiksystemen zu beschreiben (→ Methodik 4: Analyse des Systems – Experimente). Abschließend betont Forneck die Probleme bei einer möglichen Spaltung der Didaktik der Informatik, wenn die anwendungs- und benutzerorientierten Ansätze zunehmend in der Sekundarstufe I, der algorithmenorientierte Ansatz in der Sekundarstufe II vorherrschen. Er fordert, dass ein wissenschaftspropädeutischer Ansatz immer einer lebenspraktisch und damit ganzheitlich ausgerichteten Konzeption vorgeschaltet sein muss, da letztere allein nicht Selbstzweck

einer informationstechnischen Bildung sein dürfe (vgl. Forneck 1990, S. 49). Da das Ziel der vorliegenden Arbeit Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II ist, muss die Verbindung der Wissenschaftspropädeutik mit den Informatiksystemen in der Lebenswelt gelingen, ohne dabei hauptsächlich auf die Algorithmik zurückzugreifen. Die Wissenschaftspropädeutik vorzuziehen widerspricht jedoch den Kompetenzzielen. Für Basiskompetenzen liegt der Schwerpunkt auf der Anwendung von Informatiksystemen, in dem aber informatische Konzepte von Schülern identifiziert und verstanden werden müssen (→ Methodik 3: Verbindung von Verhalten und Struktur). Somit ist die im anwendungsorientierten Informatikunterricht vorherrschende Trennung von Anwendung und Entwicklung dahingehend zu modifizieren, dass für Kompetenzentwicklung mit Informatiksystemen Modellbildungsaspekte bei der Anwendung verstärkt hinzugefügt werden (→ Methodik 2: Kognitive Modelle).

Kompetenzentwicklung mit Informatiksystemen in Modrows anwendungsorientiertem Ansatz

Modrow setzt 1991 im Zuge der sich wandelnden Stellung der Informatik im Fächerkanon den informatischen Problemlöseprozess zur Begründung des Fachs Informatik in den Mittelpunkt. Denn problemlösendes Denken stellt in der Informatik keine Überforderung von Schülern dar, da in Programmen nur ein kleiner Weltausschnitt betrachtet wird:

> „Weiterhin stellen Computer ein Medium dar, in dem der Problemlösungsprozeß dynamisch ist, also durch ‚trial-and-error', ‚debugging', Fehlersuche, Tests und schrittweise Verbesserungen auch für durchschnittliche Schüler/-innen erfolgreich durchführbar" (Modrow 1991, S. 21).

Es ist festzuhalten, dass Modrows „Zur Didaktik des Informatikunterrichts" ebenso wie Baumanns „Didaktik der Informatik" (Baumann 1996) weniger von fachdidaktischer Theorie geleitet, sondern pragmatisch und methodisch angelegt ist (Abschnitt 4.2.3). Für den Informatikanfangsunterricht nennt Modrow folgende Lernziele: das Erlernen des Umgangs mit Rechner und Programmentwicklungssystem und Einblick in typische Arbeitsmethoden der Datenverarbeitung, Entwicklung von Algorithmen zur Lösung kleiner Probleme, Projektarbeit, gesellschaftliche Folgen der Datenverarbeitung sowie Nutzung des Rechners als Textverarbeitungssystem (Modrow 1991, S. 72). Diese begründet er damit, dass Problemanalyse, Begriffs- und Modellbildung, Auswahl von Datenstrukturen, Beschreibung von Problemlösungen und dazugehöriger Methoden, Realisierung, Test, Beurteilung und Dokumentation der Lösung nicht ohne Kenntnisse über das benutze Rechnersystem und seine Grenzen möglich ist (vgl. Modrow 1991, S. 34).
Modrow nennt folgende relevante Einsatzgebiete von Informatiksystemen, so dass eine Kategorisierung für den Informatikunterricht anhand ihrer Erscheinungsformen entsteht: Rechner, Datenverarbeitungssystem, Roboter, symbolverarbeitendes System (Computeralgebra, Sprachverarbeitung), Kommunikationsmittel, Planungsinstrument, Entscheidungsmaschine, künstliche Intelligenz (Modrow 1992,

S. 196ff). Diese Repräsentanten von Informatiksysteme können im Unterrichtmodell zu Informatiksystemen und Kompetenzentwicklung eingesetzt werden (→ Inhalt 1: Typische Repräsentanten).

Um Rechner anwenden zu können, müssen die Schüler Hardwarekenntnisse über Zentraleinheit mit Prozessor, Speicher, Eingabe und Ausgabe haben und die Abarbeitung von Programmen im Von-Neumann-Rechner kennen (vgl. Modrow 1991, S. 94), also ein Maschinenmodell (→ Inhalt 2: Strukturmodelle). Kenntnisse über das Betriebssystem werden als kaum erforderlich angesehen, da sie zu speziell und nicht allgemein bildend sind (Modrow 1991, S. 94). Argument ist, dass Schüler von der Existenz des Betriebssystems im Idealfall nichts bemerken. Historisch ist die Argumentation im Sinne der Anwendungsorientierung nachvollziehbar, denn darin wird der Softwareentwicklungsprozess meist für kleine Anwendungsprogramme besonders betont, nicht aber Betriebssystementwicklung im Speziellen. Hinsichtlich der Kompetenzentwicklung mit Informatiksystemen ist diese Auffassung Modrows jedoch kritisch zu sehen, denn gerade das Betriebssystem verwaltet Hard- und Software sowie die Netzverbindungen, die charakteristisch für Informatiksysteme sind. Im Zuge der Wissenschaftspropädeutik und bezüglich Kompetenzentwicklung mit Informatiksystemen sind Betriebssysteme somit ein relevantes Themengebiet. Bei dem Schwerpunkt Hardware zieht Modrow den Schaltungsentwurf mit dem Ziel der Entwicklung heran:

> „Zur Beschreibung derart komplexer Systeme [Rechner; Anm. d. V.] gehört die Benutzung von Blockschaltbildern, das Denken in Komponenten und die Beschreibungsformen der Rechnerarchitekturen. [...] Gehen wir davon aus, daß nicht die Einzelheiten des gerade benutzten Prozessors und seiner Komponenten behandelt werden [...], sondern prinzipielle Strukturen, dann ist das Blockschaltbild eines Von-Neumann-Rechners nicht sehr kompliziert und in kurzer Zeit zu erlernen" (Modrow 1992, S. 328).

Assemblerprogrammierung lehnt er ab, wenn sie nicht mittels (Modell-) Assembler die Brücke vom Rechnermodell zur höheren Programmiersprache schließt. Andernfalls sieht er darin nur eine weitere, überflüssige und komplizierte Programmiersprache (Modrow 1992, S. 329).

Das Programmentwicklungssystem in Kombination mit einem Texteditor wird von Modrow als das entscheidende Werkzeug angesehen, um nicht nur „Bedienerprobleme" im Informatikunterricht zu behandeln (Modrow 1991, S. 95). Die Problemlösung erfolgt somit auf höhersprachlichem Niveau und lässt das Problemlösen selbst zentrale Aufgabe des Unterrichts werden, statt immer wieder kleinste Programme zu schreiben (vgl. Burkert 1994b, S. 89). Jedoch bezweifelt Modrow indirekt die Übertragbarkeit von Erkenntnissen zur Hardware auf die Software und umgekehrt:

> „Computer sind ein typisches Beispiel für Systeme, die sich auf sehr unterschiedlichen Ebenen beschreiben und benutzen lassen. Dabei läßt die Kenntnis einer Ebene nur sehr bedingt Aussagen über Probleme zu, die auf einer anderen relevant sind. Konkret hilft die Kenntnis technischer Einzelheiten des Arbeitenden Mikroprozessors

> nichts, wenn mit einem Datenbanksystem gearbeitet werden soll; und selbst große Erfahrung mit Programmiersprachen ist unnütz für das Löten eines Druckerkabels" (Modrow 1991, S. 93).

Dieses Argument ist im historischen Kontext aber durchaus kritisch zu sehen, denn es setzt voraus, dass das Löten eines Druckerkabels gleichberechtigter Inhalt der Informatik ist, wie z. B. Datenbanksysteme. Dem kann mit den Kriterien für fundamentale Ideen der Informatik widersprochen werden (Abschnitt 4.2.2). Außerdem kann die logische Äquivalenz von Hardware und Software (Tanenbaum und Goodman 2001, S. 27) herangezogen werden, um den nur scheinbar unmöglichen Transfer von Erfahrungen mit technischen Aspekten auf die Softwareebene doch zu begründen.

Hervorzuheben ist, dass Modrow auf Standardsoftwaresysteme im Informatikunterricht nicht verzichten will, auch wenn er die herkömmlichen Programmentwicklungssysteme aufgrund ihrer Universalität (→ Methodik 1: Universalität) an erste Stelle setzt:

> „Werden also Datenbank-Fragestellungen behandelt, dann sollte auch irgendwann ein ‚echtes' Datenbanksystem gezeigt werden, ohne auf alle seine Spezialitäten einzugehen" (Modrow 1991, S. 62f).

Für Datenbanksysteme gibt Modrow einen möglichen und erprobten Unterrichtsgang an. Bemerkenswert daran ist, dass er nach der Vorstellung des Szenarios zur Erkundung eines Systems, eine „naive" Analyse des Systems bezüglich Teilfunktionen, deren Zusammenwirken und der Datenwege fordert, soweit dies von Schülern und Lehrern möglich ist ((Modrow 1991, S. 39), (Modrow 1992, S. 270ff)). Danach werden Teilprobleme ausgewählt und von den Schülern in einem reduzierten Modell implementiert:

> „Die mangelnde Effizienz und die Unvollständigkeit der entwickelten Programme kann eigentlich von den Beteiligten nicht als Nachteil empfunden werden, weil ja bewußt mit einem reduzierten Modell gearbeitet wird, das **Verständnis** für Datenbanken fördern soll, aber nicht unbedingt **funktionieren** muß" (Modrow 1991, S. 39; Hervorh. im Original).

Schüler müssen Informatiksysteme als Modell erkennen, das die Realität nicht vollständig abbilden kann. Für Kompetenzentwicklung mit Informatiksystemen ist es deshalb gut nachvollziehbar, dass Unvollständigkeit des reduzierten Modells keinen Nachteil darstellen muss. Schwierig hingegen ist der Aspekt des „Funktionierens" eines Programms, denn gerade Verhalten und Funktion sind unverzichtbare Charakteristika eines Informatiksystems (Abschnitt 3.2.2).

Zusammenfassend werden Datenbanksysteme, Textverarbeitungssysteme und Programmentwicklungssysteme als Werkzeuge und typische Repräsentanten von Informatiksystemen genannt. Der von Modrow formulierte Schritt des naiven Erkundens unter Verwendung einer Versuch-Irrtum-Strategie macht den Bedarf an systematischen Vorgehensweisen zur Erkundung von Informatiksystemen offensichtlich

(→ Methodik 4: Analyse des Systems – Experimente): Die Versuch-Irrtum-Strategie sollte vermieden werden (vgl. Brinda 2004b).

Als Fazit dieses ersten Abschnitts ist anzuführen, dass Entmystifizierung von Rechnern bereits als ein Ziel des Informatikunterrichts in den Publikationen formuliert wird. Vor allem die Vernetzung wird im anwendungsorientierten Informatikunterricht aufgegriffen. Dazu gehört auch eine ganzheitliche Sicht auf das Informatiksystem bzw. Anwendungen, in der nicht nur die Softwareentwicklung im Mittelpunkt steht. Technische Informatik und Algorithmik zur Erklärung von Rechnern rücken in den fachdidaktischen Publikationen in den Hintergrund. Dennoch sind die analysierten Publikationen hilfreich hinsichtlich der Kompetenzentwicklung mit Informatiksystemen. Anwendungsorientierung beispielsweise stellt Analogien zwischen Softwareentwicklung und Softwarenutzung her. Außerdem wird darauf hingewiesen, dass Informatiksysteme die ihnen zugrunde liegenden Entwurfsentscheidungen zumindest prinzipiell widerspiegeln. Gesellschaftliche Fragen des Einsatzes werden unterrichtsrelevant. Offen bleiben jedoch Fragen zu Schülervorgehensweisen bei der Erkundung von Informatiksystemen ohne Schwerpunkt auf Softwareentwicklung, und es fehlen Unterrichtskonzeptionen, die stärker auf bewusste Anwendung von Informatiksystemen fokussiert und gleichzeitig wissenschaftspropädeutisch sind.

4.2.2 Legitimation des Bildungswertes von Informatiksystemen durch fundamentale Ideen der Informatik

1993 transferiert Schwill (1993a) das erziehungswissenschaftliche Konstrukt der fundamentalen Ideen (Bruner 1960) zur Auswahl von Bildungsinhalten auf die Informatik, das den nichtspezifischen Transfer fördert. Dazu konkretisiert er Kriterien nach Bruner, Schreiber (1983) und Schweiger (1982), denen ein Konzept der Informatik genügen muss, um fundamentale Idee zu sein (Horizontal-, Vertikal-, Zeit-, Sinnkriterium). Schwill ergänzt das Zielkriterium, das zur Annäherung an eine idealisierte Zielvorstellung dient, die möglicherweise unerreichbar ist (vgl. Schubert und Schwill 2004, S. 85). Des Weiteren entwickelt Schwill eine heuristische Vorgehensweise zur Identifizierung von fundamentalen Ideen, indem die Kriterien ggf. mehrfach iterativ auf Inhalte der Informatik angewendet werden, um zu den Ideen als Abstraktion zu gelangen. Diese werden anschließend in Beziehung zueinander gesetzt. Zur Formulierung eines Katalogs von fundamentalen Ideen zieht Schwill den Softwareentwicklungsprozess heran, identifiziert in ihm wichtige Konzepte der Informatik und prüft sie auf die definierten Kriterien. Die in mehreren Phasen des Softwareentwicklungsprozesses auftretenden Ideen Algorithmisierung, Strukturierte Zerlegung und Sprache werden durch Clusterung zu Masterideen, die als Wurzelknoten die höchste Hierarchieebene ihres Teilbereiches darstellen.

Baumann beschreibt für den Informatikunterricht unabhängig von Schwill drei fundamentale Ideen der Informatik: Formalisierung, Automatisierung und Vernetzung

(Baumann 1996, S. 153), (Baumann 1998). Er leitet seine fundamentalen Ideen aus den nichttechnischen Einsatzgebieten des Rechnens, des Kommunizierens sowie des Steuerns und Regelns ab, was dadurch begründet ist, dass er die Informatik als Referenzwissenschaft für alle „Informationswissenschaften" sieht (Baumann 1996, S. 64). Auf den ersten Blick scheinen die fundamentalen Ideen nach Baumann damit enger an den Hauptfunktionen von Informatiksystemen nach Denning (Abschnitt 3.2.2) zu liegen. So stellt Humbert (2003) fest, dass Elemente der Technischen Informatik in den fundamentalen Ideen der Informatik nach Schwill unterrepräsentiert sind und ergänzt, dass ein Zugang zu sozio-technischen Systemen fehlt (→ Inhalt 4: Sichten):

> „Schwerwiegender erscheint dem Autor der fehlende Zugang zu soziotechnischen Systemen und den damit verbundenen Anwendungsbereichen" (Humbert 2003, S. 64).

Nachteil der fundamentalen Ideen der Informatik nach Baumann ist jedoch das Fehlen von Kriterien. Somit mangelt es ihnen an einem Auswahlmechanismus, der weniger subjektiv geprägt ist und die Auswahl von Inhalten nachvollziehbar macht. Darüber hinaus ist die Argumentation auf Basis aller Informationswissenschaften kritisch zu sehen, da durch sie weniger eine Orientierung am Fach Informatik zur Auswahl von Inhalten vorgenommen wird, als es nach Schwill der Fall ist.

Zur Bestimmung von Informatikinhalten zu Informatiksystemen und Kompetenzentwicklung ist es also notwendig, auf Kriterien zurück zu greifen, die selektiv sind. Da über den Wert der fundamentalen Ideen der Informatik nach Schwill in der Fachdidaktikforschung ein gewisser Konsens herrscht, der beispielsweise durch eine zunehmende Anzahl an Publikationen zu dem Thema belegt ist ((vgl. Schubert 2007), (Modrow 2002)), werden die Kriterien nach Schwill in der vorliegenden Arbeit zu Informatiksystemen und Kompetenzentwicklung eingesetzt (→ Inhalt 3: Fundamentale Ideen).

Hartmann et al. (2006) beziehen sich auf die Kriterien nach Schwill und geben zusätzlich das Repräsentationskriterium an: Eine fundamentale Idee soll sich enaktiv begreifen, ikonisch visualisieren und symbolisch beschreiben lassen können. Das Repräsentationskriterium liefert jedoch eine starke inhaltliche Überschneidungen mit dem Vertikalkriterium, denn wenn beispielsweise Lernende der Primarstufe ein informatisches Prinzip kennen lernen, dann geschieht dies es entweder enaktiv, d. h. für sie aktiv begreifbar repräsentiert, oder aber die Lernenden verstehen es auch ikonisch bzw. symbolisch so dass das Repräsentationskriterium überflüssig wird. Als Kriterium wird es in der vorliegenden Arbeit nicht herangezogen, dennoch ist die Wichtigkeit der Repräsentationsform unbestritten.

Von Schwill wird somit ein Fundament für die Begründung des allgemeinen Bildungswertes im Informatikunterricht gelegt. Er betont, dass fundamentale Ideen den nichtspezifischen Transfer unterstützen, d. h. unbekannte Varianten einer Problemstellung können als Spezialfälle eines vertrauten Grundkonzeptes erkannt und

Lösungsmethoden angepasst werden (Schubert und Schwill 2004). Damit unterstützen fundamentale Ideen den Anspruch an Kompetenz, dass eine Problemlösung in variablen Situationen erfolgreich eingesetzt werden kann, weshalb die Kriterien für fundamentale Ideen in der vorliegenden Arbeit zur Auswahl von Unterrichtsinhalten gewählt wurden. Frage ist, inwieweit sich fundamentale Ideen der Informatik bezüglich der Basiskompetenzen zu Informatiksystemen ausfindig machen lassen und wie sie im Lehr-Lernprozess vernetzt werden können (Abschnitt 5.3). Bereits der Titel dieses Abschnitts macht ein weiteres Dilemma der fundamentalen Ideen der Informatik deutlich. Anhand der Kriterien sind einzelne Konzepte auf ihren Bildungswert hin überprüfbar. Die Vernetzung der fundamentalen Ideen im Unterricht wird jedoch der Informatiklehrperson überlassen. Elegant lässt sich dieses Problem durch das Durchlaufen des Softwareentwicklungsprozesses lösen, denn darin wurden die fundamentalen Ideen der Informatik nach Schwill identifiziert und der Entwicklungsprozess bildet die logische Vernetzung. Für Kompetenzentwicklung mit Informatiksystemen ist die Lösung jedoch nicht mehr praktikabel, sind neben der Konstruktion doch auch das nach außen sichtbare Verhalten und die inneren Struktur von Informatiksystemen Schwerpunkte (→ Methodik 9: Vernetzung der Unterrichtsinhalte). In einer informatischen Bildung mit dem Ziel, Kompetenzentwicklung mit Informatiksystemen zu fördern, müssen gerade solche zugrunde liegenden informatischen Ideen genutzt werden, die unerwartetes Verhalten von Informatiksystemen erklärbar machen. In der Konsequenz sind Kandidaten für fundamentale Ideen der Informatik auch im Verhalten von Informatiksystemen bzw. im Anwendungsprozess zu identifizieren (→ Methodik 3: Verbindung von Verhalten und Struktur), auf die Kriterien nach Schwill zu prüfen und mit weiteren fundamentalen Ideen der Informatik bezüglich Informatiksysteme und Kompetenzentwicklung zu vernetzen (Abschnitt 5.4).

4.2.3 Kompetenzentwicklung mit Informatiksystemen in systemorientierten Ansätzen

Der systemorientierte Ansatz nach Baumann

Baumann führt erstmals einen systemorientierten Ansatz für die Schulinformatik ein (Baumann 1990), (Baumann 1993), (Burkert 1994a, S. 56). Zur theoretischen Fundierung seines aus der Schulpraxis kommenden Konzeptes bezieht er sich auf den Systembegriff als Strukturbegriff aller Wissenschaft. Die Struktur in Form von Ordnung und Organisation wiederum sei ein Synonym für Information (Baumann 1993, S. 12). Er folgert, dass das Pflichtfach Informatik stellvertretend für alle „Informationswissenschaften" wie Linguistik, Kognitionstheorie, Kybernetik, allgemeine Systemtheorie, Mathematik und Logik die notwendigen Begriffe, Denkmethoden zur Informationsverarbeitung vermitteln sollte (Baumann 1990, S. 13). Baumann sieht in der Anwender- und der Benutzerorientierung die Gefahr, dass Informatik

zur Sozialwissenschaft wird und folgert, dass ein Rechner nicht als isoliertes Einzelgerät, sondern als Teil umfassender Systeme gesehen werden muss (Baumann 1993, S. 12). Er erklärt, dass die „Ganzheitlichkeit" der Anwendungsorientierung überschätzt wurde und zur Legitimation des Informatikunterrichts nicht (mehr) nutzbar ist (Baumann 1993, S. 11), denn dergleichen beraube dem Informatikunterricht seines Fachprinzips:

> „Denn jenem Bestreben liegt ein gedanklicher Kurzschluß bzw. ein Irrglaube zugrunde – nämlich der, daß eine Wissenschaft, die sich der Gesellschaft verpflichtet weiß, indem sie sozial nützliche bzw. menschengerechte Informatiksysteme entwirft und gestaltet, darum bereits Sozialwissenschaft oder gar eine ‚Wissenschaft vom Menschen' sei" (Baumann 1996, S. 3).

Neben die Hard- und Softwareseite tritt im systemorientierten Ansatz die Perspektive des Benutzers (→ Inhalt 4: Sichten). Baumann (1996) fasst die Informatik als konstruktive Wissenschaft des Entwurfs und der Gestaltung von Informatiksystemen auf. Ein Informatiksystem definiert er folgendermaßen:

> „Unter einem **Informatiksystem** versteht man ein verteiltes, heterogenes technisches System, das Wissen unterschiedlicher Art und Herkunft repräsentiert, diese Wissensrepräsentationen in Gestalt von Daten und Programmen verarbeitet und den Benutzern in geeigneter Form zur Verfügung stellt" (Baumann 1996, S. 164; Hervorh. im Original).

Der Gestaltung der Benutzungsoberfläche kommt somit entscheidende Bedeutung zu (vgl. Baumann 1993, S. 12). Für Kompetenzentwicklung mit Informatiksystemen ist dies wieder ein Hinweis auf den engen Zusammenhang von Verhalten und Struktur (→ Methodik 3: Verbindung von Verhalten und Struktur). Baumann integriert den Ansatz der Informatik als Wissenstechnik (vgl. (Luft und Kötter 1994), (Schubert und Schwill 2004, S. 12f)), um Informatiksysteme zu erklären:

> „Darunter sind technische Systeme zu verstehen, die Wissen enthalten,
> - das Systemanalytiker und Wissensingenieure in Zuge einer Anforderungsanalyse gewonnen,
> - Systemarchitekten und Programmierer verstanden, und das
> - in Form von Daten und Programmen mit Hilfe geeigneter Werkzeuge (Programmiersprachen, Übersetzer, Datenhaltungssystemen) implementiert sowie
> - mit Hilfe von Zugangs-, Betriebs- und Nutzungssystemen verfügbar gemacht worden ist" (Baumann 1996, S. 94).

Damit formuliert er die Auffassung, dass Informatiksysteme Wissen im Sinne der Künstlichen Intelligenz repräsentieren. Grundlage seines Konzeptes sind neben allgemeinen Bildungszielen vier Grundkategorien der Informatik, nämlich Information, System, Modell und Programm, sowie die drei fundamentalen Ideen Formalisierung, Automatisierung und Vernetzung (Baumann 1996, S. 153). Letztere tritt beispielsweise auf bei parallelen und verteilten Systemen, Aktoren und Agenten sowie bei Rechnernetzen. Diese Ideen dürfen jedoch nicht verwechselt werden mit

den fundamentalen Ideen der Informatik nach Schwill (1993a) (Abschnitt 4.2.2). Als „Gesichtspunkte" der Didaktik der Informatik formuliert er A) Problemlösen mit Informatiksystemen, B) Wirkprinzipien von Informatiksystemen und C) Grundlagen und Grenzen technischer Wissensverarbeitung (Baumann 1993, S. 12). Ab 1996 spricht er von „Leitlinien". Ihnen ordnet er nachfolgende Leitfragen zu:

> A) „Wie können durch Entwicklung, Gestaltung und Anwendung von Informatiksystemen Probleme der Lebenswelt gelöst werden?
> B) Wie sind Informatiksysteme aufgebaut, welches sind die Prinzipien des Zusammenwirkens ihrer Komponenten und wie ordnen sie sich in größere Systemzusammenhänge ein?
> C) Wo liegen die Grenzen formaler bzw. technischer Wissensverarbeitung, und wie kann – angesichts der fortschreitenden Automatisierung geistiger Tätigkeiten – die kognitive Autonomie menschlicher Subjekte gewahrt werden?" (Baumann 1996, S. 120), (vgl. auch Baumann 1993, S. 12).

In der Leitlinie „Problemlösen mit Informatiksystemen" sollen Schüler typische Einsatzbereiche und exemplarische Anwendungen der Informatik in der Gesellschaft diskutieren und reflektieren, Methoden der Modellierung von Realitätsausschnitten kritisch nutzen, Werkzeuge problemangemessen auswählen, Entwurfsmethoden für Informatiksysteme kennen und anwenden sowie Probleme und Algorithmen hinsichtlich Komplexität beurteilen können (Baumann 1996, S. 171f).

In der Leitlinie „Wirkprinzipien von Informatiksystemen" sollen Schüler Digitalisierung als Prinzip der Informationsverarbeitung kennen, anwenden und bewerten sowie die Verarbeitung durch programmgesteuerte Automaten und die entsprechende Rechnerarchitektur (Von-Neumann-Rechner, Parallelarchitekturen) verstehen. Außerdem sollen sie Aufgaben, Funktionen und Datenmodelle von Informationssystemen kennen. Dazu müssen sie einfache Datenbanksysteme entwerfen können, Struktur und Funktion von Netzen bei der Informationsverarbeitung kennen und als sozio-technische Systeme verstehen sowie Risiken komplexer Hard- und Softwaresysteme hinsichtlich ihres Einsatzes einschätzen können (Baumann 1996, S. 172). Interessant ist an dieser Stelle die Benutzung des Begriffs des Informationssystems, das die Aufgabe hat, Daten zu speichern, zu verwalten sowie deren Auswertung zu unterstützen und daher meist ein Datenbanksystem umfasst. Baumann betont, dass jedes Informationssystem ein Informatiksystem sei, dies jedoch nicht umgekehrt gelte (Baumann 1996, S. 287). Mit dem Hinweis auf sozio-technische Systeme stellt Baumann heraus, dass Informatiksysteme aus der Gesellschaft heraus entstehen und in diese zurückwirken (Baumann 1993, S. 12). Für Kompetenzentwicklung mit Informatiksystemen ist gerade diese Leitlinie zu beachten, denn wechselwirkende Teilsysteme, deren Zusammenschluss in Netzen und die freie Programmierbarkeit sind zentral (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise; → Methodik 1: Universalität).

Lernziele in der Leitlinie „Grundlagen und Grenzen der technischen Wissensverarbeitung" sind, dass die Schüler Rechner als universelle symbolverarbeitende Ma-

schine begreifen lernen und in die Entwicklung von Technik und Kultur einordnen können. Neben der Universalität, die auch bei den Wirkprinzipien thematisiert wird, sind die Grenzen der Problemlösung mit Rechnern in dieser Leitlinie Hauptthema. Außerdem sollen sie formale und natürliche Sprachen in Zusammenhang bringen und Grenzen formaler Kommunikation kennen (→ Ziel 3: Bewusste Anwendung). Grundlegende Konzepte symbolischer Informationsverarbeitung und prinzipielle Grenzen der Berechenbarkeit und Effizienzsteigerung sollen sie kennen sowie Grenzen des Einsatzes von Informatiksystemen aufgrund individueller und gesellschaftlicher Verantwortung beachten (Baumann 1996, S. 172f). Exemplarisch skizziert er eine Kursfolge für sechs Schulhalbjahre der Sekundarstufe II, die sich an Anwendungsgebieten orientiert: Datenbanken und Informationssysteme, Sprachverstehen mit natürlichsprachlicher Datenbankabfrage, elementare Computergraphik, Strategiespiele, Compilerbau und Sicherheit im Datennetz mit kryptologischen Algorithmen (Baumann 1996, S. 198).

Fasst man Baumanns Ausführungen zusammen, dann beschreibt er einen stark an theoretischen Begriffen orientierten Ansatz für die Schulinformatik. Die konkrete Auswahl der Unterrichtsinhalte ist von ihnen ausgehend argumentativ nachvollziehbar, sie ist im Einzelfall jedoch nicht ausgrenzend und verschiebt den Schwerpunkt des Informatikunterrichts einseitig zu Themen der Künstlichen Intelligenz und der damit verbundenen Informations- und Wissensverarbeitung. Systemorientierung versteht Baumann als Synergetik, d. h. das Zusammenwirken. Es ist das grundlegende Prinzip, das Baumann zu thematisieren versucht: das Zusammenwirken von Teilsystemen untereinander, von verschiedenen Wissensformen und bei der Systemgestaltung die Kooperation zwischen Auftraggeber und Auftragnehmer (Baumann 1996, S. 115). Hinsichtlich der Kompetenzentwicklung mit Informatiksystemen ist Baumanns Leistung vor allem in der Nutzung des Begriff des Informatiksystems zu sehen, womit er den Vernetzungsaspekt besonders hervorhebt (→ Methodik 9: Vernetzung der Unterrichtsinhalte).

Der systemanalytische Anfangsunterricht nach Lehmann

Der Berliner Lehrer und Informatikfachseminarleiter Lehmann (1993) stellt einen systemanalytischen Einstieg in den Informatikunterricht vor, der sich am damaligen Berliner Informatiklehrplan orientiert, in dem komplexe Systeme im Mittelpunkt des Anfangsunterrichts stehen. Lehmann publiziert in Folge mehrere Unterrichtsbeispiele in der Zeitschrift LOG IN und spricht von der fundamentalen Idee „Arbeiten mit komplexen Systemen" (vgl. Lehmann 1995), die er allerdings nur oberflächlich auf die Kriterien nach Schwill prüft (→ Ziel 2: Arbeiten mit komplexen Systemen). Somit kann es nicht als Nachweis angesehen werden, dass es sich um eine fundamentale Idee handelt, zumal einerseits die per Definition geforderte Komplexität hinsichtlich des Vertikalkriteriums kritisch zu sehen ist, und weil es sich andererseits um einen sehr weit gefassten Bereich handelt, in dem sich viele feingranularere fundamentale Ideen der Informatik finden lassen.

Schwerpunkte und Phasen der Konzeption nach Lehmann sind Benutzung (die Sicht von außen), Analyse (der Blick in das System), Wartung des Systems mit kleinen Anpassungen (Änderung) und Konstruieren eines neuen Systems (Konstruktion) (Lehmann 1993, S. 134), (Lehmann 1995, S. 31). Durch Anfangsunterricht zu komplexen Systemen wird vernetztes Denken gefördert (→ Methodik 9: Vernetzung der Unterrichtsinhalte), denn es werden von Anfang an sowohl größere Zusammenhänge im Softwaresystem als auch mit dessen gesellschaftlichem Kontext thematisiert (vgl. Lehmann 1993, S. 134). Dabei ist jedoch die Kritik anzubringen, dass auch die von ihm beschriebenen komplexen Systeme nur relativ einfache Unterrichtsbeispiele sind. Er selbst hat den Ansatz mehrfach erprobt (Lehmann 1993, S. 135). Den Schülern muss das komplexe System kompiliert mit Dokumentation und als Quelltext vorliegen. Die Benutzungsoberfläche soll klar und der Funktionsumfang begrenzt sein, damit ihre Erkundung hilfreich für eine Detailanalyse des Systems ist:

> „An dem Hauptmenü des Lottosystems erkennen wir die hauptsächlichen Funktionen [...]. Damit dürfte deutlich geworden sein, daß bereits ein Blick auf die Benutzeroberfläche viel vom Aufbau des Systems verrät und eine nachfolgende Detailanalyse erleichtert" (Lehmann 1995, S. 31).

Dabei bezieht sich die Aussage Lehmanns vor allem darauf, dass sich einzelne Funktionalitäten ebenso wie unterschiedliche Menüs identifizieren lassen, die wiederum intern im System in verschiedenen Module gekapselt sind (→ Methodik 3: Verbindung von Verhalten und Struktur). Für das zu analysierende komplexe System fordert Lehmann folgende Eigenschaften (Abschnitt 5.6.2):

- „Klare Benutzerführung,
- gute Strukturierung mit deutlich erkennbarer Modulhierarchie,
- relativ begrenzter Funktionsumfang, der sich z. B. in den Menüs widerspiegelt,
- gut verständliche Dokumentation (kann auch im Programm integriert sein),
- einige für eine Überarbeitung leicht zugängliche Programmteile" (Lehmann 1993, S. 135).

Nach Lehmann bedeutet „verständiger Computereinsatz" (Lehmann 1995, S. 37) u. a. Auswahl geeigneter Anwenderprogramme, Zielgerichtetheit, Angemessenheit der Eingaben und der Reaktion auf Ausgaben des Rechners, planmäßiges Auswählen und Dokumentieren von Ergebnissen sowie ggf. das Konstruieren eines kleinen Programms (→ Ziel 3: Bewusste Anwendung) mit Hilfe passender Bausteine wie Prozeduren, Funktionen oder Datentypen (Lehmann 1993, S. 136).

Konkrete Lernziele werden für das erste Unterrichtsjahr genannt. Im Bereich Benutzung eines dokumentierten Systems (10 Stunden) sollen die Schüler ein Programmsystem benutzen, System- und Programmstruktur erkennen, wieder verwendbare Bausteine finden, mit einem Editor arbeiten und kleine Programmänderungen bzw. Wartungsarbeiten mit Lehrerhilfe durchführen können. Im Bereich Konstruktion von Teilalgorithmen zu Anwendungsfällen (35 Stunden) sollen sie algorithmische

Elemente kennen lernen, Elemente der Programmiersprache erlernen und Bausteine verwenden können. Weitere Inhalte sind Anwendungen und Auswirkungen der Datenverarbeitung (20 Stunden), Rechnerorganisation (10 Stunden) und Entwicklungsgeschichte der Datenverarbeitung (5 Stunden). Im zweiten Unterrichtsjahr stehen u. a. die Konstruktion eines Programmsystems zur Dateiverwaltung und Anwendung eines relationalen Datenbanksystems an. Im dritten Jahr ist die weitgehend selbständige Konstruktion eines komplexen Systems im Lehrplan vorgesehen (Lehmann 1995, S. 32).

Gründe dafür, dass diese Erfolg versprechende Top-down-Vorgehensweise von der Anwendung zur Maschine (→ Methodik 7: Von der Anwendung zur Maschine) kaum Verbreitung fand, liegen vermutlich darin, dass unterschiedliche Vorkenntnisse der Schüler ungenügend berücksichtigt werden und sie erheblichen Vorbereitungsaufwand der Lehrpersonen erfordert. Denn der Ansatz bedarf eines geeigneten Softwaresystems und wieder verwendbarer Programmbausteine, die Blick auf und in das System in Einklang bringen sowie Wartung und Konstruktion vereinfachen. Darüber hinaus ist die Vorgehensweise offenbar nur von Lehmann erprobt worden, so dass die Abhängigkeit von der Lehrperson nicht ausreichend analysiert wurde. Für ein Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung, das auf Softwareentwicklung in weiten Teilen verzichtet, bietet der systemanalytische Ansatz (→ Methodik 4: Analyse des Systems – Experimente) mit Einbeziehung des Modifizierens (→ Methodik 6: Modifikation statt Entwicklung), Veränderns und systematischen Testens vorhandener Informatiksysteme jedoch wichtige Anregungen.

Foegens systemorientierte Didaktik

Foegen entwickelt 1996 im Rahmen einer Diplomarbeit ein Konzept für eine systemorientierte Didaktik der Informatik, die das Gestalten sozio-technischer Systeme zum Leitbild hat. Als Grundlage seiner Konzeption nutzt er die allgemeine Systemtheorie, für die er die Erfüllung der Kriterien für fundamentale Ideen der Informatik nach Schwill nachweist. Kritik an dem Nachweis ist jedoch dahingehend zu üben, als dass einer Universaltheorie wie der Systemtheorie das Informatikspezifische fehlt. Ebenso könnte die Systemtheorie in einem Fach wie der Biologie Grundlage für die Betrachtung biologischer Systeme sein oder in einem anderen Fach hinsichtlich sozialer Systeme. An den Anfang setzt Foegen das Modul „Grundlagen des Verstehens und Gestaltens von Systemen", betont aber insbesondere soziale und die in ihnen eingebetteten technischen Systeme, explizit schließt er reine Softwaresysteme aus (Foegen 1996, S. 38). Er betont dabei die wichtige Rolle von Strukturierungsmustern in Systemen, die z. B. Softwaremuster sein können (Gamma et al. 1995) oder in sozialen Systemen Handlungsmuster und Rollenverhalten. Nacheinander werden anschließend die Module „Systeme als Dienstleister", „Auftraggeber und Auftragnehmer", „Informationen strukturieren", „Akteure", „Wissen und seine

Grenzen", „Kommunikation", „Rechner als Ausführungsorgane" sowie „Gestalten von Systemen" abgearbeitet. Letzteres verläuft in Teilen parallel zur ansonsten sequentiellen Gliederung. In dem vorletzten Modul „Rechner als Ausführungsorgane" betrachtet Foegen Rechner als Systeme aus Hard- und Software und thematisiert Betriebssystem, Übersetzer, Schaltwerke und deren Zusammenhang zu Automatentheorie und Robotik (→ Inhalt 3: Fundamentale Ideen):

> „Im Mittelpunkt dieses Kapitels steht die Frage, wie ein Rechner und seine Teile als dienstleistende Systeme aufgefaßt werden können und wie diese zusammenarbeiten. Dabei wird absteigend vorgegangen, indem zunächst jedes Teil (-System) von außen betrachtet und seine Funktion analysiert und danach zur Innenansicht übergegangen wird" (Foegen 1996, S. 76).

Diese Vorgehensweise eignet sich ebenfalls für Kompetenzentwicklung mit Informatiksystemen (→ Methodik 7: Von der Anwendung zur Maschine). Zusammenfassend ist zu sagen, dass der systemorientierte Ansatz von Foegen sowohl bei der Betonung der allgemeinen Systemtheorie als auch bei dem für alle Ingenieurwissenschaften geltenden Gestalten sozio-technischer Systeme nicht deutlich macht, warum dafür gerade der Informatikunterricht zuständig ist. Darauf aufbauend kommt Foegen dennoch zu Schlussfolgerungen, die für Kompetenzentwicklung mit Informatiksystemen hilfreich sind. So betont er Funktion, Verhalten und Zweck eines Systems ebenso wie die Wichtigkeit von Strukturierungsmustern (→ Inhalt 2: Strukturmodelle; Abschnitt 5.4). Darüber hinaus gliedert er den Informatikunterricht von einfachen zu komplexen Systemen und betont sowohl passive als auch aktive Systeme, deren Akteure Erkenntnisse der Künstlichen Intelligenz nutzen (vgl. Brauer und Brauer 1995).

Kompetenzentwicklung mit Informatiksystemen im informationszentrierten Ansatz

Hubwieser und Broy stellen 1997 den informationszentrierten Ansatz für den Informatikunterricht an Gymnasien vor. Zur Legitimation des Unterrichts wird in der Informationszentrierung das Grundschema der Informationsverarbeitung mit Repräsentation, Verarbeitung, Transport und Interpretation von Information in Anlehnung an das EVA-Prinzip (Eingabe-Verarbeitung-Ausgabe) herangezogen. Sie stellen fest, dass Informations- und Kommunikationssysteme im Unterricht unterschiedliche Rollen erfüllen: Unterstützung von Lernvorgängen, Schulung von Bedienerfertigkeiten und Unterstützung der Aneignung langlebiger Informatikgrundlagen (Hubwieser und Broy 1997a, S. 40f), (Hubwieser 2007a, S. 43f). Nach Hubwieser und Broy ist es der letztgenannte Aspekt, der den Informatikunterricht im eigentlichen Sinne ausmacht, d. h. dass im Unterricht Konzepte und Methoden zur Planung, Konstruktion, Beschreibung und Bewertung abstrakter Systeme thematisiert werden.

Begrifflich werden die Informations- und Kommunikationssysteme darauf aufbauend von Hubwieser allgemein als Informatiksysteme bezeichnet. Diese gliedern sich

in das Grundschema der Informationsverarbeitung ein, so dass ihre Wirkungsweise in drei Bereiche eingeteilt werden kann:

- „automatische Verarbeitung,
- Vernetzung (Transport von Repräsentation, räumliche Verteilung von Verarbeitung oder Repräsentationen) und
- Interaktion (Repräsentation und Interpretation)" (Hubwieser 2007a, S. 80).

Hubwieser sieht dieses Grundschema (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise) als Abstraktion vorheriger fachdidaktischer Ansätze, die auf Informatiksysteme abzielen, wie die Leitlinien von Friedrich (1995a) zu Umgang mit Information, Wirkprinzipien von Informatiksystemen, Problemlösen mit Informatiksystemen und Arbeiten mit Modellen (Hubwieser 2007a, S. 82). Um mögliche Unterrichtsinhalte zu identifizieren, werden die Phasen des Grundschemas sowie die Wechselwirkungen von Informatiksystemen mit ihrer Umgebung untersucht, woraus nach Hubwieser jedoch nur eine Inhaltsauflistung ohne Kategorisierung entsteht (Hubwieser 2007a, S. 81). Die Darstellung von Information erfordert beispielsweise die Aufteilung eines Systems in Subsysteme, die Betrachtung von Datenflüssen zwischen Subsystemen und zur Umwelt sowie Modelle von Informatiksystemen inklusive zeitlicher Abläufe. Bei der automatischen Verarbeitung von Informationsrepräsentationen sind Anwendungsmöglichkeiten von Systemen zu betrachten sowie die zeitliche und räumliche Struktur von Informatiksystemen mit Komponenten, Verteilung, Kooperation und Abläufen. Hinsichtlich der Interaktion sind Möglichkeiten zum Schutz vor unerlaubten oder unerwünschten Interpretationen mittels Datenschutz, Zugriffsrechten und Verschlüsselung zu betrachten. Es ist auf Fehlinterpretationen durch Manipulationsmöglichkeiten und Darstellungsfehler einzugehen. Zur Vorbereitung der Schüler auf zukünftige Lebenssituationen zieht Hubwieser auch mögliche Rollen des Menschen gegenüber Informatiksystemen heran. Aus den Rollen wiederum lassen sich Anforderungssituationen zur Formulierung von Basiskompetenzen zu Informatiksystemen herleiten (→ Inhalt 4: Sichten). Diese sind: Entscheider in Firmen zu Kauf oder Eigenentwicklung von Informatiksystemen, Planer im Sinne von Projektleitern bei der Entwicklung eines Systems, Entwickler, Administratoren, direkte und indirekte Nutzer sowie Betroffene, deren Arbeitsplätze sich durch Informatiksysteme stark ändern (Hubwieser 2007a, S. 63).

Dabei betont Hubwieser, dass für professionelle Benutzer, Administratoren und Entwickler von Informatiksystemen Aus- und Weiterbildungsangebote sowohl als Produktschulungen als auch im Sinne einer fundierten Informatikausbildung vorhanden sind, alle anderen aber durch Schule auf ihre Aufgaben vorbereitet werden müssten (Hubwieser 2007a, S. 62). Aus Hubwiesers Forderung ergibt sich in der Konsequenz, dass Informatikunterricht darauf vorbereiten muss, Kaufentscheidungen aufgrund informatischer Bildung zu treffen, nichtprofessionellen Anwendern eine bewusste Nutzung von Informatiksystemen zu ermöglichen und Betroffene in die Lage zu versetzen, Veränderungen nachvollziehen und mitgehen zu können. Für

Kompetenzentwicklung mit Informatiksystemen kann Hubwiesers Forderung dahingehend präzisiert werden, dass auch Produktschulungen eines Grundverständnisses für Informatiksysteme bedürfen, um nachhaltig zu sein. Ebenso benötigen Planer, die Anforderungen an Informatiksysteme erstellen, Kenntnisse zu Möglichkeiten und Grenzen von Informatiksystemen. Eine konkrete Ableitung von Unterrichtsinhalten, Lehr-Lernmethodik und Lernmedien aus den jeweiligen Rollen nimmt Hubwieser nicht vor.

Zur Auswahl der Unterrichtsinhalte im informationszentrierten Ansatz begründen Hubwieser und Broy eine Klassifizierung nach Anwendungsbreite: In Klasse 1 fallen Unterrichtsinhalte der Informatik, die auch außerhalb von elektronischen Rechenanlagen Anwendung finden. Klasse 2 beinhaltet Konzepte, die charakteristisch für alle elektronischen Informations- und Kommunikationssysteme sind. In Klasse 3 befinden sich Konzepte, die nur für eine Systemklasse charakteristisch sind. Klasse 4 umfasst Konzepte, die nur spezielle Subsysteme betreffen (Hubwieser und Broy 1997a, S. 44). Nur Themen der Klassen 1 und 2 sind laut Hubwieser uneingeschränkt für den Informatikunterricht geeignet. Themen der Klasse 3 sind nur für wichtige Systemklassen, Inhalte der Klasse 4 allenfalls als Grundlage für das Verständnis von Themen der niedrigeren Klassen verwendungsfähig (→ Methodik 7: Von der Anwendung zur Maschine). Die Klassifikation wird für das Unterrichtsmodell aufgegriffen (Abschnitt 5.3).

Außerdem zieht Hubwieser die drei Masterideen Algorithmisierung, Sprache und strukturierte Zerlegung nach Schwill heran und stellt heraus, dass sie „ungefähr" den Bereichen Verarbeitung, Darstellung und Verteilung von Information entsprechen (Hubwieser 2007a, S. 83). Zur konkreten Auswahl von Unterrichtsinhalten werden die Kriterien Lebensdauer und Vermittelbarkeit in Anlehnung an die Kriterien für fundamentale Ideen formuliert und um das Exemplarische ergänzt. Davon ausgehend wird die Modellierung als inhaltlicher Kern der Informationszentrierung definiert. In dem durch die vorliegende Arbeit angestrebten Unterrichtsmodell soll jedoch Kompetenzentwicklung mit Informatiksystemen den inhaltlichen Kern bilden. Fragen der Modellierung von Aspekten wie der Zerlegung eines Systems in Subsysteme, zur Darstellung der Kommunikation der Subsysteme und zur Beschreibung der inneren Struktur informationsverarbeitender Subsysteme, die Hubwieser und Broy aufwerfen, sind dessen ungeachtet wichtig bezüglich der inneren Struktur und des Verhaltens von Informatiksystemen (Hubwieser und Broy 1997a, S. 44):

> „Am Anfang [der Systembeschreibung; Anm. d. V.] steht eine Partition des zu behandelnden Systems in Subsysteme (Objekte), deren innere Struktur (Attribute), Verhalten (Operationen) und hierarchische Klassifizierung (Klassen und Vererbungsbeziehungen, Instanzen) dann zu beschreiben sind. Zusätzlich muß das Verhalten des Gesamtsystems von außen gesehen und die Kommunikation zwischen den einzelnen Objekten (Assoziationen) bestimmt werden." (Hubwieser und Broy 1997b, S. 44)

Damit sind nach außen sichtbares Verhalten und innere Struktur als Charakteristika von Informatiksystemen direkt angesprochen und müssen miteinander in Beziehung gesetzt werden (→ Methodik 3: Verbindung von Verhalten und Struktur):

> „Das Verhalten des Gesamtsystems entsteht aus dem Zusammenspiel seiner Objekte, die durch Beziehungen verbunden sind und gegenseitig ihre Methoden durch Botschaften aktivieren" (Hubwieser 2007a, S. 95).

So wird betont, dass in Informationsrepräsentationen aller Informations- und Kommunikationssysteme abstrakte Datentypen als gemeinsame Grundstrukturen auftauchen, die als Bausteine komplexerer Strukturen genutzt werden können. Als Beschreibungstechniken fordern Hubwieser und Broy einprägsame grafische Notationen und nennen neben statischen Modellen auch Objekt- und Zustandsübergangsdiagramme, Sequenzdiagramme, Datenflussmodelle und Aktionsgraphen (→ Inhalt 2: Strukturmodelle), also funktionale bzw. Verhaltensmodelle zur Darstellung von Kommunikation der Subsysteme (Hubwieser und Broy 1997a, S. 46).

Hubwieser nennt, Bezug nehmend auf Winograd und Flores (1988), drei Kriterien für Software, die Ansatz für eine wirksame Bedienerschulung im Sinne einer informatischen Bildung sein können und damit für Kompetenzentwicklung mit Informatiksystemen interessant sind: (1) Zuhanden sein zur Lösung tatsächlicher Probleme sowie (2) Bewusstsein von Blindheit der Software gegenüber Problemen außerhalb ihres Einsatzbereichs, für den sie modelliert wurde. Dazu kommt (3) die Vorwegnahme von Pannen:

> „Ebenso [...] sollten auch bei der Bedienerschulung möglichst viele Fälle von Fehlfunktionen angesprochen werden. Damit wird die Umgebung der Systeme ebenso wie ihre innere Struktur durchleuchtet. Dem Nutzer werden die Grenzen ihrer Anwendbarkeit und ihre Leistungsfähigkeit klar (Hubwieser 2007a, S. 47).

Für Kompetenzentwicklung mit Informatiksystemen folgt daraus, dass gerade Fehler die Kombination der Sichten des nach außen sichtbaren Verhaltens und der inneren Struktur erfordern (→ Methodik 3: Verbindung von Verhalten und Struktur). 1999 beschreibt Hubwieser ein pragmatisches Gesamtkonzept für das Gymnasium beginnend in Jahrgangsstufe 6 zur Vorbereitung informatischer Denkweisen. Die Funktionsweise von Rechenanlagen soll beispielsweise in Blockschaltbildern veranschaulicht werden (Hubwieser 2007a, S. 101). Er betont, dass es für Schüler erst ab diesem Alter aufgrund der Abstraktionsfähigkeit möglich ist, über das konkret verwendete Informatiksystem hinaus gehend Grundprinzipien zu verstehen (Hubwieser 2007a, S. 100f). Wahlmodule in den Jahrgangsstufen 7-9 (Hubwieser 1999, S. 171) ermöglichen die spielerische Beschäftigung mit Informatik. Für Kompetenzentwicklung mit Informatiksystemen bietet darin der Bereich Kommunikation in Rechnernetzen mit Diensten sowie Informationssuche im Internet Ansatzpunkte. Gleiches gilt für den Bereich der Konzeption von Rechenanlagen, in dem Aufbau und Funktionsweise von Rechnern und Netzen sowie Konzeption und Planung von lokalen

Netzwerken mit Zugriffs- und Sicherungsstrategien thematisiert werden (Hubwieser 2007a, S. 103). In den Allgemeinbildungsmodulen der Jahrgangsstufen 10-11 werden neben Grundlagen zu Datenbanksystemen auch das Client-Server-Prinzip und Datenflussdiagramme von Rechenanlagen angefertigt. Ein Automatenmodell des Von-Neumann-Rechners kann Variablen und Verarbeitungsvorschriften durch Zustände von Speicherzellen thematisieren (Hubwieser 2007a, S. 105). In der Oberstufe sind neben der Softwareentwicklung Vertiefungen zu Maschinenmodellen aus der Theoretischen Informatik (→ Inhalt 2: Strukturmodelle) sowie Fragen der Parallelverarbeitung mögliche Themen (Hubwieser 2007a, S. 106f).

Durch Unterrichtsbesuche im Rahmen eines Schulversuchs wird das Erreichen des Bildungsziels, d. h. die Unterstützung einer fachgerechten Sicht auf Informatiksysteme durch objektorientiertes Modellieren (OOM), bestätigt. In der externen Evaluation des Ansatzes wurden Unterrichtsbeobachtungen und auswertende Gespräche mit den Fachlehrern an Gymnasien durchgeführt:

> „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? [...] 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, S. 213f).

Viele der aktuellen Bayerischen Schulbücher für die Sekundarstufe I sind der Informationszentrierung nach Hubwieser zuzurechnen. Dementsprechend wird z. B. bei den Themen Standardsoftware sowie Ordner- und Hypertextstrukturen in einem ersten Analyseschritt der Schüler eine objektorientierte Darstellung der Sachverhalte vorbereitet, z. B. Baumdarstellung der Ordner. Anschließend werden die Darstellungen in ein objektorientiertes Modell überführt (vgl. Frey (2003), Breier und Hubwieser (2002), Voß (2006); Abschnitt 4.2.4). Systemanalyse und die Vorwegnahme von Pannen sind für Kompetenzentwicklung mit Informatiksystemen sehr viel versprechende Ansätze. Die positive Beantwortung obiger Evaluationsfrage liefert eine Begründung für die Integration von OOM in das Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung. Bei der Fehlervorwegnahme sollte jedoch die im informationszentrierten Ansatz kaum explizit benannte Hypothesenbildung der Schüler stärker im Vordergrund stehen, um Versuch-Irrtum-Strategien in unbekannten Situationen zu vermeiden (→ Methodik 4: Analyse des Systems – Experimente).

Dekonstruktion von Informatiksystemen als fachdidaktischer Ansatz

Auf der INFOS 1999 (Schwill 1999) stellen Magenheim, Schulte und Hampel den didaktischen Ansatz der Dekonstruktion von Informatiksystemen vor (Hampel et al. 1999). Sie betonen insbesondere sozio-technische Aspekte von Informatiksystemen (→ Inhalt 4: Sichten):

> „Informatiksysteme beinhalten soft- und hardwaretechnische Komponenten, die ihrerseits fundamentale Methoden und Ideen der Informatik und zugleich in digitaler Form materialisierte Modelle eines Realitätsauschnitts repräsentieren. Diese Modelle werden in Modellierungsprozessen entwickelt und bilden ein wesentliches Element der Systemgestaltung [...]. Die soziale Interaktion im Kontext eines existierenden oder zu konstruierenden technischen Informatiksystems erweitert dieses zu einem sozio-technischen System" (Hampel et al. 1999, S. 152).

Die Schwerpunktverlagerung auf die Interaktion wird u. a. mit Wegners Überlegungen „Why Interaction Is More Powerful Than Algorithms" begründet (Wegner 1997). Wegner weist darauf hin, dass Algorithmen als formalisierbare Dimension von Informatiksystemen nicht ausreichen, um Prozesse in Rechnern, parallele Prozesse oder verteilte Systeme in ihrer ganzen Komplexität zu erfassen. Menschen und Maschinen machen Eingaben in Informatiksysteme und beeinflussen damit die Systemzustände und den auf Protokollen basierenden Datenaustausch. Das Systemverhalten ist damit im Detail kaum noch deterministisch nachvollziehbar.

Dekonstruktion ist ursprünglich eine Vorgehensweise zur Beurteilung von Texten. Der informatikdidaktische Ansatz der Dekonstruktion von Informatiksystemen basiert auf der Verschmelzung von Analyse und (Re-) Konstruktion bestehender, hinreichend komplexer Systemen durch Schüler. Im Unterricht wurde bisher nur die objektorientierte Modellierung für die Dekonstruktion eingesetzt. Die Schüler sollen Einsicht in wichtige Methoden der Gestaltung und Bewertung von Informatiksystemen bekommen. Programmierung steht somit nicht mehr im Mittelpunkt eines Konstruktionsprozesses, sondern am Ende eines Modellierungs-, Formalisierungs- und Abstraktionsprozesses bei der Analyse eines speziellen Softwaresystems:

> „Dekonstruktion ist [...] mehr als Lesen von Quellcode einer Software, zum Zwecke des Erlernens der Syntax einer Programmiersprache anhand eines Beispiels. Modellannahmen und Entwurfs- und Designentscheidungen können hypothetisch extrahiert, nicht aber eindeutig belegt werden. Damit werden aber Spielräume für Gestaltungsalternativen offengelegt" ((Magenheim 2001, S. 6) zitiert nach (Engbring 2004, S. 181)).

Die von Magenheim genannten Hypothesen zu den getroffenen Entwurfs- und Designentscheidungen sind Grundlage der Dekonstruktion und fordern von den Schülern ein Verständnis für den Aufbau des Informatiksystems. Gleichzeitig sind die Entwurfsentscheidungen nicht mehr eindeutig zurück zu verfolgen und mindern den Erfolg des didaktischen Ansatzes der Dekonstruktion.

Schulte und Block (2002) geben sieben Schritte zur Dekonstruktion eines Informatiksystems an: In der ersten Phase, der Analyse der Software, werden die Perspektiven des Anwenders (1), Designers (2) und Testers (3) auf die Software genutzt. In der nächsten Phase, der Erweiterung als Synthese (4), soll mit diesem Wissen das System erweitert, verändert oder das neue Wissen auf ein unbekanntes soziotechnisches Informatiksystem transferiert werden. Zuletzt werden während einer

Bewertung und Reflexion die Fehlerfreiheit (5), Wartbarkeit (6) und Brauchbarkeit (7) durch die Lernenden betrachtet. Aus den Rollen wiederum lassen sich Anforderungssituationen zur Formulierung von Basiskompetenzen zu Informatiksystemen herleiten. Darüber hinaus fällt auf, dass die drei Perspektiven Anwender, Designer und Tester fast deckungsgleich zu den Untersuchungsebenen von Informatiksystemen sind: nach außen sichtbares Verhalten (Anwender) und innere Struktur (Designer). Der Tester verbindet die Sicht auf das Verhalten und auf Implementierungsdetails (→ Inhalt 4: Sichten; → Methodik 7: Von der Anwendung zur Maschine). Beim ersten Schritt des Schemas wird die Anwenderperspektive eingenommen, um durch entdecken und erkunden Anwendungsfälle herauszuarbeiten. Darüber hinaus sollen die Schüler

> „versuchen, anhand dieses Erkundungsvorgangs auf die verwendeten Fachklassen und deren Beziehungen zu schließen (dies geht allerdings über die Anwender-Perspektive hinaus und ragt in die Designer-Perspektive hinein)" (Schulte und Block 2002, S. 7).

Das Zitat macht deutlich, dass die Fixierung auf die drei Perspektiven hinderlich sein kann, da durch die Personifizierung Überschneidungen der Perspektiven vorkommen. In der vorliegenden Arbeit wird speziell Kompetenzentwicklung mit Informatiksystemen für Anwender angestrebt, die auch ein Grundverständnis von Entwürfen und Systemtests haben müssen, aber weit von der Perspektive eines Designers oder Testers entfernt sind. Daher wird im Folgenden weiterhin die Unterteilung nach innerer Struktur, nach außen sichtbarem Verhalten und ausgewählten Implementierungsaspekten beibehalten, um keine Missverständnisse hinsichtlich der Zielgruppe zu unterstützen. Dennoch sind Rollen von Menschen gegenüber Informatiksystemen (vgl. auch Hubwieser 2007a) hinsichtlich ihrer notwendigen Kenntnisse in Beziehung zu setzen, z. B. können sie anhand ihrer Haupttätigkeiten den Ebenen des Informatikturms nach Nievergelt zugeordnet werden (siehe Seite 42). Insgesamt lassen sich aus den Rollen Anforderungssituationen ableiten, die zur Kompetenzbeschreibung beitragen können. Zusätzlich ist für die Rollen der Anwendungskontext näher zu bestimmen (Abschnitt 5.3).

Zentral für die Dekonstruktion ist die Produkt-Prozess-Relation, also die Verbindung des sozio-technischen Informatiksystems mit dem dazugehörigen Entwicklungsprozess, der von den Lernenden in der Dekonstruktion nachvollzogen und in Teilen umgesetzt wird (Magenheim und Schulte 2006). Die von Magenheim und Schulte betonte Produkt-Softwareentwicklungsprozess-Relation hingegen entspricht nicht den in der vorliegenden Arbeit angestrebten Basiskompetenzen zu Informatiksystemen. Vielmehr bietet es sich an, eine Produkt-Anwendungsprozess-Relation zu postulieren, die für Kompetenzentwicklung mit Informatiksystemen anzustreben ist, um den für Kompetenzbeschreibungen geforderten Anforderungssituationen gerecht zu werden (→ Methodik 3: Verbindung von Verhalten und Struktur).

Sie schlagen des Weiteren didaktische Linsen im Sinne von Sichten auf Informatiksysteme vor, die technische und soziale Aspekte kombinieren (→ Inhalt 4: Sichten): Automatisierung (Automation), Interaktion (Interaction), Informationsverarbeitung (Information processing), Vernetzung (Networking), Normen und Recht (Norms, regulation and law), Gesellschaft und Ethik (Societal and ethical aspects) (Magenheim und Schulte 2006, S. 328). Dafür berufen sie sich u. a. auf Denning (2003), aber es wird eine deutliche Schwerpunktverlagerung auf sozio-technische Aspekte vorgenommen. In der vorliegenden Arbeit können die didaktischen Linsen als sinnvolle Ergänzung der fachlich bestimmten Hauptfunktionen nach Denning (2007) verwendet werden. Die Auswirkungen des Einsatzes von konkreten Informatiksystemen auf die Gesellschaft werden in der vorliegenden Arbeit in Abschnitt 5.5 im Rahmen einer systematischen Erkundung des Systemverhaltens aufgegriffen.

Zur Analyse eines sozio-technischen Informatiksystems ist sein Kontext relevant und neben Abstraktion, Reduktion und Formalisierung wird insbesondere die Dekontextualisierung im didaktischen Ansatz der Dekonstruktion betont:

> „Contextualising learning with application areas supports claims from situated learning and cognitive flexibility" (Magenheim und Schulte 2006, S. 325).

Magenheim und Schulte fordern, typische Anwendungen zur Kontextualisierung zu nutzen. Übertragen auf Informatiksysteme und Kompetenzentwicklung können die Anwendungen genutzt werden, um Anforderungssituationen zu beschreiben. Tabelle 4.1 zeigt die zeitabhängige Auswahl. Kriterien für das Ausschließen anderer Bereiche werden nicht genannt.

Der Einsatz von Class-Responsibility-Collaborator-Karten (CRC-Karten) und die Prüfung von Anwendungsfällen im Rollenspiel werden als Lehr-Lernmethoden vorgeschlagen (→ Methodik 2: Kognitive Modelle). Übertragen auf Informatiksysteme und Kompetenzentwicklung sind Rollenspiele zu Anwendungsfällen eines Möglichkeit, das Systemverhalten systematisch zu analysieren (→ Methodik 7: Von der Anwendung zur Maschine). Für Kompetenzentwicklung mit Informatiksystemen folgt daraus, dass sich aktives Verhalten von Systemkomponenten oder gekapselten Subsystemen durch Interaktion von Lernenden gut nachempfinden lässt. Ähnlich wie bei der Objektorientierung hat jede Systemkomponente Verhalten und Zustand. Statische Modelle von Informatiksystemen geben die teilnehmenden Rollen bzw. Personen an, und dynamische Modelle wie Anwendungsfälle oder Sequenzdiagramme liefern das Drehbuch des Rollenspiels. Insbesondere Prozessbeziehungen eignen sich für Rollenspiele (Schubert und Schwill 2004, S. 272). Darüber hinaus eignen sich Rollenspiele zur Simulation von Gesprächen zwischen Kunden und Entwicklern.

Zur Gestaltung von Unterricht zu sozio-technischen Informatiksystemen beschreiben Magenheim und Schulte einen fachdidaktischen Rahmen (Tabelle 4.1) und

gestalten Beispiele zur Computerspielentwicklung und Wikis im Informatikunterricht. Vorteil, aber auch Schwierigkeit des Ansatzes ist, dass er allzu umfassend ist. Begrifflich fällt eine Problematik auf: Dekonstruktion wird einerseits als Unterrichtsmethode aufgeführt (vgl. Hampel et al. 1999), sie wird aber auch als umfassende Vorgehensweise gesehen, die Konstruktion umfasst und andere Methoden wie Rollenspiele nutzt (vgl. Schulte und Block 2002). Die eingesetzten Medien dienen u. a. der Visualisierung (→ Medien 1: Visualisierung verborgener Prozesse).

Tabelle 4.1: Informatiksysteme im Unterricht nach (Magenheim und Schulte 2006, S. 332)

Was?		Wie?	
Anwendungsgebiet	Didaktische Linsen	Medien	Methoden
Produktion	Automatisierung	Videos	Rollenspiele
Logistik	Interaktion	Software	Konflikte identifizieren
Bildung	Informationsverarbeitung	Visualisierungen	Analysieren
Gesundheitswesen	Vernetzung	Dokumentation	Dekonstruktion
Unterhaltung	Normen	Quelltext	Konstruktion
Forschung	Gesellschaftliche Fragen	...	...
Militär			
E-Demokratie			

Eine Gegenüberstellung von Informationszentrierung nach Hubwieser und Dekonstruktion nach Magenheim nimmt Engbring (2004) vor. Er stellt fest, dass der Kontext in der Dekonstruktion überbetont wird. Engbring fordert eine stärkere Unterscheidung zwischen technischer und nichttechnischer Ebene, um nicht politische oder sozialwissenschaftliche Bildung zu betreiben und durch die klare Trennung ggf. Kooperationen mit anderen Fächern zu ermöglichen, die der Aufgabenteilung bedarf (Engbring 2004, S. 195). Der Bildungserfolg durch die Umsetzung der Dekonstruktion ist abhängig von der Verfügbarkeit angemessen komplexer soziotechnischer Informatiksysteme als Gegenstand. Schwierigkeit ist in verstärktem Maße die schon der Objektorientierung zugesprochene Diskrepanz zwischen komplexen, sinnvoll erweiterbaren Systemen einerseits und einem für Schüler angemessenen Komplexitätsgrad andererseits. Dies impliziert nach Magenheim, dass professionelle Entwicklungsumgebungen im Sinne von „Cognitive Tools" weit reichende Berücksichtigung erfahren (Magenheim 2003). Die Arbeit mit Cognitive Tools bezeichnet in der Schule den Einsatz von (Standard-) Software, die weitgehend nicht

an Fachinhalte gebunden und deshalb an spezielle Problemstellungen zur Wissensverarbeitung und -weitergabe anzupassen ist (vgl. Eberle 1996). Dazu kommt für die Lehrperson ein nicht zu vernachlässigender Aufwand bei der Gestaltung eines geeigneten Informatiksystems, das dekonstruiert wird. Technische Aspekte als wesentlicher Teil der Wirkprinzipien von Informatiksystemen werden bei der Dekonstruktion nicht betrachtet (Humbert 2003, S. 69). So wird zwar bei Informatiksystemen die Einheit aus Hard- und Software betont, aber beispielsweise werden bei der Beschreibung der Produkt-Prozess-Relation nur bei der Sicht auf das Produkt kurz Sensoren und Aktoren als physikalische Komponenten genannt, ohne daraus Konsequenzen zu ziehen (Magenheim und Schulte 2006, S. 322f). Im Entwicklungsprozess werden ausschließlich Fragestellungen der Softwareentwicklung betrachtet. Die de facto Einschränkung der Dekonstruktion auf objektorientierte Softwareentwicklung zeigt die Forschungslücke, dass auch eine strukturierte Beschreibung der technischen Aspekte des Von-Neumann-Rechners ebenso wie Netzverbindungen zu thematisieren ist (→ Inhalt 2: Strukturmodelle).

Mit den Ansätzen der Informationszentrierung und der Dekonstruktion werden auf informatischer Modellierung beruhende Alternativen für die im Informatikunterricht vorherrschende Implementierung vorgeschlagen. Dafür wird die Verknüpfung von informatischem Modellieren mit der Analyse existierender Informatiksysteme vorgenommen. Die Dekonstruktion strebt dabei das Ziel an, anhand des vollständigen Informatiksystems mittels Hypothesenbildung und -prüfung Entwurfsentscheidungen nachzuvollziehen. Der bislang ausbleibende Erfolg des ambitionierten didaktischen Ansatzes ist sicherlich in erster Linie darauf zurückzuführen, dass aus dem Produkt nicht mehr eindeutig Rückschlüsse auf Entwurfsentscheidungen gezogen werden können: Es lassen sich viele Alternativlösungen finden. Arbeiten wie die von Schulte (2004) und Ullenboom (2005) liefern mittlerweile didaktisch vorbereitete Materialen für den Einsatz im Informatikunterricht. Für Informatiksysteme und Kompetenzentwicklung bietet der Ansatz der Dekonstruktion besonders hinsichtlich der Produktebene, der Anwendungsbereiche sowie Hypothesenbildung und -prüfung Potential (→ Methodik 4: Analyse des Systems – Experimente).

In einer Diplomarbeit erstellt Ullenboom eine Software „Media Player", die Entwurfsmuster nach Gamma et al. (1995) für die Ausbildung aufgreift. Zielgruppe sind Schüler eines Leistungskurses Informatik in der Sekundarstufe II und Studierende der Informatik (Ullenboom 2005, S. 21). Kritisch zu sehen ist die Vermengung von Schülern und Studierenden als Zielgruppe. Lernende sollen in der Dekonstruktionsphase über den Quellcode, die UML-Diagramme und Dokumentation des „Application Programming Interface" (API) die Entwurfsmuster entdecken. Anschließend wird eine Lösung erarbeitet, mit der Umsetzung in der Software verglichen und reflektiert. Eine empirische Überprüfung der Software in der Lehre fand nicht statt (Ullenboom 2005, S. 94). Die Entwicklung von Unterrichtsmaterialien zu Entwurfsmustern konkretisiert den Ansatz der Dekonstruktion. Durch die Zielgruppe

des Leistungskurses Informatik, die starke Prägung durch die Hochschulinformatik und fehlende empirische Erprobung bleibt er jedoch für Schule als ambitioniert zu bezeichnen.

Schulte (2008) nutzt die didaktischen Linsen in Verbindung mit einer Gegenüberstellung von Funktion und Struktur zur Analyse und Rekonstruktion von Software. Am Beispiel der Textverarbeitung skizziert er die Rekonstruktion der Dualität von Struktur und Funktion durch ein spiralförmiges Anwenden aller didaktischen Linsen. Diese Dualität ist konform zur Schwerpunktsetzung auf dem nach außen sichtbaren Verhalten und der inneren Struktur von Informatiksystemen, die der Autor in seinem Unterrichtsmodell (Kapitel 5; (Stechert 2007c)) erprobt und daraufhin verfeinert hat (Kapitel 7; (Stechert und Schubert 2007)).

Kompetenzentwicklung mit Informatiksystemen im ideenorientierten Informatikunterricht

Nach Schubert und Schwill hat das Bildungziel der Entmystifizierung von Rechnern nicht an Aktualität eingebüsst (Schubert und Schwill 2004, S. 253). Der relativ neue Terminus Informatiksystem bringt ihrer Meinung nach jedoch mehr Klarheit bezüglich dessen, was die informatische Bildung leisten muss, als die Begriffe Rechner oder Computer:

> „Es geht nicht um das Inbetriebnehmen von Geräten, wie Fernseher oder das Programmieren von Geräten wie DVD-Player oder das Steuern von Maschinen wie Autos. Es geht um die Organisation der eigenen geistigen Tätigkeiten, die Schüler erlernen müssen" (Schubert und Schwill 2004, S. 253).

Damit entkräften Sie Argumente zur prinzipiellen Unbeherrschbarkeit von Informatiksystemen wie Brunnstein sie vorbringt hinsichtlich ihrer Konsequenzen für die Schulinformatik (Abschnitt 4.2.4). Schubert und Schwill beziehen sich auf die Definition des Informatikdudens (Claus und Schwill 2006, S. 314) und betonen, dass Informatiksysteme aus Schichten bestehen, die alles Geheimnisvolle ausschließen (Schubert und Schwill 2004, S. 254). Fehler jedoch, die darauf zurückzuführen sind, dass eine Anwendungssituation bei der Entwicklung des Informatiksystems nicht einkalkuliert wurde, verleihen dem Informatiksystem etwas Bedrohliches:

> „Schwere Fehler treten scheinbar zufällig auf. In Wirklichkeit sind sie an seltene Konstellationen im System gebunden, dadurch kaum analysierbar und entsprechend schwer zu beheben. Genau diese Alltagserfahrung verleiht Rechnern in den Augen von Schülern etwas Mystisches (Rechenberg, 1994)" (Schubert und Schwill 2004, S. 254).

Es ist gerade das Prinzip der freien Programmierbarkeit, das diese Fehlerzustände ermöglicht bzw. befördert (→ Methodik 1: Universalität). Denn es wird versucht, die Welt so genau wie möglich im Informatiksystem zu beschreiben, dabei aber vernachlässigt, dass Informatiksysteme nur einen Realitätsausschnitt abbilden können

und somit ein Modell darstellen (vgl. Schubert und Schwill 2004, S. 255). Schubert und Schwill kritisieren, dass Informatikunterricht es zurzeit oft versäume, die technische Basis von Rechnern zu thematisieren. So treten bestimmte Modellierungsfehler immer wieder auf, wenn die Schüler keine Vorstellung von Speicherung und Datenaustausch im Rechner haben (Schubert und Schwill 2004, S. 200). Als Unterrichtsleitlinien nennen Schubert und Schwill die drei Bereiche Pläne, Sprachen und Systeme (Schubert und Schwill 2004, S. 45ff). In dem Bereich „Systeme" sind dementsprechend der technische Aufbau von Informatiksystemen und die Thematisierung wichtiger Repräsentanten von Systemen verankert. In der Leitlinie „Pläne" geht es um Fragen der Berechenbarkeit, in der Leitlinie „Sprachen" um Interaktion mit Informatiksystemen, z. B. mittels formaler Sprachen.

Zur Gestaltung der Unterrichtsleitlinie „Systeme" empfehlen Schubert und Schwill für Unterricht zu den Wirkprinzipien von Informatiksystemen die Thematisierung des engen Verhältnisses zwischen realer, abstrakter und virtueller Maschine (Schubert und Schwill 2004, S. 269). Als reale Maschine nennen sie beispielhaft den Von-Neumann-Rechner, als abstrakte Maschine den mathematischen Funktionsbegriff und logisches Schließen über Aussagen. Eine virtuelle oder gedachte Maschine zeichnet sich durch Abstraktion von Speicherstrukturen, Operationseinheiten und Kontrollmechanismen zur Lösung eines Problems aus (Claus und Schwill 2006, S. 727). Sie kann anschließend durch Programme auf einem Rechner realisiert werden (→ Inhalt 2: Strukturmodelle).

Für Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe I empfehlen Schubert und Schwill neben den Aspekten reale, abstrakte und virtuelle Maschine dazu komplementär eine Struktur des Lehr-Lernprozesses, die die Analyse eines Von-Neumann-Rechners, Betriebssysteme und typische Anwendungen einschließt:

> „Die Schüler sollen sich im Informatikunterricht ein Grundverständnis zu Systemen aneignen, indem sie schrittweise zuerst den Rechner analysieren (Rechnerarchitektur, Betriebssystem), danach Rechner logisch zu einem Rechnernetz verbinden, auf dem die Ausführung einer typischen Anwendung untersucht wird (z. B. Informationssystem / Datenbanksystem). Wir empfehlen folgende Schwerpunkte:
>
> - Rechnerarchitektur: kognitives Modell zu Aufbau und Arbeitsweise eines Rechners, Von-Neumann-Prinzipien, Schichten-Architektur (z. B. MVC-Architektur (model – view – controller)),
> - Betriebssystem: Prozesse, Betriebsmittel, Schichten-Modell (z. B. Geräteverwaltung mit Unterbrechungsbehandlung, Gerätetreiber, Zuteilung und Freigabe von Geräten, Eingabe-Ausgabe-Bibliotheken),
> - Rechnernetz / verteiltes System: Adressierung, Protokoll, Schichten-Modell (z. B. Schichtenmodell der Internet-Technologie [...]),
> - Informationssystem / Datenbanksystem: Daten, Information, Wissen, Schichten-Modell (z. B. Drei-Ebenen-Konzept eines Datenbankmanagementsystems mit internem, konzeptionellem und externen Schema).

> Dieser Lernprozess sollte in den Jahrgangsstufen 5 bis 8 der Sekundarstufe I schulartspezifisch stattfinden, indem ein Spiralcurriculum von den Anwendungen zu den Wirkprinzipien führt" (Schubert und Schwill 2004, S. 269f).

Methodisch wird damit der Weg von der Anwendung zur Maschine empfohlen (→ Methodik 7: Von der Anwendung zur Maschine). Diese Übersicht skizziert einerseits die Verbindung von Hardware und Software, andererseits die Wichtigkeit von Schichtenmodellen für Kompetenzentwicklung mit Informatiksystemen. Darüber hinaus weisen Schubert und Schwill den zusammenwirkenden Prozessen eine Schlüsselrolle zu:

> „Der Begriff Prozess hilft Anwendern von verteilten Informatiksystemen, deren Wirkprinzipien besser zu verstehen. Prozesse sind Folgen von Ereignissen, denen Aktionen zugeordnet werden, die zur Lösung gehören. Die Ereignisse stehen in kausaler Beziehung. Daraus folgen die möglichen Varianten für eine zeitliche Ordnung, die das Neben- und Nacheinander festlegt" (Schubert und Schwill 2004, S. 270).

Die vielfältigen von Schubert und Schwill vorgestellten Vorgehensweisen für Unterricht zu Wirkprinzipien von Informatiksystemen müssen nun in ein konsistentes Unterrichtsmodell zu Informatiksysteme und Kompetenzentwicklung einfließen und ggf. im konkreten Unterricht auf ihre Tragfähigkeit überprüft werden. Das Prozesskonzept beispielsweise unterstützt direkt die Vernetzung und Verteiltheit heutiger Informatiksysteme. Darüber hinaus ergänzt es explizit die Beschreibung der inneren Struktur um die zu ihrem Verständnis notwendigen Abläufe (Abschnitt 3.2.3). Dabei ist jedoch begrifflich zwischen den Beziehungen und Abläufe zwischen den Strukturkomponenten einerseits, die im allgemeinen Sprachgebrauch ebenfalls als Prozesse bezeichnet werden, und dem informatischen Prozesskonzept andererseits zu unterscheiden. Schlussfolgerung der hier skizzierten Vorgehensweise für Kompetenzentwicklung mit Informatiksystemen ist auch, dass Schichtenmodellen als logische Verknüpfung zwischen Hardware, Software und Vernetzung zur Förderung der Kompetenzentwicklung mit Informatiksystemen dienen können (Abschnitt 9.3). Bei Schichtenmodellen ist jedoch zu bedenken, dass sie in der Informatikpraxis nicht zwangsläufig so trennscharf implementiert werden, wie ihre Spezifikation es vorschreibt. Strukturmodelle werden deshalb für das Unterrichtsmodell analysiert (→ Inhalt 2: Strukturmodelle).

4.2.4 Weitere Entwicklungen zu Informatiksystemen seit den 1990er Jahren

Kompetenzentwicklung mit Informatiksystemen in der Schulinformatik bis zur Jahrtausendwende

Tom van Weert (1993), der an internationalen curricularen Projekten der ATEE und der IFIP mitarbeitete, begründet auf der INFOS 1993 (Troitzsch 1993) Informatik als Teil der Allgemeinbildung und unterscheidet dafür technische Systeme, die eine mechanische Automatisierung des primären Betriebsprozesses, also

des Produktionsprozesses vornehmen. Denen gegenüber sieht er Informationssysteme als Ausgangspunkt der Automatisierung bei den sekundären, administrativen Prozessen. Aus den gesellschaftlichen Veränderungen durch Informationstechnologie, Automatisierung (Unterstützung des Verwaltungsprozesses), Informatisierung (persönliche Qualifizierung) und Kommunikatisierung (persönliche Qualifizierung im Prozess), leitet er gesellschaftliche Forderungen an den Informatikunterricht ab. In ihm soll das Lernen über Automatisierung auf der ersten Stufe in Form einfacher Programmerstellung und Organisation eines Rechenzentrums thematisiert werden. Auf der zweiten Stufe, der Informatisierung, soll Informatik als Anwendung der Informationstechnologie behandelt werden. Auf der dritten Stufe, der Kommunikatisierung, sieht er die Ausprägung eines neuen Alphabetismus und beruft sich dafür auf Ershov (Abschnitt 4.3.1):

> „Ershovs Metapher läßt sich weiterentwickeln: der traditionelle Alphabetismus wird erweitert werden um die Fähigkeit, komplexe dynamische konzeptuelle Modelle in exekutierbaren Symbolen (generalisierten Programmen) auszudrücken [...]. Dies wird durch eine Gesellschaft mit immer komplizierteren, auf Kommunikationsnetzen basierten Organisationsformen erfordert" (Weert 1993, S. 17).

Weert fordert dementsprechend den „Alphabetismus in der Informationstechnologie für alle" (Weert 1993, S. 17). Elemente der Informatik, die dazu für die Allgemeinbildung notwendig sind, umfassen seiner Meinung nach Methoden der Modellierung, der Programmierung und den Entwurf von Datenstrukturen. Als Vertiefung in einem Wahlfach schlägt er Software Engineering und Techniken der Mensch-Maschine-Kommunikation vor. Für Kompetenzentwicklung mit Informatiksystemen ist die Klassifizierung der Informatiksysteme anhand ihres Einsatzes in Produktions- und Verwaltungsprozessen interessant (→ Inhalt 4: Sichten). Insbesondere letztere lassen sich weiter abstufen, z. B. in Unterstützung von Routineprozessen hoher Standardisierung, Regelprozessen mit individuellen Eingriffen und Einzelfallprozessen, deren Abläufe kaum planbar sind (Picot und Rohrbach 1995). Jede Stufe benötigt ein tieferes Verständnis für das Informatiksystem und den entsprechenden Arbeitsprozess.

Die Fachdidaktikforscher Koerber und Peters (1993) diskutieren auf der INFOS 1993 die Stellung des Informatikunterrichts gegenüber der informationstechnischen Grundbildung. Dazu beziehen sie sich unter anderem auf die Nutzung von Informationstechnologie als das Gemeinsame der Aspekte informatischer Bildung, für die sie ein Gesamtkonzept fordern. Darin enthalten sein müssen Reduktion der Realität durch Modellbildung und Abstraktion. Dazu betonen sie, dass komplexes Wissen zur Systemkonstruktion nicht Voraussetzung für kompetente Anwendung von Informatiksystemen sein darf:

> „Aber es ist bisher selten oder fast gar nicht beachtet worden, daß zu einer kompetenten Beurteilung und Anwendung eines informationstechnischen Systems ebenfalls systemanalytische und teststrategische Kompetenzen notwendige Voraussetzungen

> sind. [...] es kann nicht davon ausgegangen werden, daß zuerst das komplexe analytische Wissen um alle Probleme der Konstruktion und Anwendung vermittelt worden sein muß, um irgendwann diese Kompetenzen nutzen zu können" (Koerber und Peters 1993, S. 111f).

Diese Erkenntnis ist konform zu der in der vorliegenden Arbeit getroffenen Entscheidung, Softwareentwicklung nur am Rande zu thematisieren. Koerber und Peters (1993) schlussfolgern, dass im Unterricht Informatiksysteme, wie komplexe Anwendungen oder vorgegebene Programme, auf ihre Funktionalität analysiert werden müssen. Dabei fordern sie als Lehr-Lernmethodik die Thematisierung informationstechnischer Systeme aus dem historischen Kontext heraus (→ Methodik 7: Von der Anwendung zur Maschine), also ausgehend von dem früher von Menschen durchgeführten und nun von der Maschine übernommenen Handlungsablauf (Koerber und Peters 1993, S. 112).

Die GI legt 1993 „Empfehlungen für das Fach Informatik in der Sekundarstufe II allgemeinbildender Schulen" vor (Schulz-Zander et al. 1993). Es werden drei leitende Sichtweisen genannt: A) Mensch-Computer, B) Formalisierung und Automatisierung geistiger Arbeit sowie C) Informatiksysteme, Gesellschaft und Umwelt. Gegenüber der GI-Empfehlung von 1976 wird auf veränderte Bedingungen und Sichtweisen hingewiesen:

> „Informatik verlangt neben reinen mathematisch-formal geprägten Methoden und algorithmischen Sichtweisen weitere Bearbeitungsperspektiven. [...] Dies setzt ein Behandeln von Problemlösungs- und Gestaltungsmethoden sowie deren kritische Beurteilung, eine Förderung des Denkens in Abläufen und Zusammenhängen, eine Förderung der Kommunikations- und Kooperationsfähigkeit und eine Darstellung der Probleme und Methoden, komplexe Systeme zu überschauen, im Informatikunterricht voraus" (Schulz-Zander et al. 1993, S. 206).

Damit sind für Kompetenzentwicklung mit Informatiksystemen die Themen „Arbeiten mit komplexen Systemen" und „Vernetzte Systeme" neu gewichtetet oder aufgenommen (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise; → Ziel 2: Arbeiten mit komplexen Systemen). Relevant für Informatiksysteme und Kompetenzentwicklung ist, dass eine verstärkte Umorientierung des Informatikunterrichts zu den lebenspraktisch-orientierten Ansätzen inklusive gesellschafts- und arbeitsweltlichen Abhängigkeiten stattfindet (→ Inhalt 4: Sichten), wenngleich nicht von einer Abkehr von der Algorithmik gesprochen werden kann.

Friedrich (1995b) fordert, im Informatikunterricht Aufbau, Funktion und Wirkungsweise der maschinellen Informationsverarbeitung und Problemlösungsstrategien mit Werkzeugen der Informatik zu behandeln (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise). Er stellt auf der INFOS 1995 (Schubert 1995) Thesen für die Informatikdidaktik vor, die er aus der informationstechnischen Grundbildung ableitet, die den schnellen technischen Entwicklungen nicht gewachsen ist (vgl. Brauer 1990):

> „Ein ernsthafter Fachunterricht in der Informatik sollte ein solideres Fundament für den vernünftigen Umgang mit Informatiksystemen legen sowie ein wesentlich besseres Verständnis der Anwendungen der Informatik [...] ermöglichen. Dieser Anspruch wurde nicht nur aus den Augen verloren, sondern in manchen Grundpositionen zu Zielen informatischer Bildung ungenügend berücksichtigt" (Friedrich 1995b, S. 34).

Speziell fordert er, über Benutzungsfertigkeiten einerseits und spezielle Programmiersprachenkenntnisse andererseits hinauszugehen:

> „Also steht mehr die Frage, die Grundprinzipien der Informatik zu beschreiben, die unabhängig von aktuellen Entwicklungen im Bereich der Hard- und Software Allgemeingut für den Absolventen der jeweiligen Schulart sein müßten. Das sind z. B.:
> - Grundprinzipien des Aufbaus, des Funktionierens und der Wirkung von Maschinen zur Verarbeitung von Informationen;
> - Strategien der Problemlösung mit Werkzeugen der Informatik, Betrachtung zu deren Möglichkeiten und Grenzen;" (Friedrich 1995b, S. 38).

In der didaktischen Untersetzung dieser fachwissenschaftlichen Bildungsinhalte und -ziele (→ Ziel 3: Bewusste Anwendung), einschließlich frühzeitiger Begriffsbildung bei Schülern, sieht Friedrich die Aufgabe der Fachdidaktik. Wie eine entsprechende Modellbildung umgesetzt werden soll, bleibt jedoch offen. Dies gilt ebenso für Kompetenzentwicklung mit Informatiksystemen.

Engbring (1995) begründet kultur- und technikgeschichtlich, dass Fähigkeiten und Fertigkeiten im Umgang mit Rechnern im Kontext menschlichen Handelns, Denkens und Lernens in der Allgemeinbildung zu vermitteln sind:

> „Betrachtet man also [...] den Umgang mit Computern in Zusammenhang mit geistigen Tätigkeiten und den dazu erforderlichen Techniken (technikgenetische Sichtweise), ist dieses Herangehen geeignet, *kulturelle Kohärenz* für Computernutzung und Informatik zu stiften. [...] ‚Wissenschaftliche' Inhalte der Informatik sind dann nur Mittel zum Zweck, diese Zusammenhänge zu verstehen" (Engbring 1995, S. 75f; Hervorh. im Original).

Ziel der technikgenetischen Sichtweise ist damit Entmystifizierung technischer Phänomene (→ Ziel 4: Entmystifizierung), wobei „Technik" sowohl Fertigkeiten und Kulturtechniken meint als auch Erklärung, Einschätzung, Beurteilung und Bewertung von Technik, speziell von Rechnern (Engbring 1995, S. 76). Dazu fordert er unterrichtspraktische Überlegungen und Konsequenzen für die Lehrerbildung. Engbring (1995) warnt nachdrücklich vor einer Festlegung auf eine der Metaphern „Werkzeug" oder „Medium", da beide zu kurz greifen. Solche Hilfsmittel müssten, wie Engbring mit Bezug zu Keil-Slawik (1992) feststellt, kultur- und technikgeschichtlich betrachtet werden, um Gestaltungsrichtlinien und -kriterien für interaktive Systeme abzuleiten. Auf diese Weise gewonnene Informatikunterrichtsinhalte seien sowohl allgemein bildend als auch den Zielbereichen Algorithmik und Programmierung, Informatik und Gesellschaft und Computerkunde zuzuordnen (Engbring 1995, S. 75). Computerkunde umfasst nach Engbring den Umgang mit An-

wendungen und Grundprinzipien über den Aufbau von Hard- und Software (Engbring 1995, S. 70). Ergänzend muss jedoch erwähnt werden, dass die Werkzeug-Metapher auch zu einer Unterschätzung der Universalität des Rechners führt (Schubert und Schwill 2004, S. 229). Die technikgenetische Sichtweise (vgl. Abschnitt 5.2.2) kann für das Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung dahingehend interpretiert werden, dass nicht nur unter Laborbedingungen verschiedene typische Repräsentanten von Informatiksystemen integriert werden sollen, sondern vielmehr universelle Informatiksysteme in einem ansprechenden Kontext unter Thematisierung menschlichen Denkens und Handelns einzubeziehen sind (→ Methodik 1: Universalität).

Friedrich, Schubert und Schwill skizzieren 1996 den Wandel des Schulfachs Informatik, der bei den fachdidaktischen Gesprächen zur Informatik in Königstein diagnostiziert wurde und für den folgende Ziele des Informatikunterrichts formuliert wurden:

> „Zu den auszuprägenden Qualifikationen gehören insbesondere:
> - Wissen um die allgemeinen Wirkprinzipien von Informatiksystemen und deren grundlegender technischer Realisierung.
> - Fähigkeiten zur Auswahl und der problembezogenen selbständigen Nutzung geeigneter Systeme sowie deren Anpassung.
> - Haltungen und Einstellungen zu Auswirkungen der Nutzung von Informatiksystemen, einschließlich deren Möglichkeiten und Grenzen" (Friedrich et al. 1996, S. 29).

Für die Fachdidaktik folgt daraus, dass dahingehend Fragen zu beantworten sind zu Gegenständen, Denkweisen und Methoden, zur Stellung eines Unterrichtsgegenstands im Fach und in der Alltagswelt, zu den Zugängen, zur Begriffsbildung auf unterschiedlichen Schulstufen, zu Lernschwierigkeiten und zur Umsetzung in Lerneinheiten (Friedrich et al. 1996, S. 31). Darüber hinaus wird eine Strukturierung des Schulfachs aus Sicht der Fachwissenschaft vorgeschlagen, die die wesentlichen Grundbegriffe, Prinzipien, Werkzeuge und Methoden umfasst und durch Überprüfung auf fundamentale Ideen der Informatik hinsichtlich ihrer Relevanz gesichert wird. Da dies jedoch nicht ausreiche, ist die Strukturierung aus Sicht der Fachwissenschaft hinsichtlich wissenschaftstheoretischer Sichtweisen und Fragen der Allgemeinbildung zu ergänzen (vgl. Friedrich et al. 1996, S. 32). Für Kompetenzentwicklung mit Informatiksystemen sind bewusste Anwendung, Strukturmodelle und Bereitschaften zur Anwendung zu betrachten (→ Ziel 3: Bewusste Anwendung; → Ziel 5: Bereitschaften; → Inhalt 2: Strukturmodelle)
Schubert und Schwill führen als Beispiel die Telekommunikation an, die von einer Arbeitsgruppe in Königstein auf ihren fachlichen Gehalt geprüft wurde. Dabei identifizieren sie hinter dem Teilbereich des elektronischen Bezahlens im World Wide Web (WWW) eine Vielzahl fundamentaler Ideen, jedoch ohne Nachweis der Kriterien: Im Bereich der Algorithmisierung sind es Verschlüsselungsverfahren, Protokollverfahren, Routing-Verfahren, Nebenläufigkeit, Fairness, Konsistenz, Authen-

tifizierung und Suchen. Bei der strukturierten Zerlegung finden sich Netzwerktopologie, Protokollhierarchie (Schichtenmodell der „International Organization for Standardization" (ISO); auch: „Open Systems Interconnection Basic Reference Model" (OSI-Referenzmodell)), Transportstrategien und verteiltes System. Im Bereich Sprache sind es Syntax, Dokumentendarstellung und Übersetzung. Ausgangspunkt der Analyse war eine Anwendungssituation, in der ein Warnhinweis zur unsicheren Datenübertragung auftrat. Für Kompetenzentwicklung mit Informatiksystemen ist dieses Vorgehen sehr hilfreich, da es aus der Anwendungssituation heraus fundamentale Ideen zur Begründung des Bildungswertes liefert (→ Inhalt 3: Fundamentale Ideen). Es wird jedoch auch festgestellt, dass durch die fundamentalen Ideen fast ausschließlich formalisierbare Aspekte der Kerninformatik gefiltert werden. Allgemeinbildungsaspekte bleiben, wie angedeutet, außen vor (Friedrich et al. 1996, S. 33). Außerdem ist für Kompetenzentwicklung mit Informatiksystemen der Bezug zu Informatiksystemen und den Sichten auf sie herzustellen (Abschnitt 3.2.2).

Vernetze Informatiksysteme als Unterrichtsgegenstand und Medium setzen Witten und Penon in einem Halbjahreskurs „Telekommunikation" ein (Witten und Penon 1997). Dabei wird Projektarbeit angewendet, Modellieren von Informationen mit der Hypertext Markup Language (HTML) als Sprache wird anhand von Anforderungen an gut gestaltete Webseiten thematisiert und Fragen der gesellschaftlichen Auswirkungen werden integriert. Sie merken kritisch an, dass durch den Pilotcharakter ihres Kurses einige informatische Grundlagen der Telekommunikation wie Schichtenmodelle, Protokolle und Netzarchitekturen im Sinne der fundamentalen Idee der strukturierten Zerlegung zu kurz kamen (→ Inhalt 2: Strukturmodelle).

Neupert und Friedrich (1997) betonen, dass Informatikunterricht immer wieder von Medienerziehung abzugrenzen ist, in der Informatiksysteme als Werkzeuge im Sinne einer Black-Box gelehrt werden, denen der Anwender letztlich ausgeliefert ist. Sie stellen die Frage, ob Schüler informatische Bildung benötigen und belegen die positive Antwort anhand von Fehlermeldungen bei Nutzung des Internets, die ohne entsprechende Kompetenz die bewusste und effiziente Anwendung verhindern. Als Themen bezüglich der Arbeit mit Netzen nennen sie für die Sekundarstufen im Bereich Struktur von Rechnernetzen die Topologien von Netzen, Schichtenmodelle und Netzdienste sowie exemplarische Kommunikationssysteme (→ Inhalt 2: Strukturmodelle). Interessant für Kompetenzentwicklung mit Informatiksystemen ist an der Aufteilung von Neupert und Friedrich, dass der Struktur von Rechnernetzen ein eigener Bereich zu kommt. Aber sie nennen weitere Bereiche wie die Wahl der Informationsdarstellung, der Datensicherheit und gesellschaftliche Entwicklungstendenzen, die im Kontext der Netze spezifisch sind. Letztere sind nicht getrennt in Verhalten und innere Struktur von Informatiksystemen. Als Fazit ist zu ziehen, dass erstens die Werkzeugsicht im Sinne der Black-Box nicht gleich zu setzen ist mit dem Verstehen des Verhaltens eines Informatiksystems, denn Kompetenzentwicklung mit Informatiksystemen bedarf des Wissens um deren Grundprinzipien,

die nur eine informatische Bildung leistet (→ Methodik 3: Verbindung von Verhalten und Struktur). Zweitens liefern gerade die Fehlersituationen Ansatzpunkte für allgemein bildenden Unterricht zur Förderung der Kompetenzentwicklung mit Informatiksystemen (→ Ziel 4: Entmystifizierung).

Auf der INFOS 1997 (Hoppe und Luther 1997) betrachtet Krämer Rechner aus der philosophisch-medienwissenschaftliche Perspektive unter Werkzeug-, Denkzeug- (vgl. Koerber 1978) und Spielzeugaspekt (→ Inhalt 4: Sichten). Den Rechner charakterisiert sie in Analogie zum Menschen:

> „Mensch und technisches Instrument weisen – zumindest funktionell – gleichwertige Eigenschaften auf; die Technik wird zur Erweiterung unseres Körpers und seiner Vermögen: Sie gilt als prothetische Verstärkung, Entlastung unserer motorischen, sensorischen und kognitiven Organe" (Krämer 1997, S. 7).

Sie betont dabei den apparativen Vollzug von Kulturtechniken (vgl. Nake 1998). Auf den Vernetzungsaspekt von Medien ging Krämer in einem Vortrag verstärkt ein, den sie 2006 im Forschungskolleg 615 „Medienumbrüche" der Deutschen Forschungsgemeinschaft (DFG) hielt, an dem der Autor im Teilprojekt A8 „Informatikunterricht und E-Learning zur aktiven Mitwirkung am digitalen Medienumbruch" mitforscht (vgl. (Schubert et al. 2005a), (Schubert et al. 2005b), (Schubert und Stechert 2008)). Für Kompetenzentwicklung mit Informatiksystemen ist durch den apparativen Vollzug von Kulturtechniken eine weitere Begründung hinsichtlich der allgemeinen Bildung zu sehen (→ Methodik 7: Von der Anwendung zur Maschine).

Fasst man den Abschnitt zusammen, so fällt auf, dass durch die Empfehlungen der GI und in weiteren Publikationen Informatikunterricht zu Wirkprinzipien von Informatiksystemen gefordert wird. Während van Weert aus den Arbeitsprozessen Anwendungssituationen extrahiert, begründen Friedrich und Neupert anhand von Fehlermeldungen bei der Nutzung des Internets die Notwendigkeit einer informatischen Bildung. Gerade solche Fehlermeldungen können in einem Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung aufgegriffen und als Ausgangspunkt eines systematischen Erkundens des Systemverhaltens genutzt werden (Abschnitt 5.5). Foegens Konzeption einer Didaktik der Informatik anhand der universellen Systemtheorie zeigt im Nachhinein, wie wichtig eine begriffliche Trennung des Systembegriffs vom Informatiksystembegriff ist (Abschnitt 3.2.1), die im Unterrichtsmodell zu beachten ist.

Informatiksysteme im Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen

In den Empfehlungen der GI für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen (GI 2000) wird als übergeordnetes Ziel der Schulinformatik formuliert, allen Schülern einen

> „gleichberechtigten Zugang zu informatischen Denk- und Arbeitsweisen und modernen Informations- und Kommunikationstechniken zu öffnen, informatische Bildung zu vermitteln und damit auch auf lebenslanges Lernen, d. h. auf die Möglichkeiten der ständigen Wissensreorganisation, vorzubereiten. Informatische Bildung ist das Ergebnis von Lernprozessen, in denen Grundlagen, Methoden, Anwendungen, Arbeitsweisen und die gesellschaftliche Bedeutung von Informatiksystemen erschlossen werden" (GI 2000, S. 1).

Die Empfehlung ist durch vier Leitlinien strukturiert. Insbesondere werden „Wirkprinzipien von Informatiksystemen" als eine Leitlinie neben der „Interaktion mit Informatiksystemen", der „informatischen Modellierung" und der „Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft" formuliert. Dabei wird betont, dass Interaktion mit Informatiksystemen als Werkzeug oder Medium nur bei bewusster Thematisierung informatischer Aspekte zur informatischen Bildung gehört (vgl. GI 2000, S. 1). Im eklatanten Gegensatz zu dieser inhaltlichen Forderung bezüglich Informatiksysteme steht der sich in der Empfehlung offenbarende Mangel an definierten Kompetenzen: So werden Informatiksysteme bei gegebener Beschreibung von Fach-, Methoden-, Sozial- und Selbstkompetenz nur bei den beiden letztgenannten Kompetenzen explizit erwähnt.
Gemäß der Lernziele zur Leitlinie „Interaktion mit Informatiksystemen" sollen Schüler in lokalen und globalen Netzen zielorientiert navigieren können, sich Gestaltungsmöglichkeiten erarbeiten, geeignete Informatiksysteme zur Problemlösung auswählen und anwenden sowie Anforderungen an die Gestaltung von Informatiksystemen formulieren können (vgl. GI 2000, S. 3).
Zu den Wirkprinzipien sollen Schüler Aufbau, Funktionsprinzipien und das Zusammenspiel der Systemkomponenten in größere Zusammenhänge einordnen können (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise):

> „Dazu lernen sie grundlegende Ideen und Konzepte (wie z. B. die Digitalisierung und die Kodierung, die universelle Maschine), die Wirkungsweise wichtiger Bestandteile heutiger Informatiksysteme (z. B. Prozessor, Speicher, Netze), Prinzipien, Verfahren und Algorithmen (beispielsweise Suchverfahren) und den prinzipiellen Aufbau komplexerer Basissysteme (beispielsweise Betriebssysteme, Datenbanksysteme, Netzsoftware) kennen" (GI 2000, S. 3).

Es fällt auf, dass das Systemverhalten nicht zur Analyse genutzt wird. Vielmehr werden ausschließlich Fachkonzepte mit Bezug zur inneren Struktur von Informatiksystemen aufgezählt (→ Inhalt 2: Strukturmodelle; → Inhalt 3: Fundamentale Ideen). Außerdem lernen die Schüler, Informatiksysteme als Produkt eines Modellierungsvorgangs zu sehen und die Komponenten eines Informatiksystems in Modellen zu beschreiben:

> „Informatische Modelle spielen bei der Konstruktion und Analyse von Informatiksystemen die Rolle von Bauplänen. Die 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" (GI 2000, S. 3).

Die gesellschaftlichen Auswirkungen von Informatiksystemen sollen anhand von Kriterien diskutiert werden und Gefühlen des Ausgeliefertseins (→ Ziel 4: Entmystifizierung) entgegenwirken (vgl. GI 2000, S. 4).
Als Konsequenz aus den Inhalten der Leitlinien wird bereits für die Primarstufe ein intuitiver, aber fachlich korrekter Unterricht gefordert, der Fertigkeiten in der Anwendung von Informatiksystemen als Werkzeug und Medium, sowie Grundkenntnisse zu Systemkomponenten und Funktionen einschließlich Internetdiensten vermittelt (vgl. GI 2000, S. 5).
Für die unterschiedlichen Schulstufen werden die Bildungsinhalte und -ziele präzisiert. Für die Sekundarstufe I soll den Schülern ein angemessenes Objektmodell helfen, um „Phänomene im Zusammenhang mit Informatiksystemen" (GI 2000, S. 6) zu verstehen. Beispielsweise wird darauf hingewiesen, dass sich bei der Anwendung von Standardsoftware Objekte identifizieren und Zusammenhänge herstellen lassen. Ähnlich argumentiert auch Hubwieser im didaktischen Ansatz der Informationszentrierung (Abschnitt 4.2.3). Kritik ist jedoch dahingehend zu üben, dass Schülern ein Objektmodell sicherlich hilft, aber ein handlungsorientiertes Vorgehen notwendig erscheint, um die Objekte zu entdecken, z. B. durch Experimente mit einem Informatiksystem (vgl. Seite 37; → Methodik 4: Analyse des Systems – Experimente).
Die Vernetzung von Dokumenten in Hypertextstrukturen, Fragen der Etikette bei der Nutzung von E-Mail-Diensten, Grundlagen der Digitalisierung, automatisierte Datenverarbeitung und Datenstrukturierung sowie Auswahl, Benutzung, Gestaltung, Konstruktion und Bewertung problemangemessener Informatiksysteme sollen Schüler sich aneignen (GI 2000, S. 6). Dazu sind vertiefte Kenntnisse zum Aufbau typischer Informatiksysteme und deren Klassifikation notwendig, die jedoch nicht näher erläutert werden:

> „Die problembezogene Auswahl, Benutzung, Analyse, Gestaltung, Konstruktion und Bewertung geeigneter Anwendungssysteme führt zum Aneignen und Vertiefen informatischer Kenntnisse über Aufbau, Arbeitsweise und Klassifikation typischer Informatiksysteme und einer darauf aufbauenden soliden Handlungs- und Beurteilungskompetenz" (GI 2000, S. 6).

Wiederum wird das Systemverhalten nicht explizit für Kompetenzentwicklung mit Informatiksystemen herangezogen. Gerade Analyse und Bewertung sollten sich daher neben der Komponenten der inneren Struktur auch auf das Systemverhalten beziehen. Einen Hinweis darauf, was typische Informatiksysteme sind, liefern die in der Empfehlung genannten Informatiksysteme: E-Mail-Systeme, Betriebssysteme, Datenbanksysteme, Kommunikationssysteme, Informationssysteme, Netzsoftware, Lernsoftware, Textverarbeitungssysteme, Grafikprogramme und verteilte Systeme (→ Inhalt 1: Typische Repräsentanten). Ob mit Klassifikation von Informatiksystemen eine Zuordnung zu den genannten Repräsentanten und eine einfache Unterteilung in Anwendungs- und Systemsoftware gemeint ist, bleibt jedoch offen.

In der Sekundarstufe II sind Anwendung, Analyse, Modifikation und Bewertung Grundlage der Beschäftigung mit Informatiksystemen (→ Methodik 6: Modifikation statt Entwicklung; → Methodik 7: Von der Anwendung zur Maschine):

> „Die Schülerinnen und Schüler eignen sich die Basiskonzepte ausgewählter Informatiksysteme durch **Anwendung, Analyse, Modifikation und Bewertung** an. Die Aufgaben eines Betriebssystems bei der Verwaltung von Betriebsmitteln werden modellhaft skizziert. Rechnernetze und verteilte Systeme werden durch geeignete Modelle (Schichtenmodell, Protokolle, Adressierung) charakterisiert und auf schultypische Aufgabenstellungen angewandt. Die Struktur und Funktionsweise von Rechnern wird ausgehend vom von-Neumann-Modell verallgemeinert" (GI 2000, S. 7f; Hervorh. durch den Autor).

Die Auswahl der Schichtenmodelle und des Von-Neumann-Modells stellt die Frage nach weiteren Modellen, die Kompetenzentwicklung mit Informatiksystemen fördern. Somit sind weitere Modelle auf ihre Eignung zu untersuchen. Darüber hinaus müssen Schüler Fragen der Berechenbarkeit und Entscheidbarkeit beantworten können und Konzepte der Software-Ergonomie zur Bewertung von Informatiksystemen und Problemstellungen einsetzen (vgl. GI 2000, S. 8).

Kritik am Begriff Informatiksystem in der GI-Empfehlung üben Bartke und Maurer:

> „Der von der GI geprägte Begriff des ‚Informatiksystems' mit seiner verwaschenen, je nach Kontext beliebig interpretierbaren Semantik eignet sich nicht als Abstraktion für die Beschreibung von Zielen und Inhalten des Informatikunterrichts" (Bartke und Maurer 2000, S. 1).

Thomas weist diese Kritik jedoch zurück. Dazu nennt er drei Gründe: Erstens, die vorhandenen Definitionen von Informatiksystemen, nicht zuletzt durch die GI und den Informatikduden, zweitens die verbreitete Nutzung des Begriffs, und drittens betont er, dass auch andere Wissenschaften keine präzise Definition von ihren Gegenständen haben (Thomas 2002, S. 2).

Kompetenzentwicklung mit Informatiksystemen in der aktuellen Diskussion von 2001 bis 2007

Brunnstein (2001) legt auf der INFOS 2001 (Keil-Slawik und Magenheim 2001) dar, dass komplexe Informatiksysteme nur schwer zu durchschauen und kaum zu beherrschen seien, was aber im Informatikunterricht nicht zur Sprache komme:

> „Die bisherige Entwicklung der Schulinformatik in Deutschland hat es versäumt, den Lehrern wie auch den Schülern ein angemessenes Verständnis der Risiken und der Un-Beherrschbarkeit heutiger Computer- und Netzsysteme nahe zu bringen" (Brunnstein 2001, S. 9).

Verstärkt werde dies dadurch, dass Schüler Elemente des Programmierens, die Prinzipien von Hardware, Software, Betriebs- und Netzsystemen kennen lernen

und zusätzlich wesentliche Anwendungsfelder und Techniken zur Nutzung von Informationstechnik im Informatikunterricht erfahren. Damit wird nach Brunnstein die prinzipielle Möglichkeit des Verstehens impliziert. Jedoch sieht Brunnstein im Gegenteil schon die Grundannahme als unzutreffend an, man „verstehe" Geräte, Systeme und Programme durch Informatikunterricht, da nur wenige Informatikspezialisten mit viel Erfahrung einzelne Systeme beherrschen (Brunnstein 2001, S. 11). Seine Schlussfolgerung ist, dass der risikobewusste Gebrauch einschließlich des Verzichts auf riskante Nutzungen für Schüler anzustreben wäre:

> „Es wäre nunmehr spannend zu sehen, wie sich die Curricula verändern müssten, damit das naive Verständnis heutiger Schulabgänger durch ein **aufgeklärtes Verständnis [...] durch eine risikobewusste Nutzung, z. T. auch eine Einschränkung oder Vermeidung riskanter Nutzungen abgelöst** wird" (Brunnstein 2001, S. 12; Hervorh. im Original).

Diese Sichtweise von Brunnstein unterstützt einerseits die Notwendigkeit des Informatikunterrichts. Andererseits greift sie die Schwierigkeit auf, dass informatische Prinzipien wie das Geheimnisprinzip bei der Gestaltung von Informatiksystemen eingesetzt werden. Deren Ziel ist gerade das Verbergen von Wirkprinzipien und Teilen der inneren Struktur vor dem Anwender. Aber auch andere Entwickler können aufgrund von Kapselung beispielsweise von primitiven Datenstrukturen abstrahieren. In der Umkehrung kann die Schlussfolgerung jedoch nicht sein, dass wissenschaftsorientierter Informatikunterricht bezüglich Basiskompetenzen zu Informatiksystemen nicht möglich ist, weil bereits der Entwickler von Informatiksystemen Lösungen zur Systemgestaltung versteckt. Deshalb gilt es, Erkenntnisse der Informatik einzusetzen, um mittels informatischer Konzepte wie strukturierter Zerlegung sowie Mechanismen zur Abstraktion und Formalisierung konkrete Informatiklösungen und -strukturierungen hervorzuheben (→ Methodik 2: Kognitive Modelle, vgl. auch Abschnitt 4.2.3). Genau dies ist Ziel des Informatikunterrichts:

> „Es besteht die Gefahr der Einengung auf zu schlichte Bildungsziele, da die komplexen Modellhintergründe zur Undurchsichtigkeit der Systeme führen. Informatikdidaktik stellt sich deshalb die Aufgabe, die Transparenz komplizierter Algorithmen und Datenstrukturen zu fördern, damit menschliche Verantwortung tatsächlich wahrgenommen werden kann" (Schubert und Schwill 2004, S. 52).

Thomas (2002) stellt Verbindungen zwischen Informatiksystemen und Modellen der Informatik her. Er untersucht den Modellbegriff in der Informatik durch Auswertung universitärer Skripte und bezieht sich dabei auf die Allgemeine Modelltheorie nach Stachowiak (1973). Danach ist ein Modell immer auch als System im Sinne der Systemtheorie auffassbar, ein System aber nicht zwingend ein Modell (vgl. Thomas 2001, S. 176). Er identifiziert im Sprachgebrauch der Kerninformatik Hauptmodelltypen (Tabelle 4.2). Alle Hauptmodelltypen stehen in engem Bezug zu Informatiksystemen. Während Architekturmodelle der inneren Struktur von Informatiksystemen zugrunde liegen, beschreiben Vorgehensmodelle im Wesentlichen

den Ablauf bei der Softwareentwicklung. Dabei kommen insbesondere Entwurfsmodelle als Beschreibungstechniken zum Einsatz. Untersuchungsmodelle werden genutzt, um vollständige (Teil-) Systeme zu testen. Außerdem nennt er mentale Modelle, denen er Metaphern und fundamentale Ideen der Informatik nach Schwill (Thomas 2002, S. 62) zuordnet. Gerade bei diesem letzten Modelltyp ist sicherlich kritisch zu hinterfragen, inwieweit die beispielhaft genannten Metaphern informatikspezifisch und über deren nachweisbares Auftreten im Sprachgebrauch hinaus einen vergleichbaren Anteil an der Strukturierung der Kerninformatik haben, wie die vier erstgenannten Hauptmodelltypen. Dennoch betont Thomas die Wichtigkeit der so verstandenen mentalen Modelle für die Anwendung und Entwicklung von Informatiksystemen:

> „Es zeigt sich zusammenfassend, dass mentale Modelle einen bedeutenden Stellenwert bei der Anwendung und Entwicklung von Systemen der Informatik haben. Mentale Modelle bilden die Grundlage zum Verständnis und für die Entwicklung von externen semantischen und technischen Modellen. Das Zusammenstellen und Evaluieren von nützlichen Metaphern und konzeptuellen Modellen zum Aufbau mentaler Modelle in der Informatik und im Informatikunterricht sollte Ausgangspunkt für weitere zahlreiche Arbeiten sein, um eine zukunftsorientierte Modellierungskompetenz im Allgemeinen und beim Umgang mit Informatiksystemen zu erzeugen" (Thomas 2002, S. 45).

Thomas beruft sich auf Dutke (1994), der den Aufbau mentaler Modelle durch Exploration von Informatiksystemen exemplarisch im Bereich Software-Ergonomie analysiert hat (Abschnitt 5.5). Der von Thomas betonte Zusammenhang von Modellierungskompetenz und dem bewussten Anwenden von Informatiksystemen ist für Kompetenzentwicklung hervorzuheben. Dabei ist begrifflich fest zu halten, dass Thomas unter Modellierung in diesem Fall nicht ausschließlich Systementwicklung, sondern Modellbildung versteht.

Tabelle 4.2: Hauptmodelltypen und Beispiele zitiert nach (Thomas 2003, S. 147)

Hauptmodelltypen	**Beispiele für Untermodelle**
Architekturmodelle	Rechnerarchitekturen (Von-Neumann, SISD, MIMD, neuronale Netze), theoretische Maschinenmodelle (Turingmaschine, Automaten), Rechenmodelle (imperativ, logisch-deklarativ, funktional), Referenzmodelle (Client-Server, OSI-Schichten)
Vorgehensmodelle	Wasserfallmodell, Prototypenmodell, Evolutionäres Modell, Objektorientierte Modellierung
Entwurfsmodelle	Aufgabenmodell oder Anforderungsanalyse, Modellierungssprachen (Struktogramm, Programmablaufplan, UML), Komponentenmodell, Datenmodelle (hierarchisches, relationales, logisches, objektorientiertes)
Untersuchungsmodelle	Analytisches Modell (Verifikation, betriebswirtschaftliche Modelle), Simulationsmodell (Blackbox Testen; deterministisch, stochastisch)
Mentale Modelle	Metaphern, Konzeptuelle Modelle, Fundamentale Ideen

Als Beispiel zieht er den Von-Neumann-Rechner heran, der in Informatikvorlesungsskripten als technisches Idealmodell unter Vernachlässigung technischer Details beschrieben wird. Damit kann

> „ein Modell sowohl Abbild ‚von etwas' (hier: Denkmodelle) als auch Vorbild ‚für etwas' (hier: Computergerät) sein" (Thomas 2001, S. 177).

Da Thomas die Notwendigkeit mentaler Modelle hervorhebt, um Informatiksysteme anwenden und konstruieren zu können (→ Methodik 2: Kognitive Modelle), und gleichzeitig fundamentale Ideen der Informatik zu den mentalen Modellen zählt, zeigt er implizit die enge Verbindung zwischen Kompetenzentwicklung mit Informatiksystemen und fundamentalen Ideen der Informatik (→ Inhalt 3: Fundamentale Ideen).

Humbert (2003) entwickelt in seiner Dissertation ein Modulkonzept für die Schulinformatik und beschreibt darin das Modul „Informatiksysteme verstehen und verantwortlich nutzen". Bezüglich Informatiksysteme stellt er fest:

> „Heute ist Informatik u. a. gekennzeichnet durch die objektorientierte Konstruktion von Informatiksystemen. Der Begriff Informatiksystem wird zur Bezeichnung der Einheit von Hard- und Software unter Betonung ihrer interaktiven Eigenschaften verwendet. Informatiksysteme werden zunehmend unter stärkerer Berücksichtigung verteilter und vernetzter Systeme gestaltet.
> Die beiden Aspekte objektorientierte Modellierung und verteilte, vernetzte Systeme stellen eine Umorientierung des informatischen Wissenschaftsverhältnisses dar, die als Paradigmenwechsel (vgl. Kuhn 1969) bezeichnet werden kann" (Humbert 2003, S. 5).

Auch wenn der Autor der vorliegenden Arbeit diesen Umschwung nicht als Paradigmenwechsel bezeichnen möchte, da keine Grundlagenkrise ausgelöst wurde, so stimmt er mit Humbert darin überein, dass diese Entwicklungen in der Fachwissenschaft von der Fachdidaktik Informatik produktiv aufzunehmen und konstruktiv zu diskutieren sind (Humbert 2003, S. 5). Die fachliche Strukturierung des Modulkonzepts nach Humbert orientiert sich aufgrund der Umorientierung an den Bereichen Rechnernetze, verteilte Systeme und informatische Modellierung (Humbert 2003, S. 105). Er schlussfolgert:

> „Pragmatisch angelegte Ansätze zu den Überlegungen, welche Methoden in der Informatik zum Einsatz gebracht werden, führen zu der Schlüsselbestimmung ‚Informatische Modellierung im Kontext' und dabei insbesondere zu der Besonderheit von Informatiksystemen, die Modellierung eines Realitätsausschnitts in eben dieser Realität wirksam werden lassen" (Humbert 2003, S. 15).

Informatiksysteme können demnach auch als enaktive Modelle gesehen werden, die in die Realität einwirken ((Schubert und Schwill 2004, S. 155); (→ Methodik 2: Kognitive Modelle; Abschnitt 5.3)). Er weist dem Modul „Informatiksysteme verstehen und verantwortlich nutzen" eine Schlüsselrolle zu, da das Modul die fachliche Basis für die in allen Modulen notwendige Arbeit in vernetzten Strukturen, beispielsweise

im Schulintranet, legt (vgl. Humbert 2001, S. 123). Nach Humbert muss Problem lösender Informatikunterricht auch in der wissenschaftspropädeutisch ausgerichteten Sekundarstufe II den Schülererwartungen hinsichtlich deren eigener Perspektiven auf die Anwendung von Informatiksystemen entgegen kommen (vgl. Humbert 2001). Dies wird begleitet von seiner Forderung, für den Informatikunterricht die spielerische Interaktion als Modalität des Handelns (Krämer 1997) bezüglich Informatiksysteme verfügbar zu machen, was seiner Ansicht nach noch nicht geschehen ist (→ Inhalt 4: Sichten). Außerdem seien die Überlegungen von Thomas (2002) zur Modellierung von Modellen zu konkretisieren und auf Eignung für den Informatikunterricht zu untersuchen (vgl. Humbert 2003, S. 78).

Inhaltlich umfasst das Modul einerseits Fragen des Persönlichkeitsschutzes, z. B. bei Computer-Supported-Collaborative-Learning-Systemen (CSCL-Systemen). Aber es wird auch beschrieben, wie Betriebssysteme handlungsorientiert und plattformunabhängig beispielsweise mittels Scriptsprachen erschlossen werden können. Humbert schlägt darin eine Brücke zur Modellierung mit Schichtenmodellen, Model-View-Controller-Architektur (MVC) bzw. Entwurfsmustern:

> „Soll Informatikunterricht zukunftsweisende Impulse setzen, so ist hier durch die Trennung der verschiedenen Ebenen bei der Modellierung von Informatiksystemen, z. B. mit dem MVC-Konzept, eine Möglichkeit aufzuzeigen, wie zukünftige Entwicklungen berücksichtigt werden können, ohne die erarbeiteten Fachkonzepte obsolet zu machen. Dabei soll nicht unterschlagen werden, dass die Benutzung von Entwurfsmustern im Informatikunterricht nicht ohne Probleme möglich ist. Ein zentrales (bisher nicht zufriedenstellend gelöstes) Problem besteht in der Motivation von Entwurfsmustern auf einer schmalen fachlichen Basis ohne hinreichende Projekterfahrung (im Sinne der Informatik) und der damit verbundenen Notwendigkeit (und Nützlichkeit) dieses Abstraktionsmechanismus" (Humbert 2003, S. 107).

Entwurfsmuster werden von Humbert für den Informatikunterricht somit als hilfreiche Abstraktionen zur Strukturierung von Informatiksystemen angesehen (→ Inhalt 2: Strukturmodelle). Allerdings zieht er zur Begründung ihres Einsatzes im Lehr-Lernprozess allein die Nützlichkeit im Softwareentwicklungsprozess heran, die, wie er betont, für Informatikunterricht nicht unproblematisch ist (Abschnitt 5.4). Als Beispielthema betrachten Humbert et al. (2005) im Rahmen des Modulkonzeptes den Anmeldevorgang zu Informatiksystemen für den Informatikunterricht. Ziel ist es, isolierte Wissensinseln zu vermeiden. So stellen sie den Vorgang mittels Struktogrammen dar und nehmen ihn als Ausgangspunkt für Unterricht zu Rechnernetzen, softwareergonomischen Fragestellungen und Kryptologie. Daran zeigt sich, wie ein einfacher Vorgang beim bewussten Anwenden von Informatiksystemen genutzt werden kann, um einen inhaltlich zusammen hängenden Unterricht zur Förderung der Kompetenzentwicklung zu gestalten, der viele unterschiedliche Bereiche der Informatik aufgreift (→ Inhalt 4: Sichten; → Methodik 9: Vernetzung der Unterrichtsinhalte). Darüber hinaus ergänzt Humbert, dass Überlegungen der Theoretischen Informatik notwendig sind, um Wirkprinzipien von Informatiksys-

temen zu verstehen. Denn mit ihnen können deren Strukturen verdeutlicht und modelliert werden (vgl. Humbert 2001).

Ähnlich argumentieren Hartmann und Nievergelt bezüglich des Stellenwertes der Theoretischen Informatik für Kompetenzentwicklung mit Informatiksystemen und fordern „formale Systeme" als Grundprinzip der Informatik und der informatischen Bildung (Hartmann und Nievergelt 2002). Begründung ist, dass mathematisch streng definierte formale Systeme in der Informatik genutzt werden, um Verhaltensweisen von Maschinen zu beschreiben und auszuführen. Als Beispiele nennen sie Automaten, Turingmaschine und logische Kalküle (→ Inhalt 2: Strukturmodelle).

Auf der INFOS 2003 (Hubwieser 2003) bezeichnet Hubwieser effizientes Arbeiten und verantwortliches Anwenden von Informatiksystemen als Schlüsselkompetenzen (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise; → Ziel 3: Bewusste Anwendung):

> „Effizientes Arbeiten und verantwortungsvoller Umgang mit Informatiksystemen gehören daher zu den Schlüsselkompetenzen unserer Zeit. Dies setzt jedoch grundlegende **Kenntnisse über den Aufbau und die Funktionsweise** solcher Systeme voraus" (Hubwieser 2003, S. 4; Hervorh. durch den Autor).

Durch diese Feststellung wird für Kompetenzentwicklung mit Informatiksystemen nochmals betont, dass Architekturen und Strukturmodelle von Informatiksystemen ebenso notwendig sind wie grundlegende Informatikprinzipien, die die Funktionsweise der Systeme erklären. Darüber hinaus muss es in einem Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung gelingen, beides miteinander zu kombinieren (→ Methodik 9: Vernetzung der Unterrichtsinhalte). Der Autor schlägt dazu so genannte Wissensrepräsentationen vor (Abschnitt 5.4.1). Hubwieser begründet seine Feststellung mit der umfassenden Digitalisierung und Vernetzung und betont, dass die Informatik Fachkonzepte bereitstellt, die über Aufbau und Funktionsweise im Bereich der Informatiksysteme hinaus bei der Beschreibung und Beherrschung von komplexen Systemen (→ Ziel 2: Arbeiten mit komplexen Systemen) aller Art hilfreich sind (Hubwieser 2003, S. 4).

Zur Vorbereitung von Informatikbildungsstandards zieht Friedrich (2003) Schlussfolgerungen aus dem „Programme for International Student Assessment" (PISA) (Senkbeil und Drechsel 2004). Er skizziert eine ausschließlich theoretisch begründete und auf Basis von Arbeiten zu mathematischen und naturwissenschaftlichen Kompetenzen beruhende Kompetenzstufung für den Informatikunterricht: (1) Bedienung von Informatikanwendungen, (2) Benutzung von Informatiksystemen, (3) Kenntnis fachsystematischer Grundlagen, (4) Verständnis von Konzepten der Informatik, (5) Entwicklung und Bewertung von Informatiksystemen. Kritisch ist hierbei anzumerken, dass Benutzung von Informatiksystemen nur auf solch niedriger Stufe nicht mit der in dieser Arbeit angestrebten Kompetenzentwicklung mit

Informatiksystemen gleichzusetzen ist. Des Weiteren impliziert insbesondere die höchste Stufe, dass Entwicklung von Informatiksystemen als Ziel informatischer Bildung angesehen wird, wenngleich Friedrich dies durch den Zusatz der Bewertung von Informatiksystemen relativiert. Die Stufung kombiniert Friedrich mit den vier Leitlinien (Abschnitt 4.2.4) der GI-Empfehlung (GI 2000), so dass eine Matrix entsteht. Dabei ordnet er den Leitlinien vorab jeweils einen grundlegenden Aspekt informatischer Bildung zu: der Interaktion mit Informatiksystemen die Problemlösung, den Wirkprinzipien von Informatiksystemen die Konzepte, der Informatischen Modellierung die Abstraktion und den Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft die Allgemeinbildung (Friedrich 2003, S. 136). Diese von Friedrich vorgeschlagene Zuordnung birgt jedoch auch die Gefahr, dass eine Gleichsetzung beispielsweise von Fachkonzepten mit den Wirkprinzipien von Informatiksystemen erfolgt, die wiederum den Verlust der Fokussierung auf Informatiksysteme zur Folge haben kann. Friedrich bietet eine Begriffsklärung für Wirkprinzipien von Informatiksystemen an, die ähnlich dem Begriff „Prinzip" im Duden Fremdwörterbuch (Drosdowski et al. 2001) ist:

> „Ein WIRKPRINZIP kennzeichnet naturwissenschaftliche und technische Gegebenheiten bei der Realisierung von Vorgängen, insbesondere dabei die Gesetzmäßigkeit, die Idee, die einer Sache zugrunde liegt bzw. nach der etwas wirkt oder auch das Schema nach dem es aufgebaut ist. Man untersucht also deren Zweck, ihre Struktur und Funktionalität" (Friedrich 2003, S. 142).

Die Bildungsziele zur Leitlinie Wirkprinzipien umfassen nach Friedrich (1) die Benennung und Beschreibung des Rechnerarbeitsplatzes, (2) Wissen um Grundfunktionen eines Informatiksystems, (3) Grundlegende Fachbegriffe und Einordnung in Fachsystematik, (4) Theoretische Grundlagen sowie Fachbegriffe und Konzepte sowie (5) Anwenden und Entwickeln von Konzepten (Friedrich 2003, S. 140). Für die Leitlinie Interaktion bedeutet dies eine Ausrichtung auf Informationen und Daten, Arbeitsweisen und Methoden (Friedrich 2003, S. 142f). Schwierigkeit dieser Matrix ist, dass sich der direkte Bezug zu Informatiksystemen auf die unterschiedlichen Leitlinien kaum nachvollziehbar aufteilt: So wird „Wissen um Grundfunktionen von Informatiksystemen" der Leitlinie Wirkprinzipien auf Stufe 2 (Benutzung von Informatiksystemen) zugeordnet, aber „Umgang mit Systemen; Auswahl von Methoden" der Leitlinie Interaktion auf Stufe 4 (Verständnis von Konzepten der Informatik). Der Grund dafür, dass dieser Ansatz kaum weitere Beachtung erfahren hat, ist in dem starken Bezug zu den Leitlinien der GI-Empfehlung von 2000 zu sehen, die in der nachfolgenden Diskussion um Informatikbildungsstandards nicht weiter verfolgt werden (Abschnitt 4.2.5).

Humbert und Puhlmann (2004) geben eine Klassifikation von Phänomenen für die informatischen Bildung anhand deren Bezug zu Informatiksystemen an. Sie beziehen sich dabei auf den aus der Physik und Mathematik bekannten fachdidaktischen Ansatz der Phänomene. Diese bezeichnen dabei das Auftreten von Informatik im Alltag. Humbert und Puhlmann identifizieren drei Kategorien von Phänomenen:

(1) Phänomene mit direktem Bezug zu Informatiksystemen (bewusstes Anwenden von Informatiksystemen), (2) Phänomene mit indirektem Bezug zu Informatiksystemen (Alltagssituationen in denen Informatiksysteme nicht bewusst wahrgenommen werden) und (3) Phänomene ohne Bezug zu Informatiksystemen, aber mit informatischer Struktur. Ziel ist dabei die Alltagsbewältigung im Sinne einer „Literacy". Die beiden Phänomenkategorien mit Bezug zu Informatiksystemen sind relevant für Kompetenzentwicklung. Ein Ziel muss letztendlich das Bewusstmachen von Informatiksystemen im Alltag sein. Schlussfolgerung und Herausforderung für ein Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen im Sinne eines bewussten Anwendens von Informatiksystemen ist, dass konsequent motivierende Phänomene im Unterricht einzusetzen sind (→ Ziel 5: Bereitschaften), die Experimente mit Informatiksystemen fordern (vgl. Seite 37) und deshalb auch abstrakte und theoretische Themen tangieren (Abschnitt 5.5).

Modrow (2004) diskutiert den möglichen Beitrag eines Pflichtfachs Informatik zur Ausprägung einer „Technical Literacy" (auch: „Technological Literacy") bei Schülern. Er begründet diesen Beitrag der informatischen Bildung vor allem damit, dass Informatik valide Modelle von Hard- und Software zum Gegenstand hat sowie die Konstruktion und das Testen von Systemen. Diese „Technical Literacy" beruht nach Modrow auf einer Ausweitung der „Computer Literacy", die um eine Entscheidungskomponente ergänzt wird, so dass über den angemessenen Einsatz oder Nichteinsatz von Computern entschieden werden kann. Modrow betont in diesem Kontext die Wichtigkeit technischer Aspekte wie Rechnernetze neben Algorithmik, Theoretischer Informatik und gesellschaftlichen Fragestellungen für den Informatikunterricht, die er in den 2004 von der Kultusministerkonferenz veröffentlichten „Einheitliche Prüfungsanforderungen in der Abiturprüfung (EPA) im Fach Informatik" vermisst. Welche technischen Aspekte er explizit vermisst, bleibt jedoch unbeantwortet. In den EPA Informatik werden Kompetenzen zur Funktionsweise, zur inneren Struktur sowie zu Möglichkeiten und Grenzen von Informatiksystemen beschrieben:

> „Der Informatikunterricht in der gymnasialen Oberstufe leistet einen spezifischen Beitrag zur Allgemeinbildung, indem er den Erwerb eines systematischen, zeitbeständigen und über bloße Bedienerfertigkeiten hinausgehenden Basiswissens über die **Funktionsweise, die innere Struktur sowie die Möglichkeiten und Grenzen von Informatiksystemen** ermöglicht. Dadurch wird deren sinnvolle, kompetente und verantwortungsbewusste Nutzung und Beurteilung ermöglicht. [...] Informatische Methoden wie das Strukturieren, das **systematische Zerlegen komplexer Systeme** in überschaubare Teile, das Formalisieren und Interpretieren fördern und fordern die Abstraktionsfähigkeit und das Erfassen logischer Zusammenhänge. Bei der Modellbildung, die bei der Konstruktion und Analyse von Informatiksystemen eine entscheidende Rolle spielt, üben die Schüler in besonderem Maße, eine Situation von verschiedenen Standpunkten aus zu beurteilen" (KMK 2004, S. 3; Hervorh. durch den Autor).

Mit dem Hinweis, dass es bei der Analyse von Informatiksystemen wichtig ist, verschiedene Standpunkte einzunehmen, ist für ein Unterrichtsmodell zu Informa-

tiksystemen und Kompetenzentwicklung ein Sichtenkonzept zu entwickeln (→ Inhalt 4: Sichten). Grundlage können in einer ersten Annäherung die Kategorien des nach außen sichtbaren Verhaltens, der inneren Struktur und ausgewählter Implementierungsaspekte bilden (Abschnitt 5.3) und entsprechende informatische Modelle. Bei dem Zusammenführen der unterschiedlichen Perspektiven auf Informatiksysteme wird die Lehrperson darauf hingewiesen (KMK 2004, S. 12), dass sie möglichst zusammenhängende Teilaufgaben stellen soll, die unterschiedliche Blickrichtungen und mögliche Vernetzung fördern (→ Methodik 9: Vernetzung der Unterrichtsinhalte). Dementsprechend bieten sich Experimente als Analyseaufgaben an, die die Kombination der Perspektiven ermöglichen (→ Methodik 4: Analyse des Systems – Experimente; Abschnitt 5.5). Die Wichtigkeit von Informatiksystemen wird in den EPA auch durch den angegebenen fachlichen Inhaltsbereich „Interaktion mit und von Informatiksystemen" betont, beispielsweise sollen die Schüler verschiedene typische Informatiksysteme (→ Inhalt 1: Typische Repräsentanten) als Werkzeuge nutzen können (vgl. KMK 2004, S. 6). Somit legen die EPA Informatik Wert auf das Strukturieren und systematische Zerlegen von Informatiksystemen, auf Interaktion zwischen Informatiksystemen, beispielsweise mit Protokollen und einem Schichtenmodell, sowie auf die Mensch-Maschine-Interaktion. Insbesondere wird Modellbildung als Grundlage einer Analysefähigkeit betont, die über Bedienerfertigkeiten hinaus geht (→ Methodik 2: Kognitive Modelle). Neben dem von Modrow fest gestellten Mangel bezüglich der technischen Grundlagen der Informatik in den EPA ist die sehr starke Fokussierung auf die Beschreibung von Systemteilen in Modellen für Kompetenzentwicklung mit Informatiksystemen nicht allein ausreichend. Dies liegt vor allem daran, dass eben auch konkretes Systemverhalten betrachtet werden muss und theoretische Hintergründe aller Hauptfunktionen von Informatiksystemen zu analysieren sind (Kapitel 3). Es stellt sich die Frage, inwieweit die Anforderungsbereiche I-III der EPA zur Bestimmung von Niveaustufen in einem Kompetenzmodell für die informatische Bildung dienen können.

Auf der INFOS 2005 (Friedrich 2005) skizziert Hubwieser, wie durch Konsens über Postulate die Durchsetzung des Fachs Informatik nach dem Ansatz der Informationszentrierung in Bayern gelang (Hubwieser 2005). Darunter ist das Postulat „Tiefgreifendes Verständnis statt Benutzerfertigkeit", das auf übertragbare, langlebige Konzepte statt Bedienerschulung setzt. Das Postulat „Werkzeuge in Reichweite" wiederum fordert neben dem Verständnis grundlegender Konzepte die gedankliche Durchdringung allgegenwärtiger Softwaresysteme, wodurch praktische Arbeit am Rechner impliziert ist (Hubwieser 2005, S. 29). Damit kann auch ein möglicher Vorwurf mangelnder Kreativitätsförderung bei der Fokussierung auf Basiskompetenzen zu Informatiksystemen entkräftet werden: Die Förderung von Kreativität und Teamarbeit sind in der Informatik eben nicht ausschließlich bei der Konstruktion bzw. Programmierung von Informatiksystemen in Projektarbeit möglich. Genauso kann Unterricht diese Aspekte integrieren, indem theoretische Grundlagen durch

bewusstes Anwenden am Rechner veranschaulicht werden (→ Ziel 5: Bereitschaften).

Die Dissertationsschrift von Voß (2006) zeigt, wie Standardsoftware anhand von Modellen und informatischen Konzepten verstanden werden kann. Voß stellt ein Vorgehen zu Anwenderschulungen für Tabellenkalkulationen und Textverarbeitung in der beruflichen Weiterbildung vor, das im Kontext der Informationszentrierung auch in der Sekundarstufe II erprobt wurde (Voß 2005b). Motivation ist die zunehmende Komplexität von Standardsoftware, die durch informatische Bildung verständlich wird:

> „Eine effiziente Nutzung der zunehmend komplex gestalteten Büroanwendungen setzt umfassendes und vernetztes Wissen voraus [...]. Informatische Modelle bieten eine Möglichkeit, Dokumentstrukturen zu repräsentieren, die im Gegensatz zu den schnelllebigen Benutzungsoberflächen längerfristig Bestand haben und helfen können, die **Arbeitsweise moderner Softwaresysteme** zu erklären" (Voß 2006, S. III; Hervorh. durch den Autor).

Ihr Ziel ist ein Konzept für eine nachhaltige Benutzerschulung. Sie identifiziert informatische Modelle in der Standardsoftware:

> „Die Prinzipien der objektorientierten, zustandsorientierten bzw. funktionalen oder ablauforientierten Modellierung werden am konkreten Anwendersystem entdeckt und verstanden" (Voß 2005b, S. 286).

Kennzeichnend für ihre Arbeit ist ein objektorientiertes Re-engineering der Informatiksysteme. Daraus ergeben sich zwei Konsequenzen: Erstens werden Standardsoftwaresysteme in der vorliegenden Entwicklung des Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung ausgeklammert, da die Arbeit von Voß diesen Bereich abdeckt. Zweitens ist aus ihrer Vorgehensweise abzuleiten, dass Modellierungen von Informatiksystemteilen zu entwickeln bzw. vorhandene informatische Konstrukte und Modelle für den Informatikunterricht nutzbar zu machen sind (→ Inhalt 2: Strukturmodelle; Abschnitt 5.4). Diese können dann als Grundlage einer systematischen Erkundung eines Informatiksystems dienen, in der Wirkprinzipien entdeckt und verstanden werden (→ Methodik 4: Analyse des Systems – Experimente; Abschnitt 5.5).

Koubek (2005) greift Aufbau und Funktionsweise von Informatiksystemen in einem Kompetenzmodell auf. Er stellt zwei getrennte Kompetenzstufungen für eine informatische Allgemeinbildung vor, die im Unterricht zu kombinieren sind. Die erste beschreibt das diskursive Grundverständnis der Schüler zum informationstechnologischen Inhalt, d. h. sozio-kulturelle Themen wie Ethik, Datenschutz und Urheberrecht. Die zweite Stufung zielt auf technische Kompetenzen ab. Deren erste Ebene umfasst technische Hintergründe zu Kernthemen wie Datenbanken und Kommunikationswerkzeugen. Die zweite Stufe stellt eine Vertiefung in die Komponenten von Hard- und Software dar. „Aufbau und Funktionsweise der Informatiksysteme

verstehen" bildet die mittlere Kompetenzstufe. Beispiel ist, dass Schüler ein Verständnis für Schichtenarchitektur entwickeln und mit der Hard- und Software des Schulsystems arbeiten können (Koubek 2005, S. 59). Die nachfolgenden beiden Stufen beinhalten das Modellieren und Implementieren sowie das Lösen von Problemen der eigenen Lebenswelt mit informatischen Methoden. Koubek betont, dass diese letzten Ebenen nur in Kombination mit Kompetenzen zum diskursiven Grundverständnis erklommen werden können. Positiv an dem Modell der zwei Kompetenztreppen nach Koubek ist sicherlich die klare Trennung zwischen technischen und sozio-kulturellen Aspekten. Dennoch fehlt es an konkreten Angaben, wie die Kompetenztreppen zu verbinden sind (→ Inhalt 4: Sichten). Gelingt die Verbindung, so unterstützt sie die Gestaltung komplexer Anforderungssituationen, z. B. Datenschutz, im Sinne der Kompetenzdefinition nach Weinert (Abschnitt 2.1). Kritik ist jedoch hinsichtlich Informatiksystemen und Kompetenzentwicklung durch die Frage anzubringen, warum gerade das Implementieren als Teil der höchsten Kompetenzebene das Ziel bildet. Der ebenfalls auf dieser Kompetenzebene genannte Aspekt hingegen, dass Schüler in der Lage sein sollen, Informationstechnologien ihren Wünschen und Bedürfnissen entsprechend einzusetzen, entspricht den mit der Förderung der Kompetenzentwicklung mit Informatiksystemen verfolgten Zielen.

Zur Förderung der Kompetenzentwicklung mit Informatiksystemen werden auch Simulationen eingesetzt. Herper und Hinz (2005) beschreiben, wie Schüler ein informatisches Schienenverkehrssystem mit Signalsteuerung mittels unterschiedlicher Modellierungen analysieren und implementieren können. Dazu skizzieren sie eine Unterrichtssequenz für 26 Unterrichtsstunden. Schwerpunkte darin sind das Verstehen des Modells bestehend aus mechanischer Funktionalität, Hard- und Software sowie Sensorik und Aktorik. Nach Modellierung in einem Klassendiagramm wird eine Simulation des Systems zur Entwicklung der Signalsteuerung eingesetzt. Bezüglich Informatiksysteme und Kompetenzentwicklung ist klar zu stellen, dass für dieses Szenario eine Abgrenzung der Systembegriffe vorzunehmen ist (Abschnitt 3.2.1). Zu trennen sind das zugrunde liegende Informatiksystem und seine Umwelt, in die es einwirkt. Die Simulation des Gesamtsystems wiederum eröffnet viele Möglichkeiten des Experimentierens mit unterschiedlichen Eingaben und Systemkonstellationen (→ Methodik 4: Analyse des Systems – Experimente).

Eine Alternative zur Simulation bildet der Einsatz von Robotersystemen, wie Wiesner und Brinda ihn für den Informatikunterricht nutzen. Sie berichten insbesondere von Erfahrungen bei der Vermittlung von Algorithmen anhand eines konkreten Robotersystems (Wiesner und Brinda 2007). Dabei nutzen sie die mechanisch-technischen Komponenten vornehmlich als motivationalen Faktor (→ Ziel 5: Bereitschaften; (vgl. Hirsch et al. 2000)), haben aber das Ziel, Schülern das formale Beschreiben von Abläufen anhand des Analysierens, Modellierens, Implementierens und Reflektierens handlungsorientiert nahe zu bringen (Wiesner und Brinda 2007, S. 117). Interessant ist für Informatiksysteme und Kompetenzentwicklung darüber

hinaus, dass Robotersysteme genutzt werden können, um Verteiltheit zu thematisieren. Büdding (2007) verstärkt diesen Effekt noch, indem er Robotersysteme im Rahmen des mobilen Lernens mit tragbaren Rechnern, so genannten „Personal Digital Assistant" (PDA), im Informatikunterricht einsetzt. Roboter werden kabellos mit PDAs vernetzt und können von den Schülern über die PDAs gesteuert werden.

Wursthorn (2006) erarbeitet in ihrer Dissertation, welche informatischen Grundkonzepte für Klasse 5 der Realschule im fächerübergreifenden Unterricht geeignet sind. In ihrer Einteilung der Unterrichtsinhalte fasst sie unter Wirkprinzipien von Informatiksystemen die Bereiche Codierung im Sinne von Abbildevorschriften, Algorithmisierung und Automatisierung zusammen. Für Algorithmisierung beruft sie sich auf die fundamentalen Ideen nach Schwill und für Automatisierung auf den informationszentrierten Ansatz (Wursthorn 2006, S. 22). Warum sie genau diese Grundkonzepte wählt, bleibt jedoch trotz vorheriger Analyse des Forschungsstandes unklar. Wursthorn geht davon aus, dass diese Themen in Klassenstufe 5 von den Schülern verstanden werden können und weist anhand von Sinn-, Zeit-, Ziel- und Horizontalkriterium nach, dass es sich um fundamentale Ideen der Informatik nach Schwill handelt (→ Inhalt 3: Fundamentale Ideen). Im Anschluss an die einjährige Studie mit einer Schulklasse stellt Wursthorn eine veränderte Einstellung der Schüler zum Rechner fest. Insbesondere haben die Schüler erkannt, dass der Rechner nur Anweisungen durch ein Programm und Benutzereingaben abarbeitet:

> Das Gefühl der Kinder, dass der Computer undurchschaubar ist, hat bei einigen am Ende des Schuljahres abgenommen. [...] Völlig unerklärlich ist die Annahme fast aller Schülerinnen und Schüler, dass der Computer nur Fehler mache, wenn er falsch bedient würde. Mehrfach hatten die Kinder im Unterricht selbst erfahren, dass ihre Logo-Programme ‚Fehler' machten, weil sie von ihnen selbst nicht korrekt erstellt worden waren" (Wursthorn 2006, S. 188).

Sicherlich spielt neben der systematischen Heranführung an informatische Grundkonzepte die Beschäftigung mit dem Rechner über einen langen Zeitraum in vier Schulfächern bei der veränderten Einstellung der Schüler eine große Rolle. Wursthorn schlussfolgert, dass die Schüler nicht zwischen der Anwendung und Programmierung unterscheiden (Wursthorn 2006, S. 188). Für Informatiksysteme und Kompetenzentwicklung ist damit wiederum ein Schwerpunkt auf die Universalität zu legen und den Schülern Möglichkeiten und Grenzen aufzuzeigen (→ Methodik 1: Universalität).

Freischlad stellt 2006 sein Forschungsprojekt „Didaktisches System Internetworking" vor, das eine Kollektion abgestimmter Komponenten des Lehr-Lernprozesses zur fachdidaktischen Kommunikation und Umsetzung im Unterricht zum Ziel hat. Ausgangspunkt sind die Anforderungen, die der zunehmende Einsatz digitaler Medien im Alltag an den mündigen Bürger stellt (Freischlad 2006). Schwerpunkte sind Anwenden und Verstehen des Internets (→ Inhalt 1: Typische Repräsentanten) mit Bezug zu Strukturen, Kommunikationsbeziehungen und Informationssicherheit im

Internet (Freischlad 2007). Hinsichtlich der drei Charakteristika von Informatiksystemen ist festzustellen, dass der Bereich Strukturen im Internet sicherlich der inneren Struktur des Informatiksystem inklusive Abläufen zuzuordnen ist (→ Inhalt 4: Sichten). Protokolle und Architekturen wie Client-Server-Architektur fallen hierunter. Kommunikationsbeziehungen und Informationssicherheit sind dann sowohl auf der Ebene des nach außen sichtbaren Verhaltens als auch auf der Ebene der zugrunde liegenden Strukturen und Wirkprinzipien zu betrachten. Mehr als in anderen Informatiksystemen lassen sich beim Internet die Netzverbindungen thematisieren. Hinsichtlich der Hauptfunktionen von Informatiksystemen birgt das Internet damit unterrichtliche Anknüpfungspunkte in den Bereichen Kooperation, Koordinierung und Lokalität von Information. Andere Hauptfunktionen sind ggf. schwieriger im Unterricht zu verankern (Abschnitt 3.2.2). Freischlad analysiert Fachbücher der Hochschulausbildung und extrahiert einen Katalog von Aufgabenklassen als erste Komponente des didaktischen Systems. Die Planung und mehrfache Erprobung seines Konzeptes im Informatikunterricht der Sekundarstufe II resultieren in Beschreibungen von Wissensstrukturen. Eine umfassende und in weiten Teilen explorative Lernsoftware zur Interaktion mit den Wirkprinzipien vervollständigt als dritte Komponente das didaktische System (→ Medien 2: Interaktion mit Wirkprinzipien). Da das Internet ein allgegenwärtiges, konkretes Informatiksystem ist, wurden einige Synergien zwischen dem didaktischen System Internetworking und einem Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen erzielt (vgl. Schubert et al. 2007). Grund ist, dass das Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung mit Ergebnissen zum didaktischen System Internetworking verglichen und auf Konsistenz bzw. bereichsspezifische Abwandlungen überprüft werden kann.

Stechert präsentiert auf der INFOS 2007 (Schubert 2007) Ergebnisse der Erprobung seines Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung in der Sekundarstufe II (Kapitel 6; (Stechert 2007c)). Eine erstes Konzept für den Einsatz von objektorientierten Entwurfsmustern wurde auf der INFOS 2005 vorgestellt. Da in einigen Entwurfsmustern Potential zur Vernetzung fundamentaler Ideen der Informatik identifiziert wurde (Stechert 2006b), sind sie als Wissensrepräsentation Bestandteil des Unterrichtsmodells (Abschnitt 5.4; (Stechert 2006c)). Grundlage der Strukturierung der Lernphasen sind zu diesem Zeitpunkt das nach außen sichtbare Verhalten, die innere Struktur und ausgewählte Implementierungsaspekte von Informatiksystemen.

Verschiedene konkrete Informatiksysteme sind für Kompetenzentwicklung im Informatikunterricht zu thematisieren. Doebeli Honegger (2007) nutzt Wikis als Informatiksysteme mit Werkzeugfunktion im Sinne eines Lernmittels, aber auch als Lerngegenstand in der Schule. Dazu identifiziert er fundamentale Ideen nach den Kriterien von Schwill (1993a) in Wikis (→ Inhalt 3: Fundamentale Ideen). Obwohl Wikis netzbasiert sind, ist das Thema im Gegensatz zum gesamten Bereich Inter-

networking stärker Lernmittel denn Lerngegenstand im Informatikunterricht. Das Interessante dieses Ansatzes für Kompetenzentwicklung mit Informatiksystemen ist, dass fundamentale Ideen der Informatik zur Begründung des Bildungswertes nicht im Softwareentwicklungsprozess, sondern bei der Anwendung bzw. als Systemeigenschaften identifiziert werden.

Antonitsch (2007) stellt in der gängigen fachdidaktischen Literatur zu Datenbanken zwei Vermittlungskonzepte fest, einmal basierend auf dem Anwendungsaspekt (Werkzeugebene) und einmal auf dem Modellierungsaspekt (konzeptueller und logischer Entwurf gefolgt von Normalisierung und Abfragen). Als Alternative stellt er das Konzept „Erforschen und Anwenden" vor. Zentral darin ist das Strukturieren durch die Schüler, welches Modellieren und Anwenden integriert:

> „‚Vollständiges Strukturieren' ist in diesem Zusammenhang vielmehr als Prozess mit den Phasen ‚Erlernen von Regeln zur Strukturbildung' – ‚Anwenden von Regeln zur Strukturbildung' – ‚Datengewinnung aus erzeugten Strukturen' – ‚Hinterfragen erzeugter Strukturen' zu verstehen" (Antonitsch 2007, S. 231).

Antonitsch erstellt eine tätigkeitsorientierte Sicht auf Anwendungs- und Modellierungsebene von Datenbanken, um diese dann mit konkreten Anwendungsfällen zu untersetzen. Seine Vorgehensweise kann als Vorgehensmodell aufgefasst werden, ähnlich wie es eines für die Anwendungsorientierung gab (Abschnitt 4.2.1). Er erprobte das Konzept mit 13 Schülern in ca. sieben Doppeleinheiten à 100 Minuten (Antonitsch 2007, S. 233f). Es ist zu prüfen, ob und inwieweit das Konzept „Erforschen und Anwenden von Datenbanksystemen" auf Informatiksysteme verallgemeinert werden kann (→ Inhalt 1: Typische Repräsentanten; → Methodik 4: Analyse des Systems – Experimente). Gerade die Kombination eines entdeckenden Ansatzes zur bewussten Anwendung mit dem Erstellen entsprechender Modelle erscheint viel versprechend (vgl. (Thomas 2002), (Voß 2006)). Für Kompetenzentwicklung ist eine aussagekräftige tätigkeitsorientierte Sicht auf Informatiksysteme zu verallgemeinern (vgl. Schubert und Schwill 2004, S. 41), die Anwendungs- und Modellierungsebene anspricht und auch die Diskussion um Konsequenzen einschließt (→ Methodik 7: Von der Anwendung zur Maschine; Abschnitt 5.5).

4.2.5 Informatiksysteme und Kompetenzentwicklung in den Bildungsstandards für die Sekundarstufe I

Der Fachausschuss „Informatische Bildung in Schulen" und die Fachgruppe „Didaktik der Informatik", beide in der GI, erarbeiteten seit 2003 Bildungsstandards für die Informatik in der Sekundarstufe I. Da nur in wenigen Bundesländern Informatik in der Sekundarstufe I verpflichtend angeboten wird, ist es legitim und notwendig, die Bildungsstandards auf ihren Beitrag zur Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II zu analysieren. Die Informatikbildungsstandards beruhen sowohl auf den Mathematikstandards (Carpenter und Gorg 2000)

des US-amerikanischen Mathematiklehrerverbandes National Council of Teachers of Mathematics (NCTM) als auch auf dem Bildungsstandardbegriff, wie er in der Expertise „Zur Entwicklung nationaler Bildungsstandards" (Klieme et al. 2007) verwendet wird (vgl. GI 2008, S. 3). Erstere stellen die Inhalte des schulischen Lernens und letztere die Lernergebnisse in den Vordergrund. Die Kombination dieser gegenläufigen Sichtweisen in den Mindeststandards für die Informatik resultiert in so genannten Kompetenzbeschreibungen, die durch ihre konkreten Inhaltsvorgaben nicht mehr mit dem in der vorliegenden Arbeit verwendeten Kompetenzbegriff (Abschnitt 2.1) nach Weinert (2001) und Klieme et al. (2007) übereinstimmen. Daher werden diese im Folgenden als Bildungsziele referenziert. Während die Wichtigkeit von Informatiksystemen für den Informatikunterricht unbestritten ist, wurde bei der Entwicklung der Bildungsstandards viel über ihre Rolle in den Standards diskutiert:

> „In einer frühen Phase der Diskussion wurde auch ein Themenstrang ‚Informatiksysteme' genannt. Viele der diesem Strang zugeordneten Inhalte beziehen sich auf die praktische Umsetzung eines Aspekts aus einem anderen Inhaltsbereich. Das ist aber ein generelles Ziel des Informatikunterrichts: zwar auch ohne Computer Einsichten zu erlangen, aber diese nicht im Abstrakten zu belassen, sondern am Computer (oder ggf. mit einem anderen Informatiksystem) handelnd praktisch werden zu lassen. Ein solches generelles Ziel sollte aber nicht als Themenstrang, sondern als Prinzip des Unterrichts genannt werden" (Puhlmann 2005, S. 84).

Diese Auffassung hat sich wiederum gewandelt. Aufbau und Funktionsweise von Informatiksystemen zu verstehen, wird sowohl als übergeordnetes Ziel als auch in einem gesonderten Inhaltsbereich betrachtet. So wird als übergeordnetes Ziel formuliert, dass jeder Schüler

> „in die Lage versetzt werden [soll], auf einem der jeweiligen Schulart angemessenen Niveau den grundlegenden Aufbau von ‚Informatiksystemen' und deren Funktionsweise zu verstehen, um damit einerseits deren zielgerichtete Anwendung bei der Lösung von Problemen, aber auch die leichte Erschließung anderer Systeme der gleichen Anwendung zu ermöglichen" (GI 2008, S. 11).

Es wird betont, dass dieses Ziel nicht erreicht werden kann, wenn Informatikunterricht ausschließlich auf der Ebene der Benutzungsschnittstelle verbleibt (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise). In den Bildungsstandards Informatik wird zwischen Inhalts- und Prozessbereichen unterschieden, wie es in Diskussionen während eines Dagstuhlseminars (Magenheim und Schubert 2004) angeregt wurde (vgl. Magenheim 2005). Die Inhaltsbereiche in den Bildungsstandards nennen die inhaltlichen Anforderungen an Schüler am Ende der 7. bzw. 10. Jahrgangsstufe. Sie sind in fünf Bereiche aufgeteilt: Information und Daten, Algorithmen, Sprachen und Automaten, Informatiksysteme sowie Informatik, Mensch und Gesellschaft.
Die Prozessbereiche wiederum sind Modellieren und Implementieren, Begründen und Bewerten, Strukturieren und Vernetzen, Kommunizieren und Kooperieren sowie Darstellen und Interpretieren. Als Zugang zur Erreichung des übergeordneten

Ziels, Aufbau und Funktionsweise von Informatiksystemen zu verstehen (GI 2008, S. 11), wird folgende Reihenfolge der miteinander verzahnten Inhalts- und Prozessbereiche vorgeschlagen:

> „Den Ausgangspunkt für einen produktunabhängigen Zugang bildet daher die ‚Darstellung' bzw. Repräsentation von ‚Information' zu Problemen aus der Lebenswelt der Schülerinnen und Schüler durch ‚Daten' in Informatiksystemen verschiedener Anwendungsklassen. Dabei lernen die Schülerinnen und Schüler auch von Informatiksystemen produzierte Daten im Hinblick auf die darin enthaltene Information zu ‚interpretieren'. Weiterhin erkennen sie, dass Information in festgelegter Art und Weise, unter Verwendung bestimmter ‚Sprachen' dargestellt werden muss, damit ein Informatiksystem diese mittels ‚Automaten' und ‚Algorithmen' verarbeiten kann. Dies ermöglicht ihnen einen intuitiven Zugang zur ‚Modellierung' des grundlegenden Aufbaus und der Funktionsweise von Informatiksystemen und deren exemplarischer ‚Implementierung'. Diese Betrachtung hilft den Schülerinnen und Schülern auch, die prinzipiellen Möglichkeiten und potenziellen Gefahren und Risiken zu erkennen und darauf sachgerecht zu reagieren. Sie erkennen und bewerten damit relevante Zusammenhänge zwischen ‚Informatik, Mensch und Gesellschaft' " (GI 2008, S. 11f).

Kompetenzen bzw. Bildungsziele aus den Prozessbereichen Begründen und Bewerten, Kommunizieren und Kooperieren sowie Strukturieren und Vernetzen sollen dabei von Beginn an einbezogen werden (GI 2008, S. 12). Damit wird die Schwierigkeit der Zuordnung von Unterrichtsinhalten zu den Leitlinien der GI-Empfehlung aus dem Jahr 2000 behoben, insbesondere die Trennung der Leitlinie „Wirkprinzipien von Informatiksystemen" von der „Informatischen Modellierung" (Abschnitt 4.2.4). Die neue Aufteilung in Inhalts- und Prozessbereiche impliziert, dass der Inhaltsschwerpunkt Informatiksysteme bezüglich aller Prozessbereiche zu betrachten ist, wenngleich die oben beschriebene Verzahnung der Bereiche dies zum Teil relativiert:

> „Darüber hinaus – und für den Unterricht von großer Bedeutung – sind aber auch Inhalte und Prozesse aufeinander angewiesen. Die Prozesskompetenzen werden an der Arbeit mit den Inhalten erworben, ohne die Inhalte wären viele von ihnen nicht spezifisch für die Informatik. Umgekehrt stünden die Inhalte ohne Prozesse in der Gefahr, zu einer Wissenssammlung [...] zu verkommen" (GI 2008, S. 45).

Eine Interpretationsschwierigkeit liegt jedoch dadurch vor, dass das übergeordnete Ziel der Bildungsstandards, das Aufbau und Funktionsweise von Informatiksystemen zur zielgerichteten Anwendung und zum leichten Erschließen weiterer Informatiksysteme beinhaltet, fast gleichlautend als Ziel des Inhaltsbereichs Informatiksysteme übernommen wurde. Kritisch zu sehen ist auch die Gleichsetzung von Informatiksystemen mit Alltagsgeräten, die elektronische Komponenten besitzen:

> „Beispiele für Informatiksysteme sind Computer und Handys. Informatiksysteme sind aber auch u. a. in DVD-Rekordern, Waschmaschinen, Autos, Foto und Videokameras enthalten" (GI 2008, S. 37).

Diese Aussage steht in direktem Widerspruch zur Ansicht von Schubert und Schwill, die betonen, dass es bei der informatischen Bildung um die notwendige Organisation des eigenen Denkens geht, um Informatiksysteme bewusst anwenden zu können (Schubert und Schwill 2004, S. 253). Rechner, und mittlerweile Mobiltelefone,

sind frei programmierbar. Die Universalität stellt somit eine kognitive Hürde dar, die mittels informatischer Bildung zu überwinden ist (→ Methodik 1: Universalität; Abschnitt 3.2.1). Bei vielen Alltagsgeräten mit elektronischer Komponente ist die Notwendigkeit informatischer Bildung zur bewussten Anwendung jedoch nicht nachzuvollziehen. Sie sind allenfalls als ein motivierendes Beispiel für bestimmte Aspekte von Informatiksystemen heranzuziehen (→ Ziel 5: Bereitschaften).

Die Bildungsziele zu Informatiksystemen sind strukturiert nach der Hardware, Software und Vernetzung als Bestandteile von Informatiksystemen. In den Jahrgangsstufen 5 bis 7 sollen die Schüler wichtige Bestandteile von Informatiksystemen benennen und Eingabe, Verarbeitung und Ausgabe zuordnen. Lokale und globale Netze sollen sie unterscheiden können sowie Betriebssystem und Anwendungssoftware differenzieren (GI 2008, S. 17). Als weitere Grundlage für Fähigkeiten zum Aufbau von Informatiksystemen sollen die Schüler Daten speichern und Arten der Speicher unterscheiden können. Für das zielgerichtete Anwenden von Informatiksystemen verwenden sie Dateien und verwalten diese in Verzeichnissen, arbeiten mit grafischen Benutzungsoberflächen, bearbeiten Dokumente mit ausgewählten Anwendungen und arbeiten in Netzen (GI 2008, S. 38). Um sich weitere Informatiksysteme erschließen zu können, müssen die Schüler den Grundaufbau von Informatiksystemen wieder erkennen, die in anderen Geräten integriert sind und gleichartige Aufgaben mit unterschiedlichen Programmen der gleichen Anwendungsklasse lösen (→ Inhalt 1: Typische Repräsentanten).

In den Jahrgangsstufen 8 bis 10 sollen die Schüler Kenngrößen von Hardwarebestandteilen kennen sowie Hard- und Software klassifizieren. Für das zielgerichtete Anwenden von Informatiksystemen erweitern sie bestehende Informatiksysteme um Soft- und Hardwarekomponenten, setzen das Betriebssystem für ihre Zwecke ein, kennen unterschiedliche Dateiformate, sind in der Lage problemangemessene Anwendungen auszuwählen und nutzen Internetdienste. Außerdem sollen sie sich selbstständig neue Anwendungen und Informatiksysteme erschließen können (GI 2008, S. 39).

Die genannten Bildungsziele werden in den Standards weiter präzisiert und mit einzelnen Beispielen unterlegt. Bezüglich Informatiksysteme offenbart sich bei der Analyse des Inhaltsbereichs der Informatiksysteme jedoch auch ein Mangel an Konkretisierung. So wird eine Systematik zu Informatiksystemen angegeben, die Hardware, Software und Anwendungssoftware auflistet und jeweils potentielle Unterkategorien nennt. Dennoch fehlt eine Angabe weiterer möglicher Anwendersoftwareklassen, die im Informatikunterricht betrachtenswert sind, und eine Angabe von Gründen für die Auswahl der angegebenen Repräsentanten Textverarbeitung, Tabellenkalkulation, Datenbanken, Multimedia, Spiele und Kommunikation. Als Unterscheidungsmerkmal zwischen den Anwendungsprogrammen werden beispielsweise unterschiedliche Dateiformate angegeben, Verhalten und Funktionen werden aber nur implizit als Unterscheidungskriterien genannt:

> „Zur Auswahl eines problemadäquaten Anwenderprogramms gehört auch die grundlegende Überlegung, ob für die Lösung des Problems ein Informatiksystem überhaupt erforderlich ist. Im nächsten Schritt erfolgt die korrekte Auswahl der Anwendung. So werden zum Beispiel Texte mit einem Textverarbeitungssystem geschrieben, Berechnungen in der Tabellenkalkulation durchgeführt oder Daten in Datenbanksystemen gesammelt und verwaltet" (GI 2008, S. 40).

In der Beschreibung der Prozessbereiche wird immer auch ein Bezug zu dem Inhaltsbereich „Informatiksysteme" hergestellt. In dem Prozessbereich „Modellieren und Implementieren" wird das Modellieren durch die vier Teilschritte Problemanalyse, Modellbildung zur zweckmäßigen Problemlösung mit einem Informatiksystem, Implementierung und Modellkritik charakterisiert. In den Jahrgangsstufen 5 bis 7 sollen Schüler Informatiksysteme unter dem Aspekt der zugrunde liegenden Modellierung betrachten und Objekte mit ihren Attributen in Informatiksystemen identifizieren, um die Zweckmäßigkeit der Modellierung zu prüfen (vgl. GI 2008, S. 19). Außerdem sollen Schüler bereits implementierte Systeme mit geeigneten Werkzeugen analysieren. Eine Konkretisierung dieser Forderung bleibt jedoch aus, und es ist unklar, inwieweit Quelltextanalyse oder der Einsatz von Softwareentwicklungswerkzeugen an dieser Stelle gemeint sind. Genannt werden objektorientierte Modelle und deren Darstellung mit Klassendiagrammen, Datenmodelle und Zustandsdiagramme. Es wird allerdings betont, dass es auf dieser Jahrgangsstufe kaum möglich ist, Modelle selbst zu implementieren. Letzteres findet vermehrt in den Jahrgangsstufen 8 bis 10 statt. Daraus folgt für Informatiksysteme und Kompetenzentwicklung, dass Schüler erste bereits implementierte Systeme modifizieren sollten (→ Methodik 6: Modifikation statt Entwicklung; Abschnitt 5.5).

Der Prozessbereich „Begründen und Bewerten" setzt bezüglich des Einsatzes von Informatiksystemen die Kenntnis informatischer Sachverhalte in den Mittelpunkt, damit die Vorgehensweisen bei der Nutzung eines Systems zur Problemlösung nicht nur intuitiv oder spielerisch sind. In Jahrgangsstufe 8 bis 10 sollen die Schüler Kriterien zur Auswahl von Informatiksystemen für die Problemlösung anwenden, bewerten und Informatiksysteme verantwortlich nutzen (GI 2008, S. 48f). Daraus folgt, dass es einer systematischen Vorgehensweise zur Erkundung implementierter Systeme bedarf (→ Methodik 4: Analyse des Systems – Experimente; Abschnitt 5.5.1).

In dem Prozessbereich „Strukturieren und Vernetzen" werden Methoden zur Darstellung und Umsetzung von Systemen thematisiert, z. B. hierarchische Strukturen, um Aufbau und Funktionsweise von Anwendungen zu erfassen (GI 2008, S. 51). Dabei kommen Analogien und der Übertragung von Lösungswegen auf andere Bereiche Schlüsselrollen zu (GI 2008, S. 51). Die Nähe dieses Prozessbereichs zur fundamentalen Idee der strukturierten Zerlegung nach Schwill ist offenkundig. Daraus folgt für Informatiksysteme und Kompetenzentwicklung, dass Strukturmodelle zur Lösung von Teilproblemen in Informatiksystemen zu thematisieren sind, die in un-

terschiedlichen Systemen eingesetzt werden können (→ Inhalt 2: Strukturmodelle; Abschnitt 5.4).

Weiterer Prozessbereich ist „Kommunizieren und Kooperieren". In diesem werden Informatiksysteme hauptsächlich als Medium eingesetzt, sowohl als Mittel zur Präsentation eigener Ergebnisse als auch als Werkzeug zum kooperativen Arbeiten und zum Kommunizieren. Dabei nutzen die Schüler in den Jahrgangsstufen 8 bis 10 synchrone und asynchrone Kommunikationsmöglichkeiten und diskutieren auftretende Konflikte und Probleme. Hieraus folgt, dass die didaktische Doppelfunktion von Informatiksystemen hinsichtlich Lerngegenstand und -medium für die Kompetenzentwicklung zu thematisieren ist (→ Methodik 8: Medium und Gegenstand).

Zuletzt wird der Prozessbereich „Darstellen und Interpretieren" beschrieben. Darin geht es vornehmlich darum, unterschiedliche Darstellungsformen komplexer Sachverhalte, also ggf. auch von Informatiksystemen, kennen zu lernen und Informatiksysteme als Werkzeuge zur Erstellung solcher Darstellungen einzusetzen. Somit ergeben sich nach Auffassung des Autors mit Blick auf Informatiksysteme viele Überschneidungen mit dem Prozessbereich „Modellieren und Implementieren", vor allem in den Jahrgangsstufen 5 bis 7, in denen Implementierungen von Informatiksystemen kaum vorgenommen werden. Weitere Überschneidungen mit bzw. Anknüpfungspunkte zu anderen Inhaltsbereichen existieren ebenfalls. So sollen die Schüler im Inhaltsbereich „Sprachen und Automaten" formale Sprachen zur Interaktion mit Informatiksystemen nutzen, was auch das Interpretieren von Fehlermeldungen sowie das EVA-Prinzip als grundlegendes Arbeitsprinzip von Informatiksystemen (→ Methodik 3: Verbindung von Verhalten und Struktur) einschließt (vgl. GI 2008, S. 16).

Im Inhaltsbereich „Informatik, Mensch und Gesellschaft" werden als Bildungsziele das Benennen von Wechselwirkungen zwischen Informatiksystemen und ihrer gesellschaftlichen Einbettung, das Wahrnehmen von Entscheidungsfreiheiten im bewussten Anwenden von Informatiksystemen und das Handeln in Übereinstimmung mit gesellschaftlichen Normen sowie das angemessene Reagieren auf Sicherheitsrisiken bei der Nutzung hervorgehoben (vgl. GI 2008, S. 13).

Zusammenfassend liegt den Bildungsstandards für die Sekundarstufe I eine feingranularere Strukturierung durch Inhalts- und Prozessbereiche zugrunde als der GI-Empfehlung aus dem Jahr 2000, die sich auf vier Leitlinien berief. Wenngleich die inhaltliche Zuordnung weiterhin nicht immer eindeutig ist, so ist sie durch die beschriebenen Anknüpfungspunkte und die entsprechende Verzahnung der Bereiche nachvollziehbar. Bezüglich Informatiksysteme und Kompetenzentwicklung ist fest zu halten, dass Aufbau und Funktionsweise von Informatiksystemen ebenso thematisiert werden wie die Notwendigkeit, Fähigkeiten, Fertigkeiten und Bereitschaften zur Anwendung einer Informatiksystemklasse auf eine andere zu übertragen. An dieser Stelle jedoch fehlt nach Ansicht des Autors eine Betrachtung des Verhaltens

von unterschiedlichen Informatiksystemklassen hinsichtlich ihrer Gemeinsamkeiten und Differenzen. Für Kompetenzentwicklung sind daher typische Repräsentanten von Informatiksystemen von Schülern zu untersuchen und mit geeigneten Strukturmodellen zu erklären, um sie vergleichen und klassifizieren zu können. Außerdem sollen laut Bildungsstandards Schüler in den Jahrgangsstufen 5 bis 7 eine Zuordnung von Systemkomponenten zu Eingabe, Verarbeitung und Ausgabe vornehmen, aber die Vertiefung in den Jahrgangsstufen 8 bis 10 geschieht eher auf Ebene von Klassifikationskennzahlen, denn hinsichtlich einer weiteren Vernetzung von Komponenten und Funktionsweise. Es fehlt weiterhin sowohl eine unterrichtspraktische Erprobung bezüglich der in den Bildungsstandards angegeben Bildungszielen zu Informatiksystemen sowie ein empirisch fundiertes Kompetenzmodell (vgl. Klieme et al. 2007, S. 38f), denn die zwei angegebenen Niveaustufen erlauben nur wenig Differenzierung in der Einschätzung der Schülerleistung. Eine Unterscheidung zwischen dem nach außen sichtbaren Verhalten, der inneren Struktur und ausgewählten Implementierungsaspekte wird in den Mindeststandards nicht vorgenommen. Die Zuordnung von Komponenten eines Informatiksystems zu Eingabe, Verarbeitung und Ausgabe stellt einen Schritt in diese Richtung dar, gerade die Architektur mit den Beziehungen zwischen Komponenten wird dadurch jedoch zu wenig betont.

4.3 Analyse des internationalen Forschungsstands zu Informatiksystemen und Kompetenzentwicklung in der Schulinformatik

4.3.1 Internationale Ausgangslage

Für Kompetenzentwicklung mit Informatiksystemen wird im Folgenden die internationale Diskussion historisch nachgezeichnet. Wie am Anfang des Kapitels begründet, ist Ershovs Vortrag „Programming, the second literacy" auf der 3. IFIP World Conference on Computers in Education (WCCE) in Lausanne aufgrund seines Einflusses auf die nachfolgende Forschung als Startpunkt gewählt. Ershov impliziert mit dem Begriff „programming literacy", dass die Informatik eine neue Kulturtechnik ist. Er sieht in der Programmierung den einzigen Weg, mit einem Rechner zu kommunizieren und ihn als Werkzeug für den mündigen Menschen zu nutzen. Um seiner Forderung nach einer verpflichtenden informatischen Bildung zur Entmystifizierung von Rechnern Nachdruck zu verleihen, zieht er deren volkswirtschaftliche Bedeutung heran:

> „In two generations communicating with a computer will become the concern of practically every person involved in social production. [...] every man has learned to count and to write and secretaries have acquired new masters and new obligations. The same is bound to happen with programming: [...] ordinary programming will be mastered by everyone. This is precisely what I call the second literacy" (Ershov 1981, S. 12ff).

Durch den Wandel der Steuerung von Informatiksystemen seit 1981 von Programmbefehlen zur Menüsteuerung ist die Schlussfolgerung, dass ein mündiger Bürger heutzutage nicht programmieren können, sondern Basiskompetenzen besitzen muss, um mit Informatiksystemen produktiv und zielgerichtet arbeiten zu können.

Im Folgenden werden die internationalen Curricula der UNESCO vor allem hinsichtlich beschriebener Bildungsziele und Empfehlungen zu Unterrichtsinhalten betrachtet. Dazu kommen die Curricula der ACM zur Schulinformatik, da sie als US-amerikanische Informatikfachgesellschaft einen starken internationalen Einfluss hat. Anschließend sind die Weltkonferenzen zum Einsatz von Rechnern in der Bildung (WCCE) im Fokus, die auf Forschungs- und Erfahrungsberichte zur Förderung der Kompetenzentwicklung mit Informatiksystemen untersucht werden. Dazu kommen die seit 2005 stattfindenden ISSEP-Konferenzen (Informatics in Secondary Schools – Evaluation and Perspectives) sowie eine alle zehn Jahre stattfindenden Konferenz des Informatikweltverbandes IFIP zur Verknüpfung von Informatik und Mathematik in der Bildung, die für Kompetenzentwicklung mit Informatiksystemen viele Anregungen bot. Die Habilitationsschrift von Eberle wird aufgrund ihres Einflusses im deutschsprachigen Raum hinzugezogen, um Hinweise zur Lehr-Lernmethodik zu extrahieren. Nicht untersucht werden die international viel beachteten ACM-Konferenzen zu „Computer Science Education Research", namentlich SIGCSE (Special Interest Group on Computer Science Education: Technical Symposium on Computer Science Education; seit 1970), ITiCSE (Innovation and Technology in Computer Science Education; seit 1996) und ICER (International Computing Education Research Workshop; seit 2005), da in ihnen fast ausschließlich hochschuldidaktische Fragestellungen zur Informatik thematisiert werden. Endpunkt des Untersuchungszeitraumes ist wiederum das Jahr 2007.

4.3.2 Internationale Informatikcurricula

ACM Model High School Computer Science Curriculum

1993 wurde das „ACM Model High School Computer Science Curriculum" veröffentlicht, das eine ACM-Arbeitsgruppe bestehend aus Lehrern, Hochschulinformatikern und Administratoren verfasst hatte. Es beschreibt einen einjährigen, für alle Schüler verpflichtenden Kurs, der an der Schnittstelle zwischen den Sekundarstufen I und II angesiedelt ist und informatische Bildung erstmalig viel präziser als die Forderungen von 1981 beschreibt. Es werden fünf Pflichtbereiche genannt: Algorithmen, Programmiersprachen, Betriebssysteme, Rechnerarchitektur sowie sozialer, ethischer und professioneller Kontext. Außerdem werden die zwei optionalen Bereiche Anwendungen (z. B. Datenbanken, Computeralgebrasysteme) und Additivum (z. B. Softwaretechnik, künstliche Intelligenz) beschrieben. Von den optionalen Bereichen soll mindestens einer in den Unterricht integriert werden.

Für Kompetenzentwicklung mit Informatiksystemen relevant erscheint mit Blick auf die jeweiligen Inhalte vor allem der Bereich 3 zu Betriebssystemen als einer Informatiksystemklasse (Tabelle 4.3). Neben dem Kernthema Telekommunikation mittels Rechnernetzen ist ein empfohlenes Ziel darin das Arbeiten mit komplexen Systemen (working with large complex systems (ACM 1993, S. 4)), bei dem auch die Mensch-Maschine-Interaktion betrachtet wird:

> „They [students; Anm. d. V.] will recognize types of user interfaces and command languages viewed as a set of directives, hierarchical organization of information in records, files, directories, and the connection with other systems via networks [...] from the users point of view. [...] Students will recognize the recurring theme, managing complexity by using different levels of abstraction and information hiding" (ACM 1993, S. 7f).

Damit steht die Komplexitätsbewältigung durch Abstraktion im Vordergrund (→ Ziel 2: Arbeiten mit komplexen Systemen). Bemerkenswert ist, dass ein Zugang über die Perspektive des Anwenders gewählt wird. Im Schwerpunkt Programmiersprachen werden unterschiedliche Abstraktionsebenen von Programmiersprachen thematisiert, aber auch theoretische Maschinenmodelle (theoretical machines) (ACM 1993, S. 4). Zum Kern des Rechnerarchitekturmoduls gehört ein Grundmodell eines Von-Neumann-Rechners mit Hauptprozessor (Central Processing Unit (CPU)), Speicher und Ein- / Ausgabe. Empfohlen sind Gatter und Schaltnetze sowie Von-Neumann-Prinzipien. Datenbanken werden für das Anwendungsmodul exemplarisch genannt. Als Mittel zur Abstraktion und der damit verbundenen Komplexitätsbewältigung wird damit Strukturmodellen von Informatiksystemen eine Schlüsselrolle zugewiesen (→ Inhalt 2: Strukturmodelle; Abschnitt 3.2.3).

Tabelle 4.3: Vernetzung und das Arbeiten mit komplexen Systemen als Teil des Betriebssystemmoduls im ACM Model High School Computer Science Curriculum von 1993 (ACM 1993, S. 4)

Core topics	Recommended topics	Optional topics
3. Operating Systems and User Support		
Command languages and its use	Human-Computer interaction	Communication network implementation (e.g., graphs, protocols)
Files and disk management	*Working with large complex systems*	*Memory management and virtual memory*
Telecommunications, local and wide area networks		Operating system functions (e.g. task scheduling, interrupts, buffered I/O)
		Single and multiuser machines

Durch dieses Curriculum werden einerseits (implementierungs-) technische Details vermieden und andererseits ein wissenschaftspropädeutischer Unterricht beschrie-

ben (ACM 1993, S. 1). Dabei wird in Anlehnung an die Naturwissenschaften der Experimentbegriff herangezogen:

> „The study of computer science is composed of basic universal concepts that transcend the technology [...] students will conduct experiments and write programs that demonstrate the abstract concepts, confirm the theory and demonstrate the power of computers" (ACM 1993, S. 1).

Damit wird der Wert langfristig gültiger Informatikkonzepte betont, die gleichzeitig Potential und Grenzen von Rechnern erklären. Zur Erläuterung wird beschrieben, wie Schüler konkrete Software durch Experimente zu analysieren lernen (→ Methodik 4: Analyse des Systems – Experimente):

> „Laboratories and exercises give students an opportunity to carry out experiments that illustrate topics in a realistic setting and at the same time learn the specifics of the software used" (ACM 1993, S. 3)

Potential hinsichtlich der Förderung der Kompetenzentwicklung mit Informatiksystemen liegt in der expliziten Thematisierung der Vernetzung im Zusammenhang mit dem Betriebssystem und Rechnernetzen. Insbesondere die Betonung von Komplexitätsreduktionsstrategien, die in allen größeren Systemen genutzt werden können, sind hier zu nennen. Fazit ist daher, dass Betriebssysteme und weitere Repräsentanten unterschiedlicher Informatiksysteme im Unterricht zu behandeln sind, um das Anwenden größerer Systeme zu erlernen. Darüber hinaus sind Unterrichtsexperimente eine viel versprechende Vorgehensweise zur Analyse von Informatiksystemen (Abschnitt 5.5). Abschließend ist anzumerken, dass neben der fachwissenschaftlichen auch eine fachdidaktische Fundierung des Curriculums stattfindet: In dem Curriculum werden zu den Inhalten auch unterschiedliche Zugangsmöglichkeiten diskutiert, z. B. „Breadth Approach" und „Applications-based" (ACM 1993, S. 2), sowie ein erprobter Beispielkurs vorgestellt, der geteilt in Theorie- und Praxisphasen mit der Analyse der Komponenten eines Rechners beginnt, zu vielen Themen Programmierbeispiele vorsieht, den Schwerpunkt jedoch auf Anwendungen (→ Methodik 7: Von der Anwendung zur Maschine) legt (vgl. ACM 1993, S. 9ff).

Informatics for Secondary Education – A Curriculum for Schools

Das 1994 von der UNESCO in Zusammenarbeit mit der IFIP erstellte Curriculum „Informatics for Secondary Education – A Curriculum for Schools" sieht die Anwendung von Informatics Technology (IT) als eine neue Kulturtechnik:

> „Understanding IT and mastering the basic skills and concepts of IT are now regarded by many countries as part of the core of education alongside reading and writing. This area of study goes under the all-embracing name of informatics" (UNESCO 1994, Introduction, General Aim).

Kritisch ist die künstliche Trennung von Informatics und Informatics Technology anzumerken, bei der Informatics Technology die Anwendung von Informatik in der

Gesellschaft bezeichnet. Durch sie wird die Forderung nach einer verpflichtenden informatischen Bildung durch Ershov (1981) aufgehoben und die Anwendung in den Mittelpunkt gestellt. Es werden für die allgemeine Bildung drei Hauptziele genannt: „Computer Literacy", Anwendung von Informatiktechnologie als Werkzeug zum Problemlösen in anderen Schulfächern (Application of Informatics Technology Tools in Other Subject Areas) und Anwendung von Informatik zum Problemlösen in anderen Schulfächern (Application of Informatics in Other Subject Areas). Die Hauptziele werden in zwei Niveaustufen unterteilt.

Damit kann in dem Curriculum bezüglich Basiskompetenzen zu Informatiksystemen in erster Linie nur der Bereich „Computer Literacy" betrachtet werden, für den als Ziel die kompetente und intelligente Anwendung von Rechnern (→ Ziel 3: Bewusste Anwendung) im Alltag genannt wird (UNESCO 1994, Introduction, General Aim). Problematisch ist dies vor allem, weil in diesem Bereich eine wissenschaftspropädeutische Ausrichtung kaum erfolgt. Vier weitere Lernziele gehören zur „Computer Literacy":

> „Students should be able to:
> 1. handle the basic hardware and software facilities of a computer system;
> 2. use, control and apply application oriented software tools;
> 3. solve routine problems in an algorithmic form;
> 4. identify the most important social, economical and ethical consequences of IT" (UNESCO 1994, Section 3, The Curriculum Units).

Punkt 3 stellt dabei einen Zugang zur Informatik dar, aber nicht vordergründig zur Kompetenzentwicklung mit Informatiksystemen. Die ersten Punkte hingegen zielen nur auf Bedienung (handle) und Nutzung von Werkzeugen (use). Die Hauptlernziele zu „Computer Literacy" werden mit 26 Beispielunterrichtssequenzen unterlegt und in Kernmodule (Hardware, Systems Software Environment, Computing Trends, Introduction to Using a Computer, Text Processing, Working with a Database, Working with Graphics, Social and Ethical Issues, Choice of Software Tools) und Wahlmodule (Database Design and Use, Spreadsheet Design and Use, Careers in Informatics) unterschieden. Eine genauere Betrachtung der feineren Lernziele wird an dem späteren UNESCO ICT Curriculum (UNESCO 2002) vorgenommen, dass viele Elemente des vorhergehenden Informatics Curriculum übernommen hat. Zu großen Teilen entsprechen die Inhalte und Ziele der „Computer Literacy" damit denen, die die „Association for Teacher Education in Europe (ATEE)" für eine „literacy in information technology" erarbeitet hat (Abschnitt 4.2.1), was durch den Einfluss von Weert begründet ist, der an beiden beteiligt und für UNESCO / IFIP federführend war (Weert 1984).

ICT in Secondary Education – A Curriculum for Schools and Programme of Teacher Development

2002 veröffentlicht die UNESCO das bereits im Jahr 2000 in Zusammenarbeit mit der IFIP erstellte Curriculum „Information and Communication Technology (ICT)

in Education", das eine Weiterentwicklung des „Informatics Curriculum" von 1994 darstellt. Informatik wird darin anhand des „Information Processing System" definiert:

> „[. . .] informatics as the science dealing with the design, realization, evaluation, use, and maintenance of information processing systems; including hardware, software, organizational and human aspects, and the industrial, commercial, governmental and political implications" (UNESCO 2002, S. 12).

Umso wichtiger ist es deshalb anzuführen, dass diese Definition zwar aufgegriffen wird, aber im Wesentlichen zu einer Abgrenzung zwischen Informatik und ICT dient. So verbindet ICT das Spektrum der technologischen Anwendungen, die als Informatics Technology definiert werden, mit Kommunikationstechnologie:

> „Information and communication technology, or ICT, is defined as the combination of informatics technology with other, related technologies, specifically communication technology" (UNESCO 2002, S. 13).

Dementsprechend gibt es gegenüber dem UNESCO Curriculum von 1994 Ergänzungen hinsichtlich der neuen Kommunikationsmöglichkeiten, die die Kompetenzentwicklung mit Informatiksystemen fördern können (→ Medien 3: Kooperation und Kommunikation). Insgesamt werden vier Stufen des Lernens zur Anwendung von ICT angegeben: (1) Entdecken von ICT (discovering ICT tools), (2) Anwenden von ICT (learning how to use ICT tools), (3) Bewerten von Situationen zum Einsatz von ICT (understanding how and when to use ICT tools) und (4) berufliche Spezialisierung (specializing in the use of ICT tools) (UNESCO 2002, S. 17). Dabei ist es bezeichnend, dass von „ICT literacy skills" (UNESCO 2002, z. B. S. 31, S. 33, S. 35), also (Bedien-) Fertigkeiten statt Fähigkeiten oder gar Kompetenzen gesprochen wird.

In der ersten Stufe des Entdeckens von ICT stehen die generellen ICT-Funktionen sowie deren Anwendung im Sinne einer „ICT Literacy" im Vordergrund. Ziel ist Kompetenzentwicklung mit Informatiksystemen inklusive Betriebssystem und dessen Entmystifizierung (→ Ziel 4: Entmystifizierung):

> „Students should understand how computers and the basic operating system work and demonstrate that the computer is under their control. They should be encouraged not to be mystified by computers and should be able to understand that computers are continually being improved and why" (UNESCO 2002, S. 68).

Die Inhalte zur ICT Literacy basieren zum Teil auf dem Internationalen und Europäischen Computer Führerschein (European Computer Driving Licence (ECDL)) (UNESCO 2002, S. 18). Alltägliche ICT Anwendungen sind im Mittelpunkt:

> „This curriculum area covers the use of ICT as encountered in the daily life of many communities. Specific units include basic concepts of ICT, using computers and managing files, word processing, spreadsheets, databases, creating presentations, finding information and communicating with computers, social and ethical issues, and jobs using ICT" (UNESCO 2002, S. 18).

Als konkretes Lernziel wird angegeben, dass die Schüler Komponenten der Hardware wie Zentraleinheit, Eingabe, Ausgabe und Speicher unterscheiden können und um deren Funktion wissen sollen (UNESCO 2002, S. 67). Außerdem sollen sie Peripheriegeräte identifizieren können und deren Funktion kennen, ein lokales Netz in Bezug zum Internet setzen, z. B. anhand von E-Mail, und ein Verständnis für die Funktionalität von Systemsoftware entwickeln. Dadurch können sie System- und Anwendungssoftware unterscheiden und gezielt Rechner anwenden (using the computer), beispielsweise einfache grafische und textuelle Problemstellungen kompetent lösen:

> „students should be able to [...] demonstrate the ability to use a computer competently [...] using simple software" (UNESCO 2002, S. 69).

Einstiegssoftware kann dabei ein einfaches Grafikprogramm oder altersgemäße Lernsoftware sein. Als Unterrichtsmittel werden dafür Modelle der Rechnerkomponenten und Illustrationen ihrer Funktionalität erwartet sowie Systemsoftware zur Demonstration. Methodisch wird für das bewusste Anwenden von Systemsoftware ein vom Lehrer geführtes Vorgehen und bezüglich Anwendungssoftware entdeckendes Lernen gefordert:

> „Student-centred activities, hands-on activities, on a guided basis for the system operation activities, and on a creative, self-exploratory basis for the production activities" (UNESCO 2002, S. 70).

Das ICT Curriculum ist eines der wenigen, das derartige Aussagen zur Methodik trifft. Jedoch offenbart die Forderung nach entdeckendem Lernen zum Umgang mit dem Rechner ohne Angabe eines konkreten Vorgehens eine Lücke, die es zu schließen gilt (→ Methodik 4: Analyse des Systems – Experimente).

Weiteres Lernziel in der ersten Stufe ist, dass die Schüler einfache Datenbanksysteme anwenden können (UNESCO 2002, S. 74). Zur Erreichung des Lernziels wird für den Unterricht ein problemorientiertes Vorgehen skizziert, in dessen Verlauf Schüler Phasen des Problemlösens erkennen, in der Lage sind zu beschreiben, welche Probleme mit Datenbanken gelöst werden können, und wie Informationen gespeichert werden (UNESCO 2002, S. 74). Die Schüler sollen auch in der Lage sein, mit dem Rechner über das Internet auf andere Rechner zuzugreifen und mit anderen Menschen zu kommunizieren. Dazu gehören das Kooperieren, z. B. das Senden und Empfangen von Nachrichten und Dokumenten, Nutzung von Mailing Lists, Newsgroups und Videokonferenzen, außerdem Informationssuche im WWW inklusive Einschätzung der Qualität der Information sowie urheberrechtlicher Aspekte (UNESCO 2002, S. 78f).

Auf der zweiten Stufe des Lernens von ICT stehen das Anwenden und der Transfer des Wissens auf ICT in anderen Disziplinen. Vertiefungen können Modellierung und Simulation, Roboter, Tabellenkalkulation und Datenbankdesign behandeln.

Der Einsatz einfacher Roboter kann genutzt werden, um naturwissenschaftliche Experimente durchzuführen, und beinhaltet den Einsatz von Sensoren als Eingabe (UNESCO 2002, S. 88). Für Informatiksysteme und Kompetenzentwicklung ist die Beschäftigung mit Robotersystemen hilfreich, da Sensoren eine weitere Eingabemöglichkeit darstellen. Darüber hinaus können mehrere mobile Roboter ggf. drahtlos kommunizieren, so dass die Schüler an dieser Stelle den Eindruck von eigenständigen Akteuren bekommen, die miteinander und mit ihrer Umwelt interagieren (→ Methodik 8: Medium und Gegenstand; Abschnitt 3.2.1).

Die dritte Stufe schließt das Bewerten von Situationen ein und die Frage, ob der Einsatz von ICT gerechtfertigt ist und welche Werkzeuge hilfreich sind, um ein reales Problem zu lösen. Erst die vierte Stufe involviert informatische Konzepte zu Programmierung und Entwurf von Informatiksystemen, wird jedoch zur beruflichen Bildung gezählt (UNESCO 2002, S. 19). Neben einfachen Algorithmen steht darin der Softwareentwicklungsprozess im Vordergrund. Das fertige Produkt sollen die Schüler anschließend mittels Testdaten evaluieren:

> „Students should test their programs with given or created test data to determine correctness and limitations [...]" (UNESCO 2002, S. 127).

Darüber hinaus sollen die Schüler die Effizienz und Effektivität eines Informatiksystems ebenso wie Potential und Grenzen des Einsatzes bewerten können (UNESCO 2002, S. 141f). Dabei soll der konkrete Einsatz der Software in einem speziellen Kontext nachgespielt werden, z. B. ein Unternehmen, so dass die Schüler selbst überprüfen können, inwieweit sie in der Lage sind, die Eignung der Software zur Lösung ihres Anwendungsproblems richtig einzuschätzen (→ Ziel 3: Bewusste Anwendung). Dazu soll den Schülern die Dokumentation der Software zur Verfügung gestellt werden (→ Inhalt 4: Sichten).

Zusammenfassend lässt sich feststellen, dass weniger die Informatik, denn Anwenden und Auswirkungen von Informations- und Kommunikationstechnologien Themen des Curriculums sind. In dem ICT Curriculum sind insbesondere die Kooperation (→ Medien 3: Kooperation und Kommunikation), typische Vertreter von Softwaresystemen und durch Roboter die Vernetzung bzw. spezielle Eingabeformen betont (→ Inhalt 1: Typische Repräsentanten). Die Unterscheidung, bei Anwendungssoftware entdeckendes Lernen und bei Systemsoftware lehrergeleitetes Vorgehen einzusetzen, ist durchaus kritisch zu sehen: Anwendungen laden durch ihren speziellen Kontext zwar zum Entdecken ein, allerdings werden viele Anforderungssituationen durch unerwartetes Verhalten auf Betriebssystemebene ausgelöst. Auf das Lernen von Informatikkonzepten, die das Verhalten erklären könnten, wird im ICT Curriculum jedoch weitgehend verzichtet.

A Model Curriculum for K-12 Computer Science

2003 veröffentlicht die ACM das „Model Curriculum for K-12 Computer Science", das aus vier Ebenen (Level) besteht. Level I greift die Fertigkeiten, Fähigkeiten und

grundlegenden Konzepte auf, die für die IT Fluency (NRCCITL 1999) definiert wurden. Ab Level II erfolgt mit dem Hinweis auf die wissenschaftspropädeutischen Ausrichtung eine explizite Abgrenzung von der IT Fluency (ACM 2006, S. 13).

Die in Level I referenzierte „IT Fluency" wird in Abgrenzung zur „Computer Literacy" verwendet, da letztere oft allein mit Fertigkeiten im Umgang mit aktuellen Softwareprodukten an einem Einzelplatzrechner gleichgesetzt wird:

> „People fluent with information technology (FIT persons) are able to express themselves creatively, to reformulate knowledge, and to synthesize new information." (NRCCITL 1999, S. 2)

Dazu umfasst IT Fluency die drei Kompetenzaspekte Fertigkeiten, Konzeptwissen und Fähigkeiten. Konzeptwissen (foundational concepts) beinhaltet:

> „**computer organization**, information systems, networks, digital representation of information, information organization, modeling and abstraction, algorithmic thinking and programming, **universality**, **limitations of information technology**, and societal impact of information technology" (NRCCITL 1999, S. 4; Hervorh. durch den Autor).

Fähigkeiten (intellectual capabilities) umfassen:

> „engage in sustained reasoning, **manage complexity**, test a solution, **manage faulty systems and software**, organize and navigate information structures and evaluate information, collaborate, communicate to other audiences, **expect the unexpected**, anticipate changing technologies, and **think abstractly about IT**" (NRCCITL 1999, S. 4; Hervorh. durch den Autor).

Die Komplexitätsbewältigung greift das Ziel des ACM-Curriculums von 1993 wieder auf (→ Ziel 2: Arbeiten mit komplexen Systemen). Die Fähigkeit, abstrakt über Informatiksysteme zu denken, hilft sicherlich auch, um unerwartetes Verhalten der Systeme zu erklären. Daher sind Abstraktionsmechanismen, z. B. (Struktur-) Modelle von Informatiksystemen und Beschreibungssprachen, in ein Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung zu integrieren (Abschnitt 5.3). Geforderte Fertigkeiten (contemporary skills) sind:

> „set up a personal computer, use basic operating system features, use a word processor and create a document, use a graphics or artwork package to create illustrations, slides, and images, connect a computer to a network, use the Internet to find information and resources, use a computer to communicate with others, use a spreadsheet to model simple processes or financial tables, use a database system to set up and access information, and use instructional materials to learn about new applications or features" (NRCCITL 1999, S. 4).

Durch die Angabe der Kompetenzaspekte wird die „Computer Literacy" weiter entwickelt. Für Kompetenzentwicklung mit Informatiksystemen bilden insbesondere die geforderten Fähigkeiten eine ähnliche Zieldimension.

Besonders hervorzuheben ist im ACM-Curriculum, dass neben den Zielen auch ausführliche Unterrichtsbeispiele angegeben werden. Level I behandelt bezüglich

Informatiksysteme und Kompetenzentwicklung Grundlagen der Informatik. Sie umfassen Basisfähigkeiten und -fertigkeiten zur bewussten Anwendung von Rechnern, die mit einfachem algorithmischem Denken verbunden werden:

> „To live and work successfully in an increasingly information-rich society, K-8 students must learn to use computers effectively and incorporate the idea of algorithmic thinking into their daily problem-solving vocabulary" (ACM 2006, S. 9).

Neben dem algorithmischen Problemlösen, z. B. mit Robotersystemen, ist auch die zielgerichtete Interaktion mit dem Rechner zur Kooperation mit anderen zu thematisieren (vgl. ACM 2006, S. 9). Außerdem verstehen sie grundlegende Konzepte:

> „[. . .] students will: [. . .] Demonstrate an understanding of concepts underlying hardware, software, algorithms, and their practical applications" (ACM 2006, S. 10f).

Für Kompetenzentwicklung mit Informatiksystemen ist relevant, dass Schüler am Ende der Klassenstufe 8 Strategien zur Fehleridentifikation und -behebung bei alltäglichen Soft- und Hardwareproblemen anwenden können. Zusätzlich kennen sie die Auswirkungen sich schnell verändernder Technik auf die Gesellschaft, legen rechtlich und ethisch angemessenes Verhalten an den Tag und können bereichsspezifische Informatiksysteme anwenden. Dazu gehört, dass sie mit Partnern kooperieren und Ergebnisse mit ICT entwickeln und publizieren (→ Medien 3: Kooperation und Kommunikation). Die Unterrichtsbeispiele zu Level I beziehen sich auf grundlegende Informatikkonzepte, die ohne Rechnereinsatz zu behandeln sind.

Level II, das für viele Schüler die letzte verpflichtende Berührung mit Informatik darstellt, zielt darauf, Informatikkonzepte, -methoden und ihre Anwendung in der Wissensgesellschaft zu vermitteln (ACM 2006, S. 7). Für Kompetenzentwicklung mit Informatiksystemen ist es dahingehend relevant, dass Operationen des Rechners, Netze, Mensch-Maschine-Schnittstelle und das Erstellen einfacher Programme thematisiert werden (→ Ziel 1: Aufbau, Vernetzung und Funktionsweise):

> „A major outcome [. . .] is to provide students with general knowledge about computer hardware, software, languages, networks, and their impact in the modern world" (ACM 2006, S. 11).

Dabei ist zu betonen, dass Informatikkonzepte von den Schülern aus Sicht des Anwenders und nicht aus Sicht des Entwicklers kennen gelernt werden sollen (vgl. ACM 2006, S. 12). So soll beispielsweise neben Tastatur und Maus über Roboter und deren Sensoren ein Verständnis für Eingabegeräte geschaffen werden:

> „For instance, the idea that a robot needs a method of acquiring sensory data from its environment draws attention to the general notion of an 'input device' beyond the standard keyboard and mouse. Teaching students about various input devices currently in use should help demystify the general idea of input, and prepare students to be comfortable using devices with which they are not yet familiar" (ACM 2006, S. 12).

Das Zitat zeigt damit einen Weg, typische Eigenschaften von Informatiksystemen anhand unterschiedlicher Ausprägungen zu thematisieren, um Informatiksysteme zu entmystifizieren (→ Ziel 4: Entmystifizierung; → Inhalt 1: Typische Repräsentanten). Bei einer Übertragung der Idee auf das Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen ist dazu eine Klassifikation des Systemverhaltens notwendig, um präziser als die Einteilung in Eingabe, Verarbeitung und Ausgabe auf die Besonderheiten eingehen zu können.

Am Ende des zweiten Levels sollen die Schüler (1) ein konzeptuelles Verständnis von Rechnerarchitektur mit Eingabe, Ausgabe, Speicher, Verarbeitung, Software und Betriebssystem entwickeln sowie (2) grundlegende Schritte des algorithmischen Problemlösens von der Problemstellung über Entwurf und Implementierung zu Test und Verifikation kennen. Für Kompetenzentwicklung mit Informatiksystemen ist die Beziehung zwischen der Architektur sowie Eingabe, Verarbeitung Ausgabe wieder interessant (siehe auch Abschnitt 4.2.5). Obwohl der Entwicklungsprozess und die Algorithmik für Kompetenzentwicklung mit Informatiksystemen nicht im Mittelpunkt stehen, sind Test und Verifikation ein wichtiges Mittel, um das Verhalten von Informatiksystemen zu analysieren (→ Methodik 4: Analyse des Systems – Experimente; Abschnitt 5.5.1). Außerdem sollen die Schüler ein Verständnis für Hierarchisierung und Abstraktion einschließlich höherer Programmiersprachen, Übersetzung, maschinennaher Sprachen und Schaltkreise entwickeln. Ebenso sollen sie ethische Fragestellungen zu Rechnern und Netzen wie Sicherheit und Zuverlässigkeit von Information sowie Auswirkungen auf die Gesellschaft kennen (ACM 2006, S. 12). Die Unterrichtsbeispiele zu Level II sind für Kompetenzentwicklung besonders interessant. Für die Schüleraktivität „Setting up a computer" müssen auf einem Rechner ein neues Betriebssystem und Anwendungen installiert sowie Peripheriegeräte angeschlossen werden. In dem Beispiel „Connections inside and out" stehen die Komponenten innerhalb des Rechners, deren Vernetzung und die Einbindung des Rechners in ein Netz sowie einfache Netzkomponenten im Mittelpunkt (→ Methodik 7: Von der Anwendung zur Maschine).

Zur Konkretisierung der Inhalte werden unterschiedliche Industriekurse mit Zertifizierung angegeben und auf die Inhalte des Curriculums bezogen, z. B. zum „Certified Internet Webmaster":

> „The Foundations level exam requires competency in Internet, Web page authoring, and networking fundamentals. These concepts are introduced in Levels I and II" (ACM 2006, S. 17).

Die Aspekte der Vernetzung, die in den ersten beiden Stufen des Curriculums thematisiert werden, entsprechen in hohem Maße den Anforderungen, die per Definition an Kompetenzentwicklung mit Informatiksystemen gestellt werden. Andererseits entspricht der Inhalt des „Web page authoring" dem Ansatz, informatische Grundlagen aus der Perspektive des Anwenders zu betrachten.

Level III vertieft Wissenschaftspropädeutik und professionelle Aspekte wie ingenieurmäßiges Vorgehen, algorithmisches Problemlösen, Datenstrukturen, Software- und Hardwareentwurf, Netze und gesellschaftliche Auswirkungen (ACM 2006, S. 8). Hinsichtlich der Kompetenzentwicklung mit Informatiksystemen ist es interessant, dass die von Eberle (1996) als wichtig für den Unterricht mit der Maschine erachteten Schichten der Software bzw. Programmiersprachen wieder aufgegriffen werden (Abschnitt 4.3.3), um Eigenschaften von Compilern, Betriebssystemen und Netzen zu erklären (ACM 2006, S. 13). Hinzu kommt eine explizite Betrachtung von Systemen im Zusammenhang mit der Hardware, die ebenfalls auf das Ebenenmodell zurückzuführen ist:

> „Hardware and **systems**: logic, gates and circuits, binary arithmetic, machine and assembly language, operating systems, user interfaces, compilers" (ACM 2006, S. 13; Hervorh. durch den Autor).

In den vorgestellten Beispielaufgaben für Level III wird ausschließlich auf Algorithmik, speziell Rekursion eingegangen, die z. B. mittels Rollenspiel erarbeitet wird. Die Betrachtung des Systems fehlt.

Level IV ermöglicht eine Vertiefung in einem Bereich der Informatik (vgl. ACM 2006, S. 8). Schwerpunkte können Problemlösen durch Algorithmenentwicklung zur Vorbereitung auf ein Studium, aber auch Projektarbeiten sein, die auf grundlegenden Konzepten aufbauend professionelle Fertigkeiten schulen. Für Informatiksysteme und Kompetenzentwicklung ist die im Curriculum vorgesehene Verknüpfung der theoretischen Fundierung mit Anwendungen durch den Blick hinter die Fassade der Software relevant:

> „While some [projects; Anm. d. V.] of the project curriculum may be more skills-based, the skills need to be tied to the 'behind-the-scenes' activities of the software – particularly how each task is implemented in the software (e. g., what is happening when you click 'bold'?). Answering such questions enables students to problem-solve when software does not perform as anticipated." (ACM 2006, S. 16)

Der Blick hinter die Fassade korrespondiert mit dem Ansatz von Voß (2006) zur Betrachtung von Standardsoftware. Allerdings wird im ACM K-12 Curriculum verstärkt auf die Implementierung eingegangen, während bei Voß Reengineering und speziell objektorientierte Modelle in den Vordergrund rücken. Im ACM K-12 Curriculum ist die Vorgehensweise zur Kopplung von Fertigkeiten mit dem Blick hinter die Fassade von Software nicht konkret beschrieben. Einziges Unterrichtsbeispiel ist die Einführung in objektorientierten Entwurf. Dennoch kann der Blick hinter die Fassade eine Grundlage der Kompetenzentwicklung bilden (→ Methodik 4: Analyse des Systems – Experimente). Denn sie stützt die Annahme des Autors, dass grundlegende Konzepte und Modelle der Informatik hilfreich sind, um unerwartetes Verhalten von Informatiksystemen zu verstehen. Abschließend ist festzuhalten, dass das ACM K-12 Curriculum Wissenschaftspropädeutik und damit die Informatik in den Mittelpunkt stellt. So ist nicht die Bedienung von Systemen, sondern Anwenden der informatischen Grundlagen das Ziel.

4.3.3 Informatiksysteme und Kompetenzentwicklung in Eberles Didaktik einer informations- und kommunikationstechnologischen Bildung auf der Sekundarstufe II

Eberle veröffentlicht 1996 seine Habilitationsschrift zur informations- und kommunikationstechnologischen Bildung. Die umfangreiche fachdidaktische Arbeit liefert viele theoretische Begründungen für Ziele und Themen. Im Vergleich zu den betrachteten internationalen Curricula und den international publizierten Erfahrungsberichten und Forschungsergebnissen liegt ihr Wert jedoch vor allem in der Begründung von Lernmethoden. So gibt er „Empfehlungen zum Unterricht mit der Maschine". Danach ist Voraussetzung für ein verständnisvolles Arbeiten mit der Maschine ein Verständnis für Programmschichten und den prinzipiellen Hardwareaufbau, also ein Modell des Rechners inklusive seiner Funktionsweise und Grenzen (→ Inhalt 2: Strukturmodelle). Er schlägt vor, diese Grundlagen im Unterricht zeitlich vor möglichen Anwendungen zu behandeln, um Schüler die Universalität verständlich zu machen:

> „Diese Inhalte sind deshalb – im Prinzip – vor der Einführung in Anwendersoftware zu erarbeiten. Dazu gehören zumindest die Vermittlung eines einfachen Modells des Computers, der Systemsoftware sowie der verschiedenen Programmschichten und, durch Einblick in die Idee der Programmierung, das Verständnis für die grundsätzliche Funktionsweise und die Grenzen des Computers. Das Modell soll den Zusammenhang zwischen Eingaben und den durch diese Eingaben in der Maschine initiierten Prozessen beschreibbar und damit für den Anfänger verstehbar machen" (Eberle 1996, S. 350).

Die leichte Einschränkung „im Prinzip" macht Eberles Dilemma deutlich: Verständnisvolles Arbeiten mit der Maschine erfordert eben in erster Linie das konkrete Informatiksystem, an dem kompetent gearbeitet wird. Andererseits ist nach Meinung von Eberle die Universalität des Rechners Grund für das häufige Scheitern von Bemühungen um ein grundlegendes Verständnis der Maschine (→ Methodik 1: Universalität). Er fordert, die Universalität immer wieder bewusst zu machen und sieht es deshalb als notwendig an, den grundsätzlichen Hardwareaufbau und Programmebenen zu thematisieren (Eberle 1996, S. 343). Hinzu kommt, dass die im Sinne der Universalität erstellten virtuellen Maschinen im Gegensatz zu mechanischen Maschinen intern allein an die Regeln eines abstrakten Systems gebunden sind, z. B. formale Logik oder eine Menge an Prozeduren, und nicht an Regeln der Mechanik. Dadurch sind viele Prozeduren in einem Computer zunächst nicht auf Alltagserfahrungen zurück zu führen und vor allem nicht beobachtbar (Eberle 1996, S. 342ff):

> „Reines Black-Box-Denken reicht bei einer ‚universellen Maschine' nicht aus. Zu gross sind die Input- und Outputmöglichkeiten; dies im Gegensatz zu den Beispielen anderer technischer Geräte wie Auto und Fernsehen, die immer wieder für die Begründung der Black-Box-These herangezogen werden. Die Lernenden müssen als Voraussetzung für das Gefühl von ‚Beherrschung' verstehen, wie Computer grundsätzlich arbeiten" (Eberle 1996, S. 250).

Einen hohen Stellenwert schreibt Eberle der Bildung und fortwährender rascher Korrektur mentaler Modelle zu (→ Methodik 2: Kognitive Modelle). Die Bedeutung der auf dem Bildschirm angezeigten Objekte und Fehlermeldungen soll deshalb konsequent geklärt und einzelnen Objektkategorien sowie Programmebenen zugeordnet werden (→ Methodik 3: Verbindung von Verhalten und Struktur). Mittel dazu sind z. B. Schichtenmodelle (→ Inhalt 2: Strukturmodelle). (Fehler-) Meldungen sind Programmebenen zuzuordnen (Eberle 1996, S. 350). Die Fehlervorwegnahme im Unterricht kann zur Angstprävention dienen, indem bewusst typische und vor allem unterschiedliche Fehler erfahren werden (Eberle 1996, S. 351). Ebenso muss thematisiert werden, warum unterschiedliche Befehle die gleichen Systemeffekte erzielen und wie Benutzungsoberfläche und Navigation mit der „Bedienphilosophie" zusammen hängen. Zur Korrektur der mentalen Modelle der Schüler schlägt er unter Berufung auf Jih und Reeves (1992) vor, Lernende bei der Arbeit mit Anwendungen zu beobachten und einen Lernenden einem anderen das Programm erklären zu lassen. Die Lernenden sollen Verhaltensweisen des Programms voraussagen und beschreiben, wie sie mit dem Programm arbeiten. Dazu sind Modelle und Analogien anzubieten, die die bei der Anwendung ablaufenden Verarbeitungsschritte für den Lernenden transparent machen (Eberle 1996, S. 350). Eberle fordert, jede Aktion bei der Anwendung eines Systems mit voraussagendem Sinn zu füllen:

> „Dazu gehören Intention (Zielsetzung), Selektion (Plan, Entwurf), Aktion (Handeln, Tun) und Evaluation (Reflexion)" (Eberle 1996, S. 351).

Er begründet dies damit, dass Lernende nicht nur die möglichen Systemzustände kennen, sondern diese auch vorhersehen und das dazugehörende Systemverhalten bewerten können müssen (Eberle 1996, S. 348). Gleichzeitig folgt aus der Forderung, dass Versuch-und-Irrtum-Strategien zu vermeiden sind (Abschnitt 5.5), d. h. systematisches Vorgehen ist anzustreben (→ Methodik 4: Analyse des Systems – Experimente).

Eberle bezieht die Einführung in Standardsoftware bzw. Anwendungen in seine Konzeption ein. Er empfiehlt, eine Top-down-Vorgehensweise von den generell mit der Software lösbaren Problemen mit Grundbegriffen und Erläuterungen hin zu konkreten Problemstellungen mit detaillierten, aber möglichst produktunabhängigen Anwendungsbeispielen (→ Methodik 7: Von der Anwendung zur Maschine) (Eberle 1996, S. 334). Für Befehle, Eingaben und strukturelle Eigenschaften sind geeignete Modelle zu verwenden, um mentale Modelle möglichst nachhaltig aufzubauen. Konkrete Beispiele werden hierzu jedoch nicht genannt. Außerdem sollen analog zu Programmiersprachen nur grundlegende semantische und syntaktische Besonderheiten behandelt, diese aber klassifiziert werden, damit Lernende später gelerntes strukturieren können. Darüber hinaus sind Struktogramme und Programmablaufpläne, ähnlich zu ihrem Einsatz in der Algorithmik, geeignet, um Problemlösungsmethoden mit Standardsoftware zu visualisieren (Eberle 1996, S. 340). Dafür muss das Problem jedoch eingegrenzt, zerlegt und exakt formuliert werden.

Eberle empfiehlt des Weiteren, dass für jede Systemklasse eigene Vorgehensweisen zu nutzen sind, die an deren Grundstrukturen orientiert sind, die sich meist über Jahre hinweg nicht ändern, z. B. Zellen in einer Tabellenkalkulation (Eberle 1996, S. 340). Zu strukturellen Elementen ist nach Eberle auch die werkzeugtypische Menüstruktur zu zählen, wenn sie produktunabhängig ist (Eberle 1996, S. 336). Für Kompetenzentwicklung mit Informatiksystemen bedeutet dies, dass vor allem der Anwendungsprozess als Problemlösen erkannt wird, das auf vielfältige informatische Konzepte zurück greift und mit Darstellungsmitteln der Informatik geeignet beschrieben werden kann (→ Methodik 5: Anwendung als Problemlösen). Eine solche Vorgehensweise zur Erkundung von Informatiksystemen wird in Abschnitt 5.5 beschrieben. Da bei der Anwendung von Datenbanken deklaratives Wissen im Vordergrund steht, empfiehlt Eberle für diese spezielle Klasse, über die Thematisierung der Entwicklungsmethodik zusätzlich prozedurale Aspekte in den Lernprozess zu integrieren (Eberle 1996, S. 336).

Zur Auswahl von Unterrichtsinhalten zieht Eberle ein Raster zur Strukturierung von Informationstechnologien heran, dass von Steinbock (1993) aufgestellt wurde (→ Inhalt 1: Typische Repräsentanten). Besonders betont er Rechnerklassen (z. B. Client-Server-Architekturen, Übergang vom Von-Neumann-Rechner zu Parallelrechnern, drahtlose Kommunikation), Benutzungsschnittstelle (z. B. Daten und Funktionsumfang), Kommunikationsinfrastruktur (z. B. Netze und Dienste) und die Softwareklassen Betriebssysteme, Datenbanksysteme, Programmiersprachen sowie Anwendungssoftware (z. B. Administrations-, Büro-, Führungs-, Entwurfs-, Wissensmanagements- und Prozesssteuerungssoftware) (Eberle 1996, S. 68ff).

Fasst man Eberles Ansatz hinsichtlich der Lehr-Lernmethodik zusammen, gibt er insbesondere Empfehlungen zu Standardsoftware und generell zum Unterricht mit dem Rechner. Entscheidend ist, dass sich diese Empfehlungen zu einem großen Teil überschneiden (Abschnitt 5.6.2): Strukturelle Merkmale sollen immer mit geeigneten informatischen Modellen unterlegt werden, damit die Schüler korrekte mentale Modelle ausbilden können. Grundlagen der Hardware erachtet Eberle als notwendig, um die Universalität des Rechners in Abgrenzung zur mechanischen Maschine verstehen zu können. Im Gegensatz zu Eberles Vorschlag, zuerst die technischen Grundlagen zu lernen, impliziert der Kompetenzbegriff, Grundlagen handlungsorientiert anhand der Anwendung zu erlernen. Für Kompetenzentwicklung mit Informatiksystemen besteht damit ein Zielkonflikt, der eine enge Verzahnung von Anwendungssituation und Strukturmodellen des Informatiksystems erfordert (→ Methodik 9: Vernetzung der Unterrichtsinhalte).

4.3.4 Internationale Forschungs- und Erfahrungsberichte

Kurz nach der Veröffentlichung des UNESCO Curriculums von 1994 stellt Weert (1995) auf der 6. WCCE (Tinsley und Weert 1995) Richtungen vor, die zur Integration von Rechnern in die Bildung zukünftig eingeschlagen werden können. Hierbei

wird das Anwenden von Standardsoftware wie Statistiksoftware, programmierbarer Mathematikprogramme (z. B. Maple, Mathematica) oder von Tabellenkalkulation in den Informatikunterricht integriert (→ Inhalt 1: Typische Repräsentanten; → Methodik 7: Von der Anwendung zur Maschine). Aber auch anwendungsorientierter Informatikunterricht mit Datenstrukturen und Modellierung wird als Grundlage gesehen (backbone of the new literacy), allerdings nur als „Informatics for a few" (Weert 1995, S. 8), also nicht im Sinne einer verpflichtenden allgemeinen Bildung. Die Mehrheit der Schüler erhält somit nur die Gelegenheit, Zahlen in einer Anwendung zu manipulieren, ohne die dahinter liegende Datenstruktur zu verstehen. Fehlerhafte Berechnungen, die durch Bereichsüberschreitung bzw. zu kleine Register entstehen, können von den Schülern somit nicht verstanden werden.

Proulx (1995), die zum ACM-Curriculum von 1993 einen modulbasierten Beispielkurs mit Aufgaben entwickelte (ACM 1993, S. 13), verbindet den Ansatz der „Computer Literacy" mit dem des ACM-Curriculums. Insbesondere begründet sie anhand von Beispielen, wie die sechs Bereiche des ACM-Curriculums, Algorithmen, Programmiersprachen, Rechnerarchitektur, Betriebssysteme und Benutzerunterstützung, Anwendungen sowie sozialer, ethischer und beruflicher Kontext (Abschnitt 4.3.2), zum Verstehen der Informationsgesellschaft (Information World) beitragen und deshalb notwendiger Teil der „Computer Literacy" sind. Abschließend fordert Proulx (1995) sorgfältig gestaltete und ggf. rechnergestützte Unterrichtsmaterialien, die dynamischen Prozesse in der Rechnerarchitektur und das Verhalten eines Rechners visualisieren:

> „To study computer architecture students should be able to play with a model which allows them to feed in the input, examine the memory, see the bits going through adder and other logical circuits, see the program which is being executed, explore what happens if some parameters, statements or inputs are changed. The processes happening inside of the computer are very complex and many students have a hard time visualizing or understanding this dynamic behaviour" (Proulx 1995, S. 502).

Visualisierungen der inneren Struktur von Informatiksystemen mit den entsprechenden Prozessen sind somit für die Förderung der Kompetenzentwicklung mit Informatiksystemen in Betracht zu ziehen (→ Medien 1: Visualisierung verborgener Prozesse). Dies kann vor allem in einer Lernsoftware geschehen, aber auch durch ein entsprechendes Berechnungs- oder Rechnermodell, z. B. Registermaschine. Es ist also gerade bei Strukturmodellen eines Rechners immer der Prozess, d. h. die Interaktion zwischen den Systemkomponenten zu betrachten. Dafür sind geeignete Visualisierungen sowie enaktive Modelle zu schaffen, um Modelle direkt manipulieren zu können (Abschnitt 5.6).

Kennewell (1995) stellt Basisfähigkeiten in der Anwendung von Informatiksystemen (IT capability) in Beziehung zu konzeptuell-deklarativem und prozeduralem Wissen, wobei er sich auf Erfahrung als Lehrer im Primar- und Sekundarbereich bezieht. Dafür nutzt er den Begriff des mentalen Modells in Anlehnung an Anderson (1987):

> „Thus a person's mental model of a computer system contains an understanding of what a computer and its software can do, indicates whether it will be useful to them in a particular situation, and guides their actions when using a computer for a task" (Kennewell 1995, S. 421).

Dabei entspricht das deklarative Wissen dem Wissen über das System. Es kann genutzt werden, um Möglichkeiten und Grenzen des Systems zu erklären und Verhalten vorher zu sagen. Das prozedurale Wissen entspricht einem dynamischen Modell, das das Anwenden des Systems ermöglicht, z. B. zur Erklärung des Systemverhaltens in Abhängigkeit vom jeweiligen Zustand und ausgeführter Aktion (→ Methodik 5: Anwendung als Problemlösen). Ähnlich wie Thomas (2003) schlussfolgert Kennewell (1995), dass Analogien hilfreich sind, speziell um Schüler in kognitive Konfliktsituationen und damit zu einer Anpassung ihres mentalen Modells von Informatiksystemen zu bringen (Abschnitt 4.2.4). Rechnereinsatz hält er bei Lernerfolgskontrollen zu den Basisfähigkeiten in der Anwendung von Informatiksystemen für unumgänglich. Fazit im Rahmen des Unterrichtsmodells ist, dass die Unterscheidung zwischen erklärendem Wissen, im Sinne von Fakten und Konzepten, sowie prozeduralem Wissen darüber, wie ein Informatiksystem zielgerichtet angewendet werden kann, zur Förderung der Kompetenzentwicklung mit Informatiksystemen heranzuziehen sind. Eine Verbindung beider Wissensarten kann in Schülerexperimenten vorgenommen werden (→ Methodik 4: Analyse des Systems – Experimente), die Anwendung und erklärende Konzepte aufgreifen (Abschnitt 5.5).

Cassel et al. (1995) stellen einen Grundlagenkurs in einem universitären Bachelor of Arts vor, der sowohl literaturwissenschaftliche als auch informatische Inhalte hat, und Informatiksysteme von den Anwendungen ausgehend aufgreift. Darin verwenden sie einen entdeckenden Ansatz in dem das Ergebnis nicht vorab festgelegt ist (Cassel et al. 1995, S. 674). Eine Erkundung des abstrakten Datentyps (ADT) Baum wird anhand der Verzeichnisstruktur des Betriebssystems vorgenommen:

> „*The ADT Tree:* [...] In the laboratory, the students encountered trees in the context of the DOS [Disk Operating System; Anm. d. V.] directory structure [...]. The root of the tree was the root of the network file system. [...] Further exploration led to discovery of directories containing various classes of software (editors and word processors, mathematics packages, images, databases, even games)" (Cassel et al. 1995, S. 672).

Weitere informatische Inhalte sind formale Grammatiken und Netze bzw. Graphen. Wenngleich direkte Übertragung von der Hochschule auf Schule nur (fach-) didaktisch fundiert geschehen darf, ist der Ansatz aus drei Gründen für Kompetenzentwicklung mit Informatiksystemen inspirierend: Erstens nutzt er informatische Konzepte wie den ADT Baum, so dass Schüler Daten nicht nur manipulieren, sondern auch deren Struktur und entsprechende Konsequenzen verstehen (→ Inhalt 3: Fundamentale Ideen). Zweitens wird das informatische Konzept wie die Verzeichnisstruktur in einem konkreten Informatiksystem mittels entdeckendem Lernen erfahren (→ Methodik 4: Analyse des Systems – Experimente). Drittens können

unterschiedliche Softwareklassen auch auf Ebene der Dateien identifiziert werden, z. B. anhand der Dateiformate (→ Inhalt 1: Typische Repräsentanten).

Thurber und Stratton (1995) stellen die These auf, dass Rechner und das Internet bereits durch Entwurfsentscheidungen stark von der westlichen Kultur geprägt sind. Sie sprechen vom „Western Computer". Eine starke kulturelle Beeinflussung bei der Gestaltung von Rechnern legt nahe, dass ein Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung bei gleicher Technologie nicht ohne weiteres in einen anderen Kulturkreis zu übertragen ist. Die Forschungsrichtung Ethnocomputing (Tedre et al. 2006) unterstützt diese These.

Bezüglich Kompetenzentwicklung mit Informatiksystemen ist die 7. WCCE (Watson und Andersen 2002) geprägt von Beiträgen, die sich weniger auf Informatik, als auf die Arbeiten zur „Information Technology Fluency" (NRCCITL 1999, vgl. Abschnitt 4.3.2) und die Bildungsstandards zu Technologie im Unterricht (National Educational Technology Standards) der „International Society for Technology in Education" (ISTE 1998) beziehen (vgl. (Callegarin und Cortesi 2001), (Aiken und Sandås 2001), (Zammit und Downes 2001)). Zammit und Downes (2001) erklären auf der 7. WCCE, wie Lesekompetenz und „Information Technology Competence" (IT-Kompetenz) einander bedingen und entwickeln die Standards der ISTE (1998) diesbezüglich weiter. Die Rolle der Lesekompetenz ist bei der systematischen Erkundung der Benutzungsoberfläche von Informatiksystemen, aber auch bei der Quellcodeanalyse zu beachten.

Schwerpunkt der 8. WCCE im Jahre 2005 ist „Lifelong Learning" und die Rolle von ICT diesbezüglich (vgl. Weert und Kendall 2005). Dabei bleibt wie auch schon bei der WCCE 2001 der „IT Fluency" Ansatz verbreitet (vgl. Barron et al. 2005).

Magenheim stellt auf der WCCE 2005 eine Vorstufe eines theoretisch begründeten Kompetenzmodells der Informatik zur Diskussion (Magenheim 2005), das explizit Informatiksystemverständnis (Level of System Comprehension) als eine von drei Dimensionen ausweist. Daneben gibt es Anwendung (Level of Application) und Informatiksysteme als Medium (Usage of Media Functions of the Informatics System).

Die Dimension Anwendung reicht von der angeleiteten Nutzung von Grundfunktionen zur kombinierten Nutzung spezieller Funktionen unterschiedlicher Informatiksysteme. Die Dimension Informatiksysteme als Medium umfasst Domänenspezifik, generische Cognitive Tools, Kommunikations- und Kooperationsmedien, Explorations- und Evaluationssysteme sowie professionelle Entwicklungswerkzeuge (→ Inhalt 1: Typische Repräsentanten).

Für die Dimension Informatiksystemverständnis werden aufeinander folgende Verstehensebenen skizziert: Wissen (Knowledge), Wissenstransfer (Transfer of Knowledge), Konstruktion (Complexity of Construction), Evaluation (Assessment).

Die Wissensebene umfasst Grundfunktionen von Systemen und Basiskonzepte der Hardware. Unterschiedliche Sichtweisen auf Systeme wie Algorithmen, Quelltext, Benutzungsoberflächen, Modelle und Protokolle werden als Themenschwerpunkte vorgeschlagen. Außerdem sollen Schüler grundlegende Informatikprinzipien und abstrakte Konzepte als Metamodelle kennen lernen.
Wissenstransfer beinhaltet die Anwendung von ICT-Wissen in einem bekannten Kontext und den Transfer zu einem neuen Kontext.
Auf der Ebene der Konstruktion bedarf es eines Verständnisses von der Konsistenz eines Systems und der Fähigkeit des Re-engineering. Dazu gehört die Nutzung von Information sowie adäquaten Methoden, Konzepten und Theorien der Softwaretechnik, z. B. Entwurfsmuster, zur Gestaltung neuer Systeme.
Die letzte Ebene der Evaluation umfasst die Bewertung von Systementwürfen und von Auswirkungen des Einflusses von Informatiksystemen im sozio-technischen und gesellschaftlichen Rahmen (Magenheim 2005, S. 4). Somit werden Strukturmodelle, Informatikkonzepte- und -methoden sowie unterschiedliche Sichten thematisiert (→ Inhalt 2: Strukturmodelle; → Inhalt 3: Fundamentale Ideen; → Inhalt 4: Sichten).
Die Stufung in diesem Kompetenzmodell ist ähnlich der Lernzieltaxonomie nach Bloom. Darin sind ebenfalls Konstruktion als vorletzte und Evaluation als höchste Stufe der kognitiven Prozesse genannt. Da in der überarbeiteten Lernzieltaxonomie nach Bloom (Anderson und Krathwohl 2001) diese Ebenen getauscht sind, ist auch für dieses Kompetenzmodell eine neue Einschätzung der Schwierigkeitsgrade zu prüfen.

Eine Schwierigkeit inhaltlicher Art liegt in der Kombination von Kompetenzentwicklung mit Informatiksystemen einerseits und Softwareentwicklung andererseits. Sie impliziert, dass höhere Ebenen der Kompetenzentwicklung mit Informatiksystemen nur erreicht werden können, wenn Methoden der Softwaretechnik zur Systemkonstruktion beherrscht werden. Das aber setzt in hohem Maße Programmierung und Kenntnis von Implementierungsdetails voraus, welche wiederum nicht notwendigerweise Teil von Kompetenzentwicklung mit Informatiksystemen sein müssen (Abschnitt 3.2.1). Zusammenfassend ist anzumerken, dass die Dimension Informatiksystemverständnis durch die Nähe zur Softwareentwicklung nicht der in der vorliegenden Arbeit angestrebten Kompetenzentwicklung mit Informatiksystemen entspricht. Es bestehen jedoch Übereinstimmungen mit der Dimension der Anwendung:

> „The dimension '*Level of Application*' measures the complexity of the functions of an informatics system which are used in a given scenario of application. The category '*Guided Use of Selected Basic Functions of the IS*' [Informatics System; Anm. d. V.], for instance, is often realized in a learning software. 'Scenario Adequate Free Choice of Selected Basic Functions of the IS' is mainly linked to cognitive tools. The other categories [...] are representing an increasing complexity of the usage of the system" (Magenheim 2005, S. 5).

Damit eröffnet Magenheim eine Diskussion darüber, inwieweit Fragen der Skalierbarkeit hinsichtlich kleiner und großer bzw. einfacher und komplexer Informatiksysteme für Kompetenzentwicklung zu betrachten sind (Abschnitt 5.6.2). Die Einstufung der Anwendung nach ihrer Komplexität, die u. a. durch die Benutzerführung bestimmt ist, kann genutzt werden, um Informatiksysteme zu klassifizieren, ohne auf ihren Zweck als (alleiniges) Merkmal zurückzugreifen, z. B. Tabellenverarbeitung, Datenbank etc.

Schubert stellt eine Verbindung zwischen den Dimensionen des Magenheim'schen Kompetenzmodells und den Komponenten des Didaktischen Systems her (Schubert 2005). Sie bezieht sich auf Erfahrungen mit dem Didaktischen System für objektorientiertes Modellieren (Brinda und Schubert 2001) in der Sekundarstufe und erkennt eine Lücke zwischen Wissen über kleinste Grundstrukturen von Informatiksystemen, z. B. Attribute und Datenstrukturen, Methoden und Funktionen, Klassen und abstrakte Datentypen sowie Klassenhierarchien, einerseits und Systemverständnis durch Wissen über Systemarchitekturen andererseits. Als Zwischenebene nutzt sie deshalb in Einführungsveranstaltungen zur Informatik an der Universität objektorientierte Entwurfsmuster, die eine Abstraktion eines Subsystems darstellen (→ Inhalt 2: Strukturmodelle). Sie stellt folgende Thesen auf:

> T1: „Virtual machines and their levels and layers are not strong enough connected with the knowledge networks of an introductory informatics course Algorithms and Data Structures.
>
> T2: The application of patterns bridges the gap to higher-level abstraction of subsystems" (Schubert 2005, S. 3).

Dazu fordert sie die Untersuchung der kognitiven Beziehungen zwischen Entwurfsmustern und Schichtenmodellen für Kompetenzentwicklung mit Informatiksystemen. Außerdem formuliert sie die Forschungsfrage, inwieweit Lernende in der Lage sind, Verständnis objektorientierter Systeme auf allgemeine Informatiksysteme konsistent zu transferieren. Für Kompetenzentwicklung mit Informatiksystemen folgen daraus zwei Dinge: (1) Ein Zugang über Entwurfsmuster ist auf seine Tragfähigkeit zu prüfen. (2) Die Rolle der Objektorientierung ist für Kompetenzentwicklung mit Informatiksystemen zu prüfen. Objektorientierung ist ein Mittel für das Programmieren im Großen (Appelrath et al. 2002), so dass zusammen mit Entwurfsmustern ein Beitrag hinsichtlich der Skalierbarkeit eines Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung zu erwarten ist (Abschnitt 5.6.2).

Auf der ISSEP 2005 (Mittermeir 2005) stellen sowohl Ramsky und Rezina (2005) als auch Dreier und Hartmann (2005) vor, wie Informatikunterricht zur Funktionsweise von Suchmaschinen gestaltet werden kann. Ramsky und Rezina stellen dabei die Motivation in den Vordergrund. Sie berufen sich auf Arestova et al. (2000), die beschreiben, dass die Motivation von Internetanwendern vielgestaltig ist: kognitiv, wirtschafts-, kommunikations-, kooperations- oder freizeitbezogen, spiele-

risch, auf Gruppenzugehörigkeit, Selbsterfüllung oder Selbstbehauptung ausgerichtet (→ Ziel 5: Bereitschaften). Die Lehrperson sollte sowohl kognitive Motivation, z. B. Zugang zu Zeitschriften und Bibliothekskatalogen, als auch soziale Motivation erzeugen, z. B., dass Informationssuche eine Hauptaufgabe im Studium sein wird. Anschließend nennen Ramsky und Rezina als inhaltliche Themen unterschiedliche Arten der Informationssuche (Verzeichnisse und Suchmaschinen), die Funktionsweise mit Erfassen der Webseiten, Indizieren und Bewerten sowie die Formulierung von Anfragen mittels Boolscher Operatoren. Interessant für Kompetenzentwicklung mit Informatiksystemen ist, dass ein Programmablaufplan angegeben wird, der die Schritte der Informationssuche beinhaltet. Somit wird die Anwendung als Problemlöseprozess gesehen (→ Methodik 5: Anwendung als Problemlösen), der mittels informatischer Darstellungsmittel visualisiert wird (vgl. Eberle 1996). Er beginnt bei der Analyse der Problemstellung und anschließender Formulierung der Suchanfrage in der Suchmaschine. Falls die Suche nicht erfolgreich ist, muss die Anfrage präzisiert und gegebenenfalls um Boolsche Operatoren ergänzt werden. Falls auch dadurch das Ziel nicht erreicht wird, ist die Anfrage vollständig neu zu generieren. Die Betonung motivationaler Aspekte ist ein Ansatzpunkt, dem Kompetenzbegriff bei der Anwendung von Informatiksystemen gerecht zu werden.

Dreier und Hartmann (2005) stellen die didaktische Suchmaschine Soekia vor. Als Themen nennen sie neben dem Erfassen der Webseiten, der Indizierung und der Bewertung das „Matching", d. h. das Finden der zur Benutzeranfrage passenden Dokumente.

> „Die meisten Leute sind der falschen Ansicht, dass Suchmaschinen bei einer Anfrage das Internet in Echtzeit durchsuchen. Eine klare Vorstellung vom Aufbau und der Funktionsweise eines Indexes könnte hier vielen Missverständnissen vorbeugen" (Dreier und Hartmann 2005, S. 152).

Darüber hinaus betonen sie, dass viele Anwender die Anfragen falsch stellen und z. B. nicht zwischen offenen und geschlossenen Fragen unterscheiden sowie zu wenige bzw. unspezifische Suchbegriffe nutzen (→ Methodik 7: Von der Anwendung zur Maschine).

Ein weiterer Schwerpunkt auf der ISSEP 2005 ist Informatikunterricht zu Standardsoftware. Hodnigg (2005) stellt für die Tabellenkalkulation ein 3−Ebenen-Konzept vor, dass Wertebene, Formelebene mit Datenabhängigkeiten und Datenflussebene beinhaltet. Sie stellt die $n-$dimensionale Datendarstellung in den Vordergrund und identifiziert vier Charakteristika von Tabellenkalkulationen: die tabellarische Benutzungsoberfläche, die durch Zellen sehr lokale Sicht auf Daten, Elemente höherer Programmiersprachen wie Kontrollstrukturen und zuletzt die Tatsache, dass zur erstmaligen Anwendung kaum Vorkenntnisse vorhanden sein müssen. Fazit ist, dass die eingeschränkte Menge an Programmierkonstrukten einerseits und die starke grafische Strukturierung zur Darstellung von Daten für den Informatikunterricht

besonders geeignet sind. Die Berechnung durch die Formeln lassen die Unterscheidung in Eingabe und Ausgabe wichtig werden, während die Verarbeitung wenig transparent für den Anwender stattfindet. Um dem zu begegnen schlägt Hodnigg vor, in dem Berechnungsmodell zu betonen, dass die räumliche Anordnung der Daten und deklarative Aspekte zu verstehen sind. Sie schließt damit, dass deswegen traditionelle Datenflussbetrachtungen nicht angemessen sind. Sie präsentiert ein konzeptuelles Modell, das auf dem Beobachtermuster basiert ((Tao 2000), (Tort und Blondel 2007)). Zellen, deren Inhalt von einer anderen Zelle abhängen, werden bei deren Änderung informiert und aktualisiert (→ Inhalt 2: Strukturmodelle). Steinert nutzt Tabellenkalkulation hingegen für Datenflussmodellierungen in der Sekundarstufe I, indem zu Beginn Elemente eines Datenflussdiagramms auf die Zellen projiziert werden (Schneider 2005, S. 132). Somit wird auf die räumliche Position der Daten eingegangen. Voß (2005a) stellt einen Ansatz aus der beruflichen Bildung vor, in dem Modelle der Objektorientierung eingesetzt werden, um Standardsoftware zu unterrichten (siehe Abschnitt 4.2.4). Borchel et al. (2005) ergänzen in ihrem Artikel „Design of an Informatics System to bridge the Gap Between Using and Understanding in Informatics" den Ansatz von Voß um die Implementierung der objektorientierten Modelle der Standardsoftware. Dazu nutzen sie das Pythonmodul Ponto, das durch die Pythonschnittstelle von OpenOfficeWriter (OpenOffice.org) direkt auf ein Standardsoftwaresystem zugreifen und aus einem objektorientierten Modell ein Dokument erzeugen kann.

Antonitsch (2005) diskutiert, inwiefern Tabellenkalkulationssysteme die Eigenschaften einer „Microworld" aufweisen: (1) Reduktion der Komplexität, (2) Verstecken von Komplexität, (3) Visualisierung von Programmabläufen und Struktur, (4) Einfachheit der Programmierumgebung und (5) die Möglichkeit, Probleme aus der Lebenswelt zu lösen. Dafür schlägt er eine Neuordnung der Kriterien (1) bis (5) vor, die zu den Eigenschaften Skalierbarkeit, Übertragbarkeit und Visualisierung führt (→ Medien 1: Visualisierung verborgener Prozesse). Für die Tabellenkalkulation weist er exemplarisch nach, dass die Kriterien erfüllt sind. Dabei unterscheidet er weiter zwischen struktureller Skalierbarkeit, d. h. die Möglichkeit, eine überwältigende Funktionsfülle gemäß den Vorkenntnissen des Lernenden zu reduzieren bzw. auszublenden. Dazu zählt auch, Aspekte der Problemstellung vorübergehend zu vernachlässigen. Methodische Skalierbarkeit zielt auf die Möglichkeit der Verwendung unterschiedlicher Unterrichtsmethoden, und curriculare Skalierbarkeit auf mehrfachen Einsatz der Software in unterschiedlichen Unterrichtsstunden. Übertragbarkeit ist dann gegeben, wenn das Gelernte in anderen Bereichen wieder aufgegriffen werden kann, z. B. Datenfluss- oder objektorientierte Modellierung, die erst in der Standardsoftware und später in Softwareentwicklungsprojekten eingesetzt wird.

Die ISSEP 2006 (Mittermeir 2006) wurde unter dem Titel „Informatics Education – the Bridge Between Using and Understanding Computers" durchgeführt. Ver-

hoeff stellt dort Ergebnisse aus einem in mehreren Durchgängen erprobten zweieinhalbtägigen Softwaretechnikprojekt mit Schülern der Sekundarstufe II vor, dass einen Schritt zu der von Schubert auf dem WCCE 2005 geforderten Erforschung des Beitrags von Schichtenarchitekturen zur Kompetenzentwicklung mit Informatiksystemen leistet. Zu Beginn des zweitägigen Kurses, einer so genannten „Master Class", sollten die 12 Teilnehmer mit einem vorgegebenen Programm, einem Fahrstuhlsimulationssystem, experimentieren, dass die technischen Gegebenheiten simuliert, d. h. Fahrstuhl, Stockwerke, Türen, Knöpfe und Anzeigedisplay. Daraus sollten die Anforderungen an die Steuerungssoftware abgeleitet werden. Das Thema wurde ausgewählt, da es sich durch geringe algorithmische Komplexität auszeichnet. Somit konnte der Schwerpunkt auf die Softwarearchitektur gelegt werden (→ Inhalt 2: Strukturmodelle). Um diese Anforderungen umzusetzen, wurden Vorteile von Softwarearchitekturen in Kombination mit dem Teile-und-Herrsche-Prinzip bezüglich der Komplexitätsreduktion thematisiert (→ Ziel 2: Arbeiten mit komplexen Systemen; → Inhalt 3: Fundamentale Ideen):

> „The technique of *divide and conquer* must be applied to bridle the complexity. We briefly discuss the notion of *software architecture* as a description of the structure of a software solution (components and their relationsships), and various views of such an architecture (abstract static, abstract dynamic, code files, code execution). The main advantages of software architecture are that it facilitates
> - concurrent development and verification of components in isolation;
> - stepwise integration of components;
> - piecewise improvement by changing one or two components at a time" (Verhoeff 2006, S. 154; Hervorh. im Original).

Vorkenntnisse der Schüler waren Grundkenntnisse in der Programmiersprache Delphi wie globale und lokale Variablen und Prozeduren mit Parameterübergabe. Einige der Teilnehmer hatten keinerlei Programmiererfahrung. Daher wurde die Softwarequalität explizit betont, neben nach außen sichtbaren Anforderungen wie funktionale Korrektheit und Effizienz insbesondere Verifizierbarkeit, Wartbarkeit und Wiederverwendbarkeit. Projektmanagement wurde am Rande thematisiert. Mehrmals war es Aufgabe, kleine Teile in der 3-Schichtenarchitektur zu modifizieren (→ Methodik 6: Modifikation statt Entwicklung). Die Lehrperson gab jedoch im Verlauf des Kurses erweiterte Versionen des Programms mit Musterlösung der einzelnen Komponenten, so dass Programmierung nicht den Schwerpunkt bildete. Am Ende standen systematische Tests und Zustandsbeschreibungen der einzelnen Komponenten. Schrittweise wurden die Komponenten zusammengefügt. Verhoeff betont, dass die Methode der Master Class nur für größere Projekte geeignet ist, da die darin verwendeten Softwaretechnikmethoden andernfalls als unnötig empfunden werden. Nachteil einer solchen Master Class ist der hohe Betreuungsaufwand durch die notwendige intensive Betreuung der Kleingruppen. Dennoch ist für Kompetenzentwicklung mit Informatiksystemen der vorgestellte Ansatz sehr interessant, da explizit auf algorithmische Fragestellungen verzichtet wird. Vielmehr steht die

Zerlegung eines Systems im Vordergrund. Implementierungsaspekte wurden dadurch minimiert, dass Schüler nach einer kurzer Zeit der eigenen Entwurfsarbeit eine Architektur vorgegeben bekamen, so dass durch klar definierte Schnittstellen alle Kleingruppen unterschiedliche Teilaufgaben lösen konnten. Systematisches Testen der Komponenten und des Gesamtsystems wurde vorgenommen.

Alle 10 Jahre richten Arbeitsgruppen aus dem „Technical Committee 3: Education" des Informatikweltverbandes IFIP eine Konferenz zur Verknüpfung von Informatik und Mathematik in der Bildung aus. Auf der „Informatics, Mathematics and ICT – IMICT 2007" (Benzie und Iding 2007) war die Informatik besonders stark repräsentiert (z. B. (Hubwieser 2007b), (Brinda 2007)), so dass die Konferenz in die Betrachtungen zum Forschungsstand zur Kompetenzentwicklung mit Informatiksystemen aufgenommen wird. Bruidegom und Koolen-Wijkstra (2007) nutzen das veränderte niederländische Informatikcurriculum für die Sekundarstufe, in der Hardwareorganisation ein Wahlthema ist. Dafür adaptieren sie eine für die Hochschule entwickelte Simulationsumgebung für digitale Systeme. Sie simuliert das Verhalten eines Schaltkreises pro Zeiteinheit und stellt Werte der Eingaben, Ausgaben, an den Kanten und Speicherzellen des Schaltkreises dar. Schüler sind so in der Lage, mit grundlegenden Schaltungen spielerisch zu interagieren und sich ein einfaches Prozessormodell zu erschließen (→ Medien 1: Visualisierung verborgener Prozesse). Bruidegom und Koolen-Wijkstra heben den Datenpfad und die zeitliche Strukturierung durch Takt und Ereignisse hervor (→ Inhalt 2: Strukturmodelle).

Romeike diskutiert Einflussfaktoren für Kreativität im Informatikunterricht. Interessant für Kompetenzen zu Informatiksystemen sind die damit verbundenen motivationalen Aspekte. Das Verstehen von informatischen Konzepten sieht er dabei als Grundlage der kreativen Anwendung von Informatiksystemen (Romeike 2007, S. 10). Es ist jedoch zu erwähnen, dass Romeike die Softwareentwicklung, die er mit dem Komponieren im Fach Musik vergleicht, als wichtigen Faktor für kreativen Informatikunterricht hervorhebt. Obwohl dieser Bereich in der vorliegenden Arbeit zur Kompetenzentwicklung mit Informatiksystemen bis auf Modifikation existierender Software vernachlässigt wird, zeigt Romeike hierfür einen Anknüpfungspunkt auf:

> „Computer systems function according to precise rules. As they consist of a big amount of pieces the challenge of CS [Computer Science; Anm. d. V.] lies in building and combining them. Many pieces already exist, but to put them together certainly takes creativity" (Romeike 2007, S. 8).

Damit kann das Zusammenspiel der Konzepte in Informatiksystemen und deren Komponenten Ausgangspunkt eines kreativitätsfördernden und damit motivierenden Unterrichtsexperiments sein (→ Ziel 5: Bereitschaften; → Inhalt 2: Strukturmodelle; → Inhalt 3: Fundamentale Ideen).

Micheuz et al. (2007) beschreiben Bezug nehmend auf Hubwieser ein integratives Konzept, dass alle Kategorien der Anwendung von Rechnern in Schulen abdeckt.

Danach werden Rechner in Schulen eingesetzt als Lernmedium, zur Vermittlung von Bedienfertigkeiten für ein konkretes Werkzeug und zur Vermittlung langlebiger Informatikgrundlagen (Hubwieser 2007a, S. 43f). Micheuz et al. (2007) formulieren als Ziel, dass Schüler alltägliche Situationen durch die Anwendung von Rechnern bewältigen können, ein angemessenes Bild von der Rolle des Rechners in der Gesellschaft aufbauen und „kompetente" Anwender werden, die Potential und Grenzen des Rechnereinsatzes bewerten können, um ihn effizient und verantwortungsbewusst einzusetzen. Um später vernetzte Rechner als Werkzeug und Lernmedium anwenden zu können, fordern sie zu Beginn die Aneignung informatischer Konzepte, z. B. fundamentale Ideen der Informatik oder „Threshold Concepts":

> „The students should deepen their concrete knowledge and skills for using software tools and understand the underlying concepts and principles. With these concepts in mind the students are more likely to transfer knowledge in the wide and quickly changing field of computer applications. Finally knowledge about core concepts allows the students to reflect and evaluate the use of computers in a networked society" (Micheuz et al. 2007, S. 4).

Informatische Konzepte werden durch empirische Untersuchungen als „Threshold Concepts" identifiziert. Sie zeichnen sich dadurch aus, dass es einem Lernenden erst durch ein Verständnis dieser Schwellwertkonzepte möglich ist, einen neuen Lernbereich kognitiv zu erschließen. So gilt Objektorientierung als ein solches Konzept in der Informatik (Eckerdal et al. 2006). Es stellt sich die Frage, ob für die bewusste Anwendung von Informatiksystemen ebenfalls „Threshold Concepts" identifiziert werden können (→ Inhalt 3: Fundamentale Ideen; Abschnitt 9.3).

Weigend (2007) sieht die Ausführung von Programmen als Drama, dessen Protagonisten Daten, Namen, Objekte oder Funktionen darstellen, die im Quelltext auftreten. Diese Entitäten sind oft aus der Lebenswelt den Schülern bekannt und eignen sich für Rollenspiele oder können in einer Microworld erfahren werden, z. B. Logo. Damit ist es Schülern möglich, sich die Ausführung eines Programms vorzustellen und zu verstehen. Außerdem stellen Rollenspiele und Microworld unterschiedliche Modelle dar, so dass Sichten kombiniert werden können. Der Ansatz ist für Kompetenzentwicklung mit Informatiksystemen insofern interessant, als dass Objekte im Systemverhalten identifiziert werden können und durch die Interpretation als Drama eine Formalisierung der inneren Struktur stattfindet (→ Methodik 3: Verbindung von Verhalten und Struktur).

Freischlad und Schubert (2007) betrachten die Entwicklung von Aufgabenklassen (vgl. Brinda 2007) für Internetworking. Der Katalog, der aus der Analyse von Fachbüchern entstand, ist durch die Bereiche Anwendungen, Protokolle, Adressierung, Datenübertragung und Architekturen fachsystematisch strukturiert. Für Kompetenzentwicklung mit Informatiksystemen ist zu prüfen, wie die Unterrichtsinhalte hinsichtlich des nach außen sichtbaren Verhaltens und der inneren Struktur einzuordnen sind. Protokolle beispielsweise sind eng mit der Architektur verbunden, da

sie den Ablauf auf unterschiedlichen Ebenen beschreiben (→ Inhalt 2: Strukturmodelle). In die Aufgabenklassen fließt hinsichtlich der Zugänglichkeit auch die Schülertätigkeit einer Aufgabe ein, und zwar Beobachten, Beschreiben, Zuordnen, Entscheiden, Schlussfolgern, Berechnen und Darstellen in Diagrammform. Für Kompetenzentwicklung mit Informatiksystemen ist gerade das aktive Beobachten des Systemverhaltens ein wichtiger Schritt zur Analyse (→ Methodik 3: Verbindung von Verhalten und Struktur). Iding (2007) gibt in diesem Zusammenhang Hinweise, wie die Korrektheit bzw. Glaubwürdigkeit von Information auf Webseiten bewertet und im Unterricht thematisiert werden kann.

Stechert und Schubert (2007) werten die Ergebnisse der ersten Fallstudie zum Unterrichtmodell zu Informatiksystemen und Kompetenzentwicklung aus (Kapitel 6) und beziehen die Ergebnisse der Analyse des Stands der Forschung ein (Abschnitt 4.4). Sie diskutieren den Sichtenwechsel auf Informatiksysteme (vgl. Stechert 2007c) und speziell Schülertätigkeiten zur Verbindung des nach außen sichtbaren Verhaltens von Informatiksystemen mit deren innerer Struktur (Abschnitt 7.2).

4.4 Fazit für Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II

4.4.1 Überblick und Strukturierung der Ergebnisse

In diesem Kapitel wurden der nationale und internationale Stand der Forschung zu Informatiksystemen und Kompetenzentwicklung untersucht. Ziel war, das Erfahrungswissen der Fachdidaktiken und der Schulinformatik zu analysieren, um darauf aufbauend Bildungsanforderungen eines Unterrichtsmodells zur Förderung der Kompetenzentwicklung mit Informatiksystemen anzugeben. Die Ergebnisse liefern Anhaltspunkte für mögliche Kompetenzkomponenten und bilden einen normativ begründeten Kompetenzrahmen, der als Grundlage des Unterrichtsmodells dient (Kapitel 5).

In der nationalen Diskussion (Abschnitt 4.2) wird vornehmlich ab den 1990er Jahren in Artikeln und Bildungsempfehlungen Unterricht zu Wirkprinzipien von Informatiksystemen in den Sekundarstufen I und II gefordert (z. B. (Baumann 1993), (Friedrich et al. 1996), (GI 2000)). Fachdidaktische Konzeptionen wurden für den Informatikunterricht entwickelt, die explizit auch Informatiksystemen einen großen Stellenwert zuschreiben (z. B. (Baumann 1996), (Hubwieser und Broy 1997b), (Hampel et al. 1999), (Humbert 2003)).

Auf der internationalen Ebene sind unterschiedliche Konzeptionen für Informatikunterricht erkennbar (Abschnitt 4.3). Eberle ist mit seiner Konzeption einer Didaktik der Informatik nah an der bundesrepublikanischen Diskussion. Die vorgestellten ACM-Curricula sind sehr stark durch die Fachwissenschaft Informatik

strukturiert, so dass vor allem das Entwerfen und Gestalten von Informatiksystemen im Vordergrund steht. In den von UNESCO und IFIP veröffentlichten Curricula hingegen wird vom alltäglichen Problemlösen mit Anwendungen ausgehend ein Verständnis für ICT geschaffen. Für das in der vorliegenden Arbeit verfolgte Ziel der Kompetenzentwicklung mit Informatiksystemen müssen beide Ansätze miteinander verbunden werden, indem Bildungsziele bezüglich des Anwendens von Informatiksystemen auf Informatikthemen und -methoden bezogen werden. Plattformen für diesen Austausch bilden die Weltkonferenzen WCCE. Insgesamt wird Kompetenzentwicklung mit Informatiksystemen nach Ansicht des Autors jedoch auch dort nur unzureichend thematisiert.

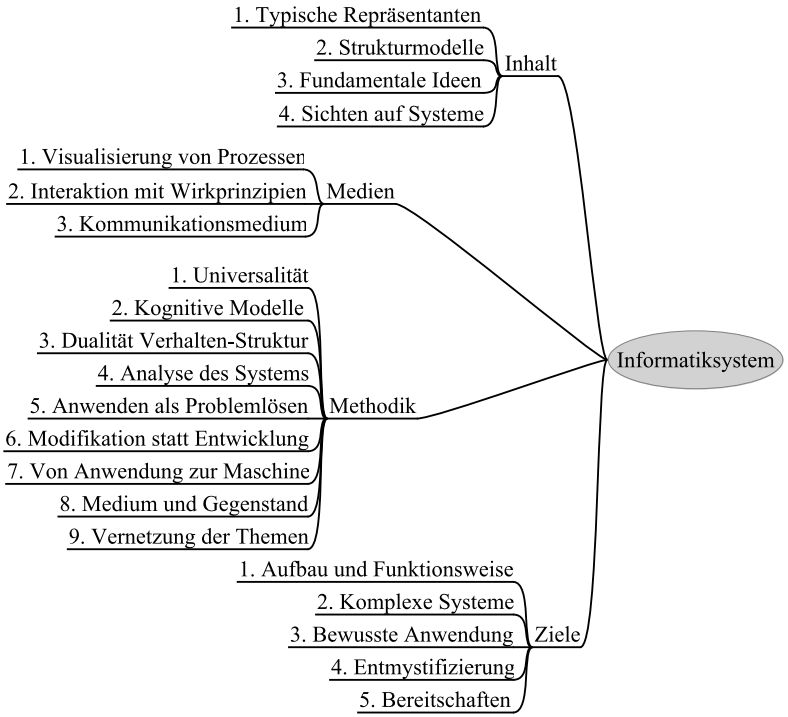


Abbildung 4.1: Die Strukturierung der Ergebnisse der Analyse des Forschungsstandes

Die Ergebnisse aus der Analyse des Forschungsstands lassen sich in die Bereiche (1) Bildungsziele, (2) Unterrichtsinhalte und Lerngegenstand, (3) Lehr-Lernmethodik und (4) Lehr-Lernmedien aufteilen. Aufgrund der Ziel-Inhalts-Mittel-Abhäng-

igkeiten, ist es nicht möglich, sie trennscharf zu formulieren (Meyer 2005, S. 92). Ergebnisse der Analyse werden im Folgenden aufgeführt und exemplarisch mit Referenzen zu fachdidaktischen Publikationen belegt. Strukturiert werden die vier Bereiche durch die in Kapitel 3 erarbeiteten Facetten des Informatiksystembegriffs. Dabei ist zu unterscheiden zwischen der Funktion eines vorliegenden Informatiksystems, seinem nach außen sichtbaren Verhalten, der inneren Struktur und den spezifikationsbedingten Eigenschaften. Abbildung 4.1 zeigt die Strukturierung der Ergebnisse der Analyse des Forschungsstandes.

4.4.2 Bildungsziele für Kompetenzentwicklung mit Informatiksystemen

In den analysierten Bildungsempfehlungen nationaler Fachgesellschaften und in den internationalen Curricula der UNESCO und ACM finden sich übergeordnete Ziele für die allgemeine Bildung in der Sekundarstufe II. Insgesamt lassen sich mehrere Schwerpunkte in der Schulinformatik zur Kompetenzentwicklung mit Informatiksystemen feststellen. Die internationalen Curricula orientieren sich entweder an der Strukturierung der Fachwissenschaft Informatik (ACM 2006) oder aber an einer eingeschränkten Sicht durch Fokussierung auf ICT (UNESCO 2002). Im deutschsprachigen Raum stellt die Veröffentlichung von Bildungsstandards für die Sekundarstufe I durch die GI eine Publikation dar, die einen ersten Schritt in Richtung einer Beschreibung von Kompetenzen zu Informatiksystemen darstellt (GI 2008). Durch den in Kapitel 3 dargestellten Filter, was Informatiksysteme auszeichnet, lassen sich die in Bildungsempfehlungen genannten Ziele folgendermaßen strukturieren.

Bildungsziel 1 (Aufbau, Vernetzung und Funktionsweise) In vielen Publikationen wird als Bildungsziel das Verstehen von Aufbau und Funktionsweise von Informatiksystemen formuliert, um sie zielgerichtet und bewusst zur Lösung von Problemen anwenden und sich weitere Systeme leicht erschließen zu können (GI 2008). Das Verstehen von Rechnern in lokalen Netzen und im Internet wird als Ziel formuliert. Dazu kommt das Wissen um Strukturmodelle von Informatiksystemen (ACM 1993).

Bildungsziel 2 (Arbeiten mit komplexen Systemen) Komplexe Systeme zu überschauen, ist ein Ziele des Informatikunterrichts (Schulz-Zander et al. 1993), (ACM 1993). Dazu sind Formalisierungs- und Abstraktionsmechanismen sowie Metaphern für Kompetenzentwicklung mit Informatiksystemen zu schaffen. Komplexe Systeme sind systematisch zu zerlegen und logische Zusammenhänge in Modellen zu erfassen (KMK 2004).

Bildungsziel 3 (Bewusste Anwendung) Schüler sollen Informatiksysteme als Werkzeug und Lernmedium bewusst anwenden können. Dazu gehören Auswahl geeigneter Anwenderprogramme, Zielgerichtetheit, Angemessenheit der Eingaben und der Reaktion auf Ausgaben des Rechners, planmäßiges Auswählen und Dokumentieren von Ergebnissen (Lehmann 1995). Wissen um Möglichkeiten und Grenzen von Informatiksystemen ist hierfür notwendig. Grundlage dessen ist eine informatische Bildung, aber auch Bedienfertigkeiten werden in einigen Empfehlungen gefordert (UNESCO 2002) und bilden eine Kompetenzfacette (Klieme et al. 2007).

Bildungsziel 4 (Entmystifizierung) Entmystifizierung des Rechners und Abwendung des Gefühls des Ausgeliefertseins sind im Unterricht umzusetzen ((Koerber 1978), (Humbert 2003), (Schubert und Schwill 2004)). Dabei sind die traditionellen Kulturtechniken und Prozesse zu berücksichtigen, die durch Informatiksysteme übernommen werden.

Bildungsziel 5 (Motivationale, volitionale und soziale Bereitschaften) Der Kompetenzbegriff nach Weinert (2001) bzw. Klieme et al. (2007) beinhaltet motivationale, volitionale und soziale Bereitschaften und Fähigkeiten, ein Problem zu lösen. Die Motivation der Schüler, Informatiksysteme für ihre Zwecke einzusetzen, soll gefördert werden (Bosse et al. 1986).

4.4.3 Empfehlungen zu Unterrichtsinhalten und -gegenständen für Kompetenzentwicklung mit Informatiksystemen

Der Softwareentwicklungsprozess wird in vielen Publikationen als wichtige Grundlage des Informatikunterrichts gesehen (z. B. (Schwill 1993a), (Baumann 1996)). Kompetenzentwicklung mit Informatiksystemen wird dann jedoch oft als Vorstufe von Fähigkeiten zur Systemkonstruktion beschrieben (Magenheim 2005). Zur Vermeidung der einseitigen Betrachtung des Softwareentwicklungsprozesses betonen Magenheim und Schulte (2006) die Produkt-Prozess-Relation hinsichtlich soziotechnischer Informatiksysteme. Offen bleibt damit, wie eine Konzeption zu Informatiksystemen und Kompetenzentwicklung aussehen muss, die nicht wesentlich auf Systemgestaltung basiert und dennoch grundlegende Informatikkompetenzen umfasst. Zur Auswahl der Bildungsinhalte für Kompetenzentwicklung mit Informatiksystemen ist das fachdidaktische Konzept der fundamentalen Ideen der Informatik (Schwill 1993a) vielversprechend. Jedoch müssen diese fundamentalen Ideen für Kompetenzentwicklung mit Informatiksystemen anhand der vorgegebenen Kriterien nicht im Softwareentwicklungsprozess, sondern im Anwendungsprozess identifiziert werden. Die allgemeinen Bildungsanforderungen zu Wirkprinzipien von Informatiksystemen sind nur realisierbar, wenn eine Konkretisierung der Bildungsergebnisse und des Bildungszugangs in Form von Lernzielen erfolgt. Hinsichtlich der

Unterrichtsinhalte und -gegenstände für Kompetenzentwicklung mit Informatiksystemen zeichnen sich im fachdidaktischen Forschungsstand nachfolgende Schwerpunkte ab.

Unterrichtsinhalt 1 (Typische Repräsentanten) Die Thematisierung unterschiedlicher Hauptfunktionen (Denning 2007) führt zum Einsatz typischer Informatiksysteme im Unterricht. In der fachdidaktischen Literatur umfasst dies Standardsoftware wie z. B. Tabellenkalkulation und Textverarbeitung, Datenbanken und Betriebssysteme. Dabei ist zwischen Anwendungssoftware und Systemsoftware zu unterscheiden (GI 2008). Aber auch die Unterscheidung hinsichtlich des Einsatzes in Produktions- und administrativen Prozessen wird vorgenommen (Weert 1993).

Unterrichtsinhalt 2 (Strukturmodelle der Informatik) Ausgewählte Strukturmodelle werden als wichtig angesehen, um ein korrektes Bild von Informatiksystemen zu vermitteln. Dazu gehören Von-Neumann-Rechner inklusive Maschinenmodell mit Zentraleinheit, außerdem Schichtenmodelle und Entwurfsmuster. Letztere stellen eine Ebene zwischen Datenstrukturen und Algorithmen einerseits sowie Architekturmodellen andererseits dar (vgl. Schubert 2005). Das Prozesskonzept und die Unterscheidung von realer, virtueller und abstrakter Maschine schließen sich an (Schubert und Schwill 2004). Damit wird auch die Wichtigkeit theoretischer Maschinenmodelle (Humbert 2003) bzw. formaler Systeme (Hartmann und Nievergelt 2002) betont. Protokolle formalisieren die Kommunikation zwischen den Systemkomponenten (Freischlad und Schubert 2007). Um Objektmodelle oder Datenflussmodelle zur Kompetenzentwicklung mit Informatiksystemen einzusetzen, werden unterschiedliche Beschreibungssprachen vorgeschlagen. Die UML dominiert in letzter Zeit diese Diskussion, zeichnet sich aber durch eine Vielzahl von Diagrammarten aus, die nach Verhaltensdiagrammen zur Beschreibung von Prozessen innerhalb des Systems sowie in Wechselwirkung mit der Umwelt und Strukturdiagrammen klassifiziert werden. In der fachdidaktischen Diskussion werden u. a. Klassen-, Objekt-, Zustands- und Sequenzdiagramme genannt sowie Struktogramme (vgl. auch Abschnitt 6.4.5).

Unterrichtsinhalt 3 (Informatikkonzepte: Fundamentale Ideen) Für Kompetenzentwicklung mit Informatiksystemen ist eine Auswahl an Unterrichtsinhalten zu treffen bzw. sind Informatikkonzepte im Einzelfall hinsichtlich ihres Beitrags zu begründen. Für den Informatikunterricht ist die fachdidaktische Diskussion zur Auswahl von Unterrichtsinhalten zurzeit von den fundamentalen Ideen der Informatik (Schwill 1993a) dominiert. Kompetenzen in der Definition von Weinert (2001) bzw. Klieme et al. (2007) beinhalten, dass Problemlösungen in variablen Situationen erfolgreich eingesetzt werden können. Fundamentale Ideen der Informatik

unterstützen den nichtspezifischen Transfer, d. h. unbekannte Varianten einer Problemstellung können als Spezialfälle eines vertrauten Grundkonzeptes erkannt und Lösungsmethoden angepasst werden (Schubert und Schwill 2004), weshalb die Kriterien für fundamentale Ideen in der vorliegenden Arbeit zur Auswahl von Unterrichtsinhalten gewählt wurden. Herausforderung ist, wie die fundamentalen Ideen der Informatik im Unterricht für Kompetenzentwicklung mit Informatiksystemen sinnvoll miteinander vernetzt werden können, wenn auf das Durchlaufen des Softwareentwicklungsprozesses bewusst verzichtet wird (Stechert 2006c).

Unterrichtsinhalt 4 (Sichten auf Informatiksysteme) Informatiksysteme zeichnen sich durch ein nach außen sichtbares Verhalten, eine innere Struktur und Implementierungsaspekte aus. Sie sind eingebettet in ihre Umwelt und speziell in gesellschaftliche Prozesse. Dadurch ist ihr Kontext, d. h. die konkrete Anwendungssituation für Kompetenzentwicklung mit Informatiksystemen relevant. Informatiksysteme sind daher aus unterschiedlichen Perspektiven zu betrachten. Diese können beispielsweise den Rollen Entwickler, Anwender, Tester, Entscheider, Planer, Administrator, und Betroffener zugeordnet werden (Hubwieser 2007a, S. 63), (Schulte und Block 2002). Die Personalisierung der Perspektiven ist jedoch auch kritisch zu sehen. Alternativ sind Informatiksysteme im Unterricht hinsichtlich ihres Zwecks im Rahmen eines Anwendungsgebietes und den Auswirkungen ihres Einsatzes zu thematisieren (Engbring 1995), (Magenheim und Schulte 2006). Die Hauptfunktionen von Informatiksystemen werden nach Denning durch sieben Kategorien der Informatik abgedeckt: computation, communication, coordination, automation, recollection, evaluation, design (Denning 2007). Sie können als Perspektive bzw. Filter zur Auswahl der Unterrichtsinhalte dienen.

4.4.4 Lehr-Lernmethodische Empfehlungen zur Kompetenzentwicklung mit Informatiksystemen

Die Analyse des Forschungsstands offenbart zwei hauptsächlich in der Unterrichtspraxis eingesetzte Vorgehensweisen. Erstens existieren viele allein stehende Unterrichtsreihen zu typischen Informatiksystemen wie Datenbanken (Antonitsch 2007), Internet (Freischlad 2006) und Standardsoftware (Voß 2006), denen jedoch die Einordnung in ein Gesamtkonzept zur Kompetenzentwicklung mit Informatiksystemen fehlt. Daher müssen unterschiedliche Informatiksystemklassen in ein Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung integriert werden (→ Inhalt 1: Typische Repräsentanten). Neben den genannten sind Kooperationssysteme für Computer Supported Cooperative Work (CSCW) und Computer Supported Cooperative Learning (CSCL) wichtig für den Informatikunterricht (vgl. Humbert 2001), da sie weitere Hauptfunktionen von Informatiksystemen betonen (Kapitel 3).

Anderen Unterrichtskonzeptionen, die Informatiksysteme besonders hervorheben ((Baumann 1996), (Hubwieser und Broy 1997b), (Hampel et al. 1999), (Humbert 2003)), fehlt es ebenfalls an verbindenden Elementen, um die zusammenhängenden Wirkprinzipien in Informatiksystemen vernetzt zu vermitteln und unterschiedlichen Sichten auf Informatiksysteme zu unterstützen. Wie müssen Inhalte zu Wirkprinzipien von Informatiksystemen repräsentiert werden, damit sie ein korrektes Bild von Informatiksystemen vermitteln und von Schülern kognitiv erfasst werden können? Im Folgenden werden Wissensrepräsentationen zur Vernetzung der Unterrichtsinhalte vorgestellt (Abschnitt 5.4). Bei der Publikation von Unterrichtskonzepten spielen fundamentale Ideen bisher oft keine Rolle. Dies ist auch darauf zurückzuführen, dass das Problem der Vernetzung der Inhalte von Lehrpersonen selbst gelöst werden muss. So ist sie oft durch langjährige Lehr-Lernerfahrungen zu Algorithmenklassen, Datentypen und Datenstrukturen, also durch die Vorgehensweise beim Programmieren im Kleinen und die eingesetzten Programmierstile (z. B. prädikativ, funktional) begründet.

Wie können Schüler Informatiksysteme erkunden und Kompetenzen handlungsorientiert entwickeln? Im Informatikunterricht fehlt das systematische Beobachten von Informatiksystemen bisher fast vollständig (Kapitel 4). Es existieren keine Unterrichtsmodelle, die das systematische Vorgehen, das zum Bewerten und Beobachten von Informatiksystemen notwendig ist, thematisieren und mit ausgewählten Bildungsanforderungen für Kompetenzentwicklung mit Informatiksystemen verknüpfen.

Hinsichtlich der Lehr-Lernmethodik zu Informatiksystemen und Kompetenzentwicklung zeichnen sich im fachdidaktischen Forschungsstand nachfolgende Schwerpunkte ab, die zu berücksichtigen sind. Dabei werden Universalität, Systemanalyse und Modifikation von Programmen nicht als Unterrichtsinhalte, sondern als Prinzipien der Unterrichtsgestaltung, d. h. als Unterrichtsmethoden aufgeführt (vgl. Hubwieser 2007a, S. 69).

Methodik 1 (Universalität) Eine der wichtigsten Eigenschaften eines Informatiksystems und gleichzeitig Lernschwierigkeit ist die freie Programmierbarkeit. Daraus folgt die Notwendigkeit der Betonung der Universalität des Rechners in Abgrenzung zu mechanischen Maschinen (Eberle 1996). Damit ist auch die Umsetzung der Universalität in einem geeigneten Modell notwendig, z. B. Von-Neumann-Architektur und Registermaschine. Die Universalität des Rechners ist ein wichtiger Unterrichtsinhalt. Die vorliegende Einordnung in die Methodik soll betonen, dass die Universalität von Informatiksystemen kontinuierlich zu thematisieren ist. Für unterschiedliche Informatiksysteme ist deren Verhalten jeweils neu zu erkunden, wodurch die Methodenentscheidungen stark beeinflusst werden.

Methodik 2 (Kognitive Modelle) Modellbildung ist ein Ziel des Informatikunterrichts (GI 2000). Methodisch ist zu beachten, dass Modelle sowohl Abbild von Informatiksystemen als auch Vorbild für Informatiksysteme sein können (Thomas 2001). Ersteres kann im Rahmen der vorliegenden Arbeit genutzt werden, um Kompetenzentwicklung mit Informatiksystemen zu unterstützen und einzelne Aspekte zu betonen. Zur Bildung korrekter kognitiver Modelle vom Rechner beim Schüler sind möglichst frühzeitig entsprechende Strukturmodelle zu thematisieren (Eberle 1996).

Methodik 3 (Verbindung von Verhalten und Struktur) Es besteht Konsens darüber, dass Informatiksysteme die von Entwicklern eingesetzten Methoden und Konzepte widerspiegeln (Magenheim und Schulte 2006), wenngleich sie nicht immer eindeutig am Produkt nachzuvollziehen sind. In der fachdidaktischen Literatur werden das Produkt, der Anwendungsprozess, der Softwareentwicklungsprozess und die Konzepte in Informatiksystemen als korrelierende Einflüsse auf das Informatiksystem beschrieben, die einem Zweck, d. h. der Lösung eines Anwendungsproblems dienen. In der vorliegenden Arbeit für Kompetenzentwicklung mit Informatiksystemen wurde die Entscheidung getroffen, die Softwareentwicklung nicht vordergründig zu betrachten. Zur Analyse des Verhaltens von Informatiksystemen werden jedoch explizit Tests und Experimente genannt (Claus und Schwill 2006). Und Methoden wie Teamarbeit werden als wichtig erachtet (Humbert 2003). Das „Inverbindungsetzen" des nach außen sichtbaren Verhaltens mit einem Blick hinter die Fassade, z. B. den Ebenen des Rechners (Eberle 1996), muss kontinuierlich geschehen (ACM 2006). Dazu gehört das Klären der Beziehung zwischen Elementen der Benutzungsoberfläche und den Komponenten in der inneren Struktur des Informatiksystems. Die bewusste Vorwegnahme häufiger Fehler im Unterricht und deren Einordnung in das Ebenenmodell des Rechners sind im Unterricht umzusetzen (Eberle 1996), (Neupert und Friedrich 1997), (Hubwieser 2007a). Dafür ist auf bewusste Anwendung zu achten, beispielsweise durch Experimente mit Hypothesenbildung (ACM 1993), um Verhalten des Systems mit den eigenen Handlungen in Verbindung zu setzen.

Methodik 4 (Analyse des Systems: Experimente) Systemanalytische und teststrategische Kompetenzen sind notwendig zur kompetenten Beurteilung und Anwendung von informationstechnischen Systemen (vgl. Koerber und Peters 1993). Experimente werden als handlungsorientierte Methode vorgeschlagen, um Informatiksysteme zu erkunden ((ACM 1993), (UNESCO 2002)). Sie implizieren Hypothesenbildung, Wiederholbarkeit und Dokumentation des eigenen Vorgehens, so dass sie das bewusste Anwenden und bewerten der Systeme unterstützen. Die Kombination aus Erforschen und Anwenden (Antonitsch 2007) kann dabei als Grundlage eines methodischen Vorgehens zur Erreichung dieses Ziels dienen.

Methodik 5 (Anwendung als Problemlösen) Die zielgerichtete Anwendung eines Informatiksystems kann als Problemlösen analog zur Algorithmik behandelt werden, z. B. Darstellung des Ablaufs durch Struktogramme oder Anwendungsfälle (Eberle 1996). Es sind unterschiedliche Systemklassen zu identifizieren und Vorgehensweisen zur bewussten Anwendung zu entwickeln (Eberle 1996). Dabei kann zwischen deklarativem Wissen als Wissen über das System und prozeduralem Wissen zur Anwendung des Systems unterschieden werden (Kennewell 1995).

Methodik 6 (Modifikation statt Entwicklung) Es wurde die Entscheidung getroffen, für Kompetenzentwicklung mit Informatiksystemen die Softwareentwicklung nicht zu betrachten. Dennoch wird ein Grundverständnis für Algorithmik und abstrakte Datenstrukturen in vielen Publikationen als notwendig für Kompetenzentwicklung mit Informatiksystemen angesehen (Nievergelt 1995), (Hubwieser und Broy 1997a). Integriert werden können sie in das angestrebte Unterrichtsmodell, indem existierende Software von den Schülern modifiziert wird (Lehmann 1995).

Methodik 7 (Von der Anwendung zur Maschine) Kompetenzen erfordern Anforderungssituationen, die zu bewältigen sind. Motivation, Fähigkeiten, Fertigkeiten und Bereitschaften sind dafür notwendig. Daher ist für Kompetenzentwicklung mit Informatiksystemen ein Weg von der Anwendung zur Maschine zu wählen, auf dem nach und nach das Black-Box-Modell durch Strukturmodelle ersetzt wird (Cyranek 1990), (Modrow 1991), (Klieme et al. 2007).

Methodik 8 (Informatiksysteme als Medium und Lerngegenstand) Die didaktische Doppelfunktion von Informatiksystemen als Lerngegenstand und Lernmedium bietet Potential hinsichtlich der Lehr-Lernmethodik (Brinda 2004a). Lernende können durch das Lernmedium an Informatikkonzepte herangeführt werden. Fortgeschrittene Lernende können auf Grundlage der so erlernten Konzepte das Lernmedium auf seine innere Struktur untersuchen und modifizieren.

Methodik 9 (Vernetzung der Unterrichtsinhalte) Informatiksysteme repräsentieren die in der Entwicklung eingesetzten Problemlösestrategien und Konzepte (z. B. Koerber et al. 1989). Wie die Informatikmethoden und -konzepte vernetzt im Unterricht thematisiert werden können, wenn nicht Softwareentwicklung gelehrt wird, bleibt weitgehend unbeantwortet. Teamarbeit und kooperatives Arbeiten mit Informatiksystemen unterstützt das vernetzte Denken (Vester 1988), (Meier 1990). Voraussetzung für Teamarbeit ist jedoch eine gemeinsame (Fach-) Sprache.

4.4.5 Empfehlungen zu Lehr-Lernmedien für Kompetenzentwicklung mit Informatiksystemen

Hinsichtlich der Lehr-Lernmedien zur Kompetenzentwicklung mit Informatiksystemen sind nur wenige direkte Hinweise zu finden. Größtenteils sind Anforderun-

gen aus den Empfehlungen zur Lehr-Lernmethodik abzuleiten: Um das nach außen sichtbare Verhalten eines Informatiksystems zu analysieren, können mediale Repräsentationen genutzt werden, um das Verhalten mit informatischen Konzepten und deren Umsetzung in der inneren Struktur des Systems miteinander in Verbindung zu setzen. Informatiksysteme als Lernmittel zur Förderung der Kompetenzentwicklung mit Informatiksystemen werden jedoch kaum in einer didaktisch aufbereiteten Form angewendet (Magenheim 2001), um die Wirkprinzipien von Informatiksystemen sichtbar zu machen. Neben Lernmedien werden in den vorherrschenden Ansätzen zur Konstruktion von Informatiksystemen oft professionelle Werkzeuge der Softwareentwicklung eingesetzt, die Schüler in der Sekundarstufe II überfordern. Hinzu kommen generische „Cognitive Tools" (Eberle 1996) und domänenspezifische Software, z. B. zur Geschäftsprozessmodellierung. Kooperationswerkzeuge und Standardsoftware sind in den Informatikunterricht zu integrieren (Humbert 2003), (Hubwieser 2007a). Freischlad (2007) nutzt eine Lernsoftware zur Verdeutlichung von Wirkprinzipien im Bereich Internetworking. Genannt werden in der fachdidaktischen Forschung nachfolgende Anforderungen.

Medien 1 (Visualisierung verborgener Prozesse) Die Darstellung der im inneren von Informatiksystemen ablaufenden Prozesse und der Struktur wird gefordert, z. B. in Animationen bzw. Simulationen (Proulx 1995). Motivation zu Lernaktivitäten soll durch Beobachtung gefördert werden.

Medien 2 (Interaktion mit den Wirkprinzipien) Lernende sollen mit den Wirkprinzipien von Informatiksystemen interagieren können, d. h. Lernmedien sollen durch Experimentieren das Entdecken fundamentaler Ideen und produktunabhängiger Strukturen ermöglichen. Fehlender Erfahrung der Lernenden mit einer systematischen Erkundung von Informatiksystemen ist durch die Lernsoftware und einen realistischen Anwendungskontext zu begegnen. Explorationsmodule nach Brinda (2004a) und die von Freischlad (2007) vorgestellte Lernumgebung für Internetworking unterstützen Interaktion mit den Wirkprinzipien und einen experimentierenden Zugang.

Medien 3 (Kooperations- und Kommunikationsmedium) Informatiksysteme können als Kooperations- und Kommunikationsmedien im Informatikunterricht genutzt werden, beispielsweise zur Unterstützung bei Projekt- und Gruppenarbeit (UNESCO 2002).

4.4.6 Schlussfolgerungen und wissenschaftliche Fragestellungen

Zusammenfassend ist eine Vielzahl an Bildungszielen, -inhalten, -methoden und -medien erkennbar, die Basiskompetenzen zu Informatiksystemen fördern kann.

Schlussfolgerung ist, dass ein Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung einschließlich Lehr-Lernmaterialien zur Unterstützung von Schülern und Lehrern auszuarbeiten ist. Dementsprechend müssen auch Lehrer in Weiterbildungsangeboten für die Notwendigkeit von Kompetenzentwicklung mit Informatiksystemen sensibilisiert werden, z. B. in Lehrerfortbildungen ((Schubert et al. 2007), (Freischlad und Stechert 2008)). Es ergeben sich folgende wissenschaftlichen Fragestellungen:

1. Wie ist ein Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen gemäß der Ergebnisse der Analyse des fachdidaktischen Forschungsstandes zu gestalten?
   Ausgehend von den in der Literatur dokumentierten fachlichen Aspekten zu Informatiksystemen und den fachdidaktisch formulierten Bildungszielen, -inhalten, -methoden und -medien wird ein Unterrichtsmodell entwickelt. Die Auswahl der Komponenten muss durch Erkenntnisse aus Informatik, Lernpsychologie, allgemeiner Didaktik und Fachdidaktik begründet werden, um Lernschwierigkeiten zu vermindern. Bei der Entwicklung des Unterrichtsmodells ist auf Stimmigkeit der Inhalte, Realisierbarkeit im unterrichtlichen Geschehen und fortwährend auf den Beitrag zu den in Kapitel 3 analysierten Eigenschaften von Informatiksystemen zu achten.
2. Wie kann das Unterrichtsmodell im Informatikunterricht der Sekundarstufe II exemplarisch erprobt werden?
   Die Forderung nach empirisch überprüften Kompetenzmodellen (Abschnitt 2.1) offenbart, dass die Entwicklung eines Unterrichtmodells nicht bei einer theoretischen Fundierung durch Erfahrungswissen der Didaktik der Informatik stehen bleiben kann, um Schülerleistungen realistisch einschätzen zu können. Ziel dieser Arbeit ist die Wegbereitung eines Kompetenzmodells zu Informatiksystemen durch Angabe eines theoretisch begründeten, durch normative Analyse des Forschungsstands praxisrelevanten Unterrichtsmodells, das exemplarisch gestaltet und im Unterricht erprobt wird. Die exemplarische Erprobung hat die Aufgabe, zwischen normativen Zielen und Inhalten sowie konkretem Informatikunterricht zu vermitteln.

5. Vorgehensweise zur Entwicklung eines Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung

5.1 Überblick

Da eine Unterrichtsmodellentwicklung nur mit einer konsensualen Verständigung über die zu erreichenden Bildungsziele und -inhalte sowie unterstützende Unterrichtsmedien und Lehr-Lernmethoden möglich ist, wurden im vorherigen Kapitel der nationale und internationale Stand der Forschung zur informatischen Bildung auf Ansätze zur Förderung der Kompetenzentwicklung mit Informatiksystemen im historischen Verlauf untersucht. In diesem Kapitel werden die vorgestellten Überlegungen zu Informatiksystemen und Kompetenzentwicklung in die Entwicklung eines Unterrichtsmodells integriert und fokussiert. Dies schließt die Integration von Erkenntnissen aus der Fachwissenschaft, der Erziehungswissenschaft und anderen Fachdidaktiken in die Didaktik der Informatik ein.

Die Gliederung der Unterrichtsmodellentwicklung nutzt die in der Berliner Didaktik (Heimann 1976) genannten Entscheidungsfelder als Strukturierung (siehe Abschnitt 5.2.2). Danach ist die Intention des Unterrichtsmodells zu bestimmen (Abschnitt 5.2). Anschließend werden die Ergebnisse aus der Analyse des Forschungsstandes aufgegriffen und eine Strategie zur Strukturierung sowie die Fokussierung im Rahmen der vorliegenden Arbeit dargelegt (Abschnitt 5.3). Ersten Schwerpunkt bildet

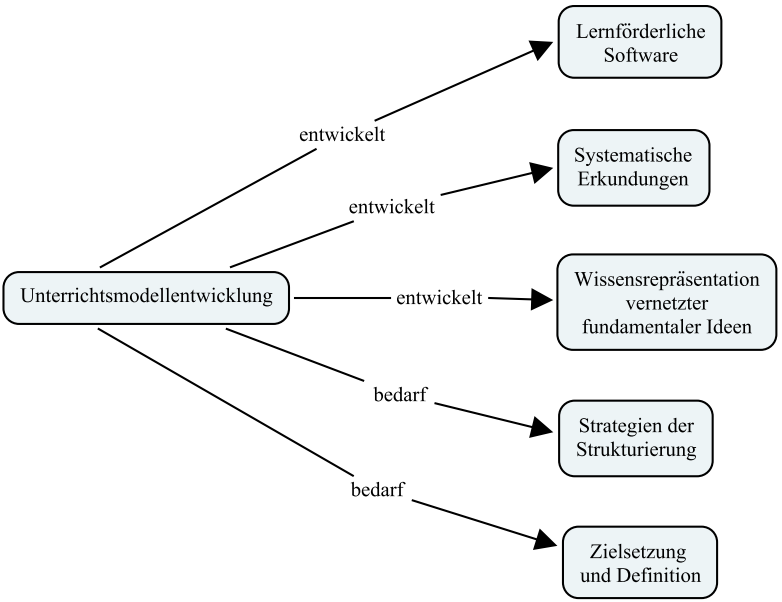


Abbildung 5.1: Struktur des Kapitels zur Unterrichtsmodellentwicklung

die Vernetzung fundamentaler Ideen der Informatik zur Förderung der Kompetenzentwicklung mit Informatiksystemen. Der Autor begründet mit Erkenntnissen der Lernpsychologie, Informatik und Informatikdidaktik, dass Entwurfsmuster informatikspezifische, extern darstellbare Wissensrepräsentationen von vernetzten fundamentalen Ideen der Informatik sind (Abschnitt 5.4). Einige nach fachdidaktischen Kriterien ausgewählte Entwurfsmuster sind gleichzeitig Lernmittel und Lerngegenstand. Als Schülertätigkeiten zur Förderung der Kompetenzentwicklung mit Informatiksystemen werden Vorgehensweisen zur systematischen Erkundung von Informatiksystemen vorgestellt (Abschnitt 5.5). Einen weiteren Teil des Unterrichtsmodells bildet Lernsoftware, die vernetzte fundamentale Ideen anhand von Wissensrepräsentationen aufgreift (Abschnitt 5.6). Abbildung 5.1 zeigt die Struktur des Kapitels.

5.2 Zielsetzung und Definition des Unterrichtsmodells

5.2.1 Motivation für Komponenten eines Unterrichtsmodells

Die Analyse des Informatiksystembegriffs sowie des nationalen und internationalen fachdidaktischen Forschungsstandes zeigen, dass es zur Förderung der Kompetenzentwicklung mit Informatiksystemen einer Strukturierung der Bildungsziele, -inhalte, Lehr-Lernmethoden und Lernmedien für den Informatikunterricht bedarf. Im Folgenden ist es das Ziel, ein Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung zu entwickeln, dass zu dem offenkundigen Bedarf geeignete Schwerpunkte setzt, um den Aneignungs- und Vermittlungsprozess zu unterstützen:

- Bedarf einer Strukturierung der Bildungsziele und möglicher Unterrichtsinhalte
  Ausgehend von den Erkenntnissen zu dem Informatiksystembegriff und Kompetenzbegriff sowie den Ergebnissen aus der Analyse des Stands der Forschung ist eine Strukturierung der Bildungsziele vorzunehmen. Die Bildungsziele sind Grundlage von Strukturierungsempfehlungen, die sowohl Vorschläge für Unterrichtsinhalte als auch zu deren Reihenfolge umfassen. In der vorliegenden Arbeit wird insbesondere fundamentalen Ideen der Informatik eine Schlüsselrolle zugewiesen (Abschnitt 5.3).
- Bedarf einer Vernetzung der Wirkprinzipien von Informatiksystemen im Unterricht
  Begründet werden Entwurfsmuster als Wissensrepräsentation vernetzter fundamentaler Ideen der Informatik, die gleichzeitig etablierte Bausteine zur Softwareentwicklung sind. Nachgewiesen wird, dass sie die lernpsychologisch begründeten Eigenschaften von Schemata besitzen und sich zusätzlich extern repräsentieren lassen. Als graphbasierte Darstellungsmittel zur Repräsentation der vernetzten fundamentalen Ideen der Informatik in Entwurfsmustern werden Wirkungsdiagramme vorgestellt, um die fachdidaktische Kommunikation zu fördern (Abschnitt 5.4).
- Lehr-Lernmethodischer Bedarf an Schüleraktivitäten und Herangehensweisen an Informatiksysteme
  Um dem Mangel an unterrichtsgeeigneten Vorgehensweisen zur Erkundung von Informatiksystemen zu begegnen, wird eine Herangehensweise in Anlehnung an Unterrichtsexperimente begründet und mit einer Stufung der kognitiven Prozesse nach Anderson und Krathwohl (2001) theoretisch fundiert (Abschnitt 5.5).
- Bedarf an Lernmedien wie lernförderlicher Software zur Unterstützung des Lehr-Lernprozesses

Gestaltungskriterien für Lernsoftware werden anhand der Erkenntnisse zu Wissensrepräsentationen erarbeitet, und Lernsoftware wird sowohl während als auch nach der Entwicklung im unterrichtlichen Geschehen erprobt. Zentral für die Vorgehensweise bei der Entwicklung von Lernsoftware sind antizipierte kognitive Hürden und erwartete Fehlvorstellungen der Schüler bezüglich Informatiksysteme. Im Rahmen des Unterrichtsmodells soll Lernsoftware die Vernetzung fundamentaler Ideen der Informatik und Schülervorgehensweisen zur systematischen Erkundung von Informatiksystemen fördern (Abschnitt 5.6).

Das erfolgreiche „Didaktische System für objektorientiertes Modellieren" (Brinda und Schubert 2001) legt nahe, auch für Kompetenzentwicklung mit Informatiksystemen ein didaktisches System zu entwickeln. Denn mit dem Ansatz der didaktischen Systeme wird eine Sammlung aufeinander abgestimmter Lehr-Lernmaterialien bereitgestellt, welche die Unterstützung der fachdidaktischen Kommunikation und Umsetzung im Unterricht zum Ziel hat. Anforderungen an Lernsoftware, Wissensstrukturen und qualitativ hochwertige Aufgaben werden für ein didaktisches System formuliert. Schubert betont, dass mit einem didaktischen System in Abhängigkeit von unterschiedlichen Unterrichtsszenarios je nach Zielgruppe sehr flexibel verschiedene Kompetenzen erreicht werden können (Schubert und Schwill 2004, S. 134).

Dennoch offenbart der Transfer des didaktischen Systems auf Kompetenzentwicklung mit Informatiksystemen das Dilemma unterschiedlicher Abstraktionsebenen. Während es Brinda (2004a) gelingt, für das klar umrissene Feld der objektorientierten Modellierung aus der vorhandenen Literatur zu OOM an Schulen und Hochschulen einen konsensfähigen Katalog an Aufgabenklassen zu extrahieren, muss für Kompetenzentwicklung mit Informatiksystemen festgestellt werden, dass das Thema weder derart eindeutig von anderen Gebieten abzugrenzen ist noch entsprechende Literatur existiert. Darüber hinaus stellt Schubert fest, dass ergänzend zum didaktischen System ein Bedarf an einer an der Fachwissenschaft orientierten Auswahl von Unterrichtsinhalten besteht, z. B. durch fundamentale Ideen der Informatik (Schubert und Schwill 2004, S. 134). Somit ist es nach Auffassung des Autors notwendig, statt eines didaktischen Systems ein Unterrichtsmodell zu entwickeln, das eine Strukturierung des Feldes vornimmt, die sowohl fachdidaktische Kommunikation und Umsetzung fördert als auch die Auswahl von Informatikinhalten und -methoden zum Ziel hat. Dennoch fließen Erkenntnisse aus dem didaktischen System in das Unterrichtsmodell ein, insbesondere hinsichtlich Lernsoftware (Abschnitt 5.6) und Aufgaben, die ein experimentierendes Vorgehen zur Systemerkundung zugrunde legen (Abschnitt 5.5). Im Folgenden werden die genannten Schwerpunkte verknüpft zu einem Unterrichtsmodell und dessen Entwicklung exemplarisch für Kompetenzentwicklung mit Informatiksystemen dargestellt.

5.2.2 Definition Unterrichtsmodell

Aufgabe des Unterrichtsmodells ist, einen fachdidaktischen Rahmen für Informatikunterricht zu schaffen, um ein Bildungsziel, wie beispielsweise Kompetenzentwicklung mit Informatiksystemen, zu fördern.

Unter einem Unterrichtsmodell sei im Folgenden ein fachdidaktisch begründetes Konzept verstanden, das Erkenntnisse aus der Lehr-Lerntheorie, der Fachwissenschaft, der Erziehungswissenschaft sowie anderen Fachdidaktiken aufgreift und mit informatikdidaktischen Konzepten

- zur Vernetzung der Grundbegriffe und Wirkprinzipien sowie zur fachdidaktischen Kommunikation (Wissensrepräsentation für vernetzte fundamentale Ideen),
- zur Kategorisierung von Bildungszielen und -inhalten (Perspektiven auf Informatiksysteme; Strategie zur Strukturierung) und
- zur Umsetzung im unterrichtlichen Geschehen (Entwicklung von lernförderlicher Software und Schülervorgehensweisen mit Handreichungen zu Motivation und Durchführung von Experimenten)

verknüpft. Die Vernetzung der Grundbegriffe und Wirkprinzipien von Informatiksystemen blieb bisher eine Aufgabe, die der Lehrer selbst lösen musste. Das Unterrichtsmodell unterstützt den Lehrer, die Unterrichtsinhalte durch Wissensrepräsentationen zu vernetzen.

Ähnlich dem didaktischen System, das als offener, erweiterbarer Verbund von Komponenten des Lehr-Lernprozesses angelegt wurde, gilt auch für ein Unterrichtsmodell, dass zielgruppen- und themenspezifische Gestaltungsanforderungen zu einer Verfeinerung und der Berücksichtigung weiterer Aspekte führen können. Der Begriff des Unterrichtsmodells wird verwendet, um anzudeuten, dass es sich um einen theoretischen Rahmen für Informatikunterricht handelt. Dabei ist eine sofortige Umsetzbarkeit des Unterrichtsmodells in Unterricht nicht das primäre Ziel, wie es bei so genannten Unterrichtskonzepten angestrebt wird:

> „Unterrichtskonzepte sind Gesamtorientierungen didaktisch-methodischen Handelns [...]. Probleme didaktischer Theoriebildung treten oft, nicht immer in den Hintergrund" (Jank und Meyer 2002, S. 305f).

Vielmehr soll durch den Begriff des Unterrichtsmodells die fachdidaktische Theoriebildung besonders betont werden. So ist es anhand des Unterrichtsmodells möglich, existierenden Unterricht auf die im Unterrichtsmodell definierten Bildungsziele hin zu untersuchen und daraufhin Anpassungen vorzunehmen. Der Begriff des Unterrichtsmodells muss jedoch abgegrenzt werden von einem didaktischen Modell, wie z. B. der Berliner Didaktik. Solch ein didaktisches Modell bildet ein Theoriegebäude, durch das didaktisches Handeln analysiert und modelliert werden kann (Jank und Meyer 2002, S. 35). Es klärt

> „theoretisch umfassend und praktisch folgenreich die Voraussetzungen, Möglichkeiten, Folgen und Grenzen des Lehrens und Lernens" (Jank und Meyer 2002, S. 35).

Diese didaktischen Modelle geben einen Rahmen für das angestrebte Unterrichtsmodell vor. Die bildungstheoretische Didaktik nach Klafki unterscheidet sieben Kategorien zur Unterrichtsplanung. Diese sind Gegenwartsbedeutung, Zukunftsbedeutung, exemplarische Bedeutung, thematische Struktur und soziale Lernziele, Erweisbarkeit und Überprüfbarkeit, Zugänglichkeit und Darstellbarkeit beispielsweise durch Medien und zuletzt die methodische Lehr-Lernprozessstruktur (Klafki 1996). Die lerntheoretische bzw. Berliner Didaktik entstand aus der Diskussion um die bildungstheoretische Didaktik. Heimann (1976) betont im Gegensatz zur bildungstheoretischen Didaktik, dass die Entscheidungsfelder voneinander abhängig seien. Die sozio-kulturellen und anthropogenen Voraussetzungen wirkten darüber hinaus in die Bedingungsfelder zur Intention, zu den Unterrichtsinhalten, zu den Methoden und zu den Medien hinein (vgl. Hubwieser 2007a, S. 27). Beide Didaktiken, die sich laut Jank und Meyer bis zur Ununterscheidbarkeit einander angenähert haben (Jank und Meyer 2002, S. 37), bilden die Grundlage des vorliegenden Unterrichtsmodells. Weitere Grundlage des Unterrichtsmodells ist die Forderung nach konstruktivistischem Lernen im Sinne des gemäßigten Konstruktivismusses (Jank und Meyer 2002, S. 301). Es basiert auf der Grundannahme, dass Menschen alles Wissen nur auf Grundlage eigener Erfahrungen konstruieren können und die Wirklichkeit letztlich nur ein Konstrukt des Gehirns und damit nicht unmittelbar zu erkennen ist (Jank und Meyer 2002, S. 286ff). Jank und Meyer betonen dabei jedoch, dass konstruktivistische Didaktiken zwar zahlreiche Anregungen bieten, aber bildungstheoretisch weiterentwickelt werden müssen (Jank und Meyer 2002, S. 301). Hubwieser führt einige konstruktivistische Strömungen an (Hubwieser 2007a, S. 10f), die spezifische Anregungen betonen und in der vorliegenden Arbeit zum Beispiel bei der Entwicklung von Lernsoftware und bei Planung und Durchführung der Unterrichtserprobungen handlungsleitend waren: Situiertheit dadurch, dass die Schüler direkt mit Informatiksystemen als Gegenstand der Aufgaben in einem konkreten Zusammenhang befasst sind (Abschnitt 5.5), Gestaltung und Einbindung eines narrativen Ankers bei der Konzeption der Lernsoftware Pattern Park (Abschnitt 5.6), kognitive Flexibilität zur Begründung unterschiedlicher Sichten auf Informatiksysteme und entsprechende Modelle (Abschnitt 5.3), und Cognitive Apprenticeship bei der Durchführung von Informatikexperimenten (Abschnitt 5.5).

Zusammenfassend sei der Begriff Unterrichtsmodell definiert als ein theoretisch begründeter Rahmen für praktische Unterrichtskonzepte. Das Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen wird für den allgemein bildenden Informatikunterricht der Sekundarstufe II konzipiert. Prinzipiell ist jedoch auch für die Sekundarstufe I angemessener Unterricht zur Förderung

der Kompetenzentwicklung mit Informatiksystemen zu fordern. Ab Abschnitt 5.4 werden die Komponenten des Unterrichtsmodells vorgestellt und diskutiert.

5.3 Strukturierung und Fokussierung der Bildungsziele, -inhalte, -methoden und -medien

5.3.1 Perspektiven auf Informatiksysteme

Die Analyse des Informatiksystembegriffs in Kapitel 3 ergab folgende Grobstrukturierung der Bildungsziele zur Förderung der Kompetenzentwicklung mit Informatiksystemen (Claus und Schwill 2006):

Die Schüler sind in der Lage, eine Anforderungssituation zu bewältigen hinsichtlich

S_A: des nach außen sichtbaren Verhaltens von Informatiksystemen,

S_B: der inneren Struktur und interner Prozesse von Informatiksystemen,

S_C: ausgewählter Aspekte der Implementierung einer konkreten Realisierung.

Bei der Erstbegegnung mit einem Sachverhalt ist das intuitive Denken und Verstehen der Schüler zu berücksichtigen. Bruner klassifiziert in seinem Repräsentationsmodell die Informationsaufnahme, -verarbeitung und -speicherung nach der Stufe der Handlung (enaktiv), der bildhaften Wahrnehmung (ikonisch) und der Sprache (symbolisch) (vgl. Bruner 1966). Es ist zu berücksichtigen, dass diese Stufen aufeinander aufbauen und ebenso wie die fundamentalen Ideen nach Bruner das Spiralprinzip unterstützen (Abschnitt 4.2.2). Hartmann et al. (2006) unterscheiden darüber hinaus zwischen enaktiv und virtuell-enaktiv. Letzteres bezieht sich auf Handlungen in einer (Lern-) Software.

Für den Lernprozess ist interessant, dass für das nach außen sichtbare Verhalten, die innere Struktur und die Implementierungsebene je eine andere Repräsentationsebene als Beschreibungsart dominiert (enaktiv, ikonisch, symbolisch), wenngleich dies nicht zwingend ist. So bietet sich für die Ebene S_A ein enaktiver oder virtuell-enaktiver Zugang an. Das heißt, Lernende können ein aktives Informatiksystem im doppelten Wortsinn begreifen. Informatiksysteme, die durch ihr Verhalten in die Realität einwirken, werden auch als enaktive Modelle bezeichnet:

> „Wir sprechen [...] von enaktiven (handlungsbezogenen) Modellen: Die Wirklichkeit wird durch Objekte modelliert, an denen man Handlungen vornehmen kann, und die selber aktiv werden und auf andere Objekte einwirken können, die folglich vom Menschen kognitiv erfasst werden wie ihre Originale" (Schubert und Schwill 2004, S. 155).

Die innere Struktur S_B hingegen ist sehr gut durch Diagramme, also ikonisch, zu beschreiben. Mit ihnen lassen sich neben statischen auch dynamische Aspekte

des Informatiksystems erfassen. Die Erarbeitung einer konkreten Realisierung S_C erfordert das Erstellen und Bearbeiten einer Spezifikation sowie Programmierung und Modifikation von Quelltext. Somit überwiegt in diesem Bereich die textuelle, also symbolische Repräsentationsform.

In der vorliegenden Arbeit wird der Schwerpunkt jedoch auf das Systemverhalten und die innere Struktur gelegt, Implementierungsaspekte spielen eine nachgeordnete Rolle (Abschnitt 3.3). Hinsichtlich der zu wählenden Modelle folgt daraus:

> „Von diesem Standpunkt aus gesehen stellen *Datenstrukturen, Verarbeitungsvorschriften* und *Systembeschreibungen* Modelle oder Teile von Modellen dar, allerdings auf unterschiedlichen Abstraktionsstufen. Während Datenstrukturen und Verarbeitungsvorschriften auf die Implementierungtechnik bezogen sind, beschäftigen sich Systembeschreibungen mit den Eigenschaften des zu modellierenden Systems" (Hubwieser 2007a, S. 87; Hervorh. im Original).

Damit sind Modelle zur Systembeschreibung zu wählen, die auf höherem Abstraktionsniveau liegen.

5.3.2 Strategie zur Strukturierung der Unterrichtsinhalte

Thematische Fokussierung

In diesem Abschnitt soll anstelle eines festen Katalogs von Unterrichtsinhalten eine Strategie zur Strukturierung begründet werden. Dabei spielen Fragestellungen zum Allgemeinbildungswert, zur Ziel-Inhalts-Mittel-Relation und zur fundierten Auswahl von vernetzten fundamentalen Ideen eine Rolle.

In der Didaktik der Informatik ist besonders der Ansatz nach Bussmann und Heymann (1987) weit verbreitet (vgl. Hubwieser 2007a, S. 57). Danach ist mindestens einer der folgenden Punkte für Unterrichtsinhalte zu erfüllen: (1) Vorbereitung auf zukünftige Lebenssituationen, (2) Stiftung kultureller Kohärenz, (3) Aufbau eines Weltbildes, (4) Anleitung zum kritischen Vernunftgebrauch, (5) Entfaltung eines verantwortlichen Umgangs mit den erworbenen Kompetenzen, (6) Stärkung des Schüler-Ichs (vgl. Engbring 1995, S. 71). Bezüglich der Vorbereitung auf zukünftige Lebenssituationen ist sicherlich ein Blick auf die Rollen des Menschen gegenüber Informatiksystemen zu nennen. Diese sind Entscheider, Planer, Entwickler, Administratoren, Nutzer und Betroffene (Hubwieser 2007a, S. 63). Stiftung kultureller Kohärenz gelingt beispielsweise durch Vereinheitlichung der Fachsprache, die bei der Anwendung von Informatiksystemen verwendet wird (Hubwieser 2007a, S. 63), aber auch durch Darlegen einer systematischen Herangehensweise an Informatiksysteme (vgl. Abschnitt 5.5). Der Aufbau eines Weltbildes wird gestärkt durch Ausbildung einer Sach-, Handlungs- und Beurteilungskompetenz zur Anwendung von Informatiksystemen. Die Anleitung zum kritischen Vernunftgebrauch kann nach Hubwieser aus den Möglichkeiten und Grenzen der Systeme durch Berechenbarkeit und Komplexität, Funktionsweise und durch Bewusstmachen des Phänomens

der Blindheit von Informatiksystemen erwachsen. Die Stärkung des Schüler-Ichs geschieht unter anderem durch Vermeidung von Computergläubigkeit, z. B. durch Thematisierung der Grenzen von Informatiksystemen und Vorwegnahme von Fehlern im Unterricht.

Im bildungstheoretischen Ansatz der Göttinger Schule nach Klafki (1996) müssen Themen für den Fachunterricht ausgewählt werden, die als Besonderes das Allgemeine enthalten. Die Leitfragen sind:

> „Welche exemplarische Bedeutung hat der Unterrichtsgegenstand? Wie bedeutend ist er für die Gegenwart? Welche Bedeutung für die Zukunft lässt sich vermuten? Wie ist die Struktur des Inhalts? Wie steht es mit der unterrichtlichen Zugänglichkeit?" (Hubwieser 2007a, S. 25f).

In ihnen wird wie auch bei der Berliner Didaktik nach Heimann (1976) besonders das Exemplarische betont.

Die Analyse von Bildungsempfehlungen und Curricula ergibt eine Vielzahl möglicher Unterrichtsinhalte, die zur Kompetenzentwicklung mit Informatiksystemen für notwendig erachtet werden. In solchen Empfehlungen werden oft jedoch nur Lernziele formuliert, ohne sie in einen größeren Kompetenzzusammenhang zu bringen (Abschnitt 4.4).

Um die Unterrichtsinhalte für ein Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung zu strukturieren, beziehen sich Stechert und Schubert (2007) auf ihre Forschungsresultate zu Schichtenmodellen, Entwurfsmustern und vernetzten fundamentalen Ideen der Informatik. Dazu werden folgende Kategorien identifiziert:

A) Typische Repräsentanten von Informatiksystemen,
   - Systemsoftware mit besonderer Berücksichtigung der Hardware,
   - Anwendungssoftware mit besonderer Berücksichtigung der Vernetzung;

B) Strukturmodelle von Informatiksystemen mit geeigneten Darstellungsformen,
   - Von-Neumann-Rechner inklusive Maschinenmodell,
   - Schichtenmodelle,
   - Entwurfsmuster;

C) Vernetzte fundamentale Ideen der Informatik in Informatiksystemen.

Dabei ist zu beachten, dass an typischen Repräsentanten und Strukturmodellen wiederum fundamentale Ideen – durch den exemplarischen Charakter möglicherweise auf einem niedrigeren Abstraktionsniveau – zu lernen sind. Der Bereich B) ist einerseits strukturiert durch das Ebenenmodell der Rechnerarchitektur, das unterschiedliche Abstraktionsebenen von der digitalen Logikebene (reale Maschine)

zur höheren Programmiersprache beschreibt (Tanenbaum und Goodman 2001), andererseits durch den Zusammenhang zwischen realer, abstrakter und virtueller Maschine (Schubert und Schwill 2004).

Im Folgenden wird die Auswahl der Unterrichtsinhalte diskutiert und für das Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung weiter begründet.

Typische Repräsentanten von Informatiksystemen

Es gibt unterschiedliche Ausgestaltungen von Informatiksystemen, z. B. Betriebssystem, Datenbanksystem und Textverarbeitungssystem. Das zu inspizierende Informatiksystem im Unterricht ist ein durch die Lehrperson ausgesuchter exemplarischer Lerngegenstand. Beschäftigung mit dem Lerngegenstand soll Kompetenzentwicklung zu ausgewählten Aspekten von Informatiksystemen fördern. Es reicht daher nicht, allein das Verhalten eines Systems zu kennen. Zur Begründung liefern Hubwieser und Broy eine entscheidende Klassifikation der Unterrichtsinhalte nach ihrer Allgemeingültigkeit, bei der nur die beiden erstgenannten für die Schulinformatik uneingeschränkt geeignet sind:

- Anwendung auch außerhalb von Informatiksystemen,
- charakteristisch für alle Informatiksysteme,
- charakteristisch für eine Klasse von Informatiksystemen,
- charakteristisch für ein konkretes Informatiksystem (vgl. Hubwieser 2007a, S. 83).

Unterrichtsinhalte sollen also möglichst für alle Informatiksysteme relevant sein. Typische, ausgewählte Repräsentanten von Informatiksystemen können dazu dienen, diese Unterrichtsinhalte oder, bei entsprechender Begründung, die für eine Systemklasse relevanten Unterrichtsinhalte zu thematisieren. Vom Autor betreut wurden Datenbanksysteme und Betriebssysteme in Seminararbeiten von Dittich (2008) und Gerding (2008) für Kompetenzentwicklung mit Informatiksystemen betrachtet. Beispielsweise bei Datenbanksystemen lassen sich zentrale Konzepte der Modellbildung anschaulich verdeutlichen (Dittich 2008). Dazu wird die Funktionalität eines Datenbankmanagementsystems auf verschiedenen Abstraktionsebenen, also als Schichtenarchitektur modelliert (vgl. Antonitsch 2007): vom Sekundärspeicherzugriff bis zu deklarativen Datenbanksprachen (vgl. Tanenbaum und Goodman 2001). Skalierbarkeit ist hier sowohl hinsichtlich der Benutzeranzahl als auch hinsichtlich der Datenmenge zu gewährleisten. Es kann auf Datenmengen eingegangen werden, wie sie beispielsweise in Datenbanken von Suchmaschinen anfallen. Auch konkurrierender Zugriff und dessen Vermeidung durch das ACID-Prinzip (Atomicity, Consistency, Isolation, Durability) lassen sich thematisieren. Da typische Informatiksysteme wie Datenbanksysteme häufig im Unterricht thematisiert werden, erfolgt in der vorliegenden Arbeit eine Fokussierung auf Eigenschaften, die

für möglichst viele Informatiksysteme zutreffen, z. B. Zugriffskontrolle. Zur Argumentation werden neben obiger Klassifikation hinsichtlich der Allgemeingültigkeit auch die Kategorien der Informatik nach Denning (2007) genutzt und im Sinne von Hauptfunktionen von Informatiksystemen angewendet.

Strukturmodelle von Informatiksystemen

Strukturmodelle bilden einen Zugang zur inneren Struktur von Informatiksystemen. Durch die Annahme, dass durch die grobe Art der Zerlegung von Systemen in wenige Architekturelemente, bzw. aufeinander aufbauende Schichten, das Verstehen von Systemcharakteristika gefördert wird, ist eine Auswahl aus den Strukturmodellen zu treffen. Dabei sollten sowohl Hardwarearchitekturen als auch Softwarearchitekturen berücksichtigt werden. Eine Grundannahme ist hierbei das Konzept der strukturierten Zerlegung, speziell der Modularisierung:

> „Hinter der Modularisierung verbirgt sich die Zielvorstellung, jedes System ließe sich durch die Eigenschaften seiner Teile vollständig erklären und als Summe total voneinander unabhängiger Teile (*bottom-up*) auffassen bzw. in total voneinander unabhängige Teile zerlegen (*top-down*). Diese Zielvorstellung ist zwar faktisch unerreichbar, denn die Eigenschaften eines Systems lassen sich nicht nur aus den Eigenschaften der Subsysteme allein ableiten, sondern sie werden auch durch ihre Beziehungen untereinander beeinflusst; die normative (nach Plato) bzw. regulative (nach Kant) Funktion dieser Idee äußert sich jedoch in dem Bestreben, die Abhängigkeiten zwischen den Teilsystemen (Schnittstellen) dann wenigstens so gering wie möglich zu halten" (Schubert und Schwill 2004, S. 93; Hervorh. im Original).

Begründung ist, dass Lernende durch Strukturmodelle, seien es Rechnerarchitektur, Softwarearchitektur oder generell die Zerlegung von Systemen in wenige Komponenten, ein kognitives Modell des Systems aufbauen können. Dementsprechend bieten sich Von-Neumann-Rechner, Schichtenmodelle und Entwurfsmuster zur Strukturierung an. Schichtenmodelle können im Unterricht die Verbindung zwischen den verschiedenen Strukturmodellen begründen:

> „To describe computer-computer interaction as well as the gap between human-computer interaction and binary machine code, i. e., between virtual machine and real machine, 'models of layers' are essential" (Stechert und Schubert 2007, S. 6)

In der vorliegenden Arbeit werden Entwurfsmuster (Abschnitt 5.4) und Architekturmuster (Abschnitt 7.3) als Strukturmodelle eingesetzt. Ein Ausblick auf das Blockmodell des Von-Neumann-Rechners und Schichtenmodelle wird in Abschnitt 9.3 gegeben. Schichtenmodelle stellen eine hierarchische Strukturierung von Informatiksystemen dar. Entwurfsmuster wiederum bilden viel feinere Strukturelemente zur Beschreibung von Systemkomponenten, die sich nicht notwendigerweise an der Schichtenarchitektur orientieren. Darüber hinaus unterstützen Entwurfsmuster den Lernvorgang im Sinne des Cognitive Apprenticeship (kognitive Handwerkslehre; (vgl. Hubwieser 2007a, S. 11)), da sie als Problemlösemuster von Experten auf dem

Gebiet der Softwareentwicklung eingesetzt werden und somit authentische Methoden darstellen. Informatische Strukturmodelle sollen sowohl für das Verstehen der Strukturierung von Informatiksystemen einsetzbar als auch als Wissensrepräsentation für vernetzte fundamentale Ideen der Informatik einen stark fachdidaktisch begründeten Einsatz erfahren.

Vernetzte fundamentale Ideen der Informatik in Informatiksystemen

Während der Analyse des Forschungsstandes wurde die Entscheidung getroffen, für das Unterrichtsmodell fundamentale Ideen der Informatik zu nutzen, da sie auf pädagogisch motivierten Kriterien basieren und gleichzeitig den Bildungswert sowie die Orientierung am Fach garantieren. Darüber hinaus ergab die Analyse, dass die Vernetzung von Unterrichtsinhalten für Kompetenzentwicklung mit Informatiksystemen wichtig ist. Brauer und Brauer fordern mit Blick auf komplexe Informatiksysteme die Informatik auf, vernetztes Denken und Handeln mit zu gestalten:

> „Ja, es wird immer klarer, daß sequentielle geschlossene Systeme sehr grobe Idealisierungen darstellen, daß aber konkrete Systeme i. a. verteilt, offen (interaktiv) und nichtsequentiell sind. Selbst die simple Turingmaschine ist nicht wirklich sequentiell realisierbar (worauf schon Petri in [Pet 62] hinwies). [...] Deshalb ist die Änderung der Denkgewohnheiten nötig; vernetztes, nicht länger sequentielles Denken wird gebraucht [...]. Vernetztes Denken und Handeln wird unterstützt durch die Informatik: durch Soft- und Hardware-Systeme, aber auch durch Begriffe, Konzepte, Modelle, Verfahren" (Brauer und Brauer 1992, S. 17).

Für das Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen wurde deshalb die Entscheidung getroffen, nicht isolierte, sondern vernetzte fundamentale Ideen der Informatik in den Mittelpunkt zu stellen.

Vernetzte fundamentale Ideen der Informatik sind zentral für Kompetenzentwicklung mit Informatiksystemen. Fundamentale Ideen der Informatik nach Schwill wurden anhand des Softwareentwicklungsprozesses identifiziert und sind somit eng mit Entwurf und Konstruktion von Informatiksystemen verknüpft (Schwill 1993a). Konsens herrscht darüber, dass sich informatische Konzepte, die bei der Konstruktion eingesetzt wurden, auch in den Produkten widerspiegeln (vgl. Kapitel 4).

Wie kann eine sinnvolle Auswahl geeigneter fundamentaler Ideen der Informatik vorgenommen werden, die für Kompetenzentwicklung mit Informatiksystemen wichtig sind?

Durch die Kriterien ist eine Einschränkung auf den von Schwill vorgegebenen Katalog fundamentaler Ideen nicht zwingend:

> „Stets ist aber zu berücksichtigen, dass jede Liste ein gewisses subjektives Element enthält, weil die beteiligten Begriffe und Bedingungen nicht formal definiert sind und auch nicht formal definiert werden können (regulative versus konstitutive Funktion von Ideen), also immer Interpretationsfreiheiten lassen. Auch gibt es kein Kriterium, um nachzuweisen, ob eine Ideenkollektion alle Elemente der Wissenschaft vollständig und in angemessenem Umfang beschreibt" (Schubert und Schwill 2004, S. 87).

Nievergelt schlägt vor, nicht pauschal, sondern im Einzelfall zu prüfen, ob ein Konzept zielführend ist (Abschnitt 3.2.3). Zur Begründung der Auswahl fundamentaler Ideen für Kompetenzentwicklung mit Informatiksystemen bieten sich drei Alternativen an. Erstens kann über fachdidaktische Kriterien argumentiert werden. So könnte in einer Auslegung des Zielkriteriums für fundamentale Ideen das (möglicherweise unerreichbare) Ziel „Beherrschung von Informatiksystemen" (vgl. Brunnstein 2001) herangezogen werden, denn dieses philosophische Kriterium soll ermöglichen, dass man

> „sie [die Informatik, A. d. V.] gegen andere Wissenschaften abgrenzen [kann]" (Schubert und Schwill 2004, S. 85).

Als zweites kann über die Sichtweisen auf Informatiksysteme argumentiert werden. Hier gibt es das nach außen sichtbare Verhalten, die innere Struktur und die Erstellung einer konkreten Realisierung. Im Rahmen des Unterrichtsmodells wird ein Zugang über das nach außen sichtbare Verhalten gewählt, daher bietet sich die Analyse des Anwendungsprozesses auf fundamentale Ideen an. Eine Auswahl kann hier einerseits über typische Repräsentanten von Informatiksystemen gelingen, indem Interaktionsformen bzw. Ein-Ausgabe-Anforderungen betrachtet werden. Andererseits kann über Qualitätsanforderungen (in Form von Normen) eine Auswahl begründet werden.

Zusätzlich kann normativ vorgegangen werden. Einige grundlegende Begriffe der Informatik, die in Bildungsempfehlungen oft genannt werden, sind eng mit Informatiksystemen verbunden. Dazu zählen beispielsweise das EVA-Prinzip, Formalisierung, Speicherung, freie Programmierbarkeit und Skalierbarkeit von Informatiklösungen. Außerdem kann über Hauptfunktionen von Informatiksystemen argumentiert werden (Denning 2007), z. B. spielt bei fast allen Informatiksystemen Zugriffskontrolle eine Rolle. Freie Programmierbarkeit ist ein Von-Neumann-Prinzip und dementsprechend häufig vertreten. Die strukturelle Einsicht, dass Informatiksysteme nur dass verarbeiten können, was formalisierbar ist, wird oft in Kombination mit formalen Sprachen und Automaten angeführt.

Zur Auswahl und Repräsentation vernetzter fundamentaler Ideen werden im Folgenden ausgewählte Entwurfsmuster der Objektorientierung analysiert.

5.4 Entwurfsmuster als Wissensrepräsentation für vernetzte fundamentale Ideen der Informatik

5.4.1 Entwurfsmuster zur Förderung der Kompetenzentwicklung mit Informatiksystemen

Motivation und Ziel

Die Analyse des Forschungsstandes ergab eine Übersicht der fachdidaktischen Erkenntnisse zu Wirkprinzipien von Informatiksystemen bezüglich deren Aufbau, Ar-

beitsweise und Vernetzung (Abschnitt 4.4). Besondere Aufmerksamkeit kommt fundamentalen Ideen der Informatik, der Vernetzung von Informatikkonzepten und Strukturmodellen von Informatiksystemen zu (Abschnitt 5.3). Potential zur Verknüpfung dieser Aspekte liefern nach Ansicht des Autors objektorientierte Entwurfsmuster nach Gamma et al. (1995). Entwurfsmuster sind Strukturmodelle von Informatiksystemen. Bisher werden sie nur in wenigen Publikationen zur Schulinformatik aufgegriffen, aber es wird in ihnen viel Potential gesehen (vgl. Humbert (2003), Schubert (2005), Magenheim (2005), Ullenboom (2005)).

Kompetenzentwicklung mit Informatiksystemen bedeutet unter anderem, dass Schüler Aufbau und Funktionsweise verstehen, Ursache-Wirkungsbeziehungen kennen und ablaufende Prozesse zur Erklärung und Bewältigung von Anforderungssituationen anwenden können. Dafür wird eine fachdidaktische Klassifikation angestrebt, deren Stärke in der informatiktypischen Repräsentation von Informatikkonzepten durch Entwurfsmuster liegt. Vorteil eines Entwurfsmuster-basierten Zugangs ist die vorliegende Musterbeschreibung. Schüler können jedes Detail bei Interesse untersuchen, wodurch Schülerzentrierung unterstützt wird. Darauf aufbauend können Schüler sich mit Variationen einzelner Entwurfsmuster beschäftigen und deren Kombinationsmöglichkeiten selbst erarbeiten. Für Kompetenzentwicklung mit Informatiksystemen ist hervorzuheben, dass Entwurfsmuster eines Kontextes bedürfen, d. h. eines objektorientierten Modells. Daraus folgt beispielsweise, dass das Kompositummuster aufgrund der objektorientierten Denkweise eine informatiktypische Repräsentation ist, um Rekursion darzustellen, während Fibonacci-Zahlen mathematiktypisch sind. Die informatiktypische Repräsentation erlaubt es Lernenden, ein kognitives Modell von fundamentalen Ideen der Informatik als Teil eines Informatiksystems zu entwickeln. Deshalb bieten Entwurfsmuster nach Meinung des Autors ein hohes Potential, um Aufbau und Funktionsweise von Informatiksystemen zu verstehen und Kompetenzen zu entwickeln. Sie müssen aber trotz und gerade wegen des für die Systemstrukturierung notwendigen hohen Abstraktionsgrads für Schüler in der Sekundarstufe II zugänglich sein, und eine Auswahl muss fachdidaktisch begründet werden.

Ein weiteres Beispiel ist, dass die Schüler mit grafischen Benutzungsoberflächen arbeiten und ihre Elemente kennen (GI 2008, S. 38). Humbert (2003) nutzt in diesem Zusammenhang das Architekturmuster Model-View-Controller (MVC) und das Beobachtermuster als Mittel, um die Trennung von Datenhaltung und Darstellung in der Benutzungsoberfläche zu vermitteln (Abschnitt 4.2.4). Das MVC wird in dem Kontext auch im Informatiklehrplan für die Sekundarstufe II der Stadt Hamburg genannt. Explizit wird gefordert, dass Schüler des Leistungskurses in der Lage sind, Entwurfsmuster und speziell das MVC-Muster zur Gestaltung von Benutzungsoberflächen einzusetzen (Bluhm et al. 2004). Damit ist der Einsatz von Entwurfsmustern auf einen kleinen Bereich der Modellierung beschränkt und nur für leistungsstarke Schüler vorgesehen. Dennoch finden sich zu dem Muster auch

auf mehreren Bildungsservern und Webseiten von Studienseminaren Unterrichtsmaterialien (vgl. Ullenboom 2005, S. 25) sowie in der Zeitschrift LOG IN, z. B. von Hermes (2003). Im Folgenden wird nachgewiesen, dass ausgewählte Entwurfsmuster vernetzte fundamentale Ideen repräsentieren und wie sie Kompetenzentwicklung mit Informatiksystemen fördern können. Sie werden als Wissensrepräsentationen bezeichnet. Zur Klassifikation der Entwurfsmuster sind Kriterien notwendig. Daher werden aus der Analyse des Forschungsstandes sowie fachwissenschaftlichen und fachdidaktischen Klassifikationen von Entwurfsmustern Kriterien abgeleitet, bei deren Erfüllung ein Beitrag zur Förderung der Kompetenzentwicklung mit Informatiksystemen von Entwurfsmustern erwartet wird, z. B.

→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik.

Die Ergebnisse werden in Abschnitt 5.4.1 ab Seite 171 vorgestellt. Nach Forneck erfordert Verstehen, dass Schüler selbst Verantwortung für ihr Lernen übernehmen (Forneck 1997, S. 25). Dies führt zur Lernerzentrierung des Informatikunterrichts und heißt unter anderem, dass Schüler ihr Lernen selbst organisieren müssen. Deshalb müssen drei Ebenen betrachtet werden:

a) Schüler der Sekundarstufe II als Lernende,

b) Didaktik der Informatik als Teil der Informatik,

c) Informatik als Fachwissenschaft.

Dabei sind a) und b) interessiert an Lernstrategien und neuen kognitiven Zugängen, während b) und c) Informatiksysteme zum Gegenstand haben.

Der Beitrag von Entwurfsmustern als Strukturmodelle zur Kompetenzentwicklung mit Informatiksystemen

Strukturmodelle von Informatiksystemen wurden als ein Mittel zur Förderung der Kompetenzentwicklung mit Informatiksystemen identifiziert (Abschnitt 5.3). In Anlehnung an eine Definition aus der Architektur (Alexander et al. 1977) wurden Mitte der Neunziger Jahre für die Softwaretechnik Entwurfsmuster entwickelt (Gamma et al. 1995). Heide Balzert charakterisiert sie folgendermaßen:

> „Ein Entwurfsmuster gibt eine bewährte, generische Lösung für ein immer wiederkehrendes Problem an, das in bestimmten Situationen auftritt. Es lassen sich klassen- und objektbasierte Muster unterscheiden. Klassenbasierte Muster werden durch Vererbungen ausgedrückt. Objektbasierte Muster beschreiben in erster Linie Beziehungen zwischen Objekten (, die zur Laufzeit geändert werden können; Einschub von S. 283)" (Balzert 1999, S. 53).

Damit nimmt Balzert Bezug auf die Klassifikation von Entwurfsmustern nach Gamma et al. (1995). Der Autor ist aufgrund der Analyse des Informatiksystembegriffs

und des fachdidaktischen Forschungsstandes zu Informatiksystemen und Kompetenzentwicklung davon überzeugt, dass Software- und Rechnerarchitekturen sowie die dazugehörenden Wirkprinzipien Kompetenzentwicklung mit Informatiksystemen befördern. Sie umfassen Beschreibungen des Systemverhaltens und Repräsentationen von Daten, Operationen und Funktionseinheiten. Die Struktur von Komponenten und ihren Schnittstellen wird durch sie für den Lernenden verständlich. Darüber hinaus sind jedoch weitere Betrachtungen zu Struktur der Funktionseinheiten notwendig. Insbesondere Schichtenarchitekturen und objektorientierte Architekturen bieten das Potential, die Vernetzung in und von Informatiksystemen zu beschreiben. Ein traditioneller Zugang wie mit Blockmodellen des Von-Neumann-Rechners allein verfehlt dieses Ziel zur Förderung der Kompetenzentwicklung mit Informatiksystemen, da er meist einen isolierten Rechner impliziert und die Hardwarekomponenten überbetont.

Für Kompetenzentwicklung mit Informatiksystemen soll im Folgenden theoretisch begründet und praktisch evaluiert werden, inwieweit Entwurfsmuster den Lehr-Lernprozess fördern. Generell sind solche Struktur- und vor allem Problemlösemuster in der informatischen Bildung von besonderer Bedeutung. Beim algorithmenorientierten Ansatz waren solche Strukturen beispielsweise Summenbildung, Sortieren und Berechnung des Maximums bzw. Minimums (Schubert und Schwill 2004, S. 224). Sowohl bei Standardarchitekturen wie auch bei Entwurfsmustern nach Gamma et al. (1995) liegen Regeln vor, wie gute Strukturen zu bilden sind, insbesondere hinsichtlich eines softwaretechnischen Qualitätsbegriffs. Diese Kriterien gehen einher mit Charakteristika der Objektorientierung. Es sind vor allem die Bündelung zusammen gehörender Aufgaben in einem Objekt, Entkopplung von Objekten, Trennung von Schnittstelle und Implementierung sowie Abstraktion. Entwurfsmuster basieren zudem auf Heuristiken, d. h. Erfahrungen aus der Softwareentwicklung hinsichtlich bewährter und wieder verwendbarer Lösungen.

Ein kognitiver Zugang über Entwurfsmuster als vorgegebene Systemstrukturierungen reduziert Komplexität, indem Wissen zur Problemlösung strukturiert wird. Wiederkehrende Probleme in objektorientierten Softwareentwürfen sind charakteristisch für bestimmte Teilsysteme. Vorarbeiten zu Entwurfsmustern in der Hochschullehre wurden von Schubert (2005) geleistet (Abschnitt 4.3.4). Darin wird der Beitrag der Entwurfsmuster zur Erklärung von Strukturen auf einer mittleren Ebene zwischen den elementaren Konzepten wie Datenstrukturen und der umfassenden Architektur beschrieben. Die Abstraktion von Implementierungsaspekten ist ein wichtiger Beitrag für Kompetenzentwicklung mit Informatiksystemen, der durch Strukturmodelle wie Entwurfsmuster geleistet wird (→ Kriterium 1: Abstraktion von Implementierungsaspekten).

Entwurfsmuster liegen somit auf einer Mesoebene der Strukturmodelle. Sie zeichnen sich dadurch aus, dass sie kombiniert werden können (pattern language), um ein Teilsystem zu beschreiben. Schichtenarchitekturen und Prozesse werden von

Schubert und Schwill gefordert für Unterricht zu Wirkprinzipien von Informatiksystemen. Architekturen, Schichtenmodelle und Entwurfsmuster lassen sich für Kompetenzentwicklung mit Informatiksystemen verknüpfen. Gerade Entwurfsmuster betonen die Vernetzung (→ Kriterium 4: Zusammenhänge mit anderen Strukturmodellen).

Davon ausgehend sollen Entwurfsmuster in die Konzipierung von Lehr-Lernprozessen einbezogen werden. Humbert stellt die Abstraktion zur Beschreibung von Strukturen als wichtiges Prinzip der Informatik heraus und schlägt den Bogen zu Entwurfsmustern als Abstraktion von wiederkehrenden Problem-Lösungsansätzen:

> „Zur Entwicklung von dekontextualisierten, operationalen Strukturen hat die Informatik besondere Abstraktionsmechanismen, namentlich die Prozessabstraktion und die Datenabstraktion entwickelt. Diese sind i. d. R. als parametrisierte Algorithmen beschrieben, die auf allgemeinen Datenstrukturen operieren. Für den Bereich der Objektorientierung konnten mit Entwurfsmustern (vgl. Gamma et al. (1995)) erste Ansätze für die Darstellung und Katalogisierung typischer wiederkehrender Lösungsmuster vorgelegt werden" (Humbert 2003, S. 20).

Humbert unterstreicht, dass solche

> „Lösungsmuster [...] ein allgemeineres Hilfsmittel für die Softwareentwicklung dar[stellen] und [...] zudem programmiersprachenunabhängig ausgelegt [sind] (auch wenn gewisse Muster ihren Bezug zu Defiziten konkreter Sprachen nicht verleugnen können – z. B. Singleton-Muster" (Humbert 2003, S. 20).

Damit wird unterstrichen, dass die meisten Entwurfsmuster von Implementierungsaspekten abstrahieren, eine Prüfung darauf ist jedoch vorzunehmen (→ Kriterium 1: Abstraktion von Implementierungsaspekten). Bisher sind Entwurfsmuster im Schul- und Hochschulbereich weitgehend hinsichtlich der Zielsetzung Softwareentwicklung eingesetzt worden. Für Kompetenzentwicklung mit Informatiksystemen muss dies nicht der einzige Zugang sein. Denn Entwurfsmuster leisten einen Beitrag zur Beschreibung der Vernetzung eines Teilsystems. In diesem Sinne unterstützen Entwurfsmuster vernetztes Denken auf einem hohen Abstraktionsgrad zur Problemlösung.

Fasst man diesen Abschnitt zusammen, so führt die Eigenschaft eines Entwurfsmusters, ein Strukturmodell zu sein, zur Auswahl von zwei Kriterien: → Kriterium 1: Abstraktion von Implementierungsaspekten; → Kriterium 4: Zusammenhänge mit anderen Strukturmodellen. Damit wird die Programmiersprachenunabhängigkeit gestärkt und die Bedeutung der inneren Struktur gegenüber Implementierungsaspekten hervorgehoben. Außerdem unterstützen Hinweise auf Zusammenhänge mit anderen Modellen die Unterrichtsgestaltung dahingehend, dass die Bildung von Wissensinseln vermieden werden kann.

Fundamentale Ideen und vernetztes Denken als Beitrag zur Kompetenzentwicklung mit Informatiksystemen

Für die Schulinformatik ist es unabdingbar, dass Architekturen und Muster nicht nur durch die typischen Veränderungen in Anforderungen bei der professionellen Entwicklung von mittelgroßen und großen Systemen motivierbar sind, sondern ein Allgemeinbildungswert erfüllt ist. Damit ist klar, dass nicht alle Muster gleichermaßen für den Informatikunterricht in der Sekundarstufe II geeignet sein werden.

Vernetzte fundamentale Ideen der Informatik bilden einen Schwerpunkt des vorliegenden Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung (Abschnitt 5.3). Sie werden als ein Kriterium zur Klassifikation von Entwurfsmustern herangezogen (→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik), da sie den Bildungswert für die informatische Bildung und Kompetenzentwicklung mit Informatiksystemen garantieren. Das Lernen von isolierten fundamentalen Ideen der Informatik entspricht nicht den Anforderungen, die an Informatikunterricht zur Kompetenzentwicklung mit Informatiksystemen gestellt werden (Abschnitt 4.4). Deshalb ist bei Entwurfsmustern deren Abstraktion einer Problemlösung mittels vernetzter fundamentaler Ideen herauszustellen. Damit die Problemlösung für Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II angemessen ist, wird als weiters Kriterium ein Lebensweltbeispiel gefordert (→ Kriterium 6: Lebensweltbezug).

Fundamentale Ideen sind in den Betätigungsfeldern der Didaktik der Informatik (b) und Informatik (c) wichtig (siehe Seite 161), denn sowohl in der Informatikdidaktik als auch im Fach Informatik werden Beziehungen und Verbindungen zwischen fundamentalen Ideen erforscht und begründet. Vernetzte fundamentale Ideen liefern Potential zur Entwicklung des vernetzten Denkens der Schüler. Für den Informatikunterricht in der Sekundarstufe II rückt das vernetzte Denken auch durch Lehrplanforderungen in den Vordergrund:

> „Wissenschaftspropädeutisches Arbeiten erfordert problem- und prozessbezogenes Denken und Denken in Zusammenhängen" (MSWWF 1999, S. XII).

Dabei kommt der Verteiltheit und der Interaktion von Subsystemen eine besondere Rolle zu. Objektorientierte Entwurfsmuster sind Lernmittel für vernetzte Informatiklösungen, in denen durch Aufgabenteilung der teilnehmenden Objekte vernetztes Denken erforderlich ist.

Entwurfsmuster stellen ein objektorientiertes Modell einer Problemlösung dar, die in der Regel nicht zentral verwaltet wird. Guzdial (1995) spricht, unter Bezug auf Resnicks Dissertationsschrift (Resnick 1992), davon, dass Informatiklernende an Hochschulen oft zentralgesteuerte Problemlösungsansätze präferieren würden (centralized mindset). Diese zentralgesteuerten Problemlösungen seien einfach zu verstehen und bei kleinen Systemen effizient, entsprächen aber in der Denkweise nicht

den Ansprüchen an objektorientiertes Modellieren. Guzdial schlägt Reflexion der programmierten Lösungen, Quelltextevaluation, Kenntnis der Bibliotheken und das Bereitstellen dezentraler Beispiellösungen vor. Gerade letzteres ist zur Förderung der Kompetenzentwicklung mit Informatiksystemen sicherlich hilfreich, da dahinter nicht nur Implementierungsaspekte stehen müssen. Dezentrale Modelle führen zu einem besseren Verständnis verteilter und paralleler Prozesse wie zum Beispiel die Kooperation von Objekten mit klarer Aufgabenteilung. Auch bei der Analyse von Informatiksystemen ist es wichtig, dezentral und somit vernetzt zu denken:

> „Gemeint ist damit ein integrierendes, zusammenfügendes Denken, das auf einem breiteren Horizont beruht, von grösseren Zusammenhängen ausgeht und viele Einflussfaktoren berücksichtigt, das weniger isolierend und zerlegend ist als das übliche Vorgehen. Ein Denken also, das mehr demjenigen des viele Dinge zu einem Gesamtbild zusammenfügenden Generalisten als dem analytischen Vorgehen des auf ein enges Fachgebiet beschränkten Spezialisten entspricht" (Ulrich und Probst 1988, S. 11).

Mit Hinweis auf Möller (1999) kommt Schulte für die Objektorientierung zu folgendem Fazit:

> „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" (Schulte 2001, S. 6).

Denn Ziel sei die Erstellung eines objektorientierten Systems in dem die interagierenden Objekte sinnvoll miteinander vernetzt werden müssen. Im Gegensatz zu imperativen Programmen ist die Dezentralisierung Grund für die Vernetzung. Schulte betont allerdings nur für Softwareentwickler die Notwendigkeit des vernetzten Denkens. Nach Meinung des Autors kann das gleiche jedoch auch für Kompetenzentwicklung mit Informatiksystemen gefordert werden. Unterstützt wird diese Sicht durch Wallingford, der zur Stärkung einer vernetzten Lösung für die Anfangskurse in der Hochschulinformatik objektorientierte Entwurfsmuster vorschlägt, die aber nicht aus dem Katalog von Gamma et al. (1995) kommen:

> „One of the key goals in this effort [is] to help students overcome the centralized mindset that they bring to problem solving. [...] OO patterns provide a mechanism for seeing solutions to problems in terms of **distributed objects with distributed control**" (Wallingford 1996, S. 30; Hervorhebung durch den Autor).

Mit diesen Zitaten wird die Vermutung gestützt, dass Entwurfsmuster die in der Analyse des Forschungsstandes identifizierte Vernetzung (Abschnitt 4.4) unterstützen können. Die Vernetzung von Informatikkonzepten ist wissenschaftspropädeutisch notwendig und liegt in Informatiksystemen vor. Der Bezug zum Denken in Modellen und zur Erfassung von Dynamik ist auch wichtig. Darüber hinaus schlagen Entwurfsmuster eine Verbindung zur Objektorientierung, in der ebenfalls das

Potential gesehen wird, vernetztes Denken zu fördern. Die didaktische Herausforderung besteht darin, ein erfolgreiches Unterrichtsmodell zu Informatiksystemen und Kompetenz zu entwickeln, dass eben dieses vernetzte Denken fördert. Im Folgenden werden die Klassifikationskriterien von Entwurfsmustern in der Fachwissenschaft analysiert und auf ihre Eignung als Kriterium zur Förderung der Kompetenzentwicklung mit Informatiksystemen geprüft.

Zusammenfassend führt dieser Abschnitt zur Auswahl und Begründung von zwei Kriterien: → Kriterium 3: Vernetzte fundamentale Ideen der Informatik; → Kriterium 6: Lebensweltbezug. Fundamentale Ideen dienen als Qualitätsmerkmale und sichern den „Bildungswert". Vor allem ist über fundamentale Ideen der Einsatz von Entwurfsmustern in der Sekundarstufe II begründbar. Die Vernetzung wiederum vermeidet das Lernen isolierter fundamentaler Ideen zur Erklärung eines Systems. Eine Analogie aus der Lebenswelt der Schüler, die das Zusammenwirken der Komponenten des Strukturmodells erklärt, unterstützt den Lernprozess. Das Lebensweltbeispiel kann, muss aber nicht notwendiger Weise direkten Bezug zu einem Informatiksystem und dessen Verhalten haben.

Klassifikation von Entwurfsmustern in der Fachwissenschaft

In diesem Abschnitt werden fachliche Musterkataloge kurz vorgestellt und wie Entwurfsmuster darin fachlich klassifiziert sind. Die Kriterien zur Systematisierung der Muster in der Fachwissenschaft werden darauf geprüft, ob sie zur fachdidaktischen Klassifikation einen Beitrag leisten können. Neben der Klassifikation von Entwurfsmustern nach Gamma et al. (1995), die als Wegbereiter gilt, gibt es auch weitere in der Fachwissenschaft, z. B. von Buschmann et al. (1996), Zimmer (1995) und Tichy (1997), die im Folgenden kurz anhand ihrer Klassifikationskriterien beschrieben werden. Die Auswahl der Kataloge erhebt dabei keinen Anspruch auf Vollständigkeit, da ihre Anzahl zu immer spezialisierteren Anwendungsbereichen weiterhin ansteigt, so existieren mittlerweile auch Muster für den Hardwareentwurf (Rincon et al. 2005).

In Entwufsmusterkatalogen werden Muster einheitlich beschrieben mithilfe eines Beschreibungsschemas mit Mustername, Problemabschnitt, Lösungsabschnitt und Konsequenzenabschnitt. Diese sind bei Gamma et al. (1995) zur Beschreibung eines Musters weiter unterteilt in Zweck, alternative Bezeichnungen, Motivation, Anwendbarkeit, Struktur, Teilnehmer, Interaktionen, Konsequenzen, Implementierung, Beispielcode, bekannte Verwendungen sowie verwandte Muster.

Bei Gamma et al. (1995) wird auch der Gesamtkatalog unterteilt in die zwei Dimensionen Gültigkeitsbereich (Vererbung oder Objektaggregation) und Zweck bzw. Aufgabe (Erzeugungs-, Struktur- oder Verhaltensmuster).

> „Wir klassifizieren Muster mittels zweier Kriterien [...]. Das erste Kriterium namens **Aufgabe** gibt wieder, was das Muster macht. Muster können entweder eine

> **erzeugende**, eine **strukturorientierte** oder eine **verhaltensorientierte** Aufgabe haben. [...] Das zweite Kriterium, der **Gültigkeitsbereich**, legt fest, ob ein Muster sich primär auf Klassen oder auf Objekte bezieht. [...] Praktisch alle Muster greifen zu einem gewissen Grad auf Vererbung zurück. Somit sind die einzigen klassenbasierten Muster jene, die sich auf Klassenbeziehungen konzentrieren" (Gamma et al. 1995, S. 14f; Hervorh. im Original).

Tabelle 5.1 zeigt die Klassifikation der Muster nach Gamma et al. (1995). Für Kompetenzentwicklung mit Informatiksystemen ist sicherlich der Zweck des Entwurfsmusters relevant (→ Kriterium 2: Zweck und Einsatzgebiet). Falls er ausschließlich in der Softwareentwicklung relevant ist, kann das Kriterium als Ausschlusskriterium für ein Muster dienen. Steinert vermutet, dass Erzeugungsmuster nicht für die Schule geeignete sind, weil sie in großem Maße Kenntnisse der zugrunde liegenden Programmiersprache fordern (Schneider 2003, S. 168). Ihr Zweck ist die Generierung von Objekten, die vom konkreten Einsatz des Objektes und der konkreten Implementierung entkoppelt wird (→ Kriterium 1: Abstraktion von Implementierungsaspekten).

Tabelle 5.1: Musterkatalog nach (Gamma et al. 1995, S. 14)

		Aufgabe		
		Erzeugungsmuster	**Strukturmuster**	**Verhaltensmuster**
Gültigkeitsbereich	klassenbasiert	Fabrikmethode	Adapter	Interpreter, Schablonenmethode
	objektbasiert	Abstrakte Fabrik, Erbauer, Prototyp, Singleton	Adapter, Brücke, Dekorierer, Fassade, Fliegengewicht, Kompositum, Proxy	Befehl, Beobachter, Besucher, Iterator, Memento, Strategie, Vermittler, Zustand, Zuständigkeitskette

Bei Buschmann et al. (1996) werden ebenfalls zwei Dimensionen unterschieden:

> „A closer look at existing patterns reveals that they cover various ranges of scale and abstraction" (Buschmann et al. 1996, S. 11).

Zur Einordnung in Problemkategorie (scale) unterscheiden sie strukturelle Dekomposition, Arbeitsorganisation, Zugangskontrolle, Management, Kommunikation, Strukturierung, verteilte Systeme, adaptierbare Systeme und interaktive Systeme. Zur Verfeinerung nutzen sie den Abstraktionsgrad (abstraction), in dem sie Architekturmuster, Entwurfsmuster und programmiersprachenabhängige Idiome gegeneinander abgrenzen. Für Kompetenzentwicklung mit Informatiksystemen wurde bereits gefordert, von Implementierungsaspekten zu abstrahieren (→ Kriterium 1: Abstraktion von Implementierungsaspekten).

Betont werden die Abhängigkeiten der unterschiedlichen Teilnehmerobjekte (→ Kriterium 4: Zusammenhänge mit anderen Strukturmodellen). Für Unterricht sind

die Zusammenhänge der Muster auf Anknüpfungen für weiteren Unterricht zu untersuchen. Zum Beispiel kann das Beobachtermuster in der MVC-Architektur identifiziert werden. Die MVC-Architektur zeigt das Konzept der Schichten und seine Verbindung zu Entwurfsmustern. Die drei Schichten realisieren eine Trennung der Daten (Model), der Benutzungsoberfläche (View) und Aspekten der Interaktion (Controller). Sie repräsentieren damit die fundamentale Idee Modularisierung (→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik).

Weitere Klassifizierungen sind die von Zimmer und Tichy. Insbesondere erstgenannte ist sehr interessant, da sie veranschaulicht, welches Muster sich auf welchen anderen abstützt. Zimmer (1995) klassifizierte die Muster nach Gamma et al. (1995) grafisch anhand ihrer Beziehung mit anderen Entwurfsmustern: Muster X nutzt Muster Y in seiner Lösung, Muster X ähnelt Muster Y, Muster X kann mit Muster Y kombiniert werden. Diese Klassifikation ist in die Fachdidaktik, d. h. auf die Strukturierung des Lehr-Lernprozesses übertragbar. Die ersten beiden Beziehungen sind nützlich hinsichtlich der Beschreibung von Vorkenntnissen der Lernenden. Die dritte Relation weist auf Anknüpfungspunkte für weiteren Unterricht hin (→ Kriterium 4: Zusammenhänge mit anderen Strukturmodellen).

Tichy (1997) entwirft einen Katalog, der sich, ähnlich wie bei Buschmann et al. (1996), an der Art der Problemlösung orientiert, und unterscheidet beispielsweise zwischen Entkopplungsmustern und Kontrollflussmustern. Als Bereiche werden genannt: Decoupling, Variant Management, State Handling, Control, Virtual Machines, Convenience Patterns, Compound Patterns, Concurrency, Distribution. Für Kompetenzentwicklung mit Informatiksystemen ist dieser Bezug zu bestimmten Klassen von Informatiksystemen wichtig (→ Kriterium 2: Zweck und Einsatzgebiet).

Zusammenfassend führt dieser Abschnitt zur Auswahl und Begründung eines neuen Kriteriums: → Kriterium 2: Zweck und Einsatzgebiet. Über das Kriterium ist ein Bezug zum nach außen sichtbaren Verhalten von Informatiksystemen möglich. Außerdem werden weitere Begründungen für drei bereits ausgewählte Kriterien geliefert: → Kriterium 1: Abstraktion von Implementierungsaspekten; → Kriterium 3: Vernetzte fundamentale Ideen der Informatik; → Kriterium 4: Zusammenhänge mit anderen Strukturmodellen.

Klassifikation von Entwurfsmustern in der Fachdidaktik

Aus Sicht der Hochschuldidaktik der Informatik klassifizieren Harrer und Steinert Entwurfsmuster. Sie betonen, dass sich die Klassifikation bzw. die Systematik der bei Gamma et al. (1995) und Buschmann et al. (1996) vorgestellten Muster an den Bedürfnissen der Softwaretechnik orientiert. Für beide Kataloge wurden fachliche Kriterien gefunden. Dahingegen sind aus fachdidaktischer Sicht die mit den Mustern vermittelbaren grundlegenden Konzepte der Informatik und ihr Komplexitätsgrad für eine solche didaktische Systematik unabdingbar. Nach Harrer und

Steinert ist die Klassifizierung von Gamma u. a. fachdidaktisch nicht begründet, da hier viele Muster zusammengefasst werden, die untereinander wenig Ähnlichkeit aufweisen. Die Klassifizierung anhand des Abstraktionsgrads bei Buschmann u. a. ist jedoch für den Unterricht sinnvoll, um eine Grobeinteilung der Muster vorzunehmen. In der Problemkategorie werden allerdings Systemklassen, wie verteilte und interaktive Systeme, mit Konzepten, wie Kommunikation und Zugriffsregelung, vermischt. Ausgehend von einem Top-Down Ansatz der Softwaretechnik (von der Architektur über Entwurfsmuster zur Verwaltung von Objekten mit Idiomen) entwickeln sie eine didaktische Klassifikation für die Hochschulinformatik, die die Konzepte Modularisierung bzw. Dekomposition, Wiederverwendung, Modellierung der Interaktion und Objekt-Management den Mustern zuordnet. Diese bei der Softwareentwicklung wichtigen Aspekte sind damit eine Grundlage ihrer fachdidaktischen Klassifikation für die Hochschule. Für die Konzepte Interaktion und Dekomposition geben Harrer und Steinert jeweils eine übergeordnete Musterstruktur an, d. h. eine Verallgemeinerung der in allen Mustern, denen dieses Konzept innewohnt, vorhandenen Struktur. Bei den übergeordneten Strukturen zu Dekomposition und Interaktion gelingt dies sehr gut, da sofort die Ähnlichkeit mit den aus Gamma et al. (1995) bekannten Strukturdiagrammen ins Auge fällt. Für Wiederverwendung und Management geben sie jedoch nur an, welche Muster dazu gehören, ohne eine übergeordnete Struktur zu beschreiben.

Ein weiteres wichtiges Kriterium der Klassifizierung nach Harrer und Steinert ist die Komplexität, für die drei Gesichtspunkte eine Rolle spielen: Schwierigkeit des Verständnisses bei Betrachtung der Gesamtheit des Musters, Umfang der Diskussionspunkte wie Einsatz und Auswirkungen und als letzter Punkt die Anzahl der Variationsmöglichkeiten des Musters. Schwierigkeit bei diesem Kriterium ist die mangelnde Eindeutigkeit der Einstufung. Harrer und Steinert unterscheiden die drei Stufen leicht, mittel und schwer. Zur Übertragung der Ergebnisse auf die Sekundarstufe II ist es nach Ansicht des Autors notwendig, Erprobungen im Informatikunterricht durchzuführen, um Aussagen über die Komplexität des Musters zu treffen. Allerdings kann eine theoretische Analyse der Komplexität Anhaltspunkte für den Einsatz des Musters im Informatikunterricht bieten (→ Kriterium 5: Komplexität).

Anschließend erweitern Harrer und Steinert ihre Systematik um die Dimensionen „Art", die dem Abstraktionsgrad bei Buschmann et al. (1996), und „Einsatzgebiet", die etwa der Problemkategorie bei Buschmann et al. (1996) entspricht. Somit haben Harrer und Steinert in ihrer Systematik die Systemklassen und die informatischen Konzepte sorgfältig getrennt:

> „Statt der Dimension der *Problemkategorie* definieren wir eine Dimension **Einsatzgebiet**, die ausdrückt, ob ein Muster allgemein einsetzbar ist oder für ganz spezielle Systemtypen und Umgebungen charakteristisch ist (z. B. verteilte Systeme, interaktive Systeme). Dadurch haben wir in unserer Klassifizierung Systemklassen und

> informatische Konzepte sauber getrennt, wohingegen in (Buschmann et al. 1996) eine Vermischung dieser Aspekte vorliegt" (Harrer und Schneider 2002, S. 73; Hervorh. im Original)

Um die Muster zu lehren, wenden sie das induktive Vorgehen (Bottom-up; vom Quelltext ausgehend) hauptsächlich für Erzeugungsmuster an. Das deduktive Vorgehen, das von der abstrakten Lösungsidee ausgeht, kommt bei einfachen Architekturmustern zum Einsatz. Zur Förderung der Kompetenzentwicklung mit Informatiksystemen ohne Betrachtung der Implementierungsaspekte ist dieses Vorgehen nahe liegend, da die Konstruktion eines Systems nicht im Fokus liegt. Für komplexe Architekturmuster wird von Harrer und Steinert vorschlagen, Details bzw. Komponenten nacheinander, aber mit Bezug zu den bekannten Komponenten zu behandeln, so dass die Gesamtarchitektur sich erst am Ende zusammen setzt. Abschließend stellen Harrer und Steinert den Bezug zum Schulfach Informatik her. Sie betonen, dass Entwurfsmuster insbesondere Problemstellungen bei der objektorientierten Modellierung lösen und einige Entwurfsmuster, die ausgewählte Konzepte auf einem angemessenen Schwierigkeitsgrad repräsentieren, für die Sekundarstufe II interessant sind (Harrer und Schneider 2002, S. 76). Welche Muster dies sein könnten, wird nicht erwähnt.

Steinert stellt weiterhin die Frage, inwieweit Entwurfsmuster der Objektorientierung bereits in der Sekundarstufe I in den Unterricht integriert werden können (Schneider 2003). Hierbei bezieht er sich auf den neu konzipierten Informatikunterricht an Bayerischen Gymnasien. Steinert empfiehlt die Entwurfsmuster Kompositum und Beobachter nach Gamma et al. (1995) für den Informatikunterricht in der Sekundarstufe I und formuliert Kriterien für die Auswahl von Entwurfsmustern:

> 1. „The pattern has to represent a substantial concept of informatics.
> 2. The pattern has to be of general interest. Special patterns, which are utilised in some specific context only, are normally not suited.
> 3. The pattern has to provide real support to the teaching of the represented concept and therefore provide a didactical function.
> 4. The pattern should not be too complex such that the students would not understand it easily or fully.
> 5. The understanding of the respective pattern should not require too much background information about a specific programming language" (Schneider 2003, S. 167f; Nummerierung durch den Autor).

Deren Anwendung führt ihn außerdem zur Empfehlung der Entwurfsmuster Interpreter und Zustand. Die Kriterien müssen jedoch für Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II diskutiert werden. Aus den vorherigen Überlegungen zum Stand der Forschung (Kapitel 4) ergibt sich, dass statt eines wichtigen Konzeptes wie in Ziffer 1 gefordert, vernetzte fundamentale Ideen der Informatik in dem Entwurfsmuster zu identifizieren sind. Damit ist gleichzeitig der Bildungswert begründet. Forderung 2 ist durch fundamentale Ideen ebenfalls

automatisch erfüllt. Der erste Teil von Ziffer 2 kann für Kompetenzentwicklung mit Informatiksystemen dahingehend konkretisiert werden, dass ein Beitrag zu den Hauptfunktionen von Informatiksystemen, ihrem Aufbau und Wirkprinzipien anzustreben ist. Dies kann über die Einordnung in die Klassifikation der Unterrichtsinhalte nach Hubwieser und Broy erfolgen (Hubwieser 2007a, S. 83). Ziffer 5 ist insofern konform zu den Zielen zur Kompetenzentwicklung mit Informatiksystemen, als dass Implementierungsdetails zur Softwareentwicklung nicht im Fokus der vorliegenden Arbeit liegen. In Ziffer 3 wird hervorgehoben, dass Entwurfsmuster nicht um ihrer selbst willen, sondern nur im Unterricht eingesetzt werden sollten, wenn sie eine didaktische Funktion hinsichtlich des Lernens eines Informatikkonzeptes erfüllen.

Zusammenfassung der fachlichen und fachdidaktischen Klassifikationen ist, dass die vorhandenen Entwurfsmusterkataloge nicht nach fachdidaktischen Kriterien zur Förderung der Kompetenzentwicklung mit Informatiksystemen zusammengestellt wurden und daher eine Auswahl von Entwurfsmustern vorgenommen werden muss. Dazu ist auf den Beitrag der Entwurfsmuster zur Kompetenzentwicklung mit Informatiksystemen und Angemessenheit für die Schule zu achten. Die fachwissenschaftlichen und fachdidaktischen Klassifikationen nach Buschmann et al. (1996) sowie Harrer und Steinert (Harrer und Schneider 2002) liefern jedoch Anregungen für Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II. Im nächsten Abschnitt werden die Analyseergebnisse zusammengefasst und Kriterien für eine fachdidaktische Klassifikation zur Förderung der Kompetenzentwicklung mit Informatiksystemen angegeben.

Zusammenfassend führt dieser Abschnitt zur Auswahl und Begründung eines neuen Kriteriums: → Kriterium 5: Komplexität. Dieses Kriterium ist jedoch ambivalent: Einerseits ist durch die Betrachtung von Schwierigkeitsgrad, Variationsmöglichkeiten und notwendiger Vorkenntnisse der Schüler ein Beitrag für den Lehr-Lernprozess zu erwarten. Andererseits kann die Komplexität kaum als Auswahlkriterium herangezogen werden, da keine empirischen Ergebnisse zum Einsatz von Entwurfsmustern in der Sekundarstufe II zu Informatiksystemen und Kompetenzentwicklung existieren.

Fazit: Kriterien zur Klassifikation

In Abschnitt 5.4.1 wurden fachwissenschaftliche Entwurfsmusterkataloge auf die verwendeten Kriterien untersucht. Von den fachlichen Kriterien zur Klassifikation von Entwurfsmustern sind der Abstraktionsgrad und die Beziehungen bzw. Abhängigkeiten von Entwurfsmuster zueinander für Kompetenzentwicklung mit Informatiksystemen nützlich. So ist für Kompetenzentwicklung mit Informatiksystemen sicherzustellen, dass Implementierungsaspekte nicht im Vordergrund stehen, sobald das Muster im Informatikunterricht eingesetzt wird. Außerdem ist für die abstrakte Problemlösung ein Beispiel aus der Lebenswelt des Schülers zu finden.

Das hochschuldidaktische Kriterium von Harrer und Steinert, dass Konzepte wie Management und Interaktion in Mustern enthalten sein müssen, reicht für Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II nicht aus. Es sind vernetzte fundamentale Ideen der Informatik in Entwurfsmustern zu identifizieren (vgl. (Stechert 2006c), (Ufer 2007)). Außerdem muss die Relevanz für Informatiksysteme betrachtet werden (Hubwieser 2007a, S. 83), beispielsweise durch den Bezug zu den Hauptfunktionen eines Informatiksystems (Kapitel 3). Zur fachdidaktischen Klassifikation von Entwurfsmustern zur Förderung der Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II wurden folgende Kriterien identifiziert:

Kriterium 1 (Abstraktion von Implementierungsaspekten) Nur Entwurfs- und Architekturmuster sind zur Förderung von Basiskompetenzen zu Informatiksystemen relevant. Programmiersprachenspezifische Muster, so genannte Idiome, werden dementsprechend aussortiert. Damit wird die Programmiersprachenunabhängigkeit gestärkt und die Bedeutung der inneren Struktur gegenüber Implementierungsaspekten hervorgehoben.

Kriterium 2 (Zweck und Einsatzgebiet) Der Zweck bzw. das Einsatzgebiet des Musters ist hinsichtlich der Relevanz für Kompetenzentwicklung mit Informatiksystemen zu prüfen. Dieses Kriterium betrachtet die Verallgemeinerbarkeit der Problemlösung durch das Muster. Verwendungsmöglichkeiten und Zuordnungen zu den Kategorien der Informatik, die die Hauptfunktionen von Informatiksystemen abdecken (Kapitel 3), untermauern den Beitrag zur Kompetenzentwicklung mit Informatiksystemen. Es sind möglichst Muster zu identifizieren, die in unterschiedlichen Klassen von Informatiksystemen eingesetzt werden (vgl. Hubwieser und Broy 1997a, S. 44).

Kriterium 3 (Vernetzte fundamentale Ideen der Informatik) Durch fundamentale Ideen in Informatiksystemen wird sowohl der Bildungswert begründet als auch dem Anspruch genügt, Wirkprinzipien zu vermitteln. Fundamentale Ideen dienen somit als Qualitätsmerkmale und vor allem ist über sie der Einsatz von Entwurfsmustern in der Sekundarstufe II begründbar. Die Vernetzung wiederum vermeidet das Lernen isolierter fundamentaler Ideen zur Erklärung eines Systems. Die Vernetzung der fundamentalen Ideen beschreibt, wie das Entwurfsmuster eine Problemlösung zur Erfüllung seines Zwecks umsetzt, so dass ein Beitrag zur Förderung des vernetzten Denkens erwartet wird. Durch die Prüfung auf den Beitrag des Zwecks hinsichtlich des Systemverhaltens ist damit ein indirekter und durch die geforderte Vernetzung ein direkter Beitrag des Kriteriums zur Förderung der Kompetenzentwicklung mit Informatiksystemen begründbar. Die identifizierten fundamentalen Ideen dienen zur Gestaltung des Lehr-Lernprozesses und unterstützen die Lehrperson beim Setzen von Prioritäten. Dadurch, dass fundamentale Ideen

der Informatik speziell den nichtspezifischen Transfer unterstützen, ist ein Beitrag zur Kompetenzentwicklung zu erwarten.

Kriterium 4 (Zusammenhänge mit anderen Strukturmodellen) Durch die Kombination von Entwurfsmustern gelingt ein Beitrag zur Förderung des vernetzten Denkens als übergeordnetes Ziel der Kompetenzentwicklung mit Informatiksystemen. Die Vernetzung der Muster vermeidet die Bildung einzelner Wissensinseln. Zusätzlich können Perspektiven zur Erweiterung und Verknüpfung von Lernszenarien aufgezeigt werden. Bei der Kombination von Entwurfsmustern werden die in den Mustern enthaltenen fundamentalen Ideen der Informatik miteinander vernetzt. Neben den Kombinationsmöglichkeiten wird analysiert, ob ein Entwurfsmuster eine strukturelle Ähnlichkeit zu anderen Mustern aufweist.

Kriterium 5 (Komplexität) Hinsichtlich der Gesamtheit des Musters, der Variationsmöglichkeiten in unterschiedlichen Kontexten sowie der zum Verstehen notwendigen Aspekte für den Informatikunterricht ist die Komplexität einzustufen, zum Beispiel hinsichtlich der Aspekte nach Harrer und Steinert: Schwierigkeit des Verständnisses, Umfang der Diskussionspunkte und der Variationsmöglichkeiten (Harrer und Schneider 2002). Vorab kann die Einstufung theoretisch erfolgen, sie sollte jedoch empirisch auf ihre Tragfähigkeit überprüft werden. Da empirische Ergebnisse für die Sekundarstufe II nicht vorliegen, wird auf die Anwendung des Kriteriums der Komplexität und eine damit begründete Selektion von Mustern in der nachfolgenden Betrachtungen verzichtet. In den Unterrichtsmaterialien bzw. studentischen Arbeiten, die im Rahmen des Promotionsvorhabens des Autors erstellt wurden, werden Muster jedoch auf ihre Komplexität untersucht, um den Lehr-Lernprozess zu unterstützen ((Ufer 2007), (Weyer 2007b)). In Abschnitt 5.4.4 werden die Eigenschaften von Entwurfsmustern, Schemata und extern repräsentierbar zu sein, aufgegriffen, um Komplexität im Lernprozess zu bewältigen.

Kriterium 6 (Lebensweltbezug) Ein Lebensweltbeispiel, das das Zusammenwirken der Komponenten des Strukturmodells erklärt, unterstützt den Lernprozess. Das Lebensweltbeispiel kann, muss aber nicht notwendiger Weise direkten Bezug zu einem Informatiksystem und dessen Verhalten haben.

Im Folgenden werden Entwurfsmuster anhand der erarbeiteten Kriterien analysiert.

5.4.2 Klassifikation der Entwurfsmuster

Im Folgenden werden die Softwaremuster nach Gamma et al. (1995) nach obigen Kriterien klassifiziert. Das erste Kriterium dient im Voraus zum Ausschluss einiger Muster (→ Kriterium 1: Abstraktion von Implementierungsaspekten). Falls ein Muster zum größten Teil das Verstehen von Implementierungsaspekten unterstützt,

ist es für zur Förderung der Kompetenzentwicklung mit Informatiksystemen im Rahmen dieser Arbeit nicht weiter relevant und wird nicht betrachtet. Dies gilt vornehmlich für sämtliche Idiome, d. h. programmiersprachenabhängige Muster, die von Buschmann et al. (1996) identifiziert wurden: Counted Pointer, Singleton, Fabrikmethode und Schablonenmethode (Buschmann et al. 1996, S. 380). Darüber hinaus werden Erzeugungsmuster in dieser Arbeit nicht betrachtet (vgl. Schneider 2003), da sie in hohem Maße Programmierkenntnisse erfordern. Ihr Beitrag zum Verstehen von Implementierungsaspekten eines Informatiksystems sei damit nicht bestritten.

Die verbleibenden 17 Muster werden kurz mit ihrer wichtigsten Charakteristik vorgestellt. Zur besseren Übersicht wird zwischen Verhaltensmustern und Strukturmustern nach Gamma et al. (1995) unterschieden. Zu den in ihnen vorhandenen informatischen Konzepten bzw. fundamentalen Ideen wird zwischen Konzepten der Objektorientierung und weiteren informatischen Konzepten unterschieden. Erstere sind in objektorientierten Entwurfsmustern selbstverständlich und daher zur besseren Übersicht für die Planung des Lehr-Lernprozesses genannt. Unter einer fundamentalen Idee werden im Folgenden auch Kandidaten für fundamentale Ideen zusammengefasst, die in der Wissenschaft anerkannte Konzepte sind, deren Überprüfung auf die Kriterien nach Schwill jedoch aussteht (Abschnitt 4.2.2). Vor allem der Nachweis des Vertikalkriteriums, der im Zweifelsfall eine Überprüfung der Lernbarkeit auf Niveau der Primarstufe erfordert, kann aus Zeitgründen nicht im Rahmen dieser Arbeit geführt werden. Bei Mustern, zu denen im Laufe der Forschungsarbeit Unterrichtsmaterialien ausgearbeitet wurden, z. B. in studentischen Arbeiten, wird darauf hingewiesen. Zuerst werden die Strukturmuster nach Gamma et al. (1995) analysiert.

Adapter *Zweck und Einsatzgebiet:* Das Adaptermuster dient der Kompatibilität von Schnittstellen, indem es eine Schnittstelle den Erwartungen des Klienten anpasst (Gamma et al. 1995, S. 171).
Vernetzte fundamentale Ideen der Informatik: Konzepte der Objektorientierung im Muster sind Mehrfachvererbung oder Objektkomposition, Klassenschnittstelle sowie die enthält-Beziehung. Informatische Konzepte im Muster sind Wiederverwendbarkeit (anderer Klassen), Polymorphie , Kapselung und Emulation.
Zusammenhänge mit anderen Mustern: Der Adapter weist Ähnlichkeiten mit den Mustern Brücke, Dekorierer und Proxy auf.
Lebensweltbezug: Beispiel ist das Werkzeug Knarre zum Lösen oder Festziehen von Schraubverbindungen. Durch Aufsätze, d. h. Adapter, können unterschiedliche Schraubengrößen gelöst werden (Duell et al. 1998, S. 8).
Unterrichtsbeispiel im Rahmen des Unterrichtsmodells: Sülz (2007) betrachtet in seiner vom Autor betreuten Seminararbeit das Adaptermuster hinsichtlich der Kompatibilität als Qualitätsanforderung an Informatiksysteme.

Brücke *Zweck und Einsatzgebiet:* Das Brückmuster trennt eine Abstraktion von ihrer Umsetzung, indem sowohl eine Klassenhierarchie der Schnittstellen als auch Klassenhierarchien der jeweiligen Ausprägung angelegt werden, um beide getrennt voneinander anpassen zu können (Gamma et al. 1995, S. 186). Ein Klient interagiert mit einer konkreten Umsetzung einer Abstraktion. Die Kommunikation läuft dabei über die Abstraktion, die Anfragen des Klienten zu einem Objekt der geeigneten Umsetzung weiterleitet. Das Muster ist beispielsweise geeignet, eine Klassenbibliothek für verschiedene Datenstrukturen zu beschreiben.

Vernetzte fundamentale Ideen der Informatik: Konzepte der Objektorientierung im Muster sind Objekte, zwei parallele Vererbungshierarchien, Selbstverantwortlichkeit von Objekten, Abstrakte Klasse. Weitere Eigenschaften sind Wiederverwendbarkeit und das Geheimnisprinzip durch die Trennung der beiden Klassenhierarchien. Damit sind als informatische Konzepte die fundamentale Idee der Modularisierung und Schichtung in dem Entwurfsmuster enthalten, und es erfüllt Aspekte der Masteridee der strukturierten Zerlegung. Darüber hinaus sind zum Verständnis des Brückemusters Aspekte der Algorithmisierung notwendig. Über Parameter wird festgelegt, welche Umsetzung zu wählen ist. Damit sind gleichzeitig die fundamentale Idee der Alternative und auch das Prinzip der Polymorphie bzw. des dynamischen Bindens enthalten. Polymorphie ist ein relevantes Konzept, um Systemverhalten zu analysieren: Abhängig von Kontext und Parametern wird im System dynamisch entschieden, welches Verhalten zu wählen ist. Schüler werden mit verändertem Systemverhalten konfrontiert. Polymorphie ist deshalb wichtig, um Schnittstellen und ggf. Vererbungshierarchien zu verstehen. Weitere informatische Konzepte im Muster sind Trennung von Abstraktion und Implementierung, verschachtelte Generalisierung, Entkopplung von Schnittstelle und Implementierung, unabhängige Erweiterbarkeit der zwei Ebenen und Kapselung. Konzeptbruch bzw. Widerspruch zu „guter" Entwurfsheuristik ist jedoch auch zu finden: Eng verbundene Daten sind nicht an einem Ort gebündelt. Außerdem sind Schnittstelle und Daten getrennt.

Zusammenhänge mit anderen Mustern: Es ist dem Adapter ähnlich und wirkt oft mit Singleton und Abstrakter Fabrik zusammen.

Lebensweltbezug: Als Lebensweltbeispiel kann ein elektrischer Schalter angesehen werden. Dessen Abstraktion in einem Schaltplan ist unabhängig vom konkreten Lichtschalter (Duell et al. 1998, S. 9).

Unterrichtsmaterial im Rahmen des Unterrichtsmodells: Sülz (2007) betrachtet in seiner vom Autor betreuten Seminararbeit das Brückemuster hinsichtlich der Kompatibilität als Qualitätsanforderung an Informatiksysteme, das aufbauend auf dem Adaptermuster eingesetzt werden kann.

Dekorierer *Zweck und Einsatzgebiet:* Das Dekorierermuster ermöglicht flexible funktionale Erweiterungen eines Objektes ohne Unterklassenbildung (Gamma et al. 1995, S. 199). Anwendung findet das Prinzip bei Eingabe-Ausgabe-Strömen (I/O-Streams). Es führt zu guter Testbarkeit, da Funktionalitäten gekapselt werden.
Vernetzte fundamentale Ideen der Informatik: Konzepte der Objektorientierung im Muster sind Objekte (Objektidentität), Delegation, Schnittstelle, Zustandsdarstellung in Unterklassen, Methodenaufruf, die hat-Beziehung (der Dekorierer hat eine Exemplarvariable, die eine Referenz auf die Klasse Komponente hält), Vererbung wird für die Anpassung des Typs genutzt, nicht für die Definition des Objektverhaltens. In Java erfolgt die Realisierung über ein Interface oder eine abstrakte Klasse, nicht über Objektaggregation. Informatische Konzepte im Muster sind rekursive Dekomposition (Kettenbildung), Erweiterbarkeit der Objektfunktionalität zur Laufzeit (beliebig viel), Kapselung, Mehrfachnutzung der gleichen Dekoration, Verhinderung der Überfrachtung der Oberklassen, Kapselung der dynamischen Kettenbildung von der Klientenfunktionalität, Trennung der Kettenbildung von den Kettenkomponenten (Shalloway und Trott 2002), Kapselung neuer Funktionalität und Kohäsion. Die Reihung der Dekorationen führt zu einer Listenstruktur.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Fabrik, Erbauer, Prototyp, und weist Ähnlichkeiten mit Adapter, Kompositum (Dekorierer als degeneriertes Kompositum) und Strategie auf. Konzeptbruch bzw. Widerspruch zu „guter" Entwurfsheuristik: Klassen sollen durch Vererbung erweitert werden, Vererbungshierarchien sollten in der Theorie der Objektorientierung tief sein, wenngleich sie in der Praxis Fehlerquellen darstellen, und es werden viele kleine ähnliche Objekte erzeugt.
Lebensweltbezug: Werkzeuge und Arbeitskleidung von Mitarbeitern dienen zu deren funktionaler Erweiterung in der Lernsoftware Pattern Park (Franke et al. 2007). Auch Benutzungsoberflächen von Software können durch Werkzeugleisten dekoriert werden (Duell et al. 1998, S. 14).
Unterrichtsmaterial im Rahmen des Unterrichtsmodells: Das Muster ist in die Lernsoftware Pattern Park zur Förderung der Kompetenzentwicklung mit Informatiksystemen integriert worden (Abschnitt 5.6). Weyer (2007b) klassifiziert das Entwurfsmuster in seiner Diplomarbeit, die vom Autor betreut wurde (Tabelle 5.2).

Fassade *Zweck und Einsatzgebiet:* Die Fassade dient der Zugriffskontrolle durch Anbieten einer einheitlichen Schnittstelle zu einem Subsystem (Gamma et al. 1995, S. 212), (Buschmann et al. 1996, S. 380). Anwendungen sind Übersetzer bzw. Compiler und Schichten (Black-Box, White-Box).
Vernetzte fundamentale Ideen der Informatik: Konzepte der Objektorientierung im Muster sind Schnittstelle und Delegation. Informatische Konzepte

im Muster sind Entkopplung von Systemen, Kapselung, Reduktion der Komplexität durch Vereinfachung der Schnittstelle und Zerlegung in Teilsysteme sowie Zustandslosigkeit.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Abstrakte Fabrik und Singleton. Fassade ist ähnlich dem Vermittler.
Lebensweltbezug: Ein Lebensweltbeispiel ist ein Pförtner eines Unternehmens, der Anfragen und Lieferungen weiterleitet (Franke et al. 2007).
Unterrichtsmaterial im Rahmen des Unterrichtsmodells: Das Muster ist in die Lernsoftware Pattern Park zur Förderung der Kompetenzentwicklung mit Informatiksystemen integriert worden (Abschnitt 5.6). Weyer (2007b) klassifiziert das Entwurfsmuster in seiner Diplomarbeit, die vom Autor betreut wurde, und beschreibt den Entwurf einer Lernsoftware, die das Muster enthält (Tabelle 5.2; Abschnitt 7.3.2).

Fliegengewicht *Zweck und Einsatzgebiet:* Das Fliegengewichtmuster erlaubt die effiziente Verwendung großer Mengen kleinster Objekte, z. B. Buchstaben- bzw. Zeichenobjekte in einem Textdokument (Gamma et al. 1995, S. 223).
Vernetzte fundamentale Ideen der Informatik: Informatische Konzepte im Muster sind strukturierte Zerlegung und Zustände.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Kompositum, Zustand und Strategie.
Lebensweltbezug: Als Beispiel wird die Generierung von Freizeichen und Besetztzeichen sowie deren Zuweisung auf eine große Menge Telefonkunden beschrieben (Duell et al. 1998, S. 17).

Kompositum *Zweck und Einsatzgebiet:* Das Kompositum fügt Objekte rekursiv in einer Baumstruktur in Form von Teil-Ganzes-Beziehungen zusammen (Gamma et al. 1995, S. 239). Einzelne Objekte und die Zusammenstellung von Objekten werden auf die gleiche Weise aufgerufen und behandelt.
Vernetzte fundamentale Ideen der Informatik: Analysiert man das Kompositummuster, ist die einheitliche Schnittstelle zu allen Komponenten charakteristisch. Der Klient führt eine Operation auf dem Kompositum aus, indem der Aufruf von jeder Komponente des Kompositums gemäß der eigenen Spezifikation ausgeführt wird. Die innere Struktur des Kompositums basiert auf den fundamentalen Ideen Rekursion und Polymorphie. Konzepte der Objektorientierung im Muster sind demnach Schnittstelle, Klassenhierarchien, die Unterscheidung zwischen primitiven und komplexen Objekten sowie Polymorphie. Hinsichtlich fundamentaler Ideen sind insbesondere die Rekursion und dadurch bewirkt die strukturierte Zerlegung eines zusammengesetzten Objektes zu nennen. Denn durch die Rekursion des Kompositummusters wird ein Divide-and-Conquer-Verfahren realisiert. Weitere informatische Konzepte im Muster sind Baumstrukturen (und Komponentenzugriff), Typsicherheit,

Datenstruktur Liste als Spezialfall des Baumes und Traversierung von Datenstrukturen. Durch das Kompositummuster ist auch ein Bezug zur Masteridee Sprache gegeben. Steinert beschreibt, dass textuelle Objekte durch dieses Entwurfsmuster zusammengesetzt werden können und hält dieses Vorgehen bereits für Schüler geeignet, die in der Sekundarstufe I sind, da es nah an der Lebenswelt der Schüler ist (Schneider 2003). Ein Text wird in Abschnitte unterteilt, die wiederum in kleinere Einheiten zerlegt werden können. Einzelne Zeichen bilden die Blätter der durch das Kompositum beschreibbaren Baumstruktur. Auf dem Beispiel von Steinert aufbauend kann in der Sekundarstufe II das Entwurfsmuster erweitert werden, um die dynamischen Aspekte der Algorithmisierung stärker zu betonen. Dazu können Unterklassen genutzt werden. Diese repräsentieren nichtterminale Ausdrücke unterschiedlicher Art wie Wiederholung und Auswahl. Damit wird der Kontext des Entwurfsmusters zu regulären Ausdrücken und formalen Grammatiken gerückt und im Informatikunterricht die Brücke geschlagen von der OOM zur Theoretischen Informatik. Damit ist ein Beitrag zu Möglichkeiten und Grenzen von Informatiksystemen zu erwarten. Information über Syntax ist dann in der resultierenden Vererbungshierarchie gespeichert. So ist das Kompositummuster geeignet, Sätze einer Sprache zu interpretieren. In diesem Kontext wird das modifizierte Kompositummuster auch als Interpretermuster bezeichnet, denn der abstrakte Syntaxbaum ist ein Exemplar des Kompositummusters (Gamma et al. 1995, S. 334).
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Erbauer, Prototyp, Zuständigkeitskette, Dekorierer, Fliegengewicht, Iterator, Besucher und Proxy. Es ist verwandt mit dem Interpreter.
Lebensweltbezug: Lebensweltbeispiele bilden die Zusammensetzung komplexer Grafiken aus primitiven Zeichenobjekten und die Aufteilung einer Jugendgruppe in Teilgruppen (Franke et al. 2007).
Unterrichtsmaterial im Rahmen des Unterrichtsmodells: Das Muster ist in die Lernsoftware Pattern Park zur Förderung der Kompetenzentwicklung mit Informatiksystemen integriert worden (Abschnitt 5.6).

Proxy *Zweck und Einsatzgebiet:* Der Proxy dient der Zugriffskontrolle und nutzt ein Stellvertreterobjekt (Gamma et al. 1995, S. 254), (Buschmann et al. 1996, S. 380). Das Muster wird zur einfachen Zugriffskontrolle, aber auch zur Protokollierung von Zugriffen eingesetzt, und es können Anknüpfungspunkte zur Webarchitektur identifiziert werden.
Vernetzte fundamentale Ideen der Informatik: Konzepte der Objektorientierung im Muster sind Schnittstelle, Methodenaufruf, Vererbung und Objekte. Informatische Konzepte im Muster sind Interaktionsregelung, Kapselung, Zugriffskontrolle, Platzhalter bzw. Referenz und Polymorphie.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusam-

men, z. B. Kompositum. Ähnlich ist es dem Adapter und dem Dekorierer.
Lebensweltbezug: Ein Beispiel für einen Proxy ist die Stellvertreterrolle einer Geldkarte für Geld (Franke et al. 2007).
Unterrichtsmaterial im Rahmen des Unterrichtsmodells: Das Muster ist in die Lernsoftware Pattern Park zur Förderung der Kompetenzentwicklung mit Informatiksystemen integriert worden (Abschnitt 5.6) und in den unterrichtlichen Erkundungen eingesetzt worden (Kapitel 6 und 8). Weyer (2007b) klassifiziert das Entwurfsmuster in seiner Diplomarbeit, die vom Autor betreut wurde, und beschreibt den Entwurf einer Lernsoftware, die das Muster enthält (Tabelle 5.2; Abschnitt 7.3.2).

Nachfolgend werden die verbleibenden Entwurfsmuster analysiert. Sie gehören zu den Verhaltensmustern nach Gamma et al. (1995).

Befehl *Zweck und Einsatzgebiet:* Das Befehlsmuster erstellt für Befehle eigene Objekte, um Operationen einzeln behandeln zu können, z. B. um sie rückgängig zu machen (Gamma et al. 1995, S. 273).
Vernetzte fundamentale Ideen der Informatik: Konzepte der Objektorientierung im Muster sind (Befehls-) Objekte, Methodenaufrufe, Objekterzeugung und Objektspeicherung. Informatische Konzepte im Muster sind Interaktionsregelung, Dekomposition, Parametrisierung, Schlange oder Liste zur Speicherung der Befehlsobjekte, traversieren (für Undo-Redo-Funktionalität) sowie Persistenz durch ein Logbuch. Weitere Konzepte sind Entkopplung von Auslöser, Empfänger und Anfrage, Erweiterbarkeit, Festlegung des Kontrollflusses zur Laufzeit und Zustände. Konzeptbruch bzw. Widerspruch zu „guter" Entwurfsheuristik: Operationen werden in diesem Muster zu Klassen, was im Widerspruch zu der auch in der Schule oft angewendeten Vorgehensweise steht, dass in beschreibenden Texten Subjekte zu Klassen und Verben zu Operationen werden.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. für zusammengesetzte Befehle mit dem Kompositum, außerdem mit Memento und Prototyp.
Lebensweltbezug: Als Lebensweltbeispiel kann die Sammlung von Bestellungen für einen Tisch auf einem Zettel dienen, die in einem Restaurant vorgenommen wird. Der Kellner reicht die Liste weiter an den Koch, der aus den einzelnen Elementen Handlungsanweisungen ableitet. Ein solcher Zettel zur Aufnahme der Bestellungen kann in verschiedenen Restaurants benutzt werden (Duell et al. 1998, S. 21).

Beobachter *Zweck und Einsatzgebiet:* Das Beobachtermuster nutzt 1-zu-n-Abhängigkeiten, um einen Datensatz unterschiedlich darzustellen, indem Zustandsänderungen allen Sichten mitgeteilt werden (Gamma et al. 1995, S. 287).

Anwendung findet es in Schichtenarchitekturen und dem Model-View-Controller-Architekturmuster.
Vernetzte fundamentale Ideen der Informatik: Konzepte der Objektorientierung im Muster sind Schnittstelle zur Registrierung, Objekte und Abstrakte Klasse. In Java erfolgt die Umsetzung mit Observer Interface und Observable class. Informatische Konzepte im Muster sind 1-zu-n-Abhängigkeiten, Konsistenz, Trennung von Daten und Darstellung, Kapselung, „Publish-Subscribe" (lose Kopplung), Zustände, Interaktionsregelung und Erweiterbarkeit. Der Benachrichtigungsaufwand (Komplexität) kann analysiert werden. Außerdem gibt es Polymorphie. Konzeptbruch bzw. Widerspruch zu „guter" Entwurfsheuristik: Eng verbundene Daten werden nicht an einem Ort definiert, und es besteht die Gefahr, dass es zu viele kooperierende Klassen gibt.
Zusammenhänge mit anderen Mustern: Neben dem Model-View-Controller-Architekturmuster sind Vermittler und Singleton verwandte Muster.
Lebensweltbezug: Ein Beispiel sind Funkuhren, die mit einer zentralen Uhr in Verbindung stehen (Franke et al. 2007).
Unterrichtsmaterial im Rahmen des Unterrichtsmodells: Das Muster ist in die Lernsoftware Pattern Park zur Förderung der Kompetenzentwicklung mit Informatiksystemen integriert worden (Abschnitt 5.6). Weyer (2007b) klassifiziert das Entwurfsmuster in seiner Diplomarbeit, die vom Autor betreut wurde, und beschreibt den Entwurf einer Lernsoftware, die das Muster enthält (Tabelle 5.2; Abschnitt 7.3.2). In einer vom Autor betreuten Seminararbeit wurden Aufgaben zum Beobachtermuster erstellt (Graf 2008).

Besucher *Zweck und Einsatzgebiet:* Das Besuchermuster erweitert die Funktionalität von Objekten in einer Objektstruktur, indem die Operation als Objekt gekapselt wird (Gamma et al. 1995, S. 301).
Vernetzte fundamentale Ideen der Informatik: Informatische Konzepte im Muster sind Datenstrukturen, deren Traversierung, Geheimnisprinzip und Wiederverwendung.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Kompositum und Interpreter.
Lebensweltbezug: Ein Lebensweltbeispiel ist die Begutachtung einer Organisation durch externe Gutachter. Diese werden zu Personen oder Abteilungen geführt, wo die Gutachter aktiv werden, beispielsweise können sie Interviews durchführen (Duell et al. 1998, S. 35).

Interpreter *Zweck und Einsatzgebiet:* Das Interpretermuster ermöglicht für eine definierte formale Sprache die Interpretation von Sätzen (Gamma et al. 1995, S. 319).
Vernetzte fundamentale Ideen der Informatik: Konzept der Objektorientierung im Muster ist Vererbung. Informatische Konzepte im Muster sind rekursive Dekomposition, reguläre Ausdrücke, Baum, Syntax, Grammatik und

Konkatenation.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Fliegengewicht und Besucher. Ähnlichkeit besteht mit Iterator und vor allem dem Kompositum.
Lebensweltbezug: Als Beispiel können Musiker dienen, die Musikkompositionen spielen, die aus Terminalzeichen wie Noten und zusammengesetzten Zeichen bestehen (Duell et al. 1998, S. 22).

Iterator *Zweck und Einsatzgebiet:* Der Iterator dient der Iteration und Zugriffskontrolle auf Datenstrukturen (Gamma et al. 1995, S. 335), (Buschmann et al. 1996, S. 380).
Vernetzte fundamentale Ideen der Informatik: Konzept der Objektorientierung im Muster ist die Schnittstelle. Informatische Konzepte im Muster sind Management, polymorphe Iteration, In-order und pre-order Traversierung, rekursiver Stack-Aufbau in Kombination mit einer Baumstruktur wie dem Kompositum.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. dem Proxy in C++ für einen polymorphen Iterator, Memento zur Speicherung des Iterationszustands, Kompositum und Fabrikmethode.
Lebensweltbezug: Als Beispiel dient das Durchlaufen eine Besucherliste im Pattern Park (Franke et al. 2007).
Unterrichtsmaterial im Rahmen des Unterrichtsmodells: Das Muster ist in die Lernsoftware Pattern Park zur Förderung der Kompetenzentwicklung mit Informatiksystemen integriert worden (Abschnitt 5.6). Weyer (2007b) klassifiziert das Entwurfsmuster in seiner Diplomarbeit, die vom Autor betreut wurde, und beschreibt den Entwurf einer Lernsoftware, die das Muster enthält (Tabelle 5.2; Abschnitt 7.3.2).

Memento *Zweck und Einsatzgebiet:* Das Mementomuster erlaubt die Zwischenspeicherung eines Objektzustands, ohne dessen Kapselung zu verletzen, um es später in den Zustand zurück versetzen zu können (Gamma et al. 1995, S. 354).
Vernetzte fundamentale Ideen der Informatik: Konzept der Objektorientierung im Muster ist Schnittstelle bzw. Zugriffskontrolle, denn nur Memento soll auf ein Objekt und dessen Zustände zugreifen können. Informatische Konzepte im Muster sind Kapselung, Zustände und inkrementelle Speicherung von Änderungen.
Zusammenhänge mit anderen Mustern: Es wirkt oft mit anderen Mustern zusammen, z. B. Befehl und Iterator.
Lebensweltbezug: Als Beispiel dient das Autoradio. Jeder Fahrer kann sich seinen Sender speichern und nach einem Fahrerwechsel durch einen Tastendruck seinen Sender wieder einstellen. Außerdem wird ein Foto der Einstellungen

eines Audiomischpultes zur Erinnerung an eine bewährte Reglerkombination genannt (Duell et al. 1998, S. 22).

Strategie *Zweck und Einsatzgebiet:* Das Strategiemuster macht das Austauschen einzelner, aber ähnlicher Algorithmen möglich, indem sie gekapselt werden (Gamma et al. 1995, S. 373).
Vernetzte fundamentale Ideen der Informatik: Konzepte der Objektorientierung im Muster sind Schnittstelle, Vererbung (Strategiehierarchien), Verantwortlichkeiten von Objekte und Delegation, um einen gesamten Algorithmus zu variieren. Informatische Konzepte im Muster sind Kapselung (von Regeln), algorithmische Komplexität, Polymorphie, Trennung von Strategiewahl und Implementierung, Bedingungen, Zusammenfassen von Familien ähnlicher Strategien, Dekomposition sowie Alternative.
Zusammenhänge mit anderen Mustern: Es wirkt z. B. mit dem Fliegengewicht zusammen.
Lebensweltbezug: Als Beispiel dient die Überwindung einer Strecke durch ein Transportmittel. Die Auswahl eines Transportmittels wie Taxi, Flugzeug oder Bus entspricht der Strategie (Duell et al. 1998, S. 32).

Vermittler *Zweck und Einsatzgebiet:* Das Vermittlermuster kontrolliert und koordiniert die Interaktion mehrerer Objekte (Gamma et al. 1995, S. 385).
Vernetzte fundamentale Ideen der Informatik: Konzept der Objektorientierung im Muster ist Entkopplung. Informatische Konzepte im Muster sind Interaktionsregelung, Dekomposition, 1-zu-n Abhängigkeiten, Kapselung und Zentralisierung.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Beobachter (Vermittler als Beobachter). Die Fassade wirkt sehr ähnlich, aber unidirektional.
Lebensweltbezug: Als Beispiel für einen Vermittler dient der Kontrollturm auf dem Flughafen, über den jegliche Kommunikation und die Verwaltung der Start- und Landezeiten der Flugzeuge läuft (Duell et al. 1998, S. 25).

Zustand *Zweck und Einsatzgebiet:* Das Zustandsmuster definiert mehrere allein stehende Objektzustände, damit das Objektverhalten zur Laufzeit in Abhängigkeit von seinem Zustand geändert wird (Gamma et al. 1995, S. 398).
Vernetzte fundamentale Ideen der Informatik: Informatische Konzepte im Muster sind Dekomposition, Kontextobjekte, Alternative und Zustand.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Fliegengewicht, Proxy und Singleton.
Lebensweltbezug: Beispiel kann eine Achterbahnfahrt mit den unterschiedlichen Zuständen der Bahn sein (Franke et al. 2007).
Unterrichtsmaterial im Rahmen des Unterrichtsmodells: Das Muster ist in die Lernsoftware Pattern Park zur Förderung der Kompetenzentwicklung mit

Informatiksystemen integriert worden (Abschnitt 5.6) und wurde im unterrichtlichen Geschehen erprobt (Kapitel 8). Weyer (2007b) klassifiziert das Entwurfsmuster in seiner Diplomarbeit, die vom Autor betreut wurde, und beschreibt den Entwurf einer Lernsoftware, die das Muster enthält (Tabelle 5.2; Abschnitt 7.3.2). In einer vom Autor betreuten Seminararbeit wurden Aufgaben zum Zustandsmuster erstellt (Graf 2008).

Zuständigkeitskette *Zweck und Einsatzgebiet:* Die Zuständigkeitskette fasst Objekte in einer Liste zusammen und leitet Anfragen die Kette entlang, wenn deren Adressat erst zur Laufzeit festgelegt wird (Gamma et al. 1995, S. 410).
Vernetzte fundamentale Ideen der Informatik: Informatische Konzepte im Muster sind Interaktionsregelung, Dekomposition, Iteration und Liste.
Zusammenhänge mit anderen Mustern: Es wirkt mit anderen Mustern zusammen, z. B. Kompositum.
Lebensweltbezug: Als Beispiel dient eine Münzsortiermaschine. Jeder Slot prüft, ob die Münze in ihm aufbewahrt werden kann. Falls nicht, wird sie weitergereicht (Duell et al. 1998, S. 19).

Die Auswahl der informatischen Konzepte erfolgte vornehmlich mit Blick auf fundamentale Ideen der Informatik, aber nicht für alle aufgeführten Konzepte ist bislang nachgewiesen worden, dass es fundamentale Ideen sind. Vor dem Transfer in Unterricht kann die Prüfung auf die Kriterien für fundamentale Ideen erfolgen (Abschnitt 5.4.4). Dennoch zeigt die Sammlung das Potential der Entwurfsmuster. In einer Diplomarbeit, die vom Autor betreut wurde, klassifiziert Weyer (2007b) ausgewählte Entwurfsmuster und entwickelt jeweils ein Lebensweltbeispiel im Kontext einer Arztpraxis (Tabelle 5.2; siehe auch Abschnitt 7.3.2). Die Einschätzung des Schwierigkeitsgrades ist nur sehr knapp begründet und bedarf der empirischen Überprüfung.

5.4.3 Zwischenfazit zur Klassifikation für Kompetenzentwicklung mit Informatiksystemen

In diesem Abschnitt wird auf die Einsatzmöglichkeiten der Klassifikation eingegangen. Im nächsten Abschnitt wird der Beitrag eines Entwurfsmusters, das mittels obiger Kriterien ausgewählt wurde, am Beispiel diskutiert.

Die vorliegende Klassifikation erlaubt nun die konkrete Auswahl bzw. den Vergleich von Entwurfsmustern für den Informatikunterricht zur Förderung der Kompetenzentwicklung mit Informatiksystemen. In (Stechert 2006b) werden die beiden Entwurfsmuster Kompositum und Brücke exemplarisch untersucht und verglichen. Beide Entwurfsmuster sind komplex und die Betrachtung ihrer Eigenschaften scheint auf den ersten Blick für Kompetenzentwicklung mit Informatiksystemen viel versprechend. Trotzdem überwiegen die Vorteile des Kompositummusters: Insbesondere die fundamentale Idee der Sprache wird hinsichtlich der formalen Grammatiken

Tabelle 5.2: Klassifikation ausgewählter Entwurfsmuster nach Weyer (2007b)

Name	**Fundamentale Ideen**	**Zusammenhänge mit Mustern**	**Komplexität**	**Einsatzgebiet**	**Lebensweltbeispiel: Arztpraxis**
Dekorierer	Rekursive Dekomposition, Vererbung	Adapter, Kompositum, Strategie	mittel	Erweiterung der Funktionalität eines Objektes	Erweiterung der Funktion einer Arzthelferin, indem sie mit Mikroskop und Terminkalender ausgestattet wird
Fassade	Dekomposition, Delegation, Entkopplung von Systemen, Zugriffskontrolle	Abstrakte Fabrik, Vermittler, Singleton, Proxy	einfach	Modularisierung in Systemen	Eine Arzthelferin nimmt die Rolle der Fassade ein. Sie ruft Patienten über die Sprechanlage aus dem Wartezimmer
Proxy	Zugriffskontrolle, Platzhalter / Referenz, Schnittstelle	Adapter, Dekorierer, Iterator, Kompositum	mittel	Zugriffskontrolle und intelligente Zeiger	Zugriff auf die Daten eines bestimmten Patienten
Beobachter	1-zu-n-Abhängigkeiten, Trennung von Daten und Darstellung, Publish-Subsribe	Model-View-Controller, Vermittler, Singleton	mittel	Schichten	Eine Patientendatenbank mit Beobachtern wie Kartei und Rezept
Iterator	Traversierung, Polymorphie, Stack	Kompositum, Fabrikmethode, Memento	einfach	zusammengesetzte Objekte	Durchsuchen einer Patientendatenliste
Zustand	Zustand, Dekomposition, Modularisierung	Fliegengewicht, Singleton, Memento	schwer	Kapselung der Zustandsänderungen eines Objektes	Zustände einer Blutprobe

und nicht nur durch Darstellbarkeit in der UML unterstützt. Textobjekte in Textverarbeitungssystemen, wie Steinert sie beschreibt, werden bereits in Klassenstufe 6 thematisiert. Deshalb bietet das Kompositum und mit ihm die einfache Erweiterung zum Interpretermuster großes Potential bei der Einführung in den Lehr-Lernprozess der Sekundarstufe II. Aufgrund der Erfahrungen in der Hochschullehre an der Universität Siegen (Schubert 2005), die Arbeit von Steinert (Schneider 2003)

und obige Überlegungen zu fundamentalen Ideen der Informatik wird das Kompositummuster als wertvoller Unterrichtsinhalt in der Sekundarstufe II betrachtet. Das Brückemuster hingegen ist aufgrund seiner hohen Komplexität (vgl. auch Harrer und Schneider (2002) und Unger-Lamprecht (2001) für die Hochschule) und dem weniger starken Beitrag zu fundamentalen Ideen erst gegen Ende der Sekundarstufe II einzusetzen. Darüber hinaus ist in die Überlegungen einzubeziehen, dass es weitere Entwurfsmuster gibt, die die Trennung von Aufgaben auf eine einfachere Weise veranschaulichen als das Brückemuster, z. B. das Beobachtermuster. Auch das Strategiemuster erscheint für den Anfang geeigneter, da es weniger komplex ist und die fundamentale Idee der Alternative repräsentiert.

Vorteil der angegebenen Klassifikation ist, dass mit ihr statt einzelner Entwurfsmuster auch Entwurfsmusterkombinationen und Architekturen klassifiziert und vom Schüler verstanden werden können. Die Lernsoftware Pattern Park stellt eine Aufgabe zur Kombination der Muster Kompositum, Iterator, Dekorierer und Fassade (Abschnitt 5.6.3). In Kapitel 8 wird die Kombination von Proxy und Zustand im Informatikunterricht vorgestellt. Durch die Kombination unterschiedlicher Entwurfsmuster werden die fundamentalen Ideen in ihnen weiter vernetzt, und es treten neue Abhängigkeiten auf. Ein Iteratormuster beispielsweise, das alle Elemente einer Datenstruktur aufzählt, arbeitet je nach Datenstruktur iterativ oder rekursiv. So ist bei der Kombination von Iterator und Kompositum ein rekursives Vorgehen zur Aufzählung aller Objekte notwendig.

Fasst man die Ergebnisse zusammen, sind Entwurfsmuster Träger von fundamentalen Ideen der Informatik, d. h. sie sind eine Repräsentation für diese fundamentalen Ideen der Informatik. Damit ist insbesondere für den Informatikunterricht in der Sekundarstufe II der Bildungswert von Entwurfsmustern begründbar. Dazu müssen Zugänge zur Kompetenzentwicklung mit Informatiksystemen mittels Entwurfsmustern folgende Anforderungen erfüllen: Erstens müssen sie auf ausgewählten Entwurfsmustern basieren, die für den Lehr-Lernprozess in der Sekundarstufe II gemäß der Klassifikation für Kompetenzentwicklung mit Informatiksystemen geeignet sind. Dafür müssen sie vor allem die Vernetzung fundamentaler Ideen ermöglichen. Zweitens sollten die Entwurfsmuster kombinierbar sein, um die Bildung von Wissensinseln zu vermeiden. Die durch Fokussierung auf Entwurfsmuster vorgenommene Einschränkung auf objektorientierte Modellierung wird durch die fundamentalen Ideen zum Teil egalisiert, da sie per Definition in weiten Gebieten der Informatik vorkommen.

5.4.4 Beitrag der durch die Klassifikation ausgewählten Entwurfsmuster zur Kompetenzentwicklung mit Informatiksystemen am Beispiel der Zugriffskontrolle

Erwartete kognitive Barrieren und Fehlvorstellungen zur Zugriffskontrolle

Wie fördert ein Entwurfsmuster, das nach obiger Klassifikation beschrieben wurde, die Kompetenzentwicklung mit Informatiksystemen? Im Folgenden werden erwartete kognitive Barrieren und Fehlvorstellungen zur Zugriffskontrolle aufgezeigt (vgl. Schubert et al. 2009). Zugriffskontrolle ist bei vielen Informatiksystemen unabdingbar, z. B. bei Mehrbenutzersystemen zur Kommunikation, Koordination und Datenspeicherung (Abschnitt 3.2.2). Für die Arbeit mit Informatiksystemen ist meist eine Authentifizierung einschließlich einer Passwortprüfung notwendig und den Schülern vom Schulnetz bereits vertraut (vgl. Humbert et al. 2005). Hinsichtlich der Klassifikation der Lernziele nach Hubwieser und Broy ist Zugriffskontrolle als charakteristisch für fast alle Informatiksysteme einzustufen (Hubwieser 2007a, S. 83).

Um Zugriffskontrolle zu verstehen, müssen verborgene Prozesse sichtbar und durch den Lernenden handlungsorientiert veränderbar gemacht werden. Ziel des Beispiels zur Zugriffskontrolle ist es aufzuzeigen, wie Entwurfsmuster im Unterricht zur Förderung der Kompetenzentwicklung mit Informatiksystemen eingesetzt werden können. Lehrenden wird anhand des komplexen Szenarios gezeigt, dass Entwurfsmuster bis zu einem gewissen Grad sukzessive kombiniert werden können und eine logische Verknüpfung von Themen möglich ist.

Eine erwartet Fehlvorstellung zur Zugriffskontrolle ist, dass ein Objekt den Zugriff auf sich selbst kontrolliert und überwacht. Dies ist aber in aller Regel nicht der Fall: Sei es aus Sicherheits- oder sei es aus Performancegründen wird oft, beispielsweise beim Proxymuster, ein Stellvertreterobjekt mit der gleichen Schnittstelle wie der des Originalobjekts eingesetzt. Somit kontrolliert das Stellvertreterobjekt die Zugriffsrechte, zählt bei Bedarf die Anzahl der Zugriffe und leitet letztendlich Aufrufe an das Original weiter.

Eine weitere kognitive Hürde ist die Frage, wie ein Platzhalter dasselbe Verhalten, d. h. die gleiche Schnittstelle wie das Original haben kann. Dies kann durch Vererbung realisiert werden, indem Stellvertreter und Original von der gleichen (abstrakten) Klasse erben. Dadurch ergibt sich für beide die gleiche Schnittstelle bei durchaus unterschiedlicher Spezialisierung. Eine gleichnamige Operation wird beim Original ein bestimmtes Verhalten auslösen, während beim Stellvertreter beispielsweise nur die Prüfung der Zugriffsrechte und gegebenenfalls die Weiterleitung an das Original durchgeführt werden.

Die nächste kognitive Hürde ist Formalisierung dynamischer Prozesse – ein generelles Ziel der informatischen Bildung, das sehr gut anhand der Zugriffskontrolle diskutiert werden kann. Lernende sollen hier die Lebensweltsituation in einem ersten Schritt Richtung maschinelle Verarbeitung formalisieren.

Als letzte Schwierigkeit ist genannt, dass fundamentale Ideen der Informatik, in diesem Fall z. B. Zugriffskontrolle und Vererbung, vernetzt sind und einander beeinflussen – eine weitere kognitive Hürde, denn beide können in anderen Kontexten unterschiedlich umgesetzt werden.

In der Beschreibung der erwarteten kognitiven Barrieren und Fehlvorstellungen wird bereits angedeutet, wie ihre Überwindung gelingen kann. Im Folgenden soll dies anhand des Proxymusters für Kompetenzentwicklung mit Informatiksystemen vorgestellt werden. Im Vorgriff auf Abschnitt 5.6 wird vorausgesetzt, dass eine Software das Entwurfsmuster umsetzt und dadurch die Zugriffskontrolle im Systemverhalten erkennbar ist. Folgende Gründe sprechen dafür, dass durch die Thematisierung der Zugriffskontrolle anhand des Entwurfsmusters Proxy die Kompetenzentwicklung mit Informatiksystemen gefördert wird:

- Relevanz für viele Informatiksysteme (→ Kriterium 1: Abstraktion von Implementierungsaspekten; → Kriterium 2: Zweck und Einsatzgebiet),
- Vernetzte fundamentale Ideen zur Erklärung des Systemverhaltens (→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik),
- Sichtenwechsel zwischen Verhalten und bewährter Struktur (→ Kriterium 1: Abstraktion von Implementierungsaspekten; → Kriterium 5: Komplexität),
- Vernetzte Strukturmodelle und Lebensweltbezug (→ Kriterium 4: Zusammenhänge mit anderen Strukturmodellen; → Kriterium 6: Lebensweltbezug),
- Entwurfsmuster als lernförderliche Schemata (→ Kriterium 5: Komplexität),
- Externe Repräsentation von Entwurfsmustern (→ Kriterium 5: Komplexität).

Diese Aspekte der Kompetenzentwicklung mit Informatiksystemen werden im Folgenden anhand der vorgestellten Kriterien erläutert.

Relevanz für viele Informatiksysteme

Exemplarisch soll anhand der Zugriffskontrolle mittels Proxymuster der Beitrag der durch die Klassifikation ausgewählten Entwurfsmuster zur Kompetenzentwicklung mit Informatiksystemen aufgezeigt werden. Für Kompetenzentwicklung mit Informatiksystemen wurde festgestellt, dass das nach außen sichtbare Verhalten und die innere Struktur von Informatiksystemen zu analysieren sind. Implementierungsaspekte sind für Kompetenzen nur in geringem Maße relevant. Das Entwurfsmuster Proxy abstrahiert per Definition von Implementierungsdetails (→ Kriterium 1:

Abstraktion von Implementierungsaspekten), denn es wird nicht den Idiomen zugeordnet (vgl. Buschmann et al. 1996), so dass die Strukturierung im Mittelpunkt steht.

Darüber hinaus ist Zugriffskontrolle für viele Informatiksysteme charakteristisch und in deren Verhalten erkennbar, z. B. in Mehrbenutzersystemen, so dass von dem Proxymuster als Strukturmodell ein Beitrag zur Kompetenzentwicklung mit Informatiksystemen zu erwarten ist. Das Proxymuster erklärt, wie Zugriffskontrolle mittels Stellvertreter realisiert werden kann (→ Kriterium 2: Zweck und Einsatzgebiet).

Eine der DeSeCo-Schlüsselkompetenzen (OECD 2005) ist die interaktive Anwendung von Technologien (DeSeCo: 1c). Zu ihr leistet Zugriffskontrolle einen Beitrag, da durch sie in vielen fällen die Nutzung von Systemen ermöglicht oder verhindert wird. Kooperation mit anderen und Informationsaustausch wird über Zugriffskontrolle reguliert (DeSeCo: 2). Außerdem ist anhand von Zugriffskontrolle das Thema Datenschutz und eine kritische Einstellung gegenüber Informatiksystemen im Sinne einer ICT und Information Literacy im Unterricht thematisierbar (UNESCO 2008). Da viele Entwurfsmuster als „Problemlösemuster" direkten Einfluss auf das Systemverhalten haben (→ Kriterium 2: Zweck und Einsatzgebiet) ist neben wissenschaftspropädeutischen Aspekten oft ein Beitrag zu den genannten Kompetenzen möglich (Kapitel 2). Insgesamt ist durch den Einsatz von Entwurfsmustern in Lernsoftware (Abschnitt 5.6) ein Beitrag zur Media Literacy zu erwarten (UNESCO 2008).

Vernetzte fundamentale Ideen zur Erklärung des Systemverhaltens

Ausgewählte vernetzte fundamentale Ideen (→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik) im Entwurfsmuster Proxy sind: Zugriffskontrolle, Schnittstelle, Vererbung und Kapselung (Abbildungen 5.4 und 5.6).

Da fundamentale Ideen in der Klassifikation gefordert werden, sei an dieser Stelle der Nachweis geführt, dass Zugriffskontrolle die Kriterien für fundamentale Ideen der Informatik nach Schwill (1993a) erfüllt (Abschnitt 4.2.2). Dabei ist zu betonen, dass der Ideenkatalog nach Schwill keinen Anspruch auf Vollständigkeit erhebt. Auf Zugriffskontrolle mittels Stellvertreter treffen Lernende in Form von Platzhaltern für Grafiken in Textdokumenten und beim Navigieren im Internet (Sinnkriterium). Zugriffskontrolle ist auf unterschiedlichem intellektuellen Niveau erklär- und verstehbar, da bereits Kinder Zugriffskontrolle durch ihre Eltern erfahren (Vertikalkriterium). Das Rollenspiel zu Suchalgorithmen, in dem ein Schüler den Zugriff auf die von ihm verwaltete Zahl kontrolliert und nach Annahme einer Süßigkeit die Zahl zeigt, ist ein einfaches Unterrichtsbeispiel für die Grundschule (Bell et al. 2002, S. 45f). Seit der Einführung von Mehrbenutzersystemen, z. B. UNIX, ist

Zugriffskontrolle in der Informatik relevant und wird durch die zunehmende Verbreitung von Informatiksystemen längerfristig relevant bleiben (Zeitkriterium). Zugriffskontrolle kommt in der Informatik sowohl bei Mensch-Maschine-Interaktion, bei Betriebs- und Anwendungssystemen als auch bei deren Entwurf vor, wenn Zugriff auf unterschiedliche Schichten kontrolliert wird (Horizontalkriterium).

Auch für weitere, im Unterricht thematisierte Konzepte sollte der Nachweis der Fundamentalität geführt werden: Vererbung und Kapselung, die ein Synonym für das Geheimnisprinzip ist (Claus und Schwill 1997, S. 315), wurden von Schwill als fundamentale Ideen klassifiziert (Schubert und Schwill 2004). Kalkbrenner (2007) weist nach, dass Schnittstelle selbst eine fundamentale Idee der Informatik ist. Allerdings ist sein Nachweis hinsichtlich des Vertikalkriteriums kritisch zu sehen, denn eine Erprobung in der Grundschule oder zu Beginn der Sekundarstufe I wird nicht vorgenommen. Da das Konzept der Schnittstelle jedoch eng verwandt ist mit Parametrisierung und Geheimnisprinzip, die ebenfalls von Schwill anhand der Kriterien als fundamentale Idee der Informatik bestätigt wurden, wird Schnittstelle im Folgenden als fundamentale Idee betrachtet. Dabei wird der Begriff Schnittstelle auch in dem Sinne verwendet, dass erst sie die Kommunikation unterschiedlicher Systemkomponenten, Systemteile und ganzer Informatiksysteme ermöglicht:

> „Das Konzept der Schnittstelle ist essentiell für die gezielte Abstraktion von Systemen sowie Teilsystemen und damit die Beherrschung umfangreicher Systeme der Informatik im Rahmen von Architekturen. Die Idee der Schnittstelle erlaubt eine entscheidende Abstraktion von der Implementierung zur reinen Darstellung der Wirkung eines Systemteils (anschaulich spricht man von ‚Black Box Sicht' oder dem dahinter stehenden Prinzip des ‚Information Hiding')" (Broy und Rumpe 2007, S. 9).

Die Vernetzung der fundamentalen Ideen erklärt das für Kompetenzentwicklung mit Informatiksystemen wichtige Prinzip der Zugriffskontrolle mittels Proxymuster als Strukturmodell: Ein Stellvertreterobjekt kontrolliert den Zugriff auf ein Objekt. Ermöglicht wird die Stellvertreterrolle erst durch die gleiche Schnittstelle zu dem zu schützenden Objekt. Die gleiche Schnittstelle wiederum ist durch Vererbung von einer gemeinsamen abstrakten Klasse realisiert. Dieses einfache Beispiel zeigt das Zusammenwirken fundamentaler Ideen zur Erklärung des Systemverhaltens. Damit wird ein Beitrag zu der Anforderung geleistet, Wirkprinzipien von Informatiksystemen zu verstehen. Dadurch, dass fundamentale Ideen der Informatik speziell den nichtspezifischen Transfer unterstützen, ist ein Beitrag zur Kompetenzentwicklung zu erwarten. Vernetztes Denken wiederum wird als Grundvoraussetzung für Schlüsselkompetenzen betrachtet:

> „Vernetztes Denken ist dabei eine zentrale Voraussetzung. Wir sollten lernen, die vielfältigen Verbindungen und Beziehungen zwischen Standpunkten oder Ideen zu berücksichtigen, die unter Umständen nur auf den ersten Blick widersprüchlich erscheinen mögen" (OECD 2005, S. 11).

Die Analyse vernetzter fundamentaler Ideen kann damit einen Beitrag zur Orientierung in einer zunehmend komplexer werdenden Welt leisten.

Sichtenwechsel zwischen Verhalten und bewährter Struktur

Für den Lernprozess zur Förderung der Kompetenzentwicklung mit Informatiksystemen ist die Unterstützung unterschiedlicher Perspektiven auf Informatiksysteme relevant. Daher wird im Folgenden kurz auf ihren Beitrag zu den Sichten nach außen sichtbares Verhalten S_A, innere Struktur S_B und Implementierungsaspekte S_C eingegangen.

Entwurfsmuster sind abstrakte Elemente zur Gestaltung der inneren Struktur von Informatiksystemen (S_B). In dieser Ebene unterstützen sie sowohl statische Beschreibungen, z. B. durch Klassendiagramme, als auch dynamische Beschreibungsformen wie Sequenzdiagramme. Am Ende der Formalisierung von Zugriffskontrolle durch die Schüler steht eine abstrakte Beschreibung, die als Grundlage für weitere Situationen mit Zugriffskontrolle genutzt werden kann: das Entwurfsmuster Proxy (→ Kriterium 1: Abstraktion von Implementierungsaspekten). Die Komplexität des Proxymusters wird von Harrer und Steinert für die Hochschule als moderat eingeschätzt ((Harrer und Schneider 2002); → Kriterium 5: Komplexität).

Die Übertragung der abstrakten Strukturelemente in die Implementierungsebene (S_C) wird nicht zuletzt durch die Entwurfsmusterkataloge durch Angabe von Beispielimplementierungen vorgenommen. Entwurfsmuster sind aber auch Problemlösemuster, d. h. ein Problem das ggf. im Verhalten des Systems erkennbar ist, wird durch eine bewährte Struktur gelöst. Wird ein Programm eingesetzt, dass ausschließlich auf einem Entwurfsmuster basiert, ist das nach außen sichtbare Verhalten direkt durch die Entwurfsmusterstruktur begründet (S_A; vgl. Abschnitt 5.6.2). Die vernetzten fundamentalen Ideen der Informatik können zur Erklärung des nach außen sichtbaren Verhaltens genutzt werden. Entwurfsmuster ermöglichen damit Sichtenwechsel: Einerseits zwischen Verhalten, Struktur und Implementierungsaspekten, andererseits zwischen statischen und dynamischen Repräsentationen.

Voraussetzung für soziale Kompetenz ist gemäß der DeSeCo-Schlüsselkompetenzen Empathie, d. h. Schüler müssen sich in andere Personen hineinversetzen können (DeSeCo: 2a):

> „Dies erfordert, dass die Individuen ein Niveau an sozialer Reife erlangen, das es ihnen ermöglicht, sich von sozialem Druck zu distanzieren, verschiedene Sichtweisen einzunehmen, eigenständige Urteile zu fällen und die Verantwortung für ihre Handlungen zu übernehmen" (OECD 2005).

Bezüglich Informatiksysteme sind für die Rollen Administrator, Entwickler und Anwender unterschiedliche Sichten einzunehmen. Die Schüler können sich durch den Sichtenwechsel in die Aufgaben dieser Personengruppen, die die interaktive Anwendung von Technologien maßgeblich mitgestalten, hineinversetzen (DeSeCo: 1c).

Vernetzte Strukturmodelle und Lebensweltbezug

Zur Vernetzung der Entwurfsmuster sei im Folgenden ein Beispiel zur Erklärung von Textverarbeitungssystemen betrachtet, dass die drei Entwurfsmuster Proxy, Kompositum und Iterator umfasst (→ Kriterium 4: Zusammenhänge mit anderen Strukturmodellen; → Kriterium 6: Lebensweltbezug). Dazu ist folgendes Szenario zur Veranschaulichung hilfreich: Eine Grafik besteht aus Teilgrafiken. Sie wird durch ein Kompositum beschrieben. Die Grafik soll nun in ein Textdokument eingefügt werden. Zur effizienten Anzeige des Textdokumentes wird die Grafik nicht direkt eingebunden, sondern vorerst nur durch einen Platzhalter repräsentiert, ein Proxy-Objekt. Erst wenn die Dokumentenstelle angezeigt werden soll, in der die Grafik liegt, wird die richtige Grafik geladen. In Abbildung 5.2 wird ein Klassendiagramm zur Darstellung der Vernetzung von Entwurfsmustern genutzt. Es erfasst aber nur implizit die ihnen innewohnenden fundamentalen Ideen. Die drei Entwurfsmuster Kompositum, Proxy und Iterator sind so kombiniert, dass der Iterator alle Elemente eines zusammengesetzten Objektes, des Kompositums, rekursiv iteriert. Dabei sind für die Blattobjekte und die zusammengesetzten Objekte des Kompositums Iteratoren gewählt worden. Freeman et al. beschreiben ein anschauliches Beispiel zur Kombination von Kompositum und Iterator inklusive Quelltext (Freeman et al. 2005, S. 315ff). Die Proxyklasse aus dem Proxymuster hat eine Verbindung zur abstrakten Klasse des Kompositums und ermöglicht so die Zugriffskontrolle auf das Kompositum. Iteratoren stellen in diesem Sinne ebenfalls eine Form der Zugriffsregelung dar.

Eine Kombination des Proxymusters mit dem Zustandsmuster wird in der zweiter Unterrichtserprobung eingesetzt (Kapitel 8).

Entwurfsmuster als lernförderliche Schemata

Die Komplexität von Entwurfsmustern wurde als ambivalent bezeichnet, da zwar theoretische Aussagen zu einzelnen Mustern möglich sind, eine Erprobung in der Sekundarstufe II zur Kompetenzentwicklung mit Informatiksystemen jedoch noch aussteht (→ Kriterium 5: Komplexität). Im Folgenden wird ergänzend argumentiert, dass Entwurfsmuster Schemata bilden, die lernförderlich sind.

Steinert fordert für die Sekundarstufe I, dass Entwurfsmuster nur eingesetzt werden, wenn sie eine didaktische Funktion erfüllen und somit lernförderlich sind (Schneider 2003). Der Erfolg der Förderung der Kompetenzentwicklung mit Informatiksystemen hängt davon ab, ob es gelingt, komplexe Zusammenhänge, wie sie zwischen der inneren Struktur und dem Verhalten von Informatiksystemen bestehen, zu strukturieren (→ Kriterium 1: Abstraktion von Implementierungsaspekten), um Aneignung und Anwendung des Wissens zu unterstützen. Solche Wissensorganisation fördert den Erwerb, die Kommunikation über den Wissensbereich und die Anwendung des Wissens. Nach Mandl und Fischer ist ein Schlüssel dazu die Visualisierung

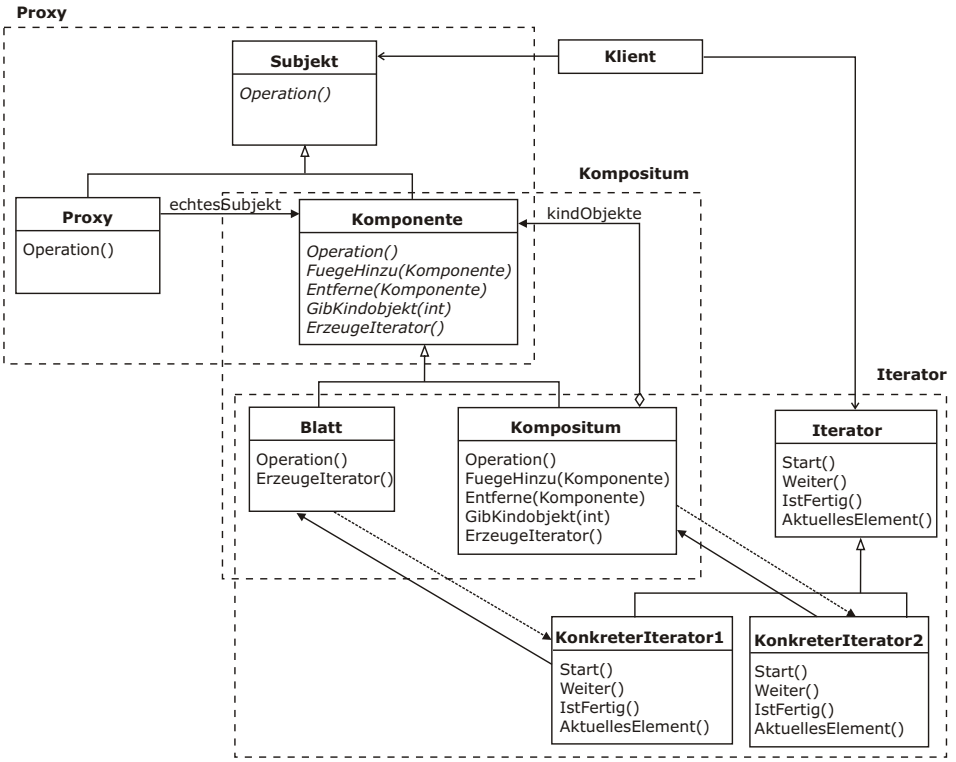


Abbildung 5.2: Klassendiagramm der kombinierten Entwurfsmuster

als externe Repräsentationsform (Mandl und Fischer 2000). Ein anderer sind Schemata als interne, kognitive Repräsentationen über einen Wissensbereich:

> „Insbesondere in wenig strukturierten Wissensdomänen beeinflussen Schemata als Strukturvorgaben die Behaltens- und Anwendungsleistung" (Kopp und Mandl 2006, S. 133).

Für das Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung bieten Entwurfsmuster die Kombination dieser beiden, sich ergänzenden Formen der Wissensorganisation, die gleichzeitig die Komplexitätsbewältigung unterstützen (→ Kriterium 5: Komplexität). Sie werden im Folgenden dargestellt. Nach Kopp und Mandl (2006) sind Wissensschemata abstrakte Wissensstrukturen (nicht zu verwechseln mit der gleichnamigen Komponente des didaktischen Systems), denen die Auffassung zugrunde liegt,

> „dass sämtliche Objekte, Situationen, Ereignisse und Handlungen vom Individuum mental erfasst und verarbeitet werden, dass ihre einzelnen Komponenten kognitiv als zusammenhängendes Konzept abgebildet werden" (Kopp und Mandl 2006, S. 127).

Sie haben zwei Hauptfunktionen:

> „Zum einen lenken sie bei der Wahrnehmung die Aufmerksamkeit, zum anderen unterstützen sie die Integration von Wissen" (Kopp und Mandl 2006, S. 129).

Eigenschaften von Schemata sind, dass

- das durch sie erworbene Wissen – sei es konkret oder abstrakt, generisch oder episodisch (dynamische Abläufe) – auf allen Abstraktionsebenen repräsentiert werden kann,
- Leerstellen vorkommen können, die nach Bedarf mit Erfahrungen oder in neuen Situationen neu belegt werden können,
- Schemata ineinander verschachtelt und hierarchisch organisiert sein können,
- Schemata erworben und in verschiedenen Situationen angewendet werden können,
- Schemata zentral beim Erwerb neuen Wissens sind, da sie darauf abzielen, bestmöglich mit den zu verarbeitenden Informationen zu korrespondieren, um Verstehen zu unterstützen (Kopp und Mandl 2006, S. 128).

An Erwerb und Veränderung von Schemata sind die Prozesse Wissenszuwachs, Feinabstimmung und Umstrukturierung beteiligt (Kopp und Mandl 2006, S. 129). Kopp und Mandl stellen fest, dass die Anwendung fremdinduzierter Schemata für den Lernprozess hilfreich ist. Sie betonen, dass solche Schemata ausgesprochen domänenspezifisch sind, da sie erst von Experten eines Fachgebietes erarbeitet werden müssen. Insgesamt unterscheiden sie drei Arten von Schemata: (1) Darstellungsschemata als abstrakte Metastruktur. Entwurfsmuster der Softwaretechnik entsprechen diesem Schematyp. (2) Problemlöseschemata als Vorstrukturierung des Problemlöseprozesses. Entwurfsmuster geben ebenfalls Hinweise zur Problemlösung. Mehrnoch entspricht jedoch die Vorgehensweise zur systematischen Erkundung des Verhaltens von Informatiksystemen diesem zweiten Schematyp (Abschnitt 5.5). (3) Falllöseschema als Spezialfall des Problemlöseschemas (Kopp und Mandl 2006, S. 131f).

Eine ähnliches Verständnis von Schemata wird beispielsweise genutzt, um kognitive Belastungen gemäß der Cognitive Load Theory (CLT) zu reduzieren (Merrienboer und Sweller 2005). Die CLT geht davon aus, dass das menschliche Arbeitsgedächtnis im Gegensatz zum Langzeitgedächtnis in seiner Kapazität begrenzt ist. Das fortlaufende Umorganisieren und Vergleichen mit bestehenden Strukturen führt zu neuem Wissen, das in Form von Schemata im Langzeitgedächtnis gespeichert wird. Vorteil dieser Schemata ist, dass sie als eine Informationseinheit aufgefasst werden und dadurch die Belastung des Arbeitsgedächtnisses reduzieren. In der CLT wird

darüber hinaus beschrieben, welche Effekte die Belastung des Arbeitsgedächtnisses hat und wie sie soweit reduziert werden kann, dass Schemata gebildet werden können, d. h. dass Lernen stattfindet (vgl. Börstler 2007).

Das Begriffspaar „Schema – Ausprägung" ist für die Informatik prägend (Wedekind et al. 2004). Wie oben angedeutet sind Ausprägungen einer universellen Beschreibung wie Handlungs- und Objektschemata zuzuführen. Objektschemata (z. B. Schüler, Lehrer) sind jedoch leichter erlernbar. Die Entwurfsmusterstruktur, bestehend aus zentralen Aspekten wie Name, Problembeschreibung, Lösungsbeschreibung und Konsequenzen, unterstützen die Schemabildung. Es ist zu unterscheiden zwischen internen Schemata und Beschreibungsschemata wie sie beispielsweise bei Entwurfsmustern vorliegen. Dennoch werden solche Beschreibungsschemata als fremdinduzierte Schemata genutzt, um die interne Schemabildung zu unterstützen. Auffällig ist die Ähnlichkeit von Entwurfsmustern zu solchen Schemata (vgl. auch Clancy und Linn 1999) wie beispielsweise dem Wissensschema „DICEOX" (Brooks und Danserau 1983), das zur Förderung des Textverständnisses auf zentrale Aspekte eines Texts aufmerksam macht. Aktiviert man beispielsweise das Schema „Entwurfsmuster Proxy", werden einzelne Komponenten und Beziehungen zur Realisierung der Zugriffskontrolle kognitiv repräsentiert. Je nach Kontext werden unterschiedliche Informationen, z. B. die Frage der Zugriffsrechte, relevant. Schemata als bewusste Lernstrategien können explizit oder implizit in den Lern-Prozess integriert werden.

Externe Repräsentation von Entwurfsmustern

Ebenso wie die Eigenschaft, ein Schema zu bilden, ist die externe Repräsentierbarkeit von Entwurfsmustern lernförderlich und hilft, Komplexität zu bewältigen (→ Kriterium 5: Komplexität). Lernstrategien der externen Visualisierung haben nach Renkl und Nückles (2006) unterschiedliche Funktionen. Eine wichtige ist die Förderung der Verarbeitungstiefe von Unterrichtsinhalten durch Herstellen von Zusammenhängen zwischen neuen Unterrichtsinhalten, Reduktion durch Identifikation von Hauptpunkten sowie Elaboration. Das heißt, die Beschreibung des Herstellens von Zusammenhängen zwischen Neuem und Vorwissen. Ein weiterer Aspekt ist die Förderung der Metakognition, da durch externe Repräsentation „Wissens- und Verstehenslücken" (vgl. Renkl und Nückles 2006, S. 136) den Lernenden bewusst werden. Externe Repräsentationen haben außerdem eine Übersetzungsfunktion von einem Repräsentationscode in einen anderen. Auf Kompetenzentwicklung mit Informatiksystemen übertragen kann dies beispielsweise der Wechsel von Quelltext zu Klassendiagrammen sein. Nach Renkl und Nückles (2006) könnten externe Visualisierungen mentaler Modelle oft besser als Stimuli für Lernende dienen, als es die mentalen Modelle selbst können (vgl. Renkl und Nückles 2006, S. 135f). Es wird zwischen bereichsunabhängigen Techniken wie Mind Maps und Concept Maps sowie bereichsspezifischen Techniken der externen Visualisierung unterschieden. Beispiele für letztere sind Klassen- und Sequenzdiagramme.

Die Wirksamkeit von Techniken der externen Visualisierung ist nach Renkl und Nückles (2006) jedoch kaum empirisch untersucht. Allein bei Concept Maps lägen die Befunde vor, dass sie Textverständnis, Metakognition und die wahrgenommene Selbstwirksamkeit des Lernens fördern. Dies gelte allerdings nur, wenn die Lernenden nicht überfordert sind. Als hilfreich hätten sich Anleitungen zum Erstellen der Repräsentationen erwiesen. Bei leistungsstarken Lernenden, die korrekte Repräsentationen erstellten, führte der Einsatz zu erhöhter Transferleistung. Renkl und Nückles sehen zwei Hauptprobleme für Lernende bei externen Visualisierungen: Sie binden kognitive Ressourcen, und es kann für Lernende schwierig sein, korrekte externe Visualisierungen anzufertigen. Sie schlussfolgern, das Lernende ggf. bei der Erstellung unterstützt werden müssen. Sie weisen darauf hin, dass Kompetenzen zum Anwenden grafischer Darstellungen durch die Nutzung externer Repräsentationen gefördert werde und verweisen auf die PISA-Studie-2000, die die Ausprägung dieser Kompetenzen fordert. Des Weiteren wird auch die Fähigkeit gefördert, unterschiedliche Repräsentationsarten aufeinander zu beziehen. Gerade für Kompetenzentwicklung mit Informatiksystemen, bei der unterschiedliche Sichten auf Informatiksysteme und innere Abläufe notwendig sind, ist dieser Aspekt essentiell. Renkl und Nückles schließen mit dem Hinweis, dass man mangels empirischer Befunde

> „aus theoretischer Perspektive genau überlegen [muss], welche kognitiven Prozesse vermutlich durch spezifische Varianten ausgelöst werden und ob man diese Prozesse anstrebt" (vgl. Renkl und Nückles 2006, S. 146).

Grafische Darstellungen dominieren die objektorientierte Modellierung. Ein Entwurfsmuster ist die Zusammenfassung mehrerer Repräsentationen wie Klassen, Sequenz- und Objektdiagrammen, aber auch textueller Beschreibung und Angabe von Quelltextbeispielen zu einer semantischen Einheit. Es ist Repräsentation und Artefakt aus der Softwareentwicklung zugleich. Vorteil der semantischen Einheit ist die damit gegebene Strukturierung des Wissens. Fasst man dies zusammen, so unterstützen beispielsweise Klassen-, Sequenz- und Zustandsdiagramme zu Entwurfsmustern durch Sichtenwechsel den Lernprozess zu Informatiksystemen.

Typischerweise bezeichnet Wissen die Kenntnis von Fakten über Objekte in einem bestimmten Bereich. Ein Entwurfsmuster impliziert Fakten bezüglich der Teilnehmer in dem Subsystem, das eine bestimmte Problemlösung beschreibt. Somit können wir Informatiksysteme und Teile von Informatiksystemen mittels Entwurfsmustern repräsentieren. Darüber hinaus beschreiben Entwurfsmuster Ereignisse, d. h. sie sind ein Repräsentationsformalismus und beschreiben den zeitlichen Ablauf einer Ereignisfolge und die zwischen ihnen bestehenden Ursache-Wirkungs-Beziehungen. Entwurfsmuster sind bereichspezifische Repräsentationen, die beispielsweise in den Fächern Architektur und Informatik eingesetzt werden, und hier wiederum deutliche fachspezifische Unterschiede aufweisen. Daraus folgt eine erforderliche Abgrenzung: Diagrammarten wie Klassen- und Sequenzdiagramm

sind ebenfalls bereichsspezifische Repräsentationen. Entwurfsmuster hingegen sind in ihrer Gesamtheit, bestehend aus Name, Problembeschreibung, Lösungsbeschreibung und Konsequenzen, zu sehen. Dafür wiederum wird ein Bündel an Klassen-, Objekt- und Sequenzdiagrammen eingesetzt. Der Einsatz einer Wissensrepräsentation erfordert, dass man mit ihr arbeiten, sie verändern kann. Genau dies leistet jedes einzelne Entwurfsmuster: Mit einem sehr viel engeren Fokus als beispielsweise ein Klassendiagramm, das ein statisches Modell eines beliebigen Softwaresystem darstellen kann, erlaubt jedes Entwurfsmuster viele Varianten einer spezifischen Problemlösung (z. B. Zugriffskontrolle mit dem Proxy). Darüber hinaus jedoch kann der Kontext der Problemlösung in einem breiteren Sinne beschrieben werden als es allein mit einem Klassendiagramm für dieses Problem möglich wäre, z. B. statische und dynamische Aspekte.

5.4.5 Darstellung der Vernetzung fundamentaler Ideen

Motivation der Darstellung zur fachdidaktischen Kommunikation

Im Folgenden wird erarbeitet, wie eine grafische Repräsentation der vernetzten fundamentalen Ideen zur Förderung der fachdidaktischen Kommunikation aussehen kann. Ziel der grafischen Darstellung ist die Veranschaulichung der Vernetzung für Akteure in Lehr-Lernprozessen und Fachdidaktikforscher. Schüler können über sie zu einer metakognitiven Auseinandersetzung mit den Unterrichtsinhalten angeregt werden. Da Entwurfsmuster in der Sekundarstufe II bisher recht selten thematisiert werden, soll die Darstellung auf einen Blick beantworten, warum ein aus der professionellen Softwareentwicklung bekanntes Informatikinstrument in der allgemeinen Bildung eingesetzt werden kann und sollte. Lernenden liefert die grafische Struktur einen Überblick und Ausblick auf nahe liegende Informatikthemen. Damit leistet die grafische Repräsentation einen Beitrag zur Analyse von Lehr-Lernprozessen. Sie verbessert die fachdidaktische Kommunikation, so dass die Unterrichtsinhalte auf fachliche Fehler analysiert werden können. Außerdem erleichtert die Abbildung der fundamentalen Ideen der Informatik in einer grafischen Repräsentation die Auswahl von Elementen, die in Lernerfolgskontrollen zu prüfen sind.

Zur fachdidaktischen Diskussion ist die Vernetzung der fundamentalen Ideen der Informatik in Entwurfsmustern geeignet zu repräsentieren. Anhand des angestrebten Bildungsziels „Kompetenzentwicklung mit Informatiksystemen" kann dann ausgewählt werden, welche fundamentalen Ideen besonders hervorgehoben werden und wie sie einander bedingen.

Es sind Kriterien für geeignete Darstellungen zu finden (vgl. auch Brinda und Schubert 2001): (1) Analog zu Wissensstrukturen des didaktischen Systems sind zur Auswahl der Darstellungsform Ausdrucksstärke und Übersichtlichkeit gefordert. Ziel ist es, die Entwurfsmuster, die oft in unterschiedlichen grafischen Darstellungen

vorliegen, um einen kompakten Überblick über die im Entwurfsmuster vorhandenen fundamentalen Ideen zu ergänzen. Die Darstellung soll für Lehrer und Schüler gleichermaßen verständlich sein und deshalb keiner großen Einarbeitung bedürfen (Anderson 2001, S. 139ff), (Brinda 2004a, S. 181). (2) Zweites Kriterium ist die Modellierung von Relationen zwischen den vernetzten fundamentalen Ideen. Im Entwurfsmuster vorkommende fundamentale Ideen sind dafür durch Knoten eines Graphen darzustellen. Relationen können als Kanten repräsentiert werden. Es ist eine geeignete und zur besseren Übersichtlichkeit geringe Menge an Relationen wünschenswert. (3) Wenn möglich sollten standardisierte Darstellungsformen genutzt werden, da durch sie die Semantik festgelegt ist und Werkzeuge zur Bearbeitung vorhanden sind. Weitere Kriterien hinsichtlich der Vorkenntnisstrukturierung sind nicht notwendig, da die Darstellung der vernetzten fundamentalen Ideen die Wissensstrukturen aus dem didaktischen System nicht ersetzen soll. Die vorhandene Vernetzung fundamentaler Ideen soll vor allem eine Erklärung des Zusammenwirkens der Ideen liefern. Damit sind Vorrangrelationen, wie bei Lernzielen möglich, nicht das Darstellungsziel.

Ossimitz betrachtet den Begriff des systemischen Denkens für die Mathematikdidaktik. Nach Ossimitz (2002) umfasst es: Vernetztes Denken, Denken in Modellen, dynamisches Denken und systemisches Handeln. Zum vernetzten Denken sind entsprechende Darstellungsmittel notwendig. Das bedeutendste Denkwerkzeug der Systemwissenschaft ist nach Ossimitz das Wirkungsdiagramm. Wirkungsdiagramme kommen aus der Kybernetik und werden von Ossimitz (2002) für die Didaktik der Mathematik eingesetzt, um Abhängigkeiten von Einflussfaktoren in einem System qualitativ zu beschreiben. Übertragen auf Kompetenzentwicklung mit Informatiksystemen kann das Wirkungsdiagramm genutzt werden, um die vernetzten fundamentalen Ideen der Informatik zu visualisieren. Es ist eine informatikunabhängige Repräsentation der durch ein oder mehrere Entwurfsmuster vernetzten fundamentalen Ideen der Informatik.

Begriffsnetze (Concept Maps) als domänenunabhängige Darstellungsform sind prinzipiell auch geeignet, das Netz der fundamentalen Ideen darzustellen. Im Gegensatz zur qualitativ eingezeichneten Beeinflussung der einzelnen Konzepte, wie im Wirkungsdiagramm, kann in Concept Maps die Art der Beziehung spezifiziert werden. Beide Darstellungsformen werden im Folgenden verglichen.

Analyse von Wirkungsdiagramm und Begriffsnetz

Zur grafischen Darstellung der vernetzten fundamentalen Ideen werden die Formalisierungen durch Wirkungsdiagramme und Begriffsnetze auf ihre Eignung untersucht und verglichen. Informatikdarstellungen wie Klassen- und Objektdiagramme werden nicht herangezogen, da sie bereits zur Entwurfsmusterbeschreibung eingesetzt werden und implizit die Vernetzung der fundamentalen Ideen repräsentieren. Eine Darstellung der Vernetzung fundamentaler Ideen durch diese Diagramme der

Objektorientierung kann im Lehr-Lernprozess unnötig verwirren. Es ist nicht ersichtlich, dass die durch Klassendiagramme darstellbaren Operationen und Attribute in diesem Kontext benötigt werden, so dass auf sie zugunsten der Übersichtlichkeit zu verzichten ist. Gegebenenfalls kann durch Vererbung die hierarchische Strukturierung der fundamentalen Ideen der Informatik nach Schwill anhand der drei Masterideen Sprache, Algorithmisierung und strukturiere Zerlegung nachgebildet werden. Allerdings geschähe dies auf Kosten der Übersichtlichkeit und liefert keinen Beitrag zur Kompetenzentwicklung mit Informatiksystemen. Die von Brinda und Schubert für Wissensstrukturen genutzten Und-Oder-Graphen sind für die Darstellung der Vernetzung fundamentaler Ideen ungeeignet, da keine hierarchische Strukturierung vorliegt. Die oben genannten Anforderungen an die grafische Darstellung vernetzter fundamentaler Ideen der Informatik, (1) Übersichtlichkeit und Ausdrucksstärke, (2) Modellierung von vernetzenden Relationen und (3) Nutzung einer standardisierten Darstellungsform, werden im Folgenden an den ausgewählten Repräsentationen, Wirkungsdiagramm und Concept Map, überprüft. Ossimitz definiert Wirkungsdiagramme wie folgt:

> „Ein Wirkungsdiagramm enthält die wesentlichen Systemelemente als Knoten und Wirkungsbeziehungen zwischen diesen Elementen als Pfeile. Dabei wird – wenn möglich – zwischen gleichgerichteten Wirkungen (Vorzeichen „+") und gegengerichteten Wirkungen (Vorzeichen „−") unterschieden" (Ossimitz 2000, S. 8).

Zu beachten ist, dass ein System der Allgemeinen Systemtheorie gemeint ist. Damit ergibt sich die Schwierigkeit, dass es zur Beschreibung vernetzter fundamentaler Ideen nur in Ausnahmefällen möglich sein wird, Wirkrichtungen einzutragen und die Art der Wirkung mit positiver oder negativer Rückkopplung zu bezeichnen. Daher sind zur Darstellung vernetzter fundamentaler Ideen der Informatik Wirkungsdiagramme zu verwenden, die auf diese, nach Ossimitz optionale Rückkopplungsbeschreibung, verzichten. Wirkungsdiagramme dienen dabei zur qualitativen Darstellung, d. h. zur Beschreibung, welche fundamentalen Ideen in dem Entwurfsmuster der Problemlösung dienen (Abbildung 5.3).

Als Beispiel wird das Wirkungsdiagramm des Proxymusters angegeben. Ein Proxyobjekt kontrolliert als Stellvertreter den Zugriff auf ein zu schützendes Objekt. Die im Proxymuster enthaltenen fundamentalen Ideen sind in Abbildung 5.4 mit einem Wirkungsdiagramm dargestellt. Es ist ersichtlich, dass die Wirkungsdiagramme sehr einfach zu erstellen und zu verstehen sind. Sie haben eine hohe Aussagekraft und Übersichtlichkeit, wenngleich ein erläuternder Text notwendig erscheint. Weitere Wirkungsdiagramme für Kompetenzentwicklung mit Informatiksystemen finden sich in der Diplomarbeit von Ufer (2007). Er hat die in Architekturmustern vorhandenen vernetzten fundamentalen Ideen der Informatik überzeugend in Wirkungsdiagrammen präsentiert.

Wirkungsdiagramme sind auch geeignet, vernetzte fundamentale Ideen in kombinierten Entwurfsmustern darzustellen. Insbesondere die Verbindungspunkte der

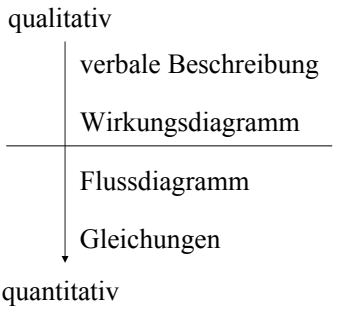


Abbildung 5.3: Wirkungsdiagramme als qualitative Darstellungsmittel zur Beschreibung von Systemen nach (Ossimitz 2002, S. 4)

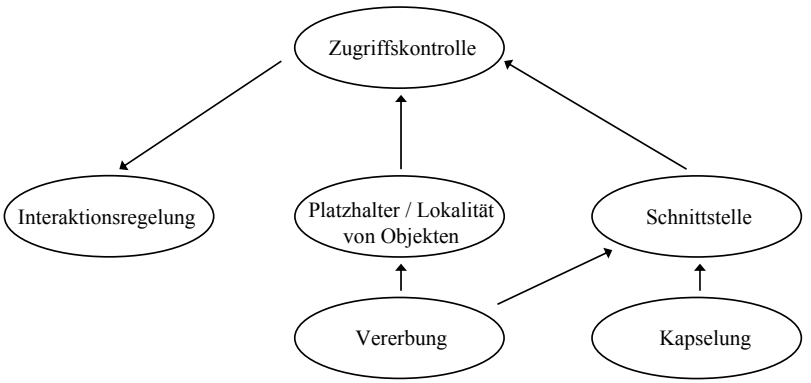


Abbildung 5.4: Wirkungsdiagramm zu den vernetzten fundamentalen Ideen im Proxymuster

Entwurfsmuster sind in Wirkungsdiagrammen beachtenswert, da sie ggf. neue Variationen verlangen. Abbildung 5.5 zeigt das Wirkungsdiagramm zu Proxy, Kompositum und Iterator gemäß des Beispiels aus Abbildung 5.2. So verändert die Kombination eines Iterators mit dem Kompositum das iterative Durchlaufen einer Datenstruktur zu einem rekursiven Traversieren der Baumstruktur des Kompositums, die eine Teil-Ganzes-Beziehung beschreibt.

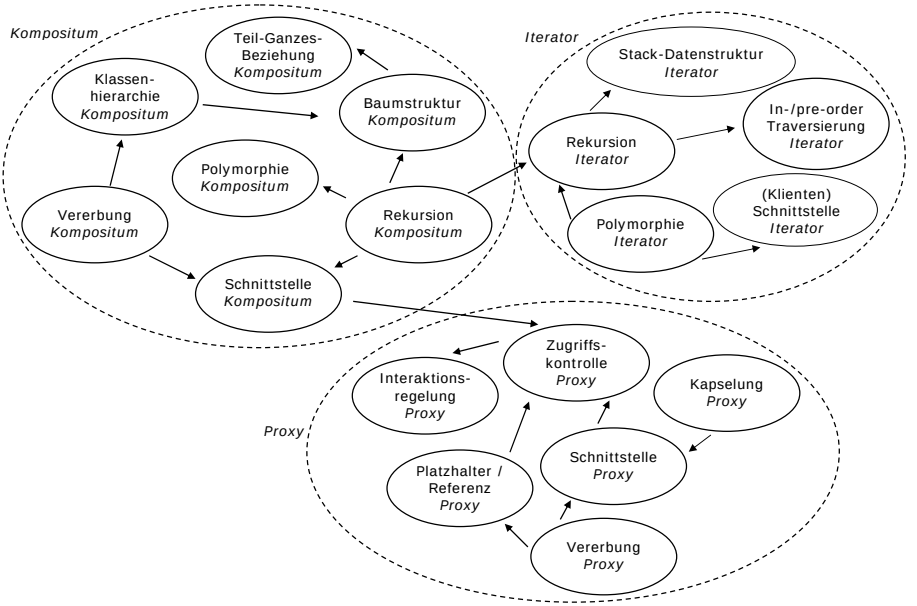


Abbildung 5.5: Wirkungsdiagramm zur Kombination der Entwurfsmuster

Nachdem Wirkungsdiagramme diskutiert wurden, werden Concept Maps untersucht. Sie sind folgendermaßen definiert:

> „A concept map is a spatial array that represents elements of knowledge by means of nodes and directionally labeled or named links, whereas the nodes represent ideas, concepts, or beliefs and the links depict relations between them" (Tergan und Keller 2005, S. 3).

Dadurch wird deutlich, dass sie beliebige n-näre Relationen darstellen können, wie es für die vernetzten fundamentalen Ideen der Informatik hilfreich ist, wenn Hinweise zu den Zusammenhängen oder zur Umsetzung im Unterricht gegeben werden sollen. Damit sind Begriffsnetze potentiell aussagekräftiger als Wirkungsdiagramme. Abbildung 5.6 zeigt eine mögliche Concept Map zu den vernetzten fundamentalen Ideen im Proxymuster. Bei der Bezeichnung der Relationen wurden ausschließlich

ermöglicht- und *erfordert*-Relationen genutzt, da sie das Zusammenwirken zur Realisierung von Zugriffskontrolle erklären und implizit Hinweise auf die Umsetzung im unterrichtlichen Geschehen erlauben. In den hier gezeigten Fällen ist es möglich, sie bei Änderung der Lese- bzw. Pfeilrichtung auszutauschen.

Alternativ dazu kann eine ausführlichere Beschreibung der Relationen im Begriffsnetz erfolgen, die jedoch dem Prinzip der Übersichtlichkeit widersprechen kann. Gleichzeitig ist anzumerken, dass auch ausführlichere Beschreibungen durch Relationen nicht äquivalent zu der Musterbeschreibung im Entwurfsmusterkatalog sein werden, in der beispielsweise Klassendiagramme und Varianten des Musters angegeben sind. Mit Bezug zu Abbildung 5.3 ist das Begriffsnetz zwischen Wirkungsdiagramm und Flussdiagramm als qualitatives Darstellungsmittel einzuordnen.

Die Analyse ergibt, dass sich sowohl vereinfachte Wirkungsdiagramme als auch Begriffsnetze zur Darstellung der vernetzten fundamentalen Ideen eignen. Letztere ermöglichen ein weites Spektrum möglicher Relationen. Empfohlen seien *ermöglicht*- und *erfordert*-Relationen, da sie explizit das Zusammenwirken der fundamentalen Ideen erklären und implizit Hinweise auf die unterrichtliche Gestaltung geben. Beide Diagrammarten stellen die Vernetzung der fundamentalen Ideen der Informatik qualitativ dar. In Wirkungsdiagrammen und Concept Maps repräsentieren die Knoten fundamentale Ideen der Informatik. Deren Kanten repräsentieren Relationen anhand derer ersichtlich wird, wie die fundamentalen Ideen im Entwurfsmuster zusammenhängen. Die Darstellung der vernetzten fundamentalen Ideen in einem Entwurfsmuster wird in der Unterrichtsvorbereitung kaum vollständig erfolgen, sondern Schwerpunkte für den Lehr-Lernprozess setzen. Die eingezeichneten Wirkbeziehungen umfassen Intentionen und sind abhängig vom angestrebten Lehr-Lernprozess.

Fazit ist, dass sowohl Concept Maps als auch die vereinfachten Wirkungsdiagramme aufgrund ihrer Ausdrucksstärke und Übersichtlichkeit geeignet sind, vernetzte fundamentale Ideen darzustellen. Bei beiden sind die Relationen, die zwischen den fundamentalen Ideen bestehen, dennoch zusätzlich zu klären. Dies geschieht im Regelfall bereits durch die Beschreibung der Entwurfsmuster, wie sie in fachwissenschaftlichen Katalogen vorliegt. Ergänzend muss, wie von Stechert (2006a), Stechert (2006b), Ufer (2007) und Weyer (2007b) durchgeführt, ein erläuternder Text zur Auswahl der fundamentalen Ideen angegeben werden, wie es für Unterrichtsentwürfe üblich ist. Fasst man diese Ergebnisse zusammen, so sind Relationen in Begriffsnetzen, die das Zusammenwirken der vernetzten fundamentalen Ideen erklären, wünschenswert, wenn die Übersichtlichkeit nicht gefährdet ist. Hinsichtlich der Auswahl von Relationen in den Begriffsnetzen ist zu beachten, dass insbesondere Vorrangrelationen, die eine Reihenfolge für den Unterricht implizieren, einer Erklärung bedürfen. Vor allem erscheinen fundamentale Ideen als Themen nicht geeignet, um die Reihenfolge im Unterricht vollständig zu begründen: Lernziele sind im Fall von Vorrangrelationen zu ergänzen, und es gibt Darstellungsformen für

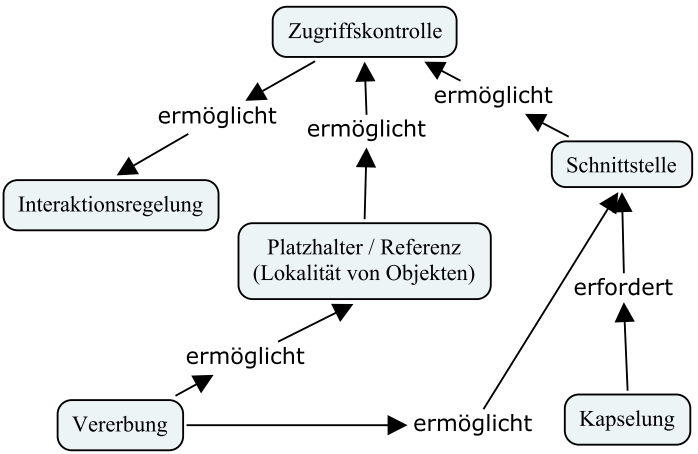


Abbildung 5.6: Concept Map zu den vernetzten fundamentalen Ideen im Proxymuster

Vorrangrelationen, die zu diesem Zweck entwickelt wurden, z. B. Wissensstrukturen des didaktischen Systems (Brinda und Schubert 2001), (Freischlad 2008) oder Lernzielgraphen (Steinert 2007).

Kritische Betrachtung von Entwurfsmustern im Informatikunterricht der Sekundarstufe II

Entwurfsmuster sind anspruchsvolle Konzepte der Softwaretechnik zur Herstellung qualitativ hochwertiger Software. In diesem Zusammenhang ist zu beantworten, warum ausgewählte Entwurfsmuster trotz Abstraktheit und Komplexität geeignet sind, Basiskompetenzen zu Informatiksystemen zu fördern. Vernetzte fundamentale Ideen als Argument sind bereits ausführlich als Begründung diskutiert worden, aber es gibt vor allem in der hochschuldidaktischen Informatikforschung Kritik am Einsatz von Entwurfsmustern. Sie werden im Folgenden aufgegriffen, und es wird diskutiert, inwieweit sie auf den Einsatz von Entwurfsmustern zur Förderung der Kompetenzentwicklung mit Informatiksystemen zutreffen.

Im Zuge der Forschung um Entwurfsmuster nach Gamma et al. (1995) sind einige Ansätze erprobt worden, Muster der imperativen Programmierung als Gegenstand der universitären Lehre (wieder) einzuführen (vgl. (Porter und Calder 2003), (Proulx 2000)).

Wallingford (1996) diskutiert sowohl imperative als auch objektorientierte Programmiermuster und schlussfolgert, dass für die Anfängerausbildung an Hochschulen einfachere Muster für die Objektorientierung entwickelt werden müssen. Seiner

Meinung nach sind die für die Softwareentwicklung beschriebenen Entwurfsmuster aus zwei Gründen ungeeignet:

> „First, that work targets primarily more advanced software practitioners. Second, it tends to emphasize design issues, whereas a first course must provide a significant amount of support for novice *programmers*" (Wallingford 1996, S. 29; Hervorh. im Original).

Deutlich wird an dieser Aussage, dass Entwurfsmuster vor allem im Kontext von Programmierkursen diskutiert werden. Diese Aussage gilt für den Informatikunterricht zur Förderung der Kompetenzentwicklung mit Informatiksystemen jedoch nicht zwangsläufig, da im Rahmen einer Wissenschaftspropädeutik durchaus die bekannten Entwurfsmuster eingesetzt werden sollten und die Entwurfsmuster von Schülern eben nicht zum Erstellen komplexer Softwaresysteme genutzt werden. Eine didaktisch begründete Auswahl einiger Entwurfsmuster ist dennoch notwendig.

Bezüglich der Objektorientierung ist Astrachan (2001) noch deutlicher. Er übt Kritik an dem Ansatz, Objektorientierung und Entwurfsmuster um jeden Preis in Anfängervorlesungen an Hochschulen zu behandeln, und fordert einen angemessen Aufgabenkontext, da Entwurfsmuster eine Methodik für komplexe Systeme sind. Dies führt zu einem Zielkonflikt um ein minimales Szenario, das komplex genug ist, um die Vorteile der Entwurfsmuster zu lehren, und nicht zu komplex für die Studierenden im Grundstudium bzw. in der Sekundarstufe II. Astrachan schließt seine Überlegungen damit, dass er Entwurfsmuster immer dann lehren wird, wenn ein für die Zielgruppe geeignetes Szenario vorliegt. Allerdings ist er bereit, auf (einige) Entwurfsmuster zu verzichten, wenn gegebene Probleme ohne Entwurfsmuster einfacher und eleganter gelöst werden können. Außerdem bevorzugt er den Vergleich von einer entwurfsmusterbasierten mit einer herkömmlichen objektorientierten Lösung eines Problems gegenüber der ausschließlichen Verwendung vieler verschiedener entwurfsmusterbasierter Problemlösungen, d. h. er verzichtet auf das Lehren eines umfangreichen Katalogs an Entwurfsmustern. Für Kompetenzentwicklung mit Informatiksystemen ist die Konsequenz, dass der Beitrag jedes Entwurfsmusters sorgfältig zu begründen ist.

Fasst man die Kritikpunkte zusammen, so beziehen sie sich vornehmlich auf den Beitrag der Entwurfsmuster zum Programmierenlernen. Die Komplexität eines Szenarios, in dem ein Entwurfsmuster für die Softwareentwicklung sinnvoll ist, wird in diesem Kontext als hinderlich betrachtet. Für Kompetenzentwicklung mit Informatiksystemen ist dies jedoch wegen der geringeren Anforderungen an die Programmierkenntnisse nicht in dem Maße relevant. Ein einzelnes Entwurfsmuster kann zum Problemraum erklärt werden, wenn es für Kompetenzentwicklung mit Informatiksystemen förderlich ist. Nach Schubert und Schwill (2004) darf eine

> „didaktische Vereinfachung die Vernetzung von Einflussgrößen nicht zerstören, kann aber sehr wohl einen Realitätsausschnitt zum Problemraum erklären, der als ‚abgeschlossene Welt' behandelt wird" (Schubert und Schwill 2004, S. 53).

So ist auch die Auswahl eines zur Erklärung eines für Kompetenzentwicklung mit Informatiksystemen relevanten Netzes fundamentaler Ideen in einem Entwurfsmuster als abgeschlossene Welt zu begründen. Die u. a. von Porter und Calder (2003) geforderte Verlagerung des Schwerpunktes von der Programmierung auf den Entwurf eines Systems ist hingegen auch für Kompetenzentwicklung mit Informatiksystemen nachvollziehbar. Insgesamt wird die Kritik am Einsatz von Entwurfsmustern in der vorliegenden Arbeit so interpretiert, dass eine didaktisch begründete Auswahl von Entwurfsmustern vorzunehmen ist und hinsichtlich der Machbarkeit eine Erprobung im Informatikunterricht stattfinden muss.

Kritische Betrachtung von Entwurfsmustern für Kompetenzentwicklung mit Informatiksystemen

Neben dem Einsatz von Entwurfsmustern in der Schule muss auch ein Beitrag zu Informatiksystemen und Kompetenzentwicklung kritisch diskutiert werden. Mit Entwurfsmustern wurde ein Mittel gefunden, das vernetztes Denken fördern kann und die Erklärung des nach außen sichtbaren Verhaltens eines Systems über eine bewährte Struktur ermöglicht. Eingeführt in das Unterrichtsmodell, um vernetzte fundamentale Ideen der Informatik zu repräsentieren, sind Entwurfsmuster dennoch Strukturierungsmuster und Heuristiken aus der Informatikpraxis. Mit ihnen finden sich für das Unterrichtsmodell sehr viele Anknüpfungspunkte an den traditionellen objektorientierten Informatikunterricht für eine Unterrichtsreihe zum Thema Informatiksysteme. Dennoch ist Kompetenzentwicklung mit Informatiksystemen nicht zwangsläufig mit der objektorientierten Modellierung der Sekundarstufe II verknüpft. Vielmehr können im Sinne eines Spiralcurriculums besonders Themen und Fragestellungen zum nach außen sichtbaren Verhalten bereits früher und ggf. ohne Objektorientierung im Informatikunterricht behandelt werden. Es bleibt festzuhalten, dass Entwurfsmuster eine besondere Art von Strukturmodellen darstellen, die aufgrund ihrer Anwendung als Problemlösemuster unter bestimmten Voraussetzungen die Kompetenzentwicklung mit Informatiksystemen fördern. Informatiksysteme und Kompetenzentwicklung sind jedoch nicht notwendigerweise an objektorientierte Entwurfsmuster gebunden (siehe auch Kapitel 4).

5.4.6 Zusammenfassung zu Entwurfsmustern als Wissensrepräsentationen

Aus der Analyse des Forschungsstandes wurde geschlossen, dass es an lernförderlichen Strukturen fehlt, die Informatikunterricht zu Wirkprinzipien von Informatiksystemen unterstützen. In Entwurfsmustern wurde Potential zur Förderung der Kompetenzentwicklung mit Informatiksystemen gesehen, und in der vorliegenden Arbeit wurden sie auf vernetzte fundamentale Ideen der Informatik untersucht. Grund für die Verknüpfung von Entwurfsmustern mit vernetzten fundamentalen

Ideen der Informatik war, dass die Analyse des Forschungsstandes ergab, dass die Vernetzung der Grundbegriffe und Wirkprinzipien von Informatiksystemen bisher eine Aufgabe blieb, die der Lehrer selbst lösen musste. Ausgewählte Entwurfsmuster wurden nach Anwendung der Klassifikation als Wissensrepräsentationen für vernetzte fundamentale Ideen bezeichnet. Aufgabe dieser Wissensrepräsentationen ist es, eine fachdidaktische Begründung zur Erreichung des Bildungsziels „Kompetenzentwicklung mit Informatiksystemen" zu liefern.

Fasst man die Diskussion zusammen, so erfüllen ausgewählte Entwurfsmuster als Wissensrepräsentation zur Förderung der Kompetenzentwicklung mit Informatiksystemen folgende Eigenschaften:

- Sie erfüllen eine didaktische Funktion und sind lernfördernd (Lehr-Lerntheorie),
- sie basieren auf bewährten Informatik(system)strukturen (Informatik),
- sie sichern den Bildungswert durch Vernetzung fundamentaler Ideen der Informatik (Didaktik der Informatik),
- sie sind im Unterricht kombinierbar mit weiteren Wissensrepräsentationen, z. B. mit weiteren Softwaremustern, virtuellen Maschinen und Schichtenmodellen.

Ziel ist es, die fachdidaktische Kommunikation zu unterstützen, um Unterricht zur Förderung der Kompetenzentwicklung mit Informatiksystemen begründen und bewerten zu können. Neben der informatischen Darstellung der Wissensrepräsentation, z. B. Musterform nach Gamma et al. (1995), wurden Wirkungsdiagramme und Concept Maps als graphbasierte Darstellungsmittel der vernetzten fundamentalen Ideen der Informatik kurz diskutiert (Abschnitt 5.4.5). Zusammenfassend ist somit das Ziel des Einsatzes von Wissensrepräsentationen im Lehr-Lernprozess das Komprimieren und Veranschaulichen, um das Aneignen und Rekapitulieren zu unterstützen.

5.5 Entwicklung von Vorgehensweisen zur Erkundung von Informatiksystemen

5.5.1 Experimente und Tests zur Erkundung von Informatiksystemen

Begründung für Experimente zur Erkundung von Informatiksystemen

Ziel dieses Abschnitts ist es, eine Schülervorgehensweise zur Förderung der Kompetenzentwicklung mit Informatiksystemen im Informatikunterricht anzugeben, die gleichzeitig mit dem Ansatz der Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik kombiniert werden kann (Abschnitt 5.4).

Winograd und Flores (1988) betonen, dass das Verhalten komplexer Systeme kaum vorhersagbar und deshalb das Systemverhalten schrittweise zu erkunden ist:

> Similarly, in a complex computer system designed and functioning properly at a higher level there is often no way to predict how a program will act, short oft running it (or simulating it step by step, which is of the same complexity). In interacting with such a system we tend to treat it in the same way we treat organisms – we operate by perturbing it and observing the results, gradually building up an understanding of its behavior" (Winograd und Flores 1988, S. 95).

Komplexe Systeme sind demnach durch Intransparenz gekennzeichnet (Dörner und Schaub 1994), und dem Experimentieren kommt eine Schlüsselrolle zu. Um das Verhalten von Informatiksystemen zu analysieren, wird sowohl in der Fachwissenschaft das Experimentieren als auch für die Schulinformatik ein experimentierender, handlungsorientierter Zugang zu Informatiksystemen gefordert: Gemäß Duden Informatik ist das nach außen sichtbare Verhalten von Informatiksystemen durch Tests und Experimente zu ermitteln (Claus und Schwill 2006, S. 677). Für die Schulinformatik wird beispielsweise 1993 im ACM-Curriculum das Experimentieren gefordert (ACM 1993, S. 3). Im aktuellen UNESCO ICT Curriculum ist Experimentieren eine empfohlene Lehr-Lernmethode:

> „Experimenting is one of the principal means to perform exploratory learning and to construct knowledge – based on experience" (UNESCO 2002, S. 94).

Konkrete Handreichungen zu Aufbau und Durchführung der Experimente zur Förderung der Kompetenzentwicklung mit Informatiksystemen werden jedoch nicht angegeben.

Definition von Forschungsexperiment und Unterrichtsexperiment

Begrifflich ist zwischen dem Forschungs- und Unterrichtsexperiment zu unterscheiden. Das Forschungsexperiment entspricht der systematischen Erforschung von Gesetzmäßigkeiten:

> „Informatiker simulieren, experimentieren, prüfen und analysieren am Modell, um zu neuen *Erkenntnissen* zu kommen. Im Unterschied zum Naturwissenschaftler gilt das Erkenntnisinteresse jedoch nicht den Gesetzen oder Phänomenen der Natur, sondern vor allem kulturellen Phänomenen (Schefe 1999, S. 123), dem intendierten Informatiksystem und den zugrunde liegenden Strukturen" (Thomas 2002, S. 64; Hervorh. im Original).

Dabei sind jedoch nicht allein intendierte Informatiksysteme das Ziel, sondern durchaus auch bestehende, unüberschaubare Systeme. In diesem Sinne ist das Experiment im Duden Informatik definiert:

> „**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" (Claus und Schwill 2006, S. 243; Hervorh. im Original).

Systematisches Testen in der Kerninformatik und im Informatikunterricht

Testen wird mit Bezug zum Experimentbegriff folgendermaßen definiert:

> „**Testen:** Überprüfung des Ein- / Ausgabeverhaltens eines Programms anhand von Experimenten und gezielten Programmdurchläufen" (Claus und Schwill 2006, S. 682; Hervorh. im Original).

In den von Thomas (2003) identifizierten Hauptmodelltypen der Kerninformatik (Architekturmodelle, Vorgehensmodelle, Entwurfsmodelle, Untersuchungsmodelle und mentale Modelle) gehört systematisches Testen zu den Untersuchungsmodellen. Das systematische Testen korrespondiert stark mit dem Qualitätsmerkmal Zuverlässigkeit von Informatiksystemen (Kapitel 3). Im Unterricht können Anforderungen an und Qualitätsmerkmale von Informatiksystemen von den Schülern erarbeitet werden und anschließend ein System systematisch getestet werden.

Für Kompetenzentwicklung mit Informatiksystemen sind die Analyse der inneren Struktur und des nach außen sichtbaren Verhaltens von Informatiksystemen notwendig. In der Fachwissenschaft wird Testen in White-Box-Testen, d. h. Implementierung und innere Struktur sind bekannt, oder aber Black-Box-Testen unterschieden (Abschnitt 3.2.1). Bei letzterem wird allein das nach außen sichtbare Verhalten untersucht. Er ist somit programmiersprachenunabhängig:

> „Beim Black-Box-Test geht man von den funktionalen Anforderungen an das Programm aus, d. h. von der Interaktion des Programms mit seiner Umgebung und seiner Wirkung auf diese. Die Auswahl der Testfälle richtet sich nach den Eingaben, Ausgaben und ihrer funktionellen Verknüpfung" (Frühauf et al. 1997, S. 47).

Fazit an dieser Stelle ist, dass zur Analyse des nach außen sichtbaren Verhaltens eine Vorgehensweise für den Unterricht in Anlehnung an den Black-Box-Test zu entwickeln ist. Man spricht von einem optimalen Test, wenn alle Kombinationen von Ein- und Ausgaben für alle funktionellen Verknüpfungen getestet werden. Aufgrund der kombinatorischen Explosion versucht man in der Praxis, gleichwertige Ergebnisse auf eine Klasse von Eingabewerten zusammenzufassen und nur einmal zu testen. Diese Klassen von Eingabewerten nennt man Äquivalenzklassen. Grenzwerte bilden hier oft eine eigene Äquivalenzklasse. Sülz (2007) hat in einer Seminararbeit betreut vom Autor eine Unterrichtssequenz zu diesem Thema beschrieben und dabei das fachliche Kriterium für gute Testfälle aufgegriffen:

> „Die für den Testfall gewählten Werte sind typisch, stehen also stellvertretend für eine ganze Klasse von Eingabewerten" (Thaller 1994, S. 99).

Dieses Kriterium ist für Kompetenzentwicklung mit Informatiksystemen relevant, läuft es doch auf eine Klassifizierung des Verhaltens von Informatiksystemen hinaus. Verifikation und Validierung werden eingesetzt, um zu überprüfen, ob ein Programm korrekt arbeitet, d. h. sie dienen zur Kontrolle der Zuverlässigkeit eines

Informatiksystems. Verifizieren ist dabei das Prüfen, ob in dem Informatiksystem vorgegebene Anforderungen erfüllt sind, während Validieren das Prüfen auf Eignung für den vorgesehenen Verwendungszweck ist (vgl. Frühauf et al. 1997). Nach Schwill ist Verifikation eine fundamentale Idee der Informatik. Dafür zeigt Schwill (1993b), dass es möglich ist, die Verifikation eines Programms im Informatikunterricht aller Stufen zu behandeln (vgl. auch Schwill 1994). Im nächsten Abschnitt wird die Entwicklung von Unterrichtsexperimenten zur Förderung der Kompetenzentwicklung mit Informatiksystemen beschrieben.

5.5.2 Unterrichtsexperimente zur Förderung der Kompetenzentwicklung mit Informatiksystemen

Im Folgenden wird die Definition des Unterrichtsexperiments nach Meyer verwendet, der betont, dass das Unterrichtsexperiment weiter gefasst ist als in der Wissenschaft üblich: Beobachtungsaufgaben und Messungen werden als Experiment bezeichnet (Meyer 2006, S. 316). Transferiert man das in der Informatik geforderte Testen und Experimentieren adäquat auf Unterricht, so kann in beiden Fällen von Unterrichtsexperimenten gesprochen werden. Grundannahme ist, dass das Verhalten von Informatiksystemen systematisch erkundet werden kann und rational, beispielsweise anhand vernetzter fundamentaler Ideen der Informatik, verstehbar wird:

> „Den Prozess, sich Informatikwissen und -können anzueignen, soll der Schüler als systematisches Testen des Informatiksystems auffassen. Er hat dann den ersten Schritt zum Verständnis der interaktiven Arbeit mit dem Rechner bewältigt" (Schubert und Schwill 2004, S. 43).

Als Konsequenz des Kompetenzbegriffs ist die aktive Bearbeitung einer Problem- bzw. Anforderungssituation durch die Schüler ein Gestaltungsmerkmal des zu entwickelnden Unterrichtsmodells. Nach Meyer führt Handlungsorientierung zu einem schülerorientierten Unterricht (Meyer 2005, S. 215). Schülerorientierter Unterricht versucht, den subjektiven und objektiven Bedürfnissen und Interessen der Schüler zu entsprechen (Meyer 2005, S. 216). Problemorientierung ist ein zur Motivation und zur Gestaltung von Anforderungssituationen hilfreiches Instrumentarium im Informatikunterricht. Ein Problem besteht dabei aus unerwünschtem Anfangszustand, erwünschtem Zielzustand und einer zu überwindenden Barriere (Edelmann 1986). Problemsituationen entsprechen den wichtigen komplexen Anforderungssituationen und sind nicht routinemäßig lösbare Aufgaben. Solches Problemlösen gehört zur allgemeinen Bildung. Gerade entdeckender Unterricht (nach Bruner) fordert, dass Schüler ein Problem analysieren, prozessorientierte Lernhilfen bekommen, Hypothesen und Lösungen formulieren, die von der Lehrperson überprüft werden. Nach Edelmann (1986) sind Probleme optimal, die sich durch Anwenden von Strategien oder systemisches Denken lösen lassen (vgl. (Hubwieser 2007a, S. 69), (Diethelm 2007, S. 48)). Hubwieser betont, dass strikte Problemorientierung den

Informatikunterricht vor Produktschulungscharakter verschont (Hubwieser 2007a, S. 69). Informatikunterricht ist deshalb so geeignet für Problemorientierung, weil Problemlösestrategien von den Schülern nicht nur angewendet, sondern für die Problemlösung mit Informatiksystemen auch zu formalisieren sind (Schubert und Schwill 2004). Dies gilt hinsichtlich der Kompetenzentwicklung mit Informatiksystemen nicht nur für die Programmentwicklung, sondern auch für den Anwendungsprozess. Übergänge zwischen dem entdeckenden Lernen und dem Experimentieren der Schüler sind fließend (Meyer 2006, S. 317). Wagenschein (1991) stellt in seiner Verallgemeinerung der bildungstheoretischen Didaktik das Exemplarische für entdeckendes Lernen in den Vordergrund, da es als Spiegel des Allgemeinen dient (Jank und Meyer 2002, S. 37).

Der typische Ablauf von Unterrichtsexperimenten in Anlehnung an Meyer (Meyer 2006, S. 318) ist:

1. Beobachtung eines Phänomens (z. B. ein unerwartetes Verhalten eines Informatiksystems),
2. Formulierung einer klaren Fragestellung durch Lehrer und Schüler (Hypothesen zur Erklärung der Beobachtung),
3. Konzeption eines Experiments zur Überprüfung der Hypothese durch Lehrer und / oder Schüler,
4. Durchführung des Experiments und Dokumentation der Ergebnisse,
5. Bestätigung oder Widerlegung der Hypothesen durch Interpretation der Ergebnisse und des Versuchsverlaufs,
6. Diskussion um Kontrollversuche, Varianten, technische und soziale Folgen des Einsatzes von Informatiksystemen.

Schubert und Schwill halten fest, dass die wesentlichen Schülertätigkeiten das Planen des Experiments, das Manipulieren und das Beobachten des Untersuchungsgegenstands sind (Schubert und Schwill 2004, S. 240). Außerdem ist eine Auswertung des Experiments vorzunehmen. Paul (1994) unterscheidet bei der Exploration zwischen den Interaktionsformen Erkundung und Experiment. Durch erkundende Interaktion verschafft sich ein Lernender einen Überblick über das Leistungsspektrum des Informatiksystems. In Experimenten überprüft der Lernende seine Hypothesen über die Wirkungsweise von Funktionen. In Unterrichtsexperimenten können sowohl Erkundung als auch Experiment nach Paul zusammengefasst werden, denn die Erkundung nach Paul liefert Beobachtungen, die Motivation für ein hypothesengeleitetes Vorgehen sein können. Beobachtungsaufgaben kommt also eine wichtige Rolle in den Experimenten für Kompetenzentwicklung mit Informatiksystemen zu:

> „Lernen durch Beobachtung bedeutet aktive Auseinandersetzung mit dem Lerngegenstand. Daher muss der Lernende aufgefordert und angeleitet werden, mitzudenken, die Tätigkeit des Vorbilds in Bestandteile aufzugliedern und gleichzeitig deren

> Zusammenhänge zu erkennen, die gesehene Tätigkeit vorstellungsmäßig mitzuvollziehen" (Hacker und Skell 1993, S. 229).

So verstandene Beobachtungsaufgaben können zur Motivation als Vorstufe eines Experimentiervorganges zum systematischen Erkunden von Informatiksystemen genutzt werden. Schülern muss es beim Experimentieren gelingen, Hypothesen über fachliche Zusammenhänge aufzustellen und diese experimentell zu überprüfen. Interaktionen eines Schülers mit dem Informatiksystem sind Ausgangspunkt für ihn, Hypothesen über den Lerngegenstand, das Informatiksystem, zu bilden und diese in Experimenten (Interaktionen mit dem System) zu überprüfen (vgl. Paul 1994).

Ziel von Unterrichtsexperimenten ist auch die Förderung der Methodenkompetenz der Schüler zum Planen von Abläufen. Sie bezieht sich nicht auf den Kompetenzbegriff nach Weinert (Abschnitt 2.1). Man unterscheidet Unterrichtsexperimente nach den Ausführenden in Lehrer- und Schülerexperiment, wobei ersteres oft eine Demonstration ist (Meyer 2006, S. 316f). Solches Experimentieren im Unterricht wird gefordert, um systematische Vorgehensweisen zu lernen und Beobachtungsfähigkeit zu schulen. Darüber hinaus werden sie nach ihrem Ziel unterschieden: Experimente zur systematischen Erkundung von Informatiksystemen können beispielsweise als Einführungsexperiment zur Motivation dienen. Oder sie sind Entdeckungstätigkeiten zur Erkundung von Beziehungen und Strukturen bzw. Bestätigungsexperimente zur Überprüfung von Hypothesen über gesetzmäßige Beziehungen und Strukturen (vgl. auch Steinkamp 1999, S. 38).

In Anlehnung an Meyer (2006) werden im Folgenden mögliche Lernziele für Kompetenzentwicklung mit Informatiksystemen skizziert, die Schüler durch systematisches Erkunden von Informatiksystemen durch Unterrichtsexperimente erreichen können. Kognitive und manuelle Lernziele sind, dass die Schüler lernen können, Systemverhalten genau zu beobachten, Beobachtungsresultate zu klassifizieren und sie Oberbegriffen zuzuordnen, Vorhersagen über den mutmaßlichen Experimentierablauf zu formulieren und die Prognosen mit dem tatsächlichen Verlauf zu vergleichen und Schlussfolgerungen auf vernetzte fundamentale Ideen aus den Beobachtungsergebnissen zu ziehen. Des Weiteren lernen sie, die im Experiment ermittelten Daten zu interpretieren, abhängige Variablen zu unterscheiden und Problemstellungen zu operationalisieren, also so zu formulieren, dass sie durch Messoperationen einer Überprüfung zugänglich werden (Meyer 2006, S. 319). Als affektiv-emotionale Lernziele können die Schüler darüber hinaus ihre Neugierde befriedigen und lernen, sich bei der Erkundung zu konzentrieren (Meyer 2006, S. 319). Diese Lernziele sind wichtig zur Kompetenzentwicklung, da Einstellungen und Bereitschaften nicht zu vernachlässigen sind (Abschnitt 2.1). Die Lernförderlichkeit des Experimentierens wird belegt durch nachfolgende Aussage:

> „Das selbsttätige oder auch vom Lehrer locker geleitete Vorbereiten, Durchführen und Auswerten eines Experiments ist auf jeden Fall eine der besten denkbaren Möglichkeiten, die methodische Handlungskompetenz der Schüler zu entwickeln, und

> zwar sowohl in kognitiver als auch in sozialer und motorisch-manueller Hinsicht" (Meyer 2006, S. 318).

Steinkamp entwickelte in einer Diplomarbeit Ansätze zur Übertragung des Experimentbegriffs aus dem naturwissenschaftlichen und technischen Unterricht in die Informatik. Darin kommt Steinkamp zu dem Ergebnis, dass nachfolgend aufgezählte Experimente im Informatikunterricht nicht hinlänglich vertreten sind:

> - „Entdeckende und erkundende Experimente
> Unter diesen Begriff fallen Experimente, deren Durchführungs- und Beobachtungsaufgaben zu Hypothesen über Eigenschaften und Funktionsmöglichkeiten des Untersuchungsobjekts führen sollen. [...]
> - prüfende Experimente
> Bei diesen Experimenten bildet eine Hypothese den Ausgangspunkt, die durch das Experiment erhärtet oder widerlegt werden soll. In diesen Bereich fallen auch güteprüfende Experimente, die zu theoretischen Abschätzungen praktische Belege liefern können [...]
> - strukturermittelnde Experimente
> Hierbei stehen Erkenntnisse über das Zusammenspiel mehrerer Komponenten im Mittelpunkt des Interesses" (Steinkamp 1999, S. 38).

Alle genannten Ansätze sind Unterrichtsexperimente nach (Meyer 2006). Brinda zieht ein positives Fazit zu dem von Steinkamp entwickelten Konzept der Informatik-Experimente und dessen Beitrag zum Erkennen von Wirkprinzipien von Informatiksystemen:

> „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" (Brinda 2004a, S. 118).

Bezug nehmend auf Bunge (1967) betont Brinda für Explorationsmodule, dass die Experimente der Lernenden qualitativ sind:

> „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" (Brinda 2004a, S. 153).

Gleiches gilt für die systematische Erkundung eines Informatiksystems.

Indem die Lernenden Informatiksysteme, die nach fachdidaktischen Kriterien gestaltet sind, systematisch durch Interaktion erkunden, können sie ihre Vorstellungen in Form von Hypothesen mit Experimenten auf Richtigkeit prüfen. Die Lernenden können sich bei der Erkundung an ihren Hypothesen mit Bezug zu fundamentalen Ideen der Informatik bzw. Leitfragen orientieren, die als Beobachtungsaufgaben durch die Lehrperson vorgegeben sind.

Bei der Vorgehensweise zur systematischen Erkundung ist es wichtig, die Strategie des Schülers zur Analyse von Informatiksystemen dahingehend zu lenken, dass er planvoll vorgeht, geeignete Mittel nutzt und wählt, Hilfen umsetzt, seine Ergebnisse

reflektiert und seinen Lösungsweg gegebenenfalls korrigiert. Die Mensch-Maschine-Schnittstelle, an der Informatiksysteme erkundet werden können, ist deren meist grafische Benutzungsoberfläche. Sie besteht in der Regel aus drei Komponenten: der Präsentation zur Darstellung von Daten, der Interaktion, durch die Dialogfunktionen bereitgestellt werden, und die Kontrolle zur Datenanalyse und Ausführung von Programmen und Dialogbefehlen (Claus und Schwill 2006, S. 89). Schubert und Schwill fordern, dass Schüler bei der Interaktion mit Informatiksystemen folgende Stufen der Dialogarbeit durchlaufen: Die Schüler können Ausgaben des Informatiksystems richtig interpretieren und vorhersehen, die Eingaben zur Steuerung der Ausgaben gezielt vornehmen sowie die jeweiligen Ergebnisse im Anwendungskontext bewerten und verantwortlich nutzen (Schubert und Schwill 2004, S. 231f). Dabei ist die Stufung nicht am Informatiksystem, sondern an den kognitiven Prozessen des Schülers orientiert.

Die Ziele der Experimente mit Informatiksystemen können in zwei Bereiche unterteilt werden: den primären, fachlichen und den sekundären, fächerübergreifenden. Der fachliche Anteil des Experiments zur systematischen Erkundung kann durch vernetzte fundamentale Ideen der Informatik erfolgen, die der Grund für ein spezifisches Systemverhalten sind und ihren Hauptfunktionen zugrunde liegen (Denning 2007). Fächerübergreifend ist das Erkennen, das Informatiksysteme, die im Alltag auftreten, systematisch erkundet werden können. Dabei erfahren die Lernenden, dass Informatiksysteme oft unerwartetes Verhalten an den Tag legen (Abschnitt 5.5.3). Wenn dies so ist, müssen sie Hypothesen für den Grund dafür aufstellen können und ggf. Fehlersuche betreiben. Ziel ist, Informatiksysteme und deren Funktionalität zu hinterfragen. Dies geht nur, wenn die Lernenden verschiedene Sichtweisen auf Informatiksysteme verstanden haben und diese miteinander kombinieren können. Ein weiteres Lernziel beim Experimentieren ist Teamarbeit, meist in Form von Partnerarbeit, um die Kommunikation über Informatiksysteme zu fördern. Darüber hinaus soll das strukturierte und planerische Denken gefördert werden, was sich insbesondere durch Überprüfung aller Sonderfälle auszeichnet. Außerdem können die Lernenden eine eigene Dokumentation des Informatiksystems erstellen, also die Entstehung von Modellen selbst nachvollziehen und eigene konstruieren. Im Folgenden wird das Experimentieren zur Analyse des nach außen sichtbaren Verhaltens und der inneren Struktur von Informatiksystemen (Abschnitt 5.5.3 und Abschnitt 5.5.4) beschrieben.

5.5.3 Vorgehensweise zur systematischen Erkundung des nach außen sichtbaren Verhaltens

In der Analyse des Forschungsstands wurden Fehler als Mittel identifiziert, Verhalten und Struktur von Informatiksystemen miteinander zu verbinden (→ Methodik 3: Verbindung von Verhalten und Struktur; S. 142; (Eberle 1996); (Hubwieser 2007a)). Nach Hubwieser ermöglichen es Fehler, die Umgebung der Systeme und

ihre innere Struktur zu durchleuchten (Hubwieser 2007a, S. 47). Fehler und Irrtümer bei der Anwendung eines Systems sind nach Dutke Ereignisse, die Schülern die Grenzen ihres Wissens bzw. ihres mentalen Modells aufzeigen. Dutke beschreibt und evaluiert die Wirksamkeit explorierenden Lernens in der Mensch-Computer-Interaktion als Mittel zum Aufbau mentaler Modelle:

> „Dabei wird kaum beachtet, daß Bedienungsfehler technischer Systeme zunächst nur Regelverstöße darstellen, Verstöße gegen artifizielle, von Menschen mehr oder weniger willkürlich konstruierten Regeln" (Dutke 1994, S. 154).

Die Vorwegnahme von Fehlern und unerwartetem Verhalten im Lehr-Lernprozess ‚immunisiert' gegen Stress in Fehlersituationen (Dutke 1994, S. 155). Die systematische Erkundung bedarf jedoch des Vorwissens über das System zur geeigneten Wahl eines Ziels (Dutke 1994, S. 156). Darüber hinaus erleichtert das Beherrschen einfacher Funktionen das Erlernen komplexerer Funktionen, wenn das System konsistent ist (Dutke 1994, S. 159). Da bei Exploration dieser Lernweg nicht vorausgesetzt werden kann, bietet sich eine Vorgabe der Schrittfolge bei ersten Erkundungen an. Fehlfunktionen sind bei der nachfolgenden systematischen Erkundung handlungsleitend. Ausgangspunkt ist der Erfahrungsbereich und der Alltag der Schüler.

Im Folgenden werden zwei Vorgehensweisen zur systematischen Erkundung von Informatiksystemen entwickelt. In Anlehnung an den Black-Box-Test bezieht sich die erste Vorgehensweise vornehmlich auf das nach außen sichtbare Verhalten des Informatiksystems. Die zweite Vorgehensweise dient der Analyse der Systemkomponenten und geht davon aus, dass beispielsweise der Quelltext eines Programms vorliegt.

In Anlehnung an die von Anderson und Krathwohl überarbeitete Bloom'sche Taxonomie (Anderson und Krathwohl 2001) wurde eine aus acht Schritten bestehende Experimentierabfolge entwickelt, die alle sechs kognitiven Prozesse anspricht: Erinnern, Verstehen, Anwenden, Analysieren, Bewerten und (Er)schaffen (vgl. (Schobel und Holdt 2004), (Fuller et al. 2007); Abschnitt 2.1.5). Eine Vertiefung der Analyse der informatischen Konzepte und deren Sonderfälle führt zu der Erweiterung auf acht Schritte. Die Schüler sollen das nach außen sichtbare Verhalten eines Informatiksystems systematisch erkunden und alle Zwischenergebnisse notieren. Bei der nachfolgenden Vorgehensweise sind in Klammern der entsprechende kognitive Prozess und die Wissensdimension nach Anderson und Krathwohl (2001) angegeben:

1. Wie lautet der Name des Systems? (Erinnern; Faktenwissen)
2. Beschreiben Sie die Benutzungsoberfläche. (Verstehen; Faktenwissen)
3. Welche Funktionalitäten vermuten Sie unter Berücksichtigung der vorherigen Beobachtungen? (Verstehen; Begriffliches Wissen)
4. Welche Beziehungen bestehen zwischen den Elementen? (Verstehen; Faktenwissen)

5. Welche informatischen Konzepte wurden eingesetzt, um die Funktionalität zu erreichen? (Analysieren; Begriffliches Wissen)
6. Analysieren Sie das Programm, indem Sie Sonderfälle ermitteln und mit ihnen experimentieren. (Analysieren und Anwenden; Begriffliches Wissen)
7. Werten Sie Fehler und unerwartetes Verhalten aus – auch mit Blick auf die Sonderfälle der Konzepte. (Bewerten; Begriffliches Wissen)
8. Hypothetischer Einsatz des Systems – Wie würde sich das Programm in anderen (realen, komplexeren) Situationen verhalten? ((Er)schaffen; Faktenwissen)

Die Schrittfolge ergibt damit ein Problemlöseschema ((Kopp und Mandl 2006, S. 131f); vgl. S. 191). Durch Schritt 3 erfolgt ein Bezug zu den Hauptfunktionen von Informatiksystemen nach Denning (2007), indem die Funktionalität des Systems beobachtet wird. Der Bezug zu informatischen Konzepten (Schritt 5) und deren Sonderfälle (Schritt 6) ermöglicht die Klassifikation des Systemverhaltens anhand typischer Fälle und Sonderfälle. Gleichzeitig bilden sie den Anknüpfungspunkt zu vernetzten fundamentalen Ideen der Informatik und zu den Wissensrepräsentationen (Abschnitt 5.4). Anhand der Hauptfunktionen von Informatiksystemen ist außerdem eine Diskussion über Informatiksysteme und Gesellschaft strukturierbar (Schritt 8).

Damit bezieht die Aufgabe sowohl die problembezogene Ebene (den Zweck des Systems) und die modellbezogene Ebene mit ein (Modellbildung). Auch wird eine (nicht formale) Verifikation vorgenommen (vgl. Abschnitt 4.2.1). Bei der Untersuchung der Informatiksysteme auf Fehler sind Sonderfälle der Konzepte und unübliche Eingaben ein Ansatzpunkt, um das Systemverhalten zu klassifizieren. Zum Beispiel können zu große Zahlen eingegeben und Datenstrukturen stark gefüllt oder überprüft werden, ob eine Operation auf einer leeren Liste ebenso funktioniert wie auf einer nicht-leeren Liste. Diese Vorgehensweise eignet sich für Partner- und Teamarbeit, denn soll für ein Systemverhalten eine Erklärung gefunden werden, kann die Lerngruppe vorab ein Brainstorming durchführen. So können in der Gruppe Beispiele diskutiert werden, die ggf. durch den Lehrer vorgegeben werden. Dies können systematische Fehler sein, d. h. Fehler, die für eine Klasse von Eingaben immer zu erwarten sind. Beispiele für Fehler von Informatiksystemen aus Tageszeitungen können für die Auswirkungen sensibilisieren (Abschnitt 6.4).

Dafür ist es wichtig, Elemente des Systems, insbesondere Objekte, zu identifizieren und zu benennen. Diese ersten Objektkandidaten lassen Rückschlüsse auf Klassen zu. Analog zum objektorientierten Entwurf können statt Klassendiagrammen auch CRC-Karten (Abschnitt 4.2.3) eingesetzt werden, die statt Verantwortungen von und Beziehungen zwischen Klassen nun Systemkomponenten und deren Verantwortungen und Beziehungen untereinander beschreiben. Der letzte Schritt der

systematischen Vorgehensweise (Schritt 8) spricht den Aspekt der Blindheit eines Systems an, also speziell die Grenzen des Systems und Aufgaben, für die es nicht vorgesehen ist (vgl. Hubwieser 2007a, S. 47).

Bei der Betrachtung der Vorgehensweise fällt auf, dass diese Art der systematischen Erkundung insbesondere dem nach außen sichtbaren Verhalten zuzuordnen ist, aber durch Hypothesenprüfung hinsichtlich zugrunde liegender fundamentaler Ideen auch ein Bezug zur inneren Struktur des Informatiksystems besteht.

5.5.4 Vorgehensweise zur systematischen Erkundung der inneren Struktur eines Informatiksystems

In der Fachwissenschaft Informatik wird der Begriff des White-Box-Tests (auch: Glass-Box-Test) genutzt, um einzelne Komponenten eines Systems bei offen liegendem Quelltext des Systems zu testen. Analog dazu wird im Folgenden eine Vorgehensweise zur Analyse des Systems bei vorliegender Beschreibung der inneren Struktur für die Schulinformatik angegeben. Eine Abbildung der Vorgehensweisen für einen White-Box-Test auf die Schulinformatik ist damit nicht intendiert. Die Schüler sollen die innere Struktur eines Informatiksystems anhand des Quelltextes systematisch erkunden und alle Zwischenergebnisse notieren. Bei der nachfolgenden Vorgehensweise sind in Klammern der entsprechende kognitive Prozess und die Wissensdimension nach Anderson und Krathwohl (2001) angegeben:

1. Wie lautet der Name des Projektes und der beteiligten Dateien? (Erinnern; Faktenwissen)
2. Identifizieren Sie beteiligte und überflüssige Klassen durch Beziehungen im Quellcode. (Verstehen; Faktenwissen)
3. Welche Operationen und Attribute gibt es? Nutzen Sie zur Analyse Kommentare im Quelltext. (Verstehen; Faktenwissen)
4. Erstellen Sie ein Klassendiagramm, dass Ihre Ergebnisse beinhaltet. (Anwenden; Begriffliches Wissen)
5. Welche informatischen Konzepte finden Sie in diesem Informatiksystem? Analysieren Sie auch die Abläufe. (Analysieren; Begriffliches Wissen)
6. Wie werden Sonderfälle behandelt? (Analysieren; Begriffliches Wissen)
7. Wo liegen Fehlerursachen? (Bewerten; Begriffliches Wissen)
8. Bewertung des Systems: Sind einfache und elegante Strukturen genutzt worden, die für andere Programme wieder genutzt werden können? (Bewerten; Begriffliches Wissen)
9. Modifikation des Systems ((Er)schaffen; Begriffliches Wissen / Verfahrensorientiertes Wissen)

Analog kann eine Vorgehensweise zur Analyse der Systemkomponenten anhand vorgegebener Diagramme statt Quelltextanalyse angegeben werden. Die ersten Schritte entsprechen dem Reverse Engineering. Dabei werden gestaltende Ideen und abstrakte Konzepte, speziell Strukturen, Nebenläufigkeit sowie Datentypen analysiert und in einer (Re-) Dokumentation zusammengefasst (Claus und Schwill 2006, S. 582), ohne auf Implementierungsdetails einzugehen:

> „Reverse engineering generally involves extracting design artefacts and building or synthesizing abstractions that are less implementation-dependent" (Chikofsky und Cross 1990, S. 15).

Ziel beim Reverse Engineering in der Fachwissenschaft ist, in einem ersten Schritt

> „die Bedeutung der existierenden Software [zu] ermitteln (‚*Software-Verstehen*'), was nicht entscheidbar ist und daher generell nicht automatisiert werden kann" (Claus und Schwill 2006, S. 570; Hervorh. im Original).

Die Modifikation des Informatiksystems fördert das Verstehen durch Variantenbildung, bei der das Bekannte variiert und auf ähnlichem Weg wie zuvor gelöst wird. Die theoretisch, anhand des Experimentiervorgangs (Meyer 2006) und kognitiver Stufen (Anderson und Krathwohl 2001) begründeten Vorgehensweisen sind im unterrichtlichen Geschehen auf ihre Tragfähigkeit zu prüfen. Ziel ist, dass die Erkenntnisse aus einem Experiment handlungsleitend für die Anwendung von Informatiksystemen sind.

Als methodische Variation können unterschiedliche Informatiksysteme in Form eines Unterrichtszirkels im Unterricht von den Schülern erkundet werden. Dazu werden an mehreren Stationen Informatiksysteme oder Modelle von Informatiksystemen platziert, die in vorgegebener Zeit von den Schülern nach den bekannten Vorgehensweisen erkundet werden müssen. Bei dieser Unterrichtsmethode spricht man vom Unterrichts- bzw. Lernzirkel oder auch vom Stationenlernen. Dabei können je nach Aufgabenstellung wiederum unterschiedliche Repräsentationsformen von Informatiksystemen vorliegen, sei es als UML-Diagramm, Animation, simuliertes System oder aktives Informatiksystem. Insbesondere zum Abschluss einer Unterrichtsreihe lassen sich Kompetenzen an bisher nicht untersuchten Informatiksystemen wie Robotern oder Textverarbeitungssystemen überprüfen. Die Schüler arbeiten weitgehend selbsttätig. Nachteil ist, dass viele Materialien erstellt werden müssen und ggf. viele Rechner zum Einsatz kommen. Möglichkeiten der Selbstkontrolle sind für Schüler in dieser Methode besonders wichtig. Der Unterrichtszirkel kann in Pflicht- und Wahlstationen aufgeteilt sein, um Binnendifferenzierung vorzunehmen.

5.5.5 Zusammenfassung und Einordnung der systematischen Erkundung in das Unterrichtsmodell

Kritische Betrachtung der systemanalytischen Vorgehensweise

Der systemanalytische Top-down-Zugang zu Informatiksystemen wurde gewählt, da er den Blick auf das System und den Blick in das System verbindet. Implemen-

tierungsaspekte können weitgehend ausgeblendet werden. Bei der Dekonstruktion von Informatiksystemen wurde jedoch kritisch angemerkt, dass aus dem Produkt nicht mehr eindeutig Rückschlüsse auf Entwurfsentscheidungen gezogen werden können. Es lässt viele Alternativlösungen zu (Abschnitt 4.2.3). Gleiches gilt – im Prinzip – für die vorgestellten Vorgehensweisen der systematischen Erkundung. Ihr Vorteil ist in der Einbindung in das Unterrichtsmodell zu sehen. Ziel ist es, über das Experimentieren erst einzelne, dann vernetzte fundamentale Ideen der Informatik in einem Informatiksystem zu entdecken. Dadurch, dass ein oder mehrere ausgewählte Entwurfsmuster Grundlage einer entsprechend gestalteten lernförderlichen Software sind (Abschnitt 5.6), besteht das Potential, auf die fundamentalen Ideen zu fokussieren. Indem zuerst das nach außen sichtbare Verhalten des Systems als Black-box erkundet wird und anschließend der Quelltext bzw. die Klassenstruktur zur Analyse der inneren Struktur hinzu kommt, kann eine Korrektur der kognitiven Modelle der Schüler erfolgen. Sie führt implizit oder explizit zu den Entwurfsmustern bzw. fundamentalen Ideen.

Eine nicht zu vernachlässigende Schwierigkeit liegt in der Schülermotivation, das Experiment durchzuführen. Die Schüler müssen erfahren, dass das systematische Vorgehen ihnen einen Vorteil bietet gegenüber einer Versuch-Irrtum-Strategie. Um dies zu erreichen, wurden in den Unterrichtserprobungen (Kapitel 6 und 8) zum einen unterschiedliche Varianten eines Programms eingesetzt, deren Verhalten sich in besonderen Situationen unterschied. Als Beispiel ist das Programm einer Arztpraxis mit dem Iteratormuster zu nennen, in dem Notfallpatienten in den Varianten unterschiedlich in die Patientenliste eingefügt wurden (Kapitel 6). Die Variantenbildung war durch die gleich bleibende innere Klassenstruktur des Programms für die Lehrperson ohne großen Aufwand zu erstellen. Zum anderen ist die Problemgröße relevant. So konnte ein Programm um ein neues Entwurfsmuster erweitert werden. In der systematischen Erkundung wurde bekanntes Verhalten aus der vorherigen Version, dies war Zugriffskontrolle mit dem Proxymuster, identifiziert und von dem neuen Verhalten abgegrenzt. Letzteres war durch die Identifikation neuer Zustände möglich, die durch das Zustandsmuster hinzugefügt wurden (Kapitel 8). Da Unterrichtsexperimente sowohl Aspekte von Instruktion und Konstruktion kombinieren, bieten sie die Möglichkeit, Informatiksysteme und fundamentale Ideen nachhaltig zu verstehen.

Die konkrete Vorgabe einer Schrittfolge zur Erkundung von Informatiksystemen berücksichtigt das zur Kompetenzentwicklung notwendige individuelle Lern-und Leistungsvermögen von Schülern. Insbesondere können auch lernschwächere Schüler Erfolge erzielen. Durch die Beobachtung unerwarteten Verhaltens bleiben den Schülern Freiheiten, eigene Lösungswege zu gehen, die aber durch die Orientierung am Unterrichtsexperiment nicht beliebig sind. Die Schritte erlauben es darüber hinaus, Zwischenlösungen als Teillösungen anzuerkennen. Nachdem die Schüler erste Erfahrungen mit der systematischen Erkundungen gemacht haben, kann das Unter-

richtsexperiment als offenere Aufgabe gestellt werden, indem die Kleinschrittigkeit verringert wird. Offene Aufgaben verlangen von den Schülern eine höhere Flexibilität beim Lösen der Aufgaben.

Im Folgenden wird daher das Beispiel Zugriffskontrolle wieder aufgegriffen, um die Verknüpfung der systematischen Analyse mit den Wissensrepräsentationen für vernetzte fundamentale Ideen zu illustrieren.

Systematische Erkundung eines Programms am Beispiel Zugriffskontrolle: Rolle der Wissensrepräsentationen

Die vorgestellten Vorgehensweisen zur systematischen Erkundung des nach außen sichtbaren Verhaltens und der inneren Struktur von Informatiksystemen sind durch theoretische Überlegungen zu Unterrichtsexperimenten und zu kognitiven Prozessen begründet. Ihr Potential für Kompetenzentwicklung mit Informatiksystemen liegt jedoch in der Kombination mit den in Abschnitt 5.4 vorgestellten Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik. Ausgewählte Entwurfsmuster fördern die Kompetenzentwicklung mit Informatiksystemen. Diese Entwurfsmuster können genutzt werden, um lernförderliche Software zu gestalten, die dann anhand der beschriebenen Vorgehensweisen systematisch erkundet werden können (Abschnitt 5.6). Für Kompetenzentwicklung mit Informatiksystemen ist es wichtig, dass ihr Verhalten klassifiziert wird (Abschnitt 5.5.1), beispielsweise durch Äquivalenzklassen hinsichtlich Ein- und Ausgabe. Zur Beschreibung der Äquivalenzklassen sind insbesondere Sonderfälle zu betrachten, die eigene Klassen bilden. Eine Klassifikation des Verhaltens ist dann besonders einfach, wenn fundamentale Ideen identifizierbar sind. Beispielsweise lassen sich bei Rekursion bzw. Baumstrukturen (Kompositummuster), Listenstrukturen (Iteratormuster) und Zugriffskontrolle (Proxymuster) anhand der Datenstrukturen oder der Unterscheidung von Zugriffsrechten (Äquivalenz-) Klassen des Verhaltens angeben. Darüber hinaus bieten Entwurfsmuster besonders elegante Beschreibungen der inneren Struktur, die ein vernetztes Denken erfordern und lernförderlich sind.

Für das Beispiel zur Zugriffskontrolle (Abschnitt 5.4) bedeutet dies – für den Fall, dass das Proxymuster als Wissensrepräsentation die strukturelle Grundlage eines untersuchten Programms bildet –, dass die fundamentalen Ideen per Definition in der Wissensrepräsentation und damit in der Software vorhanden sind. Sie unterstützen Schüler und Lehrperson, unterschiedliche Fälle im Systemverhalten zu klassifizieren. So können beim Proxymuster unterschiedliche Zugriffsrechte identifiziert werden. Aufgrund des Lebensweltbezugs der Entwurfsmuster (→ Kriterium 6: Lebensweltbezug; S. 173) ist es für Schüler über Analogien leichter möglich, Hypothesen zu bilden, z. B. zu unterschiedlichen Zugriffsrechten, die überprüft werden. Darauf aufbauend sind Hypothesen zur inneren Struktur zu prüfen. Zusammenhänge mit anderen Entwurfsmustern (→ Kriterium 4: Zusammenhänge mit anderen

Strukturmodellen; S. 173) zeigen Anknüpfungspunkte zur Erweiterung der lernförderlichen Software, so dass Ergebnisse von Experimenten vor und nach der Erweiterung durch Schüler verglichen werden können. Zugriffskontrolle bietet hinsichtlich der aktuellen politischen Debatte um Datenschutzrechte ein großes Potenzial, Auswirkungen in der Gesellschaft zu diskutieren. Eine ausführlichere Diskussion der systematischen Erkundung wird in Abschnitt 5.6.2 zur Lernsoftware vorgenommen.

Im Folgenden werden deshalb Gestaltungshinweise für Lernsoftware basierend auf Wissensrepräsentationen angegeben (Abschnitt 5.6), die Schüler anhand der vorgestellten Vorgehensweisen systematisch erkunden können. Im Laufe des Forschungsprojekts wurde die systematische Erkundung von Informatiksystemen in zwei Unterrichtserprobungen (Kapitel 6 und 8) sowie mittels Laut-Denken in einer Laborstudie evaluiert und jeweils anschließend weiterentwickelt (Kapitel 7).

5.6 Lernförderliche Software für Kompetenzentwicklung mit Informatiksystemen

5.6.1 Anforderungen an lernförderliche Software für Kompetenzentwicklung mit Informatiksystemen

Ziel dieses Abschnitts ist es, Lernsoftware zu beschreiben, die vernetzte fundamentale Ideen in Wissensrepräsentationen und Schüleraktivitäten zur systematischen Erkundung sinnvoll miteinander verknüpft.

Das Ziel „Kompetenzentwicklung mit Informatiksystemen" erfordert lernförderlich gestaltete Software. Das beinhaltet, dass die Charakteristika von Informatiksystemen – nach außen sichtbares Verhalten, innere Struktur und Implementierungsaspekte – in einem Programm zusammenhängend, beispielsweise durch Sichtenwechsel, und konsistent repräsentiert werden. Damit wird die didaktische Doppelfunktion von Informatiksystemen als Bildungsmedium und Bildungsgegenstand genutzt.

Vorteil eines im Unterricht eingesetzten, konkreten Informatiksystems ist die Situiertheit (vgl. Hubwieser 2007a, S. 10). Die Schüler gehen unmittelbar mit dem System als Gegenstand der Aufgabe um und sind mit dem konkreten Fall befasst. Außerdem gelingt soziale Einbettung, da überwiegend in Teams gearbeitet wird. Gerade die Anschlussmöglichkeiten durch die Anwendung von Modellierung, aber auch Programmierung im Sinne von Modifikation und systematischem Testen unter Einbezug des problemorientierten Lernens sind lernförderlich. Systematische Erkundung und systematisches Testen der Informatiksysteme lassen sich oft so gestalten, dass die Lernenden ihr Arbeitstempo selbst bestimmen (vgl. Wiesner und Brinda 2007). Ein ansprechender Kontext des Systems ist dafür unabdingbar.

Folgende Arten Lernsoftware werden für das Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung eingesetzt:

1. Informatiksysteme, deren innere Struktur durch Entwurfsmuster als Wissensrepräsentation für vernetzte fundamentale Ideen bestimmt ist. In Abschnitt 5.6.2 wird auf diese Art Lernsoftware weiter eingegangen.
2. Lernsoftware, die fundamentale Ideen der Informatik unter Nutzung von Entwurfsmustern als Wissensrepräsentation thematisiert. Ein geeigneter Lebensweltkontext hinsichtlich der Basiskompetenzen zu Informatiksystemen ist in der Lernsoftware darzustellen. Beispiel ist die Lernsoftware Pattern Park (Abschnitt 5.6.3).

Darüber hinaus ist Lernsoftware zu Verhalten und innerer Struktur von Informatiksystemen denkbar, die die beiden oben genannten Varianten kombiniert und in der sowohl eine Produktsicht auf ein reguläres Informatiksystem als auch eine Parametersicht mit Zugriff auf Parameter von Entwurfsmustern existieren. Parameteränderungen müssen dann zur Förderung der Basiskompetenzen in geändertem Systemverhalten resultieren und weiteres Experimentieren erlauben. Weyer (2007b) erarbeitete im Rahmen einer Diplomarbeit einen Entwurf solcher Lernsoftware (Abschnitt 7.3.2).

Grundlage der Entwicklung von Lernsoftware sind Überlegungen zu kognitiven Barrieren und Fehlvorstellungen hinsichtlich des Lerngegenstands, die mit Hilfe der Lernsoftware abgebaut bzw. korrigiert werden sollen. Für beide oben genannten Arten Software wird exemplarisch Zugriffskontrolle mit dem Proxymuster betrachtet, da es sowohl in der Lernsoftware Pattern Park umgesetzt wurde als auch Grundlage kleiner Programme ist, die im unterrichtlichen Geschehen eingesetzt wurden (Kapitel 6 und 8). Kognitive Barrieren und Fehlvorstellungen zur Zugriffskontrolle finden sich in Abschnitt 5.4.4.

5.6.2 Entwurfsmuster als strukturelle Grundlage lernförderlicher Software

Verknüpfung von vernetzten fundamentalen Ideen in Entwurfsmustern mit der systematischen Erkundung eines Programms

Es wurden Kriterien angegeben, nach denen Entwurfsmuster ausgewählt werden können, um den Bildungsprozess zu Informatiksystemen positiv zu beeinflussen (Abschnitt 5.4). Ziel ist, dass die positiven Eigenschaften der fachdidaktisch ausgewählten Entwurfsmuster dadurch für den Unterricht nutzbar gemacht werden, dass die Muster die innere Struktur von Software bilden.

Einfach zu erstellen sind kleine Programme, in denen ein Strukturmodell explizit und unverfälscht gestaltgebend ist. Der Funktionsumfang solcher Systeme kann erkundet und mit deren innerer Struktur in Verbindung gesetzt werden. Neben der direkten Beeinflussung des Systemverhaltens, die sich in der Benutzungsoberfläche

widerspiegelt, ist durch die modulare Entwurfsmusterstruktur die leichte Zugänglichkeit zu Programmteilen für deren Modifikation gegeben (vgl. Lehmann 1993, S. 135). Dies ist z. B. beim Proxymuster durch Unterscheidung der Klassen „Proxy" und „RealesSubjekt" gegeben (Abbildungen 5.7 und 5.8).

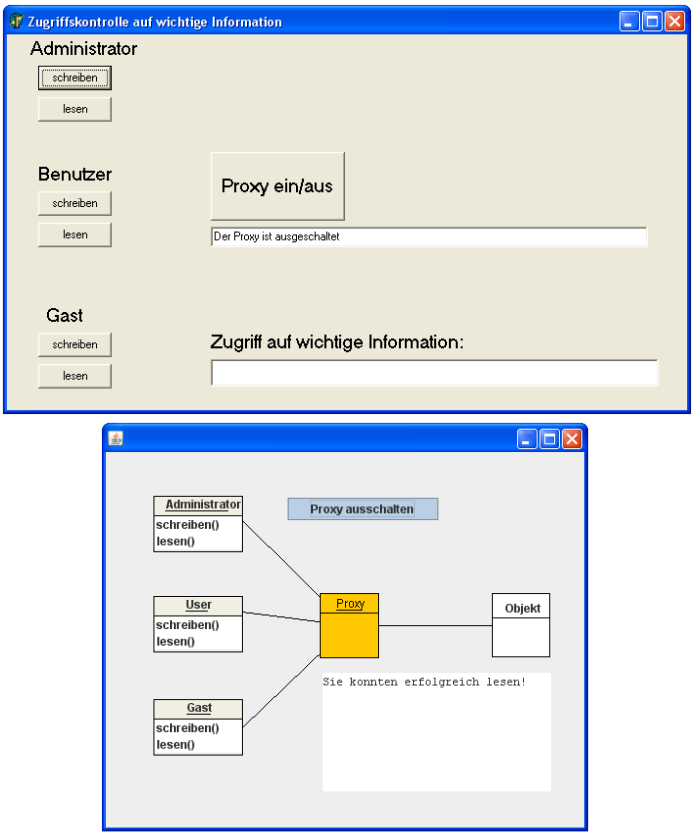


Abbildung 5.7: Umsetzung des Programms zur Zugriffskontrolle in Delphi und Java; Information bzw. Objekt korrespondieren zur Klasse „RealesSubjekt" aus Abbildung 5.8, die Rollen Administrator, Benutzer und Gast sind Klienten

Magenheim stellt für den Ansatz der Dekonstruktion von Informatiksystemen folgende Anforderungen an Lernsoftware:

1. „Hinreichende Komplexität,
2. Quellcode gut strukturiert und dokumentiert,

3. zugrunde liegende Entwurfs- und Designkonzepte widerspiegeln wesentliche Konzepte des Softwareengineerings (z. B. Objektorientierung, Ereignisorientierung, model-view-control Konzept ...),
4. Benutzungsoberflächengestaltung entspricht den zentralen Anforderungen der Softwareergonomie,
5. zentrale Ideen und Methoden über Software zugänglich,
6. Nutzungskontext der Software für Schüler erschließbar, exemplarische 'didaktische Fenster' auf tieferliegenden Schichten des Informatiksystems möglich (Symbolverarbeitung),
7. mediale Funktionen des Softwaresystems beispielhaft analysierbar,
8. ergänzende Dokumentationen über Modellierungsprozess verfügbar,
9. Visualisierung von impliziten Konzepten (z. B. Klassenhierarchien, Sequenzdiagramme etc.) mit geeigneten Entwicklungsumgebungen möglich etc." (Magenheim (2001); zitiert nach (Engbring 2004, S. 183).

Bei Ausblendung des Softwareentwicklungsprozesses und Fokussierung auf Basiskompetenzen zu Informatiksystemen wird klar, dass einige der Anforderungen eine weniger wichtige Rolle spielen. So sind hinreichende Komplexität (1) und das Vorhandensein einer ergänzenden Dokumentation über den Modellierungsprozess (8) weniger relevant, da Softwareentwicklung nicht wie im Ansatz der Dekonstruktion ein Ziel des Unterrichts ist. Es ist die Balance zwischen angemessener Komplexität der Software für die Lernenden und Angemessenheit des Einsatzes eines Entwurfsmusters zur Lösung des Anwendungsproblems zu gewährleisten.

Andere Anforderungen müssen umformuliert werden: Die zugrunde liegenden Entwurfs- und Designkonzepte müssen vernetze fundamentale Ideen der Informatik widerspiegeln (3 und 5). Außerdem müssen Visualisierungen von impliziten Konzepten in geeigneten Lernmaterialien möglich sein (9).

Beispiel: Lernförderliche Software zur Zugriffskontrolle mit dem Proxymuster

Nach außen sichtbares Verhalten, innere Struktur und Implementierungsaspekte werden als Perspektiven genutzt (Stechert 2006c), um die Informatiksysteme, z. B. in Form lernförderlicher Software, systematisch und handlungsorientiert zu erkunden. Im Folgenden soll am Beispiel der Zugriffskontrolle die Rolle der lernförderlichen Software im Unterrichtsmodell diskutiert werden. Nach der positiven Klassifikation und Auswahl des Proxymusters (Abschnitt 5.4) wurde anschließend lernförderliche Software entwickelt, in der das Muster als Brücke zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur dient.

Die lernförderliche Software zur Zugriffskontrolle hat diesbezüglich exemplarischen Charakter, da sie auf dem Proxymuster basiert, und die Dualität von Verhalten und Struktur eine systematische Erkundung unterstützt (Abbildung 5.8). Somit kann das Programm sowohl auf sein Verhalten als auch auf seine bewährte innere Struktur überprüft werden. Nach Brinda kann Lernsoftware, wie Explorationsmodule, auch zum Unterrichtsinhalt werden, denn Lernende können die Umsetzung

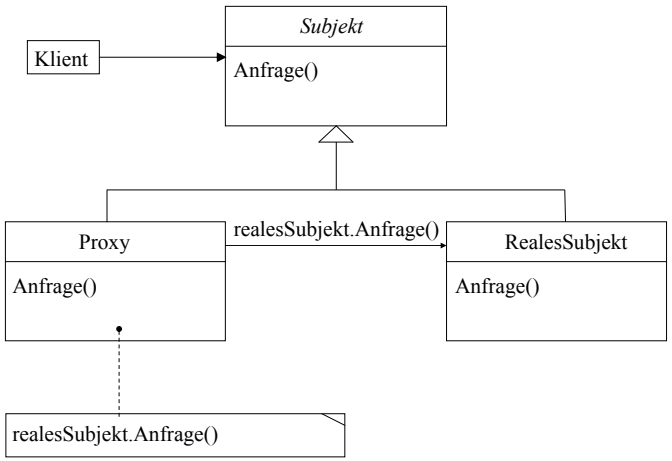


Abbildung 5.8: Das Klassendiagramm des Programms zur Zugriffskontrolle – ohne Benutzungsoberfläche – entspricht dem Proxymuster

> „informatischer Fachkonzepte (z. B. Entwurfsmuster, vgl. Gamma et al. 1996) in einem Explorationsmodul analysieren und selbst daraus resultierende, kleine Erweiterungen des Systems vornehmen" (Brinda 2004a, S. 153).

Eine Bereicherung des Lehr-Lernprozesses im Rahmen des Unterrichtsmodells wird durch die Kombination von Wissensrepräsentationen (gemäß der Kriterien zur Klassifikation der Entwurfsmuster) mit der Vorgehensweise zur systematischen Erkundung erzielt (Abschnitt 5.5.3). Für die systematische Erkundung ist das Lernen aus Fehlern, z. B. ausgelöst durch unerwartetes Verhalten von Informatiksystemen, ein Ansatzpunkt. Durch Schritt 3 (vgl. Abschnitt 5.5.3; S. 214) der Vorgehensweise erfolgt ein Bezug zu den Hauptfunktionen von Informatiksystemen nach Denning (2007). Der Bezug zu informatischen Konzepten (Schritt 5; vgl. Abschnitt 5.5.3; S. 214) und deren Sonderfälle (Schritt 6; vgl. Abschnitt 5.5.3; S. 214) ermöglicht die Klassifikation des Systemverhaltens. Gleichzeitig bilden sie den Anknüpfungspunkt zu vernetzten fundamentalen Ideen der Informatik (→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik; S. 172). Da diese per Definition in der Wissensrepräsentation und damit in der Software vorhanden sind, unterstützen sie Schüler und Lehrperson, unterschiedliche Fälle im Systemverhalten zu klassifizieren. So können beim Proxymuster unterschiedliche Zugriffsrechte identifiziert werden. In den Abschnitten 6.4 und 8.3 wird dies aufgegriffen, z. B. die Vernetzung der Zugriffskontrolle mit den fundamentalen Ideen Vererbung und Schnittstelle in der lernförderlichen Software, um Problemstellen im Unterricht zu bewältigen. Außerdem wird in Abschnitt 8.3 eine Verknüpfung des Proxymusters mit dem Zu-

standsmuster in das unterrichtliche Geschehen integriert, um Systemzustände zu betrachten.

Aufgrund des Lebensweltbezugs der Entwurfsmuster (→ Kriterium 6: Lebensweltbezug; S. 173) ist es für Schüler über Analogien leichter möglich, Hypothesen zu bilden, z. B. zu unterschiedlichen Zugriffsrechten, die überprüft werden. Darauf aufbauend sind Hypothesen zur inneren Struktur zu prüfen. Zusammenhänge mit anderen Entwurfsmustern (→ Kriterium 4: Zusammenhänge mit anderen Strukturmodellen; S. 173) zeigen Anknüpfungspunkte zur Erweiterung der lernförderlichen Software, so dass Ergebnisse von Experimenten vor und nach der Erweiterung durch Schüler verglichen werden können. Erweiterungen, d. h. sinnvolle Kombinierbarkeit eines zugrunde liegenden Entwurfsmusters als Wissensrepräsentation mit anderen einzelnen Entwurfsmustern sollte möglich sein. Denn um den Nachteil auszugleichen, dass für jedes Entwurfsmuster eine neue Software eingesetzt wird, sollten die bekannten Strukturen in späteren Phasen wieder erkennbar sein, um den Wissenstransfer zu fördern: Eine strukturell unveränderte Software mit einem anderen Kontext kann im Unterricht ohne großen Aufwand für die Lehrperson eingesetzt werden (Kapitel 8). Die Vernetzung fundamentaler Ideen kann einerseits durch den Einsatz solcher Variationen der im Kern unveränderten Software mit anderer Schwerpunktsetzung im Unterricht erfolgen. Andererseits ist es möglich, durch Kombination mit weiteren Entwurfsmustern eine für Kompetenzentwicklung mit Informatiksystemen förderliche Software zu gestalten. Bei einer Erkundung des Verhaltens des erweiterten Systems können die Schüler auf vorherige Erkenntnisse aufbauen. Anhand der Hauptfunktionen von Informatiksystemen ist außerdem eine Diskussion über Informatiksysteme und Gesellschaft strukturierbar (Schritt 8; vgl. Abschnitt 5.5.3; S. 214).

Beispiele für einfache Programme, die auf Entwurfsmustern als strukturelle Grundlage basieren, sind die in den Unterrichtserprobungen (Kapitel 6 und 8) eingesetzten Programme mit dem Lebensweltbezug der Arztpraxis sowie Programme zu Architekturmustern (Abschnitt 7.3.1; (vgl. Ufer 2007)). Dazu kommen Programme aus den Seminararbeiten von Sülz (2007) und Graf (2008).

Die Unterrichtsprojekte zeigen, dass Schüler einen experimentierenden Zugang über die lernförderliche Software annehmen und die darin enthaltene Vernetzung fundamentaler Ideen der Informatik entdecken (Kapitel 6 und 8).

Kompetenzentwicklung mit Informatiksystemen im Großen und im Kleinen

Der Einsatz von einem oder wenigen Entwurfsmustern als strukturelle Grundlage eines Informatiksystems führt zu einem begrenzten Funktionsumfang des Systems. DeRemer und Kron (1975) postulieren in ihrem wegweisenden Artikel „Programming-in-the-Large Versus Programming-in-the-Small", dass sich das Programmieren im Kleinen grundlegend vom Programmieren im Großen unterscheidet:

> „We argue that structuring a large collection of modules to form a 'system' is an essentially distinct and different intellectual activity from that of constructing the individual modules. That is, we distinguish programming-in-the-large from programming-in-the-small. Correspondingly, we believe that essentially distinct and different languages should be used for the two activities" (DeRemer und Kron 1975, S. 114).

Als „große" Programme definieren sie Systeme, die aus vielen kleinen Programmen bzw. Modulen bestehen, die wiederum von unterschiedlichen Personen geschrieben sein können (DeRemer und Kron 1975, S. 114). Aufgrund der allgemeinen Zustimmung, dass es diesen prinzipiellen Unterschied gibt (z. B. Appelrath et al. 2002), ist nun zu fragen, ob sich daraus Konsequenzen für die vorliegende Arbeit ergeben – d. h., ob eine Unterscheidung hinsichtlich der Kompetenzentwicklung mit großen Informatiksystemen wie Standardsoftware, Betriebssystemen und Datenbanksystemen einerseits und der Kompetenzentwicklung mit kleinen Informatiksystemen andererseits notwendig ist, die beispielsweise einen Algorithmus zur Berechnung des Maximums umsetzen. Die Analyse des fachdidaktischen Forschungsstandes stützt die Annahme, dass Informatiksysteme die während der Konstruktion eingesetzten Konzepte und Methoden widerspiegeln. Deshalb wird im Folgenden zusammengefasst, welche Aspekte das Programmieren im Großen ausmachen und inwieweit sie Einfluss auf Kompetenzentwicklung zu Informatiksystemen haben. Es ist zu prüfen, ob Kompetenzentwicklung mit großen Informatiksystemen anderer Strategien und Informatikkonzepte bedarf als mit kleinen Informatiksystemen.

Menschen werden im Alltag mit Informatiksystemen konfrontiert, die nach der Definition von DeRemer und Kron (1975) groß sind. Für Kompetenzentwicklung mit Informatiksystemen fehlen jedoch theoretisch begründete und empirisch überprüfte Kriterien zur Beurteilung und Stufung der Größe eines Informatiksystems. Frage ist, wie sich Programme, die auf Entwurfsmustern basieren, zwischen kleiner Software, die einfache Algorithmen umsetzt, und komplexer Software einordnen lassen.

Welche Informatikkonzepte und -methoden kommen in großen Informatiksystemen zum Einsatz? Da Programmiersprachen und -werkzeuge für die Entwicklung kleiner Programme bzw. Module existieren, stellen DeRemer und Kron (1975) fest, dass es Sprachen und Werkzeugen zur Vernetzung der Module bedarf. Daher skizzieren sie Anforderungen an eine Beschreibungssprache. Damit ist bereit offensichtlich, dass DeRemer und Kron die Beziehungen zwischen den Systemmodulen als den entscheidenden Unterschied zwischen kleinen und großen Programmen ansehen (module interconnctivity information), der Datenabstraktion und Systemtests notwendig macht. Als Methoden zur Sicherung der Zuverlässigkeit großer Programme betonen sie das Geheimnisprinzip und Schichten virtueller Maschinen.

Appelrath et al. (2002) widmen in ihrem Standardwerk zur Einführung in die Informatik das Kapitel zur Softwareentwicklung dem Programmieren im Großen. In der Gliederung des Kapitels kommt der objektorientierten Modellierung und der Qualität von Software eine Schlüsselrolle zu. Genannt werden: Korrektheit, Effizienz,

Robustheit, Benutzungsfreundlichkeit, Wartbarkeit, Bewertung des Programms gegenüber den Anforderungen und gute Verständlichkeit für jedermann (Appelrath et al. 2002, S. 106).

Das Prinzip der Zerlegung ist sowohl für das Programmieren im Kleinen als auch im Großen fundamental:

> „Während die Zerlegung beim Programmieren im Kleinen an den einzelnen Funktionalitäten des Programms ausgerichtet ist, orientiert sie sich beim Programmieren im Großen an den Elementen (Daten, Objekte) des Problembereichs. Die Datenstrukturen sowie Prozeduren zum Zugriff und zur Manipulation der Daten werden in Komponenten (Module, Klassen) zusammengefasst. Diese werden strukturiert zusammengesetzt und bilden die Architektur des Softwaresystems" (Appelrath et al. 2002, S. 106).

Hieran lassen sich bereits mehrere Unterschiede erkennen:

- Einzelne Funktionalitäten des Programms, die sicherlich im Verhalten sichtbar werden, können gegebenenfalls auf allein stehende Prozeduren und damit auf Aspekte des Programmierens im Kleinen zurückgeführt werden. Diese Zuordnung ist jedoch nicht zwingend, da andere Funktionalitäten erst durch das Wechselspiel mehrerer Komponenten (Klassen) realisiert werden, z. B. Zugriffskontrolle durch das Proxymuster.
- Während einfache Datentypen zum Programmieren im Kleinen gehören, sind abstrakte Datentypen, die Datenstrukturen und Operationen auf ihnen zusammenfassen, dem Programmieren im Großen zugeordnet, da relevante Eigenschaften in den Vordergrund treten und Implementierungsdetails vernachlässigt werden können (Appelrath et al. 2002, S. 112).

Objektorientierte Modellierung wird unter den Methoden des Programmierens im Großen aufgeführt, da sie die Vernetzung von Komponenten beschreibbar macht. Entsprechende Konzepte des Programmierens im Großen wie Objekte, Klassen, Attribute, Methoden, Kommunikation zwischen Objekten, Vererbung, Polymorphismus und dynamisches Binden sowie Datenkapselung werden explizit erörtert. Entwurfsmuster werden in diesem Kontext als hilfreich für Studienanfänger hervorgehoben:

> „Entwurfsmuster beschreiben allgemeingültige Entwurfslösungen, die von erfahrenen Softwareentwickler(inne)n gefunden wurden und in unterschiedlichen Kontexten vor allem von anderen Entwicklergruppen wiederverwendet werden können. Sie helfen damit insbesondere Anfängern und Anfängerinnen beim Erlernen der objektorientierten Softwareentwicklung" (Appelrath et al. 2002, S. 127).

Diese Aussage unterstreicht einen positiven Seiteneffekt des vorgestellten Unterrichtsmodells, nämlich den, dass die als Wissensrepräsentation für vernetzte fundamentale Ideen ausgewählten Entwurfsmuster lernförderlich für OOM sein können.

Kritisch zu sehen ist, dass diese Aussage sich nicht auf Schule, sondern den Kontext der Hochschule bezieht.

In ähnlichem Maße ist die Skalierung hinsichtlich der Anzahl der Anwender eines Informatiksystems ein Kennzeichen für große Informatiksysteme. Der Funktionsumfang ebenso wie die „Bedienphilosophie" bei großen Programmen spielen eine entscheidende Rolle zu deren Verständnis. Fragen wie „Wo bin ich?", „Was kann ich hier tun?", „Wie kam ich hier hin?", „Wo kann ich noch hin und wie komme ich dahin?" muss sich der Anwender beantworten (Nievergelt und Ventura (1983), zitiert nach (Eberle 1996, S. 351)).

Im Informatikunterricht sind strukturelle Merkmale eines Informatiksystems mit geeigneten informatischen Modellen zu unterlegen, damit die Schüler korrekte mentale Modelle ausbilden können (Voß 2006). Interessant ist, dass Eberle (1996) das Anwenden, z. B. von Standardsoftware, als Problemlöseprozess sieht, der beispielsweise mit Struktogrammen beschrieben werden kann, die der Algorithmik und damit dem Programmieren im Kleinen zuzuordnen sind. Andererseits empfiehlt er, dass Strukturen der Benutzungsoberfläche und Fehlermeldungen den verschiedenen Programmebenen zuzuordnen sind (Eberle 1996, S. 351). Das Ebenenmodell des Rechners wiederum ist ein Beispiel für Schichtenarchitekturen, die dem Programmieren im Großen zugerechnet werden. Auf die Notwendigkeit des vernetzten Denkens für Kompetenzentwicklung mit Informatiksystemen im Alltag weisen Brauer und Brauer hin (Abschnitt 5.4.1; (Brauer und Brauer 1992, S. 17)).

Damit wurden einige Informatikkonzepte identifiziert, die in großen Informatiksystemen zum Einsatz kommen. Wenn die Annahme stimmt, dass Informatiksysteme die in ihnen eingesetzten Konzepte widerspiegeln, bietet der Ansatz für Kompetenzentwicklung mit Informatiksystemen, der Entwurfsmuster als Wissensrepräsentationen für vernetzte fundamentale Ideen nutzt, viel Potential, denn wichtige Aspekte großer Informatiksysteme werden bereits angesprochen, ohne dass die Systeme konstruiert werden müssen. Eine empirische Überprüfung ist jedoch notwendig.

Offene Frage ist letztendlich die Skalierbarkeit des Unterrichtsmodells, d. h. inwieweit die diskutierten Vorgehensweisen für Kompetenzentwicklung mit Informatiksystemen sowohl für kleine Systeme als auch für große Systeme eingesetzt werden können, oder ob unterschiedliche Methoden zum Einsatz kommen müssen. Die Komplexität des Informatiksystems sei als Eingabegröße n bezeichnet. Die Frage ist, wie viele Ressourcen $f(n)$ beispielsweise die systematische Erkundung des Informatiksystems benötigt. Ressourcen seien Zeit und Kompetenzstufe zu Informatiksystemen, wobei letztere beispielsweise durch Wissen um Informatikkonzepte und deren Vernetzung in einem System gekennzeichnet ist. So ist es möglich, dass eine Vorgehensweise für große Informatiksysteme nicht zum gewünschten Ergebnis führt, während sie für kleine Programme gute Resultate liefert. Unter Skalierbarkeit sei also die Anpassungsfähigkeit des Unterrichtsmodells auf die Eingabe eines

konkreten Informatiksystems verstanden. Dazu zählt auch die Frage der Vollständigkeit, d. h., ob das Unterrichtsmodell bestimmte Aspekte großer (oder kleiner) Informatiksysteme gar nicht erfasst und diese ausgeblendet werden. Zur Erinnerung sei betont, dass Kompetenzentwicklung mit Informatiksystemen die Entmystifizierung, das Verstehen des Aufbaus, der Vernetzung und der Funktionsweise sowie bewusstes Anwenden und entsprechende Komplexitätsbewältigung zum Ziel hat, die motiviert, ein Informatiksystem für seine Zwecke zielgerichtet einzusetzen. Nicht gemeint ist die vollständige Beherrschung eines Systems wie Brunnstein (2001) sie beschreibt. So beinhaltet der hier verwendete Begriff der Skalierbarkeit sowohl Aspekte der Effizienz, d. h. Kompetenz zu Informatiksystemen schnell zu entwickeln, als auch der Effektivität, falls das Unterrichtsmodell einige Aspekte größerer Systeme vernachlässigt.

Die Skalierbarkeit des Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung ist in späteren Forschungsarbeiten mittels empirischer Untersuchungen mit Referenzsystemen durchzuführen. Unter anderem können zwei Systeme mit prinzipiell ähnlicher Funktion, aber unterschiedlichem Funktionsumfang gegenüber gestellt werden. Anhand von Kriterien und Kategoriensystemen lässt sich dann der Grad der Kompetenz zu Informatiksystemen messen. Dafür bedarf es somit eines Kompetenzmodells, das nicht mehr im Rahmen der vorliegenden Arbeit zu untersuchen ist. Der Autor greift diese Fragestellungen im Rahmen weiterer Forschung wieder auf (Abschnitt 9.3).

5.6.3 Fallstudienbasierte Entwicklung der Lernsoftware Pattern Park

Einordnung der Entwicklung in die Forschungsvorgehensweise

In diesem Abschnitt wird die Entwicklung der Lernsoftware Pattern Park beschrieben. Die Lernsoftware ist das Resultat der Arbeit einer studentischen Projektgruppe mit einem Umfang von 3600 Arbeitsstunden. Darin werden ausgewählte Entwurfsmuster gemäß des Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung für die Lernenden der Sekundarstufe II aufbereitet. Ziel der Lernsoftware ist nicht, die Schüler zu Softwareentwicklern auszubilden, sondern Entwurfsmuster als Wissensrepräsentationen für wichtige vernetzte fundamentale Ideen der Informatik lernförderlich einzusetzen. Dabei wird besonderer Wert auf die Formalisierung von statischen und dynamischen Aspekten von (Lebenswelt-) Situationen gelegt. Immer werden dazu Sichtenwechsel angeboten. Bereits entwicklungsbegleitend wurden einzelne Modulelemente exemplarisch im Informatikunterricht der Sekundarstufe II eingesetzt und evaluiert (Abschnitt 6.4.4). Dadurch gab es noch während des Entwicklungsprozesses Rückkopplung mit den Schülern, woraus notwendige Verfeinerungen abgeleitet werden konnten.

Die Entwicklung von Lernsoftware wird in einer experimentell bzw. quantitativ ausgerichteten Lehr-Lernforschung vernachlässigt, wodurch eine Innovationskrise entsteht (Reinmann 2006). Eine Orientierung an der ingenieurwissenschaftlichen Forschungsmethodik, die Softwareentwicklung einschließt und beispielsweise von Brinda (2004a) und Arnold (2007) vorgenommen wurde, fördert hingegen alltagstaugliche Innovation:

> „the engineering approach to research is directly concerned with practical impact – understanding how the world works and helping it 'to work better' by designing and systematically developing high-quality solutions to practical problems. [...] It combines imaginative design and empirical testing of the products and processes during development and in evaluation. [...] In the educational research community the engineering approach is often undervalued" (Burkhardt und Schoenfeld 2003, S. 5).

Reinmann (2006) bezeichnet dieses Vorgehen als „grundlagenorientierte Anwendungsforschung".

Die Lernsoftware Pattern Park basiert auf dem Lebensweltbezug durch das Szenario eines Freizeitparks. Dieser fungiert im Sinne der „Anchored Instruction" (Bransford et al. 1990) als interessanter und realistischer narrativer Anker (Bransford et al. 1990), also als Makrokontext (vgl. Hubwieser 2007a, S. 10).

Eingeordnet in die Strukturierung der Unterrichtsinhalte des Unterrichtsmodells unterstützt die Lernsoftware sowohl die Strukturmodelle als auch die vernetzten fundamentalen Ideen dadurch, dass sie Entwurfsmuster als Wissensrepräsentation einsetzt. Darüber hinaus legt die Lernsoftware die Schwerpunkte auf das nach außen sichtbare Verhalten und die innere Struktur von Informatiksystemen. Quelltext der Entwurfsmuster wird zwar ergänzend für fortgeschrittene Lernende angeboten, ist aber nicht manipulierbar. Vernetzte fundamentale Ideen der Informatik in Lebensweltsituationen bilden somit das Fundament.

Sichtenkonzept und Aufbau der Lernsoftware Pattern Park

Im Folgenden werden das Sichtenkonzept und der Aufbau der Lernsoftware Pattern Park beschrieben. Neben der Zugriffskontrolle, die sowohl im Proxy- als auch im Iteratormuster vorhanden ist, und den Zuständen mit dem Zustandsmuster, werden die fundamentalen Ideen Rekursion durch Kompositum und Dekorierer, Kapselung durch Fassade, Modularisierung durch Schablonenmethode, Iteration durch den Iterator, und „separation of concerns", d. h. Aufteilung der Aufgaben auf unterschiedliche Komponenten, durch das Beobachtermuster thematisiert (Tabelle 5.3). Zusätzlich zur Betrachtung fundamentaler Ideen in einzelnen Entwurfsmustern wird auch die Kombinierbarkeit der Muster aufgegriffen. Sie wurde in der fachdidaktischen Klassifikation als ein Kriterium identifiziert, das Anknüpfungspunkte für weiteren Unterricht liefert (Abschnitt 5.4.1). Daher gibt es in der Lernsoftware Pattern Park

eine so genannte Musterkombinationsaufgabe, in der die Muster Kompositum, Iterator, Dekorierer und Fassade verbunden sind. Darin wird das Lebensweltbeispiel einer zusammengesetzten Grafik zur Erstellung eines Logos des Freizeitparks, das bereits aus dem Modul zum Kompositummuster bekannt ist, wieder aufgegriffen und erweitert.

Zur Begründung des Sichtenkonzepts werden die sechs Interaktivitätsstufen von Lernumgebungen nach Schulmeister herangezogen.

Bei der Erstbegegnung mit einem Sachverhalt ist das intuitive Denken und Verstehen der Schüler zu berücksichtigen.

1. Objekte betrachten und rezipieren,
2. multiple Darstellungen betrachten und rezipieren,
3. die Repräsentationsform variieren,
4. den Inhalt der Komponente modifizieren,
5. das Objekt bzw. den Inhalt der Repräsentation konstruieren,
6. den Gegenstand bzw. Inhalt der Repräsentation konstruieren und durch manipulierende Handlungen intelligente Rückmeldung vom System erhalten (Schulmeister 2002, S. 194ff).

Folgende Sichten bzw. Aufgabentypen mit unterschiedlichen Sichten werden im Pattern Park angeboten.

Tabelle 5.3: Übersicht über die in der Lernsoftware Pattern Park umgesetzten Entwurfsmuster (Franke et al. 2007, S. 3)

Entwurfsmuster	**Lebensweltbeispiel**	**Fundamentale Ideen**
Kompositum	Grafik / Jugendgruppe	Rekursion, Datenstruktur Baum, Aggregation, Komposition
Beobachter	Eine Uhrzeit – mehrere Uhren	Assoziation (1-zu-n), Darstellung von Daten, separation of concerns
Fassade	Pförtner	Datenkapselung, Schnittstelle, Delegation
Proxy	Geldkarte	Zugriffsschutz, Schnittstelle
Dekorierer	Werkzeuge und Arbeitskleidung von Mitarbeitern	Rekursion, Vererbung
Iterator	Durchlaufen einer Besucherliste	Datenstruktur Liste, Traversierung
Schablone	Aufbau von Bahnen	Vererbung
Zustand	Achterbahnfahrt	Zustandsdiagramm, Zustände, Bedingungen, Aktionen
Musterkombination (Kompositum, Iterator, Dekorierer, Fassade)	Grafikwettbewerb	siehe einzelne Muster

Animation Jedes Entwurfsmustermodul enthält eine Animation zur Darstellung einer fundamentalen Idee, die durch ein Lebensweltbeispiel und anhand des jeweiligen Entwurfsmusters motiviert wird. Es gibt die Möglichkeit, Untertitel zu aktivieren, so dass unabhängig vom deutschsprachigen Audiokommentar weitere Sprachen durch Austausch der entsprechenden XML-Datei (Extensible Markup Language) unterstützt werden können. Zusätzlich zu den einzelnen Modulen gibt es eine Animation, die die Ziele der Lernsoftware für Kompetenzentwicklung mit Informatiksystemen zusammenfasst, und eine Animation zu Entstehung und Zweck der Entwurfsmuster nach Gamma et al. (1995), die Schülern das Prinzip der Wiederverwendung von Mustern näher bringen soll. Die Animationen können der Interaktivitätsstufe 1 nach Schulmeister zugeordnet werden.

Aufgabe im Freizeitparkkontext Aufgaben dieses Typs greifen einen Lebensweltbezug im Rahmen des Freizeitparks auf, um eine fundamentale Idee, die zu dem entsprechenden Entwurfsmustermodul gehört, handlungsorientiert zu verstehen. Die Veranschaulichung der Problemstellung steht im Vordergrund. Es werden keine weiteren Formalisierungen verlangt. Oft werden Zuordnungsaufgaben gestellt. Diese Aufgaben können der Interaktivitätsstufe 2 nach Schulmeister zugeordnet werden.

Aufgabe im Freizeitparkkontext mit Formalisierung Aufgaben diesen Typs beziehen sich ebenfalls auf eine Situation im Kontext des Freizeitparks, nutzen aber UML-Diagramme zur Formalisierung. In diesen Aufgaben werden bis zu drei Sichten dargestellt. Neben der Lebenswelt- oder Realsicht meist eine formale Sicht, z. B. Sequenzdiagramm, in der die Aufgabe zu lösen ist. Zusätzlich ist beispielsweise im Modul zum Proxymuster ein vollständiges Klassendiagramm angegeben. Die Vorgabe einer bereits formalisierten Darstellung zusätzlich zur Lebensweltsicht wurde gewählt, um die Schüler bei der Formalisierung einer Problemsituation durch eine weitere formale Sicht zu unterstützen. Im Zustandsmodul wird durch ein Zustandsdiagramm eine Achterbahnfahrt gesteuert. Insgesamt stärkt der in diesem Aufgabentyp immer vorhandene Sichtenwechsel einerseits die Verbindung des nach außen sichtbaren Verhaltens mit der inneren Struktur, andererseits ergänzt die Synchronisation einer statischen und dynamischen Sicht, z. B. durch Klassen- und Sequenzdiagramm, den Blick auf die Problemsituation (Brinda 2004a). Diese Aufgaben können den Interaktivitätsstufen 2 bis 3 nach Schulmeister zugeordnet werden.

Aufgabe mit Klassendiagrammeditor Zur formalen Darstellung der Musters wurde ein einfacher Klassendiagrammeditor, d. h. eine ikonische Repräsentation, in Pattern Park integriert. In dem so genannten UML-Puzzle gibt es ein unvollständiges Klassendiagramm und eine Aufgabenstellung, die letztlich

zu einer konkreten Umsetzung des Entwurfsmusters führt. Ein UML-Puzzle ermöglicht auch die Kombination mehrerer Entwurfsmuster in der Musterkombinationsaufgabe. Diese Aufgaben können der Interaktivitätsstufe 5 bis 6 nach Schulmeister zugeordnet werden, da die Lernsoftware Unterstützungsfunktionen anbietet.

Quelltextsicht Die Quelltextsicht ergänzt das Entwurfsmustermodul um eine symbolische Repräsentation des Musters. Sie wendet sich an interessierte Schüler, die sich auch Implementierungsaspekte ansehen möchten und ist in Java verfasst. Die Quelltextsicht ist nicht manipulierbar und kann der Interaktivitätsstufe 1 nach Schulmeister zugeordnet werden.

Arnold et al. (2005) stellen drei Anforderungen an entdeckendes Lernen im Informatikunterricht: Offenheit des Themas, Vollständigkeit des zur Verfügung gestellten Materials und Würdigung aller ernsthaften Schülerlösungen in der Bewertung. Diesem Ansatz des entdeckenden Lernens entsprechen in der Lernsoftware Pattern Park Aufgaben zum UML-Puzzle (Tabelle 5.4) insofern, dass die Problemstellungen nicht nur eine einzige richtige Lösung besitzen (Offenheit des Themas) und dass das jeweilige Modul hinreichend viel Zusatzmaterial zur Verfügung stellt, um die Problemstellung zu lösen (Vollständigkeit des zur Verfügung gestellten Materials). Die Bearbeitungszeit kann in Abhängigkeit vom Problem festgelegt werden und den Schülern kann Freiraum bei der Bearbeitung gelassen werden. Die Auswertung erfolgt nur zum Teil durch die Lernsoftware, so dass dem Lehrer die Bewertung der Schülerlösungen obliegt.

Zur Unterstützung des Lernens enthält die Modulübersicht Elemente zur Steuerung des eigenen Lernens (Abbildung 5.9). Schüler werden vorab über Lernziele, Lerndauer und notwendige Vorkenntnisse informiert und können dadurch ihren Lernprozess durch eine meta-kognitive Betrachtung selbst organisieren. Abbildung 5.9 zeigt die Modulübersicht zur Zugriffskontrolle.

Sichten und Schülertätigkeiten, um erwartete kognitive Barrieren zur Zugriffskontrolle zu überwinden

Die Lernsoftware Pattern Park bietet handlungsorientierte Zugänge an, um die Fehlvorstellungen und kognitiven Barrieren hinsichtlich der abstrakten Konzepte auszuräumen (siehe Abbildung **??** und Abschnitt 5.4.4). In diesem Abschnitt wird das Beispiel zur Zugriffskontrolle wieder aufgegriffen (vgl. Abschnitte 5.4.4 und 5.5.5).

Um die fundamentale Idee Zugriffskontrolle zu veranschaulichen, muss der Blick hinter die grafische Benutzungsoberfläche ermöglicht werden. Ein Entwurfsmuster als Wissensrepräsentation wird genutzt, um zu beschreiben, wie Struktur und Prozess zusammenspielen, um Zugriffskontrolle zu realisieren. Dazu wird der Ansatz

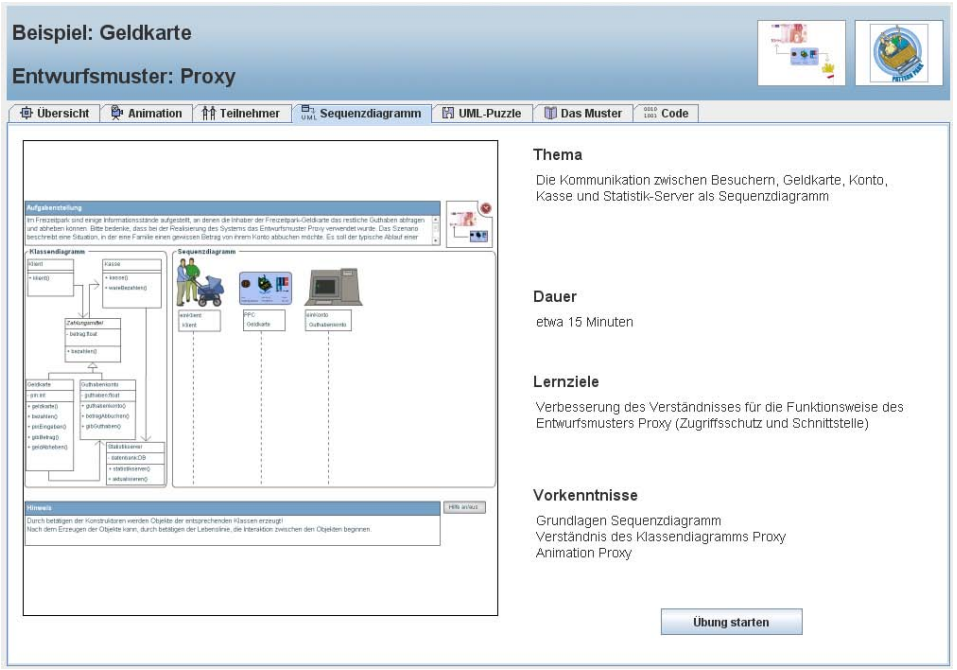


Abbildung 5.9: Übersicht einer Aufgabe zur Zugriffskontrolle mit Angabe der Dauer, Lernziele und Vorkenntnisse

des entdeckenden Lernens eingesetzt, durch das die Schüler vernetzte fundamentale Ideen erkunden können. Somit wird es in der Lernsoftware möglich, mit den fundamentalen Ideen der Informatik zu interagieren. Das Konzept der Sichtensynchronisation unterstützt das Lernen.

Das Modul zur Zugriffskontrolle startet mit einer Animation, die den Prozess und die teilnehmenden Objekte der Zugriffskontrolle mittels Stellvertreter visualisiert (S_A: nach außen sichtbares Verhalten; siehe Abschnitt 5.3). Mit der Animation wird die erste Fehlvorstellung, ein Objekt kontrolliere den Zugriff auf sich selbst, vermieden. Danach können Schüler den Prozess des Geldabhebens an einem Geldautomaten mit Hilfe unterschiedlicher UML-Diagramme beschreiben ($S_A + S_B$: Kombination des nach außen sichtbaren Verhaltens mit der inneren Struktur von Informatiksystemen). Damit gelingt die Verbindung der formalen Beschreibung eines Entwurfsmusters in Diagrammform mit der Lebenswelt. Diese Kombination der Sichten erlaubt das Überwinden der kognitiven Barriere der Vererbung in diesem

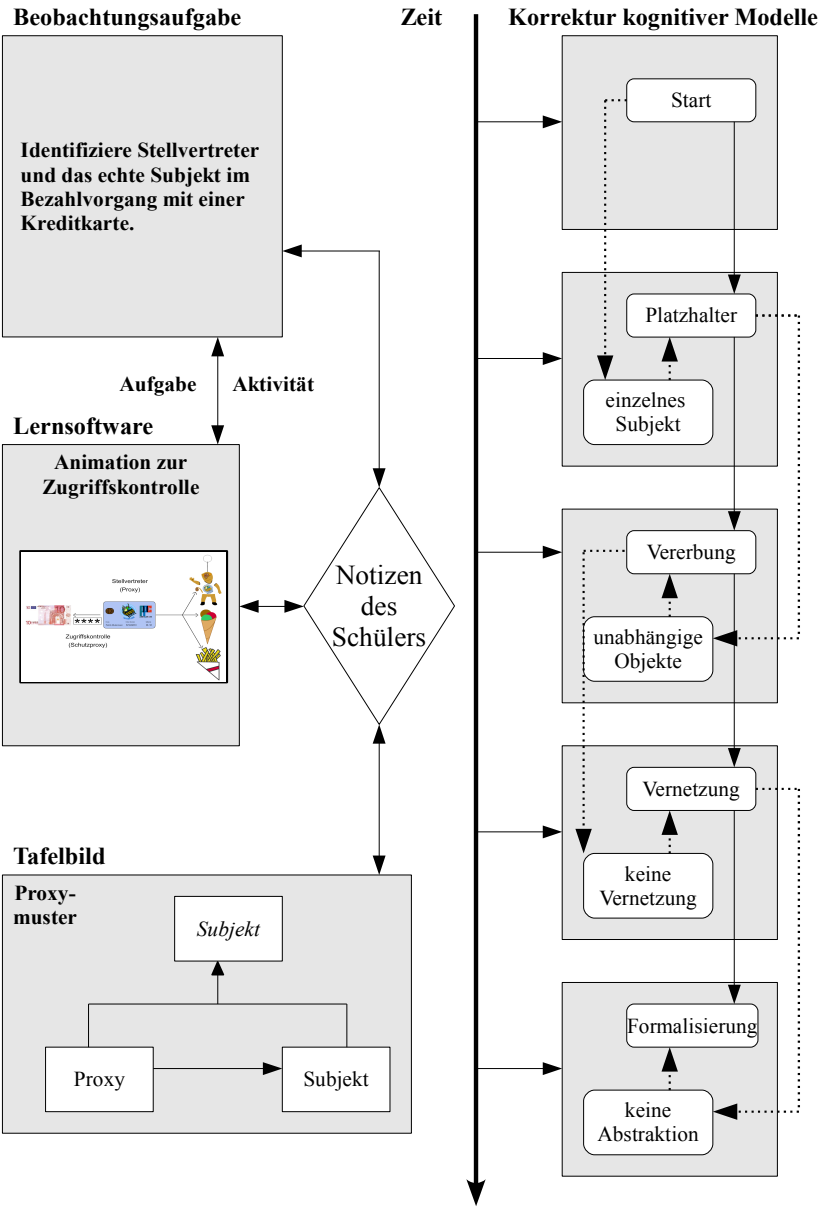


Abbildung 5.10: Korrektur von Fehlvorstellungen und kognitiven Modellen im Lehr-Lernprozess (vgl. Schubert et al. 2009, S. 3)

Kontext, durch die Originalobjekt (Geld) und Stellvertreterobjekt (Geldkarte) die gleiche Schnittstelle aufweisen.

Die kognitive Hürde Formalisierung ist schwierig zu überwinden: Das gegebene Klassendiagramm (vgl. Abbildung 6.3) ist eine weitere formale Darstellung, die die Schüler unterstützt zu formalisieren. Beistand leistet den Schülern in der Lernsoftware eine Hilfefunktion, die zu einem beliebigen Zeitpunkt im Problemlöseprozess aufgerufen werden kann. Sie stellt Problemstellung und vor allem die Elemente des Sequenzdiagramms nochmals vor, um die Schüler in die Lage zu versetzen, den Prozess der Zugriffskontrolle zu beschreiben. In Abbildung **??** wird der Lehr-Lernprozess mit Unterstützung von Lernsoftware zur Überwindung von Fehlvorstellungen und kognitiven Barrieren beispielhaft dargestellt.

Vor Fertigstellung der Lernsoftware Pattern Park wurden im Unterricht Teile des Moduls zur Zugriffskontrolle mit dem Entwurfsmuster Proxy und des Moduls zur Iteration mit dem Iteratormuster prototypisch erprobt (Abschnitt 6.4.4).

Die exemplarische Zuordnung der Aufgaben des Moduls zur Zugriffskontrolle zu den Stufen der Interaktivität zeigt Tabelle 5.4.

Tabelle 5.4: Exemplarische Einordnung der Modulaufgaben zur Zugriffskontrolle in die Interaktivitätstaxonomie nach Schulmeister (2002)

Aufgabe in der Lernsoftware Pattern Park	Stufe der Interaktivität
Animation zu Zugriffsschutz mittels Geldkarte (Zugriffsschutz, Stellvertreterfunktion, Smart Reference)	Stufe 1: Die Animation ermöglicht das Rezipieren der beteiligten Objekte und der Terminologie. Weitergehende Interaktivität ist nicht möglich.
Übung zu Lebensweltbeispielen für Situationen mit Stellvertreter, Klienten und Originalobjekten	Stufe 2-3: Die Schüler können die Objekte bewegen, jedoch nicht ihre Darstellung verändern.
Übung zur Beschreibung von Kommunikationsbeziehungen in einem Lebensweltbeispiel mittels Sequenzdiagramm bei gegebenem Klassendiagramm.	Stufe 3: Die Schüler können durch Eingabe von Parametern neue Darstellungsformen erarbeiten.
UML-Puzzle zur Zugriffskontrolle	Stufe 5-6: Mit dem Klassendiagrammeditor steht den Schülern ein Werkzeug zur Verfügung, in dem sie selbst Klassen erzeugen und beschreiben können. Durch die Lernsoftware werden die symbolischen Inhalte als sinntragende Objekte modelliert, so dass eine (einfache) Überprüfungsfunktion zur Rückmeldung an die Schüler umgesetzt ist.
Quelltextsicht	Stufe 1: Die Quelltextsicht ermöglicht das Rezipieren der Implementierung eines Entwurfsmusters.

Anforderungen an die Entwicklung interaktiver Lernsoftware

Arnold und Hartmann (2007) stellen fest, dass an interaktive Lernsoftware meist drei Anforderungen gestellt werden: Im Unterricht soll sie einen didaktischen sowie lehr-lernorganisatorischen und insgesamt einen ökonomischen Mehrwert haben. Da selten alle Anforderungen erfüllt sind, geben sie zehn pragmatische Empfehlungen, die bei der Entwicklung interaktiver Lernsoftware zu beachten sind (Arnold und Hartmann 2007, S. 177). Im Folgenden werden diese Schritte bzw. Fragestellungen aus Sicht der Lernsoftware Pattern Park kommentiert.

1. **Ist Lernsoftware bzw. Rechnereinsatz für dieses Thema notwendig?** Informatiksysteme und die ihnen innewohnenden fundamentalen Ideen der Informatik sind sehr facettenreich. Unterschiedliche Sichten auf und in ein Informatiksystem können mit der Lernsoftware geeignet unterstützt und die Sichten verbunden werden. Der übergreifende Lebensweltbezug des Freizeitparks bei dennoch kleinen Modulen zu vernetzten fundamentalen Ideen mittels Entwurfsmuster stellt einen didaktischen Mehrwert dar. ✓

2. **Ist das Unterrichtsthema auch in 10 Jahren noch relevant?** Durch die an vernetzten fundamentalen Ideen der Informatik orientierte Umsetzung der Lernsoftware ist die Langfristigkeit gewährleistet und der Entwicklungsaufwand gerechtfertigt. ✓

3. **Use-Cases: Ist Interaktivität möglich?** In zwei selbstorganisierten Seminaren zu Beginn des Projekts erarbeiteten die Projektgruppenmitglieder mögliche Aufgabenstellungen. Diese wurden mit den Betreuern diskutiert und in der Zwischenpräsentation überarbeitet vorgestellt. Jedes Modul in Pattern Park bietet Aufgaben unterschiedlich hoher Interaktivität. Von einer Animation über Zuordnungen und Aufgaben, die Wettkämpfe um beste Rundenzeiten einer Achterbahn ermöglichen, bis hin zur freien Gestaltung von Klassendiagrammen (vgl. Tabelle 5.4). ✓

4. **Paper Based Prototyping** Paper Based Prototyping der Benutzungsoberflächen zur Überprüfung von Funktionalität und Layout wurde für Pattern Park insofern eingesetzt, dass viele grafische Elemente handgezeichnet und dann in die Adobe Flash Entwicklungsumgebung übertragen wurden. Eine grafische Repräsentation ohne Funktionalität wurde so sehr schnell umgesetzt. Dadurch, dass gleichzeitig erste Beschreibungen der Aufgaben existierten, konnte geprüft werden, ob es Konflikte zwischen dem beabsichtigten Einsatz der Aufgaben und der Gestaltung der Lernsoftware gab. –/✓

5. **Rapid Prototyping** Da Flash-Filme in ein Java-Programm integriert werden sollten, war der Aufwand zur Erstellung eines Prototyps sehr gering und die

Umsetzung schnell möglich. Erste Erprobungen der Flash-basierten Aufgabenelemente wurden durch Peer-Review von Mitgliedern der Projektgruppe durchgeführt. ✓

6. **Technische Anforderungen: so einfach wie nur möglich** Pattern Park ist in Java implementiert und benötigt daher nur eine Java-Umgebung (Java Runtime Environment – JRE), die auf vielen Schulrechnern zu finden ist. Trotz der theoretischen Plattformunabhängigkeit durch Java gibt es verschiedene Versionen für Microsoft Windows, Linux und Mac OS, da die Betriebssysteme Adobe Flash unterschiedlich gut unterstützen. ✓

7. **Frühzeitige Erprobung** Im November 2006 wurde eine Unterrichtserprobung in der Sekundarstufe II mit Modulelementen zur Iteration und zur Zugriffskontrolle durchgeführt (Kapitel 6), um frühzeitig vor der endgültigen Fertigstellung der Software den generellen Aufbau der Module und die Akzeptanz der Lernenden bezüglich der unterschiedlichen Aufgabentypen in einem Modul zu prüfen. ✓

8. **Sparsamkeit bei der Benutzungsschnittstelle** Die Informatikstudierenden hatten alle das Nebenfach Medieninformatik und haben ihr Wissen zu Softwareergonomie genutzt, um die Benutzungsoberfläche ansprechend und intuitiv zu gestalten. Eine vertiefte Analyse der Benutzungsfreundlichkeit – über die Unterrichtserprobung und Peer-Reviews hinaus – fand nicht statt. –/✓

9. **Verbreitung der Lernumgebung** Der Verbreitungsprozess der Lernsoftware Pattern Park hat gerade erst begonnen. Neben der vorhandenen Webseite ist über weitere Verbreitungsmöglichkeiten wie Bildungsserver nachzudenken. Tabelle 5.5 zeigt die Zugriffsstatistik für Pattern Park im ersten Jahr nach der Fertigstellung. ✓

10. **Sicherstellung von Unterhalt und Kontinuität** Dadurch, dass die Software unter der GNU Lesser General Public License steht, ist eine – auch universitätsunabhängige – Weiterentwicklung prinzipiell möglich. –/ ✓

Analysiert man die Entwicklung der Software Pattern Park hinsichtlich der pragmatischen Empfehlungen, so ist vor allem hervorzuheben, dass frühzeitig Rückmeldungen von Schülern eingefordert und deren Anregungen integriert wurden. Das Interesse an der Lernsoftware Pattern Park zeigt die Zugriffsstatistik des Webservers (Tabelle 5.5). Für die Lernsoftware Pattern Park sind die von Hubwieser formulierten Fragen zum Medieneinsatz daher positiv zu beantworten:

1. „Ist dieses Medium eindeutig genug, um das intendierte Lernziel unmissverständlich und klar erscheinen zu lassen?
2. Repräsentiert dieses Medium den intendierten Inhalt derartig isomorph, dass es eine optimale Erfahrungsquelle für den Lernprozess schafft?

3. Ist dieses Medium attraktiv genug, um Aufmerksamkeit zu erregen?" (Hubwieser 2007a, S. 40)

Tabelle 5.5: Zugriffsstatistik des Webservers für die Lernsoftware Pattern Park binnen eines Jahres

Jahr	Monat	Downloads		
		Windows	Linux	MacOS
2007	Juli	108	5	3
	August	133	23	8
	September	114	20	12
	Oktober	106	8	6
	November	113	14	2
	Dezember	74	4	4
2008	Januar	93	16	12
	Februar	75	6	7
	März	70	18	6
	April	75	9	5
	Mai	63	7	4
	Juni	62	12	6
insgesamt		1086	142	75

Kritische Betrachtung der Lernsoftware Pattern Park

Die ersten Erprobungen im Unterricht (Kapitel 6 und 8) zeigen, dass die Lernsoftware attraktiv genug ist, um Aufmerksamkeit der Schüler zu erregen. Außerdem sind die Einbettung in einen Lebensweltkontext und das Anbieten unterschiedlicher Sichten dazu geeignet, den Unterrichtsgegenstand zu repräsentieren, und auch die Übungsaufgaben sind auf die im Lernmodul formulierten Lernziele ausgerichtet. Dennoch wird die weitere Verbreitung sehr davon abhängen, Lehrpersonen von der Lernsoftware zu überzeugen. Fundamentale Ideen als Unterrichtsinhalt können einen starken Beitrag dazu leisten. Es ist jedoch aus mehreren Rückmeldungen zu entnehmen, dass die Verwendung von Entwurfsmustern zum Teil hemmend wirkt. Dem kann durch Einbettung in gut ausgearbeitete Unterrichtssequenzen und die Hervorhebung der fundamentalen Ideen entgegen gewirkt werden. Ein weiterer Kritikpunkt zur Lernsoftware ist sicherlich dahingehend anzubringen, dass freies Experimentieren im Sinne der Interaktivitätsstufen 5 bis 6 innerhalb der Lernsoftware nur im Klassendiagrammeditor möglich ist. Die Animationen und Aufgaben im Freizeitparkkontext mit und ohne formale Darstellungen sind durch die Realsicht weitaus ansprechender gestaltetet, jedoch von geringerem Freiheitsgrad. In dem vorliegenden Unterrichtsmodell ist die Lernsoftware Pattern Park daher zur Motivation und vor allem zur Kombination unterschiedlicher Sichten hinsichtlich Verhalten und Struktur, zur Verbindung dynamischer und statischer Darstellungen und zur Veranschaulichung fundamentaler Ideen einzusetzen. Hilfefunktionen unterstützen den Lernenden im Pattern Park. Mehr Freiheiten im Lehr-Lernprozess

bieten dagegen kleine Programme, die strukturell auf ausgewählten Entwurfsmustern basieren (Abschnitt 5.6.2).

5.7 Zusammenfassung und Schlussfolgerungen zur Kompetenzentwicklung

5.7.1 Zusammenfassung

Zur Förderung der Kompetenzentwicklung mit Informatiksystemen wurde ein Unterrichtsmodell entwickelt. Dafür wurde in diesem Kapitel zunächst die Intention des Unterrichtsmodells erläutert (Abschnitt 5.2). Eine Strategie zur Strukturierung inklusive einer Fokussierung der Ergebnisse der Analyse des Forschungsstandes wurde vorgestellt (Abschnitt 5.3). Schwerpunkte des Unterrichtsmodells zur Förderung der Kompetenzentwicklung mit Informatiksystemen bilden vernetzte fundamentale Ideen der Informatik und Entwurfsmuster als Strukturmodelle. Es wurde ein systemanalytischer Top-down-Zugang zu Informatiksystemen gewählt, der den Blick auf das System und den Blick in das System kombiniert, ohne die Erstellung eines konkreten Systems zum Ziel zu haben. Damit wird der häufigen Schwierigkeit des Informatikunterrichts begegnet, dass Schüler stark unterschiedliche Vorkenntnisse hinsichtlich der Syntax einer Programmiersprache haben. Es folgte die Forderung nach informatischen Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik, die die konzeptuelle Planung des Unterrichtsmodells unterstützen. Exemplarisch erfolgte eine Ausarbeitung von Entwurfsmustern als Wissensrepräsentation (Abschnitt 5.4). Aus der Strukturierung der Basiskompetenzen zur Förderung der Kompetenzentwicklung mit Informatiksystemen wurde abgeleitet, dass Schüler in der Lage sein müssen, Informatiksysteme systematisch zu erkunden. Ein systematisches Vorgehen in Anlehnung an Unterrichtsexperimente unter Berücksichtigung unterschiedlicher kognitiver Niveaustufen wurde entwickelt (Abschnitt 5.5). Abschließend wurden Konsequenzen für Lernsoftware diskutiert, die vernetzte fundamentale Ideen über Entwurfsmuster aufgreift (Abschnitt 5.6). Damit wird eine Anleitung zum einfachen Erstellen von analysefreundlichen Programmen gegeben. Software, die auf nur einem oder zwei Entwurfsmustern basiert, ist darüber hinaus leicht in andere Kontexte zu transferieren, so dass der Vorbereitungsaufwand für die Lehrperson gering ist. Die gut dokumentierten Entwurfsmusterkataloge unterstützen die Lehrperson bei der Entwicklung. Im Rahmen der vorliegenden Arbeit wurden mehrere derartige Programme konstruiert. Ergänzt werden die Materialien durch die umfangreiche Lernsoftware Pattern Park. Abbildung 5.11 zeigt die schematische Darstellung des Unterrichtsmodells.

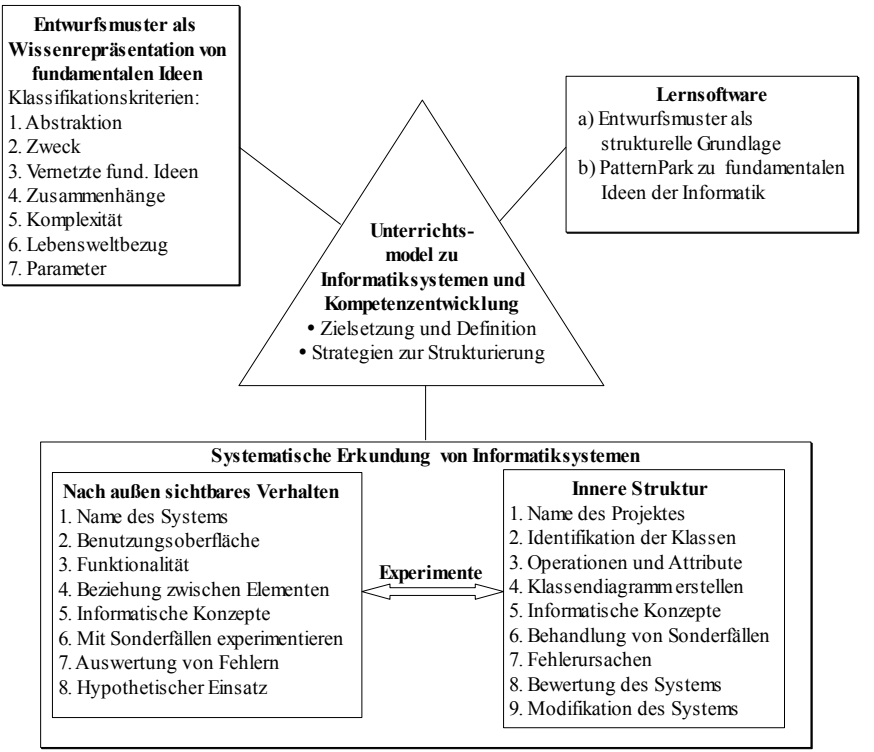


Abbildung 5.11: Schematische Darstellung des Unterrichtsmodells zur Kompetenzentwicklung mit Informatiksystemen

5.7.2 Schlussfolgerungen für die Kompetenzentwicklung mit Informatiksystemen

Förderung von Schlüsselkompetenzen im Unterrichtsmodell

Hinsichtlich der Kompetenzentwicklung sind Schülertätigkeiten und speziell Handlungen mit Informatiksystemen notwendig. Als zentrale Schülertätigkeit wurde in dem Unterrichtsmodell die systematische Erkundung von Informatiksystemen herausgearbeitet (Abschnitt 5.5). Insbesondere wurde betont, dass Fehler und unerwartetes Verhalten von Informatiksystemen es den Schülern ermöglichen, Inkonsistenzen in der Software und ihrem kognitiven Modell aufzudecken (vgl. „Anticipation of breakdown" und „Readiness-to-hand" (Winograd und Flores 1988, S. 164), (Hubwieser 2007a, S. 47)). Dadurch werden sie in die Lage versetzt, Informatiksysteme

interaktiv anzuwenden, d. h. sie erwerben die Schlüsselkompetenz (1c) gemäß DeSeCo (Abschnitt 2.1.2; (OECD 2005)). Sie sind in der Lage, Potential und Grenzen einzuschätzen und Informatiksysteme in ihren Alltag zu integrieren (OECD 2005, S. 13). Damit können sie Informatiksysteme beispielsweise als Lernmedium einsetzen und wählen zielgerichtet Anwendungen aus.

Neben der bewussten Anwendung von typischen Informatiksystemen bedarf Kompetenzentwicklung mit Informatiksystemen aber auch weitergehender informatischer Kenntnisse, um sich Fertigkeiten und Fähigkeiten zur interaktiven Nutzung von Informatiksystemen als Werkzeug anzueignen.

> „Um dieses Potenzial zu nutzen, sind weiterreichende Fähigkeiten und Fertigkeiten erforderlich, die über eine einfache Internet-Nutzung, den Versand von E-Mails usw. hinausgehen" (OECD 2005, S. 13).

Kompetenzentwicklung mit Informatiksystemen erlaubt es, deren Anwendungsbereiche für das Individuum zu erweitern. In Kapitel 2 wurde argumentiert, dass Bereitschaften und Einstellungen zur Kompetenz gehören. Damit stellt sich die Frage nach Erwartungshaltungen der Schüler gegenüber Informatiksystemen, die dann ausschlaggebend sind, ob ein Schüler sie als Werkzeug nutzt (DeSeCo: 1c). Zur Förderung der Motivation werden für die systematische Erkundung unerwartete Ereignisse und Beobachtungen als Ausgangspunkt gewählt. Durch die erwartete erhöhte Motivation ist es Schülern dann möglich, sich auf ihr Handeln zu konzentrieren und diese Fähigkeit in ihren außerschulischen Alltag zu transferieren.

Es wurde herausgestellt, dass Schüler die „Versuch-Irrtum"-Strategie für Kompetenzentwicklung nicht anwenden sollen, sondern ihr Handeln reflektieren müssen. Durch die systematische Erkundung von Informatiksystemen bekommen die Schüler eine Handreichung, um eigenständig zu handeln (DeSeCo: 3; (OECD 2005, S. 16)). Durch die Fähigkeit, sich selbst unbekannte Informatiksysteme zu erschließen, können sie an unterschiedlichen Projekten innerhalb und außerhalb der Schule aktiv teilnehmen (DeSeCo: 3b). Außerdem können sie durch vorheriges Erkunden eines Informatiksystems als Individuen die Konsequenzen ihres Handeln einschätzen, z. B. hinsichtlich der Bereitschaft, persönliche Daten im System zu speichern, um Entscheidungen zu fällen und Verantwortung zu übernehmen (DeSeCo: 3a). Reflexion über den hypothetischen Einsatz von Informatiksystemen einschließlich sozialer Implikationen wird in der systematischen Erkundung des nach außen sichtbaren Verhaltens von Informatiksystemen gefordert.

Die Notwendigkeit, die systematische Erkundung eines Informatiksystems zu dokumentieren und mit Mitschülern über das System und das geplante Experiment zu sprechen, fördert die Kooperationsfähigkeit der Schüler. Sie ermöglicht das Agieren in heterogenen Gruppen zur Bewältigung einer Anforderungssituation mit Informatiksystemen (DeSeCo: 2b).

Bei der Analyse der inneren Struktur von Informatiksystemen ist der Wechsel zwischen unterschiedlichen Rollen hilfreich, z. B. Anwender, Softwareentwickler und Auftraggeber. Durch Empathie gelingt es Schülern, unterschiedliche Anforderungen an Informatiksysteme abzuleiten und Verständnis für die gewählte Strukturierung aufzubringen (DeSeCo: 2a). Auch bei kleinen Modifikationen bestehender Systeme kann in Paaren an der Aufgabe gearbeitet werden, um die Kommunikationsfähigkeit in heterogenen Gruppen zu erlernen (DeSeCo: 2b).

Förderung der ICT Literacy durch das Unterrichtsmodell

Zur Strukturierung des Unterrichtsmodells wurden Strukturmodelle und vernetzte fundamentale Ideen besonders betont (Abschnitt 5.3). Durch die Fokussierung auf Entwurfsmuster (Abschnitt 5.4) ist hinsichtlich der von der UNESCO definierten Literacys ein Beitrag zur Software Literacy zu erwarten (UNESCO 2008). Dennoch ist das Unterrichtsmodell nicht notwendigerweise mit Softwaremustern verknüpft. Vielmehr ermöglicht die Vernetzung von fundamentalen Ideen in Strukturmodellen ebenso eine stärkere Betonung der Hardware, z. B. durch das Von-Neumann-Blockmodell oder das Ebenenmodell des Rechners (vgl. Abschnitt 9.3). Durch den Einsatz unterschiedlicher Lernsoftware (Abschnitt 5.6) ist des Weiteren ein Beitrag zur Media Literacy zu erwarten.

Einordnung des Unterrichtsmodells in den Europäischen Qualifikationsrahmen

Exemplarisch können Themen und Lernziele des Unterrichtsmodells in den Europäischen Qualifikationsrahmen für lebenslanges Lernen (EQR) eingeordnet werden. In Kapitel 2 wurde argumentiert, dass für informatische Bildung in der Sekundarstufe II nicht alle Niveaustufen des EQR zu adressieren sind. Tabelle 5.6 führt mögliche Lernziele und Unterrichtsinhalte anhand des Unterrichtsmodells bis einschließlich Niveaustufe 4 auf. Im Unterrichtsmodell selbst werden unterschiedliche Lernzielebenen gemäß der Lernzieltaxonomie nach Anderson und Krathwohl (2001) genutzt (Abschnitt 5.5).

Die theoretisch begründete Entwicklung des Unterrichtsmodells erfordert nun eine exemplarische Erprobung in der Sekundarstufe II an allgemein bildenden Schulen (Kapitel 6).

Tabelle 5.6: Einordnung des Unterrichtsmodells in den EQR bis Niveaustufe 4

Kenntnisse	**Fertigkeiten**	**Kompetenz**
1) Grundlegendes Allgemeinwissen, z. B. zur Verbreitung typischer Informatiksysteme	Grundlegende Fertigkeiten, die zur Ausführung einfacher Aufgaben mit Informatiksystemen erforderlich sind, z. B. Analyse der Beziehung zwischen Elementen der Benutzungsoberfläche und der Funktionalität des Systems	Arbeiten oder Lernen mit Informatiksystemen unter direkter Anleitung in einem vorstrukturierten Kontext, z. B. lehrergeleitetes Erkunden eines Informatiksystems
2) Grundlegendes Faktenwissen zu Informatiksystemen, z. B. zur Zugriffskontrolle	Grundlegende kognitive und praktische Fertigkeiten mit Informatiksystemen, um Aufgaben auszuführen und Routineprobleme unter Verwendung einfacher Regeln und Anwendungssoftware zu lösen, z. B. Vermeidung von Versuch-Irrtum-Strategien	Arbeiten oder Lernen mit Informatiksystemen unter Anleitung mit einem gewissen Maß an Selbstständigkeit, z. B. Einsetzen einer Lernsoftware unter Aufsicht der Lehrperson
3) Kenntnisse von Fakten, Grundsätzen, Verfahren und allgemeinen Begriffen zu Informatiksystemen, z. B. systematisches Testen	Eine Reihe kognitiver und praktischer Fertigkeiten zur Erledigung von Aufgaben und zur Lösung von Problemen, wobei die systematische Erkundung eines Informatiksystems ausgewählt und angewandt wird, z. B. Erkundung der neuen Version einer Standardsoftware	Verantwortung für die Erledigung von Arbeits- oder Lernaufgaben übernehmen, z. B. die Bereitschaft, bei der Lösung von Problemen die systematische Erkundung an unbekannte Systeme anpassen
4) Breites Spektrum an Theorie- und Faktenwissen zu Informatiksystemen, z. B. Universalität und die Rolle der Entwurfsmuster	Eine Reihe kognitiver und praktischer Fertigkeiten, die erforderlich sind, um Lösungen für spezielle Probleme zu finden, z. B. Beheben von Fehlerursachen durch Identifikation von Sonderfällen oder Modifikation von Quelltext	Selbstständiges Tätigwerden innerhalb der Handlungsparameter zu Informatiksystemen, die in der Regel bekannt sind, sich jedoch ändern können; Beaufsichtigung der Routinearbeit anderer Personen, wobei eine gewisse Verantwortung für die Bewertung und Verbesserung der Arbeits- oder Lernaktivitäten übernommen wird, z. B. bei der systematischen Erkundung und Identifikation von Sonderfällen der Software

6. Erste exemplarische Erprobung des Unterrichtsmodells

6.1 Überblick

Ein Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen wurde theoretisch begründet (Kapitel 5). In diesem Kapitel wird die erste exemplarische Erprobung des Unterrichtsmodells beschrieben. Dazu erfolgt eine Einordnung in den Forschungsverlauf (Abschnitt 6.2). Die Rahmenbedingungen einschließlich inhaltlicher Konzeption, Lerngruppe, zeitlichen und technischen Restriktionen sowie Unterrichtsmethodik werden in Abschnitt 6.3 angegeben.

Die Beschreibung und Durchführung der Erprobung geschieht in Abschnitt 6.4. In Abschnitt 6.5 werden die erhobenen Daten zur Evaluation hinsichtlich Machbarkeit und Akzeptanz durch Schüler und Lehrer herangezogen. Abschließend wird eine kurze Diskussion der Ergebnisse vorgenommen (Abschnitt 6.6). Abbildung 6.1 zeigt die Einordnung des Kapitels in den Forschungsverlauf.

6.2 Motivation der Unterrichtserkundung und Einordnung in den Forschungsverlauf

Die erste Unterrichtserprobung fand im Herbst 2006 statt (Stechert 2007c). Vorausgegangen waren die theoretische Fundierung der Entwurfsmuster als Wissensrepräsentation für vernetzte fundamentale Ideen der Informatik (vgl. (Stechert 2006a), (Stechert 2006b)) und die Unterrichtsmodellentwicklung (vgl. Stechert 2006c). Basiskompetenzen wurden im Unterrichtsprojekt in drei Bereiche $S_i, i \in \{A, B, C\}$ grob strukturiert, die im Fokus des Unterrichtsmodells zu Informatiksystemen stehen (vgl. Claus und Schwill 2006, S. 677):

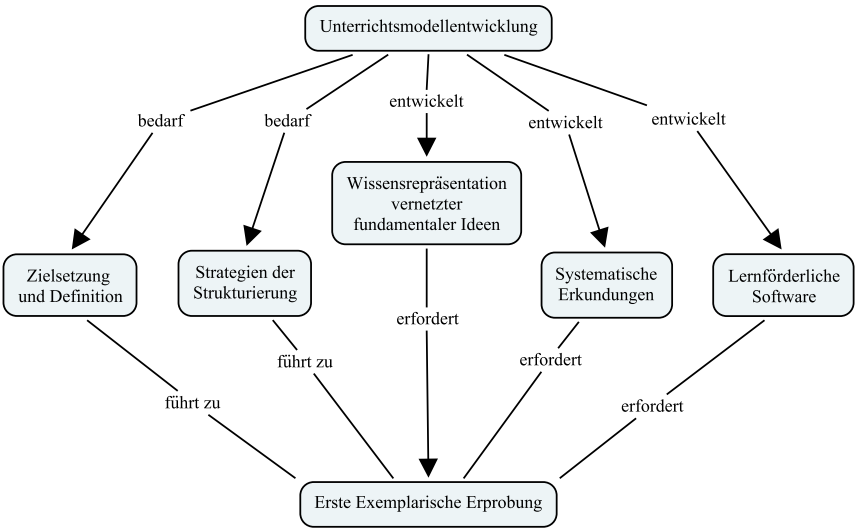


Abbildung 6.1: Einordnung des sechsten Kapitels in den Forschungsverlauf

S_A: Aspekte des nach außen sichtbaren Verhaltens.

S_B: Aspekte der inneren Struktur, die auf Strukturmodellen und vernetzten fundamentalen Ideen der Informatik basieren.

S_C: Ausgewählte Implementierungsaspekte zur Entwicklung einer konkreten Realisierung.

Im Sinne von Basiskompetenzen als Mindestanforderungen, die ein mündiger Bürger im Alltag erfüllen muss, wurde S_C gering gewichtet und wenig Lernzeit darauf verwendet (Abschnitt 2.1).

Der exemplarische Charakter der Umsetzung des Unterrichtsmodells musste trotz zeitlicher Restriktion gewährleistet werden. Annahme ist, dass die exemplarische Umsetzung von ausgewählten Teilen eines umfassenderen Forschungsprojekts verallgemeinerbare Rückschlüsse hinsichtlich Lernschwierigkeiten auf das Forschungsprojekt insgesamt erlaubt (Schubert et al. 2007). Zum Beispiel bleiben durch die begründete Auswahl von einigen wenigen Entwurfsmustern wesentliche Merkmale des Unterrichtsmodells bestehen. Dies gilt insbesondere bezüglich der Gestaltung von qualitativ hochwertigen Aufgaben zur Kompetenzentwicklung mit Informatiksystemen: So sind die ausgewählten Entwurfsmuster Wissensrepräsentationen für die in Informatiksystemen vorkommenden vernetzten fundamentalen Ideen der

Informatik. Zur Strukturierung des Unterrichts wurden Schülertätigkeiten und Unterrichtsinhalte dem nach außen sichtbaren Verhalten, der inneren Struktur und Implementierungsaspekten zugeordnet. Lernende setzen das nach außen sichtbare Verhalten eines Informatiksystems in Beziehung zu der zugrunde liegenden inneren Struktur. Trotz Einschränkung auf Softwaresysteme für Kompetenzentwicklung mit Informatiksystemen bleibt der exemplarische Charakter der Erprobung bewahrt, denn:

> „Hardware und Software sind logisch äquivalent" (Tanenbaum und Goodman 2001, S. 27).

Bei ersten Rechnern war die Grenze zwischen Hardware und Software gut erkennbar. Heutzutage jedoch sind Hard- und Software durch hinzufügen, entfernen und verschmelzen von Ebenen kaum noch zu unterscheiden. Tanenbaum und Goodman (2001) verweisen darauf, dass Softwareberechnungen auch direkt in der Hardware durchgeführt werden können und Software wiederum Hardwarefunktionen simulieren könne.

Untersuchungsschwerpunkte sind

1. die Tragfähigkeit und Auswirkung der Dreiteilung in Systemverhalten, innere Struktur und Implementierungsaspekte,
2. die Eignung ausgewählter Entwurfsmuster zur Repräsentation und Vernetzung fundamentaler Ideen der Informatik in kleinen Programmen,
3. die exemplarische Erprobung von Modulen der Lernsoftware Pattern Park,
4. die Lehr-Lernmethodik mit systematischen Erkundungen von Informatiksystemen und die Akzeptanz der Schüler.

6.3 Rahmenbedingungen und Untersuchungsmethodik

6.3.1 Inhaltliche Konzeption

Das theoretische Unterrichtsmodell muss an die Vorkenntnisse der konkreten Zielgruppe angepasst werden. Bei der Gestaltung von Lernmitteln für eine erste Erkundung des Unterrichtsmodells spielte die konkrete Lernsituation in Jahrgangsstufe 12 der Kooperationsschule eine große Rolle. In Nordrhein-Westfalen ist Objektorientierung in der Sekundarstufe II ein weit verbreitetes Konzept (MSWWF 1999), das in der Kooperationsschule eingesetzt wird.

Zur Vorbereitung wurde mit dem Informatiklehrer, in dessen Klasse das Unterrichtsprojekt durchgeführt werden sollte, und einem weiteren Informatiklehrer der

Schule vier Monate vor der Durchführung ein erster Entwurf der Unterrichtskonzeption diskutiert, um Umsetzungsprobleme frühzeitig zu erkennen. Mit Blick auf das Schulcurriculum wurden die Lernziele zusammengestellt. Die Ergebnisse dieser Diskussionen flossen auch in die Unterrichtsmodellentwicklung (vgl. Stechert 2006c) ein. Darüber hinaus hat der Autor in dem Kurs zwei Wochen vor der Durchführung hospitiert, um an vorangegangenen Unterricht anzuknüpfen. Dabei erwies es sich als unkompliziert, den bisherigen Unterricht nun nahtlos in einem Unterrichtsprojekt weiterzuführen, das dem Unterrichtsmodell zu Informatiksystemen und Kompetenzentwicklung folgte. Dies ist neben der Tatsache, dass konkrete Programme Unterrichtsgegenstand waren, auch auf die objektorientierten Entwurfsmuster zurückzuführen. So eignete sich das Iteratormuster, um an die Datenstruktur Schlange anzuknüpfen, die im Unterricht behandelt wurde. Das Thema Zugriffskontrolle ist aufgrund seiner Wichtigkeit bei der Anwendung von Informatiksystemen aufgenommen worden (Abschnitt 5.4.4). Tabelle 6.1 zeigt die in Absprache mit den Informatiklehrern der Kooperationsschule erarbeitete Konzeption. Durch die geringe zur Verfügung stehende Zeit wurde im Verlauf des Projektes von dieser Vorlage insofern abgewichen, als dass das Programm zur Zugriffskontrolle auf Musikstücke nur zur Motivation diente, und die Schüler für die Listenstruktur bei einem Programm, einer einfachen Arztpraxissoftware, blieben.

Tabelle 6.1: Struktur der ersten Unterrichtserprobung inklusive Schülertätigkeiten

Thema (Muster)	S_A: **Nach außen sichtbares Verhalten von IS** Experimente mit Black-Box-Sicht	S_B: **innere Struktur von IS** Erkundung der inneren Struktur und Modellierung mit Klassen- und Sequenzdiagramm	S_C: **ausgewählte Implementierungsaspekte von IS** Modifikation einer konkreten Realisierung
Zugriff auf eine Liste (Iterator)	Erkennen von Listeneigenschaften durch Testen von einfachen Programmen (Arztpraxis mit Warteschlange)	Objektorientierter Entwurf einer Listenstruktur; Schlange vs. Liste	Iteration durch Iterator modellieren; Programmieren von Programmfragmenten
Zugriffskontrolle (Proxy)	Textverarbeitung: Zugriff auf Grafiken (virtueller Proxy); Internet-Beispiel anhand eines Arbeitsblattes; Verhalten eines Programms mit Zugriffszähler auf Musiktitel evaluieren; Vorteile von Zugriffskontrolle	Objektorientierter Entwurf der Zugriffskontrolle; Proxy-Cache, Verbindung mit Liste als weiteres Beispiel; Zugriffskontrolle auf Liste mit Musikstücken entwerfen (Smart Reference)	Erweiterung der Zugriffskontrolle um das Zählen der Zugriffe (z. B. Zugriff auf ein Musikstück) mit Delphi; Programmierung von Programmfragmenten und Anpassung an vorgegebenen Quellcode

Tabelle 6.2: Exemplarische Übersicht der inhaltlichen Konzeption anhand der Strukturierung der Basiskompetenzen

	exemplarische Unterrichtsinhalte
$S_{A,1}$	Die Datenstruktur Schlange und Schlangentraversierung sowie weitere Anforderungen an das Entwurfsmuster Iterator.
$S_{A,2}$	Zugriffskontrolle anhand des Entwurfsmusters Proxy.
$S_{B,1}$	Das Konzept der Schnittstelle anhand der Kombination von Schlange und Iteratorentwurfsmuster sowie phänomenologischer Vorwegnahme der Zugriffskontrolle.
$S_{B,2}$	Zugriffskontrolle anhand des Entwurfsmusters Proxy sowohl statisch, d. h. via Klassendiagramm, als auch dynamisch mittels Sequenzdiagramm.
$S_{C,1}$	Unterschiedlicher Traversierungsalgorithmen durch Modifikation des Iteratormusters.
$S_{C,2}$	Zuweisung von Zugriffsrechten durch Vererbung und Modifikation von Implementierungsaspekten im Proxyentwurfsmuster.

In der durchgeführten Unterrichtserprobung lag der Schwerpunkt auf dem Zusammenspiel zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur eines zum Teil selbst erstellten Programms (S_A und S_B). Zur Formulierung der Teillernziele wurden Operatoren aus den Anforderungsbereichen I (z. B. Wiedergabe von Kenntnissen), II (z. B. Anwenden und Transferieren von Kenntnissen) und III (z. B. Problemlösen und Bewerten) der Einheitlichen Prüfungsanforderungen (EPA) Informatik (KMK 2004) sowie nach Blooms überarbeiteter Taxonomie eingesetzt (Anderson und Krathwohl 2001). Exemplarische Unterrichtsinhalte anhand der Strukturierung der Basiskompetenzen zeigt Tabelle 6.2.

Die konkrete Umsetzung wird in Abschnitt 6.4 beschrieben. Der Bereich S_C war nicht vordergründig Beobachtungsziel des Projekts. Nur vereinzelt wurde existierender Quellcode von den Schülern erweitert, um an vorhandenes Vorwissen anzuknüpfen.

6.3.2 Lerngruppe und zeitlicher Rahmen

Die Erprobung des Unterrichtsmodells fand im Rahmen eines Grundkurses Informatik in der Jahrgangsstufe 12 eines Siegener Gymnasiums statt. Der Kurs setzte sich zusammen aus vier Schülerinnen und 19 Schülern. Dadurch, dass es sich hierbei um einen regulären Grundkurs handelte, sind valide qualitative Auswertungen möglich, da der Kurs nicht nur aus hoch motivierten Schülern besteht, wie es im Rahmen von Wahlfächern oft der Fall ist. Um die Ergebnisse der Fallstudie im Sinne von Plausibilitätserklärungen weitergehend verallgemeinern zu können, sollten die Schüler Vorkenntnisse in objektorientierter Modellierung besitzen und möglichst keine Spezialisierung im technisch-naturwissenschaftlichen Bereich vorweisen. Die Schüler nutzten bereits seit einem halben Schuljahr die objektorientierte Modellierung u. a. mit der Klassenbibliothek „Stifte und Mäuse" (Czischke et al. 1999). Assoziationen, Vererbung, das Erstellen von Klassen und Erzeugen von Objekten

sowie ereignisgesteuerte Programmierung waren bekannt ebenso wie grundlegende Programmierkonzepte, z. B. Polymorphie und dynamisches Binden sowie Schleifen und Variablen.

Die Hospitation in der Klasse kurz vor Beginn des Unterrichtsprojekts machte deutlich, dass Klassendiagramme als wichtiges Strukturbeschreibungsmittel eines Informatiksystems ohne weitere Vorbereitung eingesetzt werden konnten. Allgemein fiel während der Hospitation die Schwierigkeit der Schüler auf, zwischen Daten und Darstellung zu unterscheiden – ein Abgleich zwischen Datenhaltung und Darstellungsebene fand in den Schülerlösungen dementsprechend oft nicht statt. Dies war ein erster Ansatzpunkt für die Erkundung des nach außen sichtbaren Verhaltens des Programms im Unterrichtsprojekt. Schnittstellen und Geheimnisprinzip sowie das Substitutions-Prinzip nach Liskov, welches besagt, dass Unterklassen Basisklassen substituieren, da Unterklassen alles von der Oberklasse erben, waren nur intuitiv bekannt.

Tabelle 6.3: Themenliste der ersten Unterrichtserprobung mit Einzel- (ES) und Doppelstunden (DS)

Termin	**Stundenthema**
17.10.2006 (ES)	Systematisches Erkunden des nach außen sichtbaren Verhaltens und der inneren Struktur eines Informatiksystems am Beispiel eines Arztpraxisprogramms
19.10.2006 (DS)	Zusammenhänge zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur eines Informatiksystems herstellen
24.10.2006 (ES)	Beschreiben der inneren Struktur mit Klassendiagrammen
26.10.2006 (DS)	Iteration mit Iteratorentwurfsmuster
31.10.2006 (ES)	Varianten der Iteration mit Iterator
02.11.2006 (DS)	Zugriffskontrolle mittels Proxy
16.11.2006 (DS)	Beschreiben dynamischer Abläufe bei der Zugriffskontrolle
21.11.2006 (ES)	Sequenzdiagramme zur Beschreibung von dynamischen Abläufen (studentische Lehrperson)
23.11.2006 (ES)	schriftliche Übung: Test Akzeptanzbefragung

Die Erprobung umfasste 12 Unterrichtsstunden zu drei Stunden pro Woche. Zusätzlich wurden ein Abschlusstest und eine Akzeptanzbefragung mit den Schülern durchgeführt. Tabelle 6.3 zeigt die Übersicht über die Unterrichtsstunden. Die Unterrichtsentwürfe und -materialien auf der Webseite zum Dissertationsprojekt sind entsprechend dieser Übersicht geordnet (Anhang A.3). Der Unterricht wurde bis auf eine Doppelstunde gegen Ende der Erkundung, die von einem Lehramtsstudierenden im Rahmen der schulpraktischen Studien unterrichtet wurde, von der Forschungsperson gehalten und vom Informatiklehrer sowie einem weiteren Mitarbeiter des Lehrstuhls für Didaktik der Informatik und E-Learning der Universität Siegen gezielt beobachtet, um erste Erkenntnisse über typische Bearbeitungsstrategien und Fehler zu gewinnen. Zur Methodenkritik siehe Abschnitt 1.2.2. Es wurde

angenommen, dass die Schüler keine Vorkenntnisse zur systematischen Erkundung des nach außen sichtbaren Verhaltens eines Informatiksystems aufweisen. Die Unterrichtsstunden wurden sowohl mit dem Lehrer als auch mit weiteren Mitarbeitern des Instituts für Didaktik der Informatik und E-Learning vor- und nachbereitet. Der Informatiklehrer stand abschließend für ein Leitfaden-Interview zur Verfügung. Die Erprobung wurde unterbrochen durch eine Woche, in der der Informatiklehrer (aufgrund einer Vortragsreise des Autors) unabhängig vom Unterrichtsprojekt unterrichtete. Darüber hinaus gab es eine Verzögerung durch einen beweglichen Ferientag direkt im Anschluss, so dass insgesamt eine anderthalbwöchige Pause entstand.

6.3.3 Unterrichtsmethodik und technischer Rahmen

Bei der Gestaltung des Unterrichtsprojekts wurde besonders darauf geachtet, dass die Schülertätigkeiten im Mittelpunkt standen. Somit wurde möglichst wenig in Lehrervorträgen umgesetzt, stattdessen war das Unterrichtsprojekt schülerzentriert angelegt. Lehrervorträge nahmen nur wenige Minuten pro Unterrichtsstunde ein. Partnerarbeit war die vorherrschende Arbeitsform, sowohl an den Rechnern als auch bei schriftlichen Arbeitsaufträgen. Die räumliche Gestaltung des Informatiklabors unterstützte die Trennung von Theorie und Informatikexperimenten mit Rechnern durch den zentralen Kommunikationsbereich und die dezentral an den Seiten aufgestellten Rechner. Das Schulintranet besaß eine Verbindung zum Internet und eine von allen gemeinsam nutzbare Speicherpartition des Dateiservers ermöglichte den problemlosen Austausch größerer Dateien. Der vordere Demonstrationsbereich des Lehrers war mit Whiteboard, Rechner und Projektor zur Präsentation ausgestattet.

6.4 Beschreibung und Durchführung der Erprobung

6.4.1 Lernphasen und Problemstellen im Unterrichtsprojekt

Startpunkt zur Motivation war eine Übersicht über prominente Fehlerquellen und Pannen einiger Informatiksysteme, z. B. reagierte der Bordcomputer eines Flugzeugs auf ein Leck in einem Kerosintank damit, dass er Kerosin aus dem zweiten Tank in den leeren Tank pumpte, um das Flugzeug zu stabilisieren. Das Flugzeug musste landen, da nach kurzer Zeit beide Tanks leer waren. Um Fehlvorstellungen analysieren zu können, wurden sämtliche in den Unterrichtsstunden handschriftlich oder digital erstellten Schülerlösungen eingesammelt. Nach Abschluss des Unterrichtsprojekts wurde ein Test in Form einer schriftlichen Übung durchgeführt. Um dessen Wichtigkeit für die Schüler zu unterstreichen, sollten die Ergebnisse in die

Gesamtnote einfließen. Aus diesem Grund wurde die schriftliche Übung nicht anonymisiert durchgeführt.

Logische Verknüpfung des Unterrichtsprojekts war das Wechselspiel von Erkundung des nach außen sichtbaren Verhaltens eines Programms und der Betrachtung und Erstellung von Modellen der inneren Struktur. Begonnen wurde mit einem Programm einer Arztpraxis mit einer Warteschlange, das bereits im vorhergehenden Informatikunterricht von den Schülern mittels Klassendiagramm modelliert und in der Programmiersprache Objekt-Pascal mit Delphi 6 geschrieben worden war. Es wurde erweitert um das Iteratormuster ($S_{A,1}$). Erst für das Thema Zugriffskontrolle ($S_{A,2}$) wurde ein neues Programm untersucht, das die Vergabe von Zugriffsrechten realisierte (Abbildung 6.2). Dies geschah jedoch immer mit Rückbezügen zum „Arztinformationssystem". Spiralförmig wurden somit die Lernphasen wiederholt und auf die Themen Datenstruktur Schlange, Iteration und Zugriffskontrolle angewendet (Tabelle 6.1). Es zeigte sich während der Unterrichtsfolge, dass Vererbung eine Fehlerquelle darstellte (siehe Abschnitt 6.5.1).

6.4.2 Datenstruktur Schlange und Iteratormuster

Zu Beginn des Unterrichtsprojekts wurde das nach außen sichtbare Verhalten des Arztinformationssystems systematisch erkundet, und anschließend schlossen die Schüler auf zugrunde liegende informatische Konzepte. Hierfür wurde ein Experimentiervorgang beschrieben, der aus acht Schritten (Abschnitt 5.5.3; analog zur Schrittfolge in Abbildung 6.2) besteht und die sechs kognitiven Prozesse umfasst (vgl. Stechert 2007b), die in der überarbeiteten Lernzieltaxonomie nach Bloom angegeben sind (Anderson und Krathwohl 2001). Er bestand aus dem Aufstellen von Hypothesen über das Systemverhalten, dem Beschreiben und Dokumentieren des Experimentierablaufs, den Rückschlüssen aus dem tatsächlichen Verhalten auf informatische Konzepte, dem Identifizieren von Sonderfällen wie der leeren Schlange, dem Überprüfen der Umsetzung der Sonderfälle im Informatiksystem sowie Auswirkungen des Einsatzes des Informatiksystems. Insbesondere das Erkennen von möglichen Konzepten fiel anfangs bei Schlange und Iterator schwer ($S_{A,1}$), obwohl das Listenkonzept bekannt war. Mit dem Hinzufügen des Iterators wurde die Schlange in eine Liste überführt, da nun das Entfernen von Elementen an beliebiger Stelle möglich wurde – beispielsweise bei einem Notfall oder wenn ein Patient des Wartens überdrüssig wird und die Arztpraxis verlässt. Bei der Auswertung der ersten Durchführung des Informatikexperimentes wurde nur bei etwas mehr als der Hälfte der abgegebenen Lösungen das Konzept der Schlange wieder erkannt. Der Sonderfall, dass die Schlange leer, bzw. die Frage, ob die Schlange voll sein kann, wurde im Plenum diskutiert. Die Demonstration der Informatikexperimente, sei es durch Lehrperson oder Schüler, unterstützte das Unterrichtsprojekt zu Informatiksystemen und Kompetenzentwicklung stark, da die neuen systematischen Vorgehensweisen zur Analyse erläutert werden konnten. Danach galt es, die innere

Struktur des Programms anhand des Quellcodes und eines unvollständigen Klassendiagramms zu untersuchen ($S_{B,1}$). Dafür wurde die Vorgehensweise zur Erkundung der inneren Struktur eingesetzt (Abschnitt 5.5.4). Die Umsetzung der identifizierten informatischen Konzepte in eine Klassenstruktur wurde thematisiert. Um dynamische Abläufe in der inneren Struktur beschreiben zu können, wurden während der ersten Woche in einer Partnerarbeit Objektdiagramme zu verschiedenen Zeitpunkten des Programmablaufs erstellt. Relativ überraschend war für die Schüler die Einsicht, dass der Iterator eine Variante der Zugriffskontrolle ist ($S_{B,1}$). Die Analyse des Systemverhaltens führte dazu, in der inneren Struktur Fehler schneller finden und beheben zu können ($S_{C,1}$).

6.4.3 Zugriffskontrolle

Bei der Erkundung der Zugriffskontrolle ($S_{A,2}$) war den Schülern das Vorgehen zur systematischen Erkundung des Systemverhaltens bekannt. Überraschend war für die Schüler bei der inneren Struktur der auf Vererbung basierende Ansatz zur Überprüfung der Zugriffsrechte (siehe Klassendiagramm in Abbildung 6.3). Da das Verstehen von dynamischen Aspekten essenziell für Kompetenzentwicklung mit Informatiksystemen ist, wurde zur Beschreibung der Abläufe bei der Zugriffskontrolle ($S_{B,2}$) das Sequenzdiagramm genutzt. Einstiegsaufgabe war die Erstellung eines so genannten Interaktionsdiagramms (als Vorstufe des Sequenzdiagramms), anschließend wurden einfache Sequenzdiagramme mit Elementen der Lernsoftware Pattern Park erstellt (Abschnitt 6.4.4). Die Einführung der Zugriffskontrolle mittels Proxy ermöglichte den Schülern, das Prinzip der Zugriffrechte zu verstehen und selbst für bestimmte Situationen zu gestalten. So konnten neben Administrator, Benutzer und Gast weitere Anwender (-gruppen) mit bestimmten Rechten versehen werden ($S_{C,2}$). Abbildung 6.2 zeigt eine Aufgabe aus der ersten Unterrichtserprobung und die Benutzungsoberfläche der Software, die auf dem Entwurfsmuster Proxy basiert. Es lassen sich die Rollen Administrator, Benutzer und Gast identifizieren. Falls ein Proxy eingeschaltet ist, unterscheiden sich die Schreib- und Leserechte. Bei der Erkundung des Systemverhaltens müssen die Schüler Hypothesen darüber bilden, ob Zugriffskontrolle vorliegt und wie diese umgesetzt ist, um Fehlvorstellungen zu überwinden.

Da die Schüler Vorkenntnisse zum Programmieren hatten, verfügten sie implizit über eine Vorgehensweise zur Analyse von Quelltext. Sie mussten die in Abbildung 6.2 gegebene Vorgehensweise zur Erkundung der lernförderlichen Software zur Zugriffskontrolle durchführen (→ Kriterium 6: Lebensweltbezug; S. 173). Eine erwartete Fehlvorstellung zur Zugriffskontrolle war, dass ein Objekt den Zugriff auf sich selbst kontrolliert. Meist wird jedoch ein Stellvertreterobjekt mit der gleichen Schnittstelle wie der des Originalobjekts eingesetzt (Abschnitt 5.4.4). Daher wurde die Benutzungsoberfläche der Software so gestaltet, dass die Schüler leichter erkannten, dass ein Stellvertreterobjekt eingesetzt wird (Abbildung 6.2). Im

1. Wie lautet der Name des Systems? (Erinnern; Faktenwissen)
2. Beschreiben Sie die Benutzungsoberfläche. (Verstehen; Faktenwissen)
3. Welche Funktionalitäten vermuten Sie unter Berücksichtigung der vorherigen Beobachtungen? (Verstehen; Begriffliches Wissen)
4. Welche Beziehungen bestehen zwischen den Elementen? (Verstehen; Faktenwissen)
5. Welche informatischen Konzepte wurden eingesetzt, um die Funktionalität zu erreichen? (Analysieren; Begriffliches Wissen)
6. Analysieren Sie das Programm, indem Sie Sonderfälle ermitteln und mit ihnen experimentieren. (Anwenden; Begriffliches Wissen)
7. Werten Sie Fehler und unerwartetes Verhalten aus – auch mit Blick auf die Sonderfälle der Konzepte. (Bewerten; Begriffliches Wissen)
8. Hypothetischer Einsatz des Systems – Wie würde sich das Programm in anderen (realen, komplexeren) Situationen verhalten? ((Er)schaffen; Faktenwissen)

Abbildung 6.2: Benutzungsoberfläche und Aufgabe zur Erkundung der lernförderlichen Software zur Zugriffskontrolle mit kognitivem Prozess und Wissensart gemäß Lernzieltaxonomie (Anderson und Krathwohl 2001)

weiteren Verlauf der systematischen Erkundung des Systemverhaltens waren die Schüler nicht in der Lage, die Frage, wie ein Stellvertreter das selbe Verhalten wie das Original haben kann, hypothetisch zu beantworten. Erst bei der Analyse des Quelltextes erkannten die Schüler, dass Zugriffskontrolle mit Vererbung (→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik; S. 172) realisiert wurde, indem Stellvertreter und Original von der gleichen (abstrakten) Klasse erben. Dadurch ergibt sich für beide die gleiche Schnittstelle bei durchaus unterschiedlicher Spezialisierung. Eine gleichnamige Operation „lesen()" bzw. „schreiben()" wird beim Original ein bestimmtes Verhalten auslösen, während beim Stellvertreter nur die Prüfung der Zugriffsrechte und ggf. die Weiterleitung an das Original durchgeführt wird. Zur Dokumentation der Erkundungsschritte sollten die Schüler ein

Klassendiagramm erstellen des Programms erstellen und so die Struktur des Proxymusters selbst erarbeiten. In diesem Zusammenhang ist es wichtig, Variationen des Entwurfsmusters (→ Kriterium 5: Komplexität; S. 173), z. B. die Erweiterung um die Benutzungsoberfläche, explizit im Unterricht aufzugreifen. So fragten Schüler bei der Diskussion des Klassendiagramms des Proxymusters, wie der Zugriff auf das Muster umgesetzt werden könne. Dies ist durch die Benutzungsoberfläche, oder allgemeiner, durch einen Klienten möglich, der in das Klassendiagramm integrierbar ist (vgl. Abbildung 5.8). Die Dokumentation des Systemverhaltens durch Formalisierung von Abläufen stellte sich als weitere Hürde dar.

6.4.4 Rolle der Lernsoftware

Durch Modularisierung bei der Entwicklung der Lernsoftware Pattern Park konnten für die unterrichtliche Erprobung schon vorab Animationen zu Iteration ($S_{A,1}$) und Zugriffskontrolle ($S_{A,2}$) sowie Übungsmodule mit Sequenzdiagrammen ($S_{B,2}$) eingesetzt werden. Letzteres geschah bei der Nutzung eines Proxy zur Realisierung von Zugriffsschutz. Insbesondere das Übungsmodul griff eine Lebensweltsituation handlungsorientiert auf und schlug so eine Brücke zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur eines Informatiksystems einerseits und zwischen der statischen und dynamischen Strukturbeschreibung andererseits. Die Beobachtung der Bearbeitungsweisen hat ergeben, dass gerade diese Aufgaben, die das nach außen sichtbare Verhalten von Informatiksystemen und die Analyse der inneren Struktur verknüpfen, Schlüssel zur Kompetenzentwicklung waren, denn hier wurden unterschiedliche Sichten auf das Informatiksystem kombiniert. Außerdem zeigte sich während des Unterrichtsprojekts, dass das frühzeitige Vorführen der Animation den Schülern nicht nur die Problemstellung klarer werden ließ, sondern gleichzeitig durch den Bezug zum Entwurfsmuster, einem „Lösungsmuster", eine konkretere Vorstellung von Lösungsansätzen unterstützte.

Bezüglich des oben bereits ausgeführten Beispiels der Zugriffskontrolle wurde diese in der Unterrichtserprobung mittels Stellvertreter (Proxyobjekt) eingeführt. Die Schüler kennen Stellvertreter aus ihren alltäglichen Erfahrung, z. B. Klassensprecher. Die Unterrichtseinheit begann mit der Animation, in der eine Geldkarte als Stellvertreter für Geld eingesetzt wurde (nach außen sichtbares Verhalten). Dadurch werden die Schüler sensibilisiert für weitere Arten der Zugriffskontrolle mittels Proxy. Beispielsweise ist die Funktionalität des Schutzproxys mit Überprüfung von Rechten anhand eines Passwortes zu nennen, aber auch darüber hinaus gehende Aktionen wie Datenspeicherung wann und wie oft die Karte genutzt wird und ggf. Weiterleitung der Daten an Dritte.

Sensibilisiert durch die Animation wurde in der Unterrichtserprobung nun das – vom Pattern Park unabhängige – Programm zu Zugriffsrechten vorgestellt. Darin gab es die unterschiedlichen Rollen Administrator, Benutzer und Gast (Abbildung

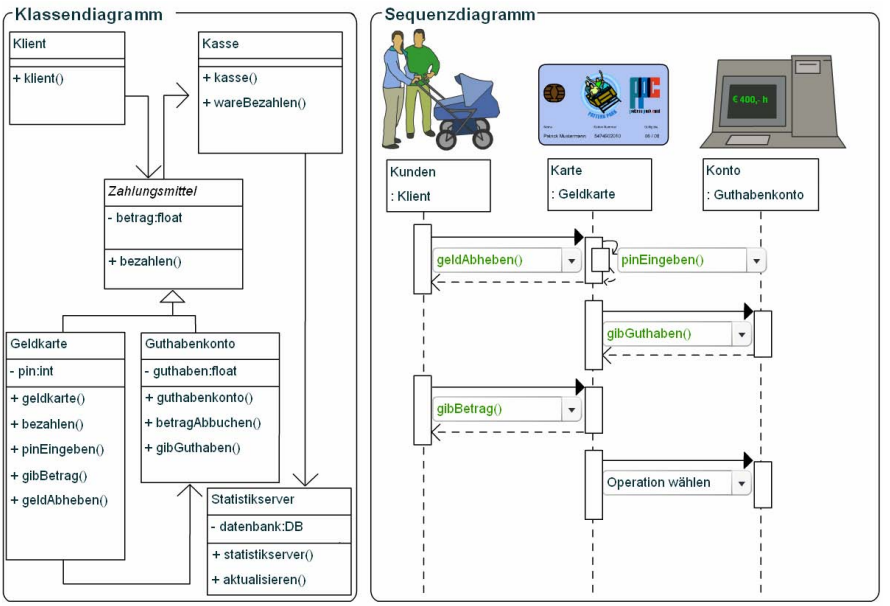


Abbildung 6.3: Klassen- und Sequenzdiagramm zum Zugriffsschutz aus der Lernsoftware Pattern Park

6.2). Bei eingeschaltetem Zugriffsschutz mit Proxy konnte nur der Administrator auf eine Datei sowohl lesend als auch schreibend zugreifen, der Benutzer nur lesend, der Gast durfte auf die Datei nicht zugreifen. Die Schüler konnten mit dem Programm insofern experimentieren, als dass das Proxyobjekt aktiviert und deaktiviert werden konnte und sich das Verhalten des Programms daraufhin änderte (S_A: nach außen sichtbares Verhalten).

Danach sollten die Schüler an die Beschreibung von dynamischen Aspekten, speziell den Ablauf bei der Zugriffskontrolle, herangeführt werden. Dazu wurde ein Modulelement aus Pattern Park mit zwei Aufgaben ausgewählt, dass die statische Sicht auf ein Klassendiagramm und Bilder zur lebensweltlichen Illustration der Situation anbietet (S_A und S_B: Kombination des nach außen sichtbaren Verhaltens mit der inneren Struktur) aus der ein Sequenzdiagramm zur Beschreibung der in den Aufgaben geschilderten Abläufe erarbeitet werden soll. Das gender-neutrale Szenario mit Geld und Geldkarte war den Schülern bereits durch die Animation bekannt.

Um die Sequenzdiagramme der Situationen effizient und ohne Versuch-Irrtum-Strategie zu erstellen, ist der ständige Rückbezug auf die durch das Klassendia-

gramm formal dargestellten Operationen unverzichtbar. Abbildung 6.3 zeigt das Klassendiagramm des Proxymusters zur Realisierung der Zugriffskontrolle. Anhand des Klassendiagramms mussten die Schüler zwei korrekte Sequenzdiagramme zweier typischer Abläufe in dem vorgegebenen Szenario kreieren. Die beiden Aufgaben haben unterschiedliche Schwierigkeitsgrade: In der ersten Aufgabe zum Geldabheben an einem Geldautomaten sind nur drei Objekte involviert, im zweiten Szenario, eine Einkaufssituation, bereits fünf. Darauf aufbauend kehrten die Schüler zum Programm mit Administrator, Benutzer und Gast zurück, führten ein Experiment zur Erkundung der inneren Struktur durch (S_B: innere Struktur) und fanden diese in Form des nun bereits bekannten Proxymusters.

6.4.5 Exkurs: Beitrag der eingesetzten informatischen Darstellungsformen

Um unterschiedliche Sichten auf Informatiksysteme einzunehmen, sind geeignete informatische Darstellungsformen für den Unterricht einzusetzen (vgl. Hubwieser und Broy 1997a, S. 46). In Abschnitt 3.2.3 wurde betont, dass zur Analyse der inneren Struktur von Informatiksystemen nicht nur die Komponenten, sondern auch die intern ablaufenden Prozesse betrachtet werden müssen, d. h. statische und dynamische Sichten sind notwendig (Brinda 2004a), (Hadar und Hazzan 2004, S. 150). Entwurfsmuster umfassen sowohl Strukturbilder also auch Beschreibungen von Prozesses. Modelle werden meist als Abstraktion eines realen oder geplanten Systems angesehen (Hubwieser 2007a, S. 86), deren Anwendungsfälle und hypothetischen Abläufe dem Modell die Semantik geben.

Als Modellierungssprache wurde für die erste Erprobung die UML mit Klassendiagramm und Sequenzdiagramm ausgewählt. Dazu ist anzumerken, dass Klassendiagramme als Vorkenntnisse der Schüler vorhanden waren. Das Sequenzdiagramm wurde als dynamisches Modell zur Ablaufbeschreibung ausgewählt:

> „Sequenzdiagramme können dazu verwendet werden, um Szenarien und (einfache) Anwendungsfälle grafisch darzustellen. Sie beschreiben nicht nur die Ausführungsreihenfolge der Operationen, sondern auch die dafür zuständigen Klassen bzw. deren Objekte. Sie stellen somit in der UML eine wichtige **Verbindung zwischen der Funktionalität und dem Klassendiagramm** dar" (Balzert 2005, S. 119; Hervorh. durch den Autor).

Objektdiagramme wurden einmal exemplarisch eingesetzt, aber wegen der offensichtlichen Objekt-Klasse-Fehlvorstellung bei den Schülern wurde im weiteren Verlauf auf sie verzichtet.

Begründungen für die Auswahl liefern Hadar und Hazzan (2004) sowie Karahasanovic und Holmboe (2006). Hadar und Hazzan (2004) fragten eine Gruppe von 42 Studierenden in einem Fragebogen, neun unterschiedliche Diagrammarten einerseits auf ihre Wichtigkeit im Softwareentwicklungsprozess und andererseits auf ihre

Nützlichkeit zum Verstehen eines unbekannten Softwaresystems einzuschätzen. Die Unterschiede zwischen den Diagrammarten waren bezüglich des Programmverstehens im Gegensatz zur Entwicklung nicht sehr groß:

> „More specifically, the order in which the four diagram types were ranked (from high to low importance) was: class, sequence, collaboration and finally, state. The differences, however, in the level of importance were smaller in the comprehension section of the questionnaire compared to the development section" (vgl. Hadar und Hazzan 2004, S. 151f).

Obwohl explizit zwischen der Softwareentwicklung und dem Verstehen eines existierenden Programms differenziert wird, können die Ergebnisse nur einen Anhaltspunkt für Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II liefern, da Studierende befragt wurden. Karahasanovic und Holmboe (2006) üben Kritik an der Studie von Hadar und Hazzan, da in ihr keine Aussage über die Rolle von UML als Hilfsmittel für das Lernen von OOM getroffen wird, denn jede Gruppe nutzte UML. In ihrer dreiwöchigen Studie in der Sekundarstufe II mussten fünf 18-jährige Schüler ohne Modellierungsvorkenntnisse vier Aufgaben bearbeiten: ein Restaurant objektorientiert modellieren sowie Klassendiagramme eines Telefonbuchs, eines Zeichenprogramms und eines Schlüsselkartenlesegeräts erstellen. Implementierung war nicht erforderlich. Die fünf Schüler wurden in zwei Gruppen aufgeteilt. Bei der Gruppe, die aktiv die UML nutzte, sahen Karahasanovic und Holmboe einen Hinweis auf verbessertes Verstehen von Konzepten der Objektorientierung. Nach der Fallstudie leiten sie aus den Ergebnissen ab, dass Lernende Schwierigkeiten haben, den Problembereich (Lebenswelt) und den Lösungsbereich aufeinander abzubilden:

> „**Understanding of a large system**
> [. . .]
> The students in our experiment appeared to have problems handling the mapping between the real world (problem domain) and the solution domain. Seemingly, they had problems understanding and analyzing the complex system they were faced with" (Karahasanovic und Holmboe 2006, S. 57; Hervorh. im Original).

In einer anderen Aufgabe, in der die Schüler die Aufgabe des Systems, ein Schlüsselkartenlesegerät, kannten, waren ihre Leistungen besser:

> „[. . .] they seemed to have fair understanding of the role of the computer system and thus of the implicit use-cases that it should accomodate" (Karahasanovic und Holmboe 2006, S. 58).

Allerdings kann auch diese Studie nur als Indiz für den positiven Effekt von UML dienen, da eine zu kleine Grundgesamtheit angesetzt wurde.

Appelrath et al. (2002) empfehlen für die Hochschule die UML zur Dokumentation, da sie Personen unterstützt, die nicht selbst am Entwicklungsprozess beteiligt waren (Appelrath et al. 2002, S. 127). Für Kompetenzentwicklung mit Informatiksystemen sind die beiden Studien insofern hilfreich, als dass sie es nahe legen,

die UML als Modellierungssprache zu nutzen, falls OOM eingesetzt wird. Außerdem sollte der Zweck für Kompetenzentwicklung mit Informatiksystemen explizit thematisiert werden, denn dann können die Schüler typische Anwendungsfälle des Systems beschreiben. Dies wird in der in Abschnitt 5.4.1 angegebenen Klassifikation der Entwurfsmuster vor allem durch das → Kriterium 2: Zweck und Einsatzgebiet (S. 172) berücksichtigt.

In der ersten Erprobung zeigten die beschriebenen Unterrichtsbeobachtungen und die Lernerfolgskontrolle (Abschnitt 6.5.1), dass die eingesetzten Diagramme den Lehr-Lernprozess unterstützen konnten.

6.5 Evaluation

6.5.1 Lernerfolgskontrolle

Wichtiges Ziel der Untersuchung war zu beantworten, ob die Schüler die intendierten Lernziele erreichen. Zur Beantwortung sind zwei Kriterien zu erfüllen. Die Schüler müssen die im Unterricht gestellten Aufgaben erfolgreich bearbeiten und die schriftlichen und praktischen Aufgaben selbständig lösen.

Für die schriftliche Übung wurden Multiple-Choice-Aufgaben und Lückentexte gewählt. Als Stimuli für die Testitems dienten u. a. Klassendiagramme. Inhaltlich waren die Schwerpunkte der schriftlichen Übung:

- nach außen sichtbares Verhalten von Informatiksystemen,
- innere Struktur von Informatiksystemen,
- Zusammenhang zwischen nach außen sichtbarem Verhalten und innerer Struktur,
- Zugriffskontrolle mittels Entwurfsmuster Proxy,
- Iteration mittels Entwurfsmuster Iterator.

Für die schriftliche Übung wurden zwei Varianten A und B eingesetzt, so dass ein Thema bei Bedarf in der einen Variante mit Lückentext, in der anderen mit Multiple-Choice-Aufgabe geprüft werden konnte. Die Bearbeitungszeit betrug 30 Minuten.

Insgesamt wurden 21 Tests bearbeitet (Variante A: 11; Variante B: 10). Die Ergebnisse sind überdurchschnittlich gut ausgefallen (gerundete Werte), obwohl die Schüler in der Akzeptanzbefragung den Stoffumfang als sehr groß eingeschätzt haben (vgl. Abschnitt 6.5.2). Im Folgenden werden die Kennzahlen in Prozent bezogen auf die maximal mögliche Punktzahl in der Lernerfolgskontrolle angegeben: Minimalwert: 53%, Maximalwert 100%, Mittelwert: 75 %, Median: 77%. Die häufigsten

Werte (Modi) waren 63%, 67% und 77% mit je drei Ausprägungen. Der Mittelwert bei den Mädchen lag mit 72% leicht unter dem Gesamtdurchschnitt, ist aber aufgrund der geringen Anzahl von vier Personen nicht aussagekräftig.

Bei den Fragen zum nach außen sichtbaren Verhalten hatten die Schüler Schwierigkeiten in Bezug auf eine systematische Erkundung. Hieraus könnte der Schluss gezogen werden, dass es kaum Vorkenntnisse bezüglich eines solchen Vorgehens gab. Im scheinbaren Widerspruch dazu steht die Aussage fast aller Schüler, dass das Erkunden der inneren Struktur schwieriger als das Erkunden des nach außen sichtbaren Verhaltens sei. Es haben alle Schüler angegeben, dass zum Verstehen des nach außen sichtbaren Verhaltens eines Informatiksystems ein systematisches Experimentieren notwendig sei und dass unerwartetes Verhalten eines Informatiksystems Rückschlüsse auf dessen innere Struktur liefere.

Mit der Vorgehensweise zum Erkunden der inneren Struktur hingegen waren im Test weniger Schwierigkeiten verbunden. Dies mag an den Vorkenntnissen zur Erstellung von Klassendiagrammen und auch zu Delphi-Projekten liegen. Es zeigte sich eine teilweise Überforderung der Lernenden bei der Verbindung des nach außen sichtbaren Verhaltens mit der inneren Struktur. Angebracht gewesen wäre hier eine Aufteilung in kleinere Teilaufgaben.

Die Hauptschwierigkeiten lagen bei der Beschreibung der inneren Struktur der Datenstruktur Schlange und des Iterator, da oftmals falsche Assoziationen angegeben wurden. Des Weiteren ist einigen Schülern die Unterscheidung zwischen Schlange und Iterator nicht gelungen. Außerdem wurden bei dem Transfer des Proxy auf eine neue Situation Fehler gemacht, obwohl die zum Unterricht ähnliche Aufgabe zur Zugriffskontrolle gelöst wurde. In weiteren Unterrichtsprojekten sollten deshalb mehr Transferaufgaben hinsichtlich ähnlicher Situationen und Strukturen zu bearbeiten sein. In der zweiten Unterrichtserprobung wird deshalb die Zugriffskontrolle zusätzlich in einer Arztpraxissoftware umgesetzt (Abschnitt 8.3).

Die Ergebnisse aus der Analyse der Explizierung und Reflexion der jeweiligen Schüler-Vorgehensweisen in der Lerngruppe lassen sich jedoch nicht eindeutig Fehlvorstellungen zu Informatiksystemen zuordnen. Insbesondere ist zu vermuten, dass typische Fehlvorstellungen der Objektorientierung sehr dominant sind und Schülervorstellungen von Informatiksystemen überlagern. Zur Realisierung der Zugriffskontrolle wurden beispielsweise oft falsche Vererbungsbeziehungen verwendet (vgl. Abschnitt 7.4).

6.5.2 Schriftliche Akzeptanzbefragung der Schüler

Im Nachgang des Unterrichtsprojekts wurden die Lernenden des Kurses schriftlich befragt. Der Akzeptanzfragebogen orientiert sich an dem von Brinda eingesetzten Fragebogen (Brinda 2004a, S. 222). Er wurde gemeinsam mit Freischlad adaptiert

und gleicht – bis auf wenige themenspezifische Fragen – dem zur Evaluation von Unterricht zum „Didaktischen System Internetworking" verwendeten Akzeptanzfragebogen (Freischlad 2007). Die Fragen und Ergebnisse befinden sich im Anhang A.1. Die Befragung der Lernenden zum Unterricht ermöglicht eine Einschätzung zur Akzeptanz und Rückschlüsse auf den Lernerfolg (Schubert et al. 2007).

Die ausgefüllten Fragebögen wurden von den Lernenden anonymisiert. Die Fragebögen enthielten 36 Aussagen aus den vier Bereichen „Informatikunterricht allgemein", „Schwierigkeitsgrad und Lernstoff", „Befragung zum Unterrichtsthema" und „Einschätzung des Lernfortschritts in den einzelnen Lernbereichen", zu denen die Lernenden durch Ankreuzen ihre Zustimmung oder Ablehnung bei (vorwiegend) vier vorgegebenen Skalenwerten äußern sollten zuzüglich der Möglichkeit sich zu enthalten. Ein Ziel war es, die Motivation der Lernenden zu analysieren. Gefragt wurde deshalb auch nach außerschulischer Beschäftigung mit dem Thema, weitergehendem Interesse sowie Einschätzung des persönlichen Nutzens. Weitere ausgewählte Einflussgrößen bzw. Einstellungen der Lernenden waren beispielsweise Geschlecht und Anzahl versäumter Unterrichtstermine, um die Antworten interpretieren zu können. In der Akzeptanzbefragung zeigte sich, dass von 23 Schülern 16 (70%) den Schwierigkeitsgrad des Unterrichtsprojekts als angemessen und nur vier (17%) ihn als hoch empfanden. Der Stoffumfang wurde von 16 (70%) als angemessen und von sechs (26%) als viel wahrgenommen. Dieser Aspekt wird auch durch zehn (43%) schriftliche Anmerkungen gestützt, welche die geringe zur Verfügung stehende Zeit für das Bearbeiten einiger Aufgaben beanstanden.

Zu dem eigenen Lernfortschritt bei den inhaltlichen Themen der Analyse, der inneren Abläufe und des Aufbaus von Informatiksystemen geben jeweils 16 (70%) oder mehr Schüler an, einiges bzw. viel dazugelernt zu haben. Die Konzentration auf die Aufgaben fiel 15 Schülern (65%) leicht. Der Aussage, dass Hilfsmittel wie Arbeitsblätter und (Lern-) Software ausreichend und in guter Qualität vorhanden waren, wurde von 17 (74%) Schülern etwas, bzw. voll zugestimmt. Dies ist sicherlich auch auf den Einsatz ausgewählter Animationen und Übungen aus der Lernsoftware Pattern Park zurückzuführen. Durchaus mit den auf den Basiskompetenzen liegenden Zielen des Unterrichtsmodell im Einklang steht, dass fünf (22%) Schüler anzeigten nichts und elf (47%) nicht viel zur Programmierung dazugelernt zu haben. Dass elf (48%) Schüler angaben eher keinen bzw. drei (13%) bestätigten, keinen Spaß daran zu haben, ihr Verständnis für das Thema zu vertiefen, offenbart ein Motivationsproblem, dem in folgenden Erprobungen durch stärkeren Bezug zu Alltagsproblemen und anderen Fächern begegnet werden muss. Die Motivation über prominente Beispiele für Fehlfunktionen von Informatiksystemen und den Bezug zur Lebenswelt über fundamentale Ideen wie Zugriffskontrolle war sicherlich ein Schritt in die richtige Richtung, aber nicht ausreichend. Sequenzdiagramme als Darstellungsform wurden von den Schülern positiv aufgenommen, da 19 (82%) angaben, über Abläufe in Informatiksystemen etwas oder viel dazugelernt zu haben.

Abschließend ist zu erwähnen, dass 16 (70%) Schüler angaben, für sich etwas dazugelernt zu haben, und dass die Mehrheit von 19 (82%) Schülern der Aussage, dass die Inhalte des Unterrichtsprojekts für den Umgang mit Informatiksystemen sehr nützlich sind, zustimmen.

6.5.3 Leitfaden Interview mit der Informatiklehrperson

Das Interview mit der Lehrperson, die die Lerngruppe in der Kooperationsschule betreut, erlaubt eine Einschätzung der Angemessenheit bezüglich Stoffumfang und Schwierigkeitsgrad. Zudem kann die Lehrperson die Ergebnisse der Intervention zu anderen Lehr-Lern-Situationen in Bezug setzen und vergleichend bewerten (Schubert et al. 2007).

Durch Befragungstechniken ermittelt man die subjektive Sichtweise von Akteuren über vergangene Ereignisse, Zukunftspläne etc. (Bortz und Döring 2002). Insbesondere qualitative Befragungen arbeiten mit offenen Fragen und berücksichtigen auch die Eindrücke und Deutungen des Interviewers, hier des Forschers, als Informationsquellen.

> „Charakteristisch für diese Befragungsform ist ein Interview-Leitfaden, der dem Interviewer mehr oder weniger verbindlich die Art und die Inhalte des Gesprächs vorschreibt" (Bortz und Döring 2002, S. 239).

Das Interview zur Unterrichtserprobung, für das der Informatiklehrer abschließend zur Verfügung stand, bestätigte beispielsweise, dass der Schwierigkeitsgrad des Unterrichtsprojekts angemessen sei. Auch die Konzeption zur Kompetenzentwicklung mit Informatiksystemen mit dem Wechsel der unterschiedlichen Sichtweisen auf Informatiksysteme wurde befürwortet. Ein zusätzliches Ergebnis war, dass weitere Kooperationsformen, insbesondere Gruppenarbeit, verstärkt eingesetzt werden sollten. Dies wird in der zweiten Unterrichtserprobung aufgegriffen (Kapitel 8) und in einigen vom Autor betreuten Seminararbeiten weiter untersucht (Gerding 2008), (Graf 2008), (Dittich 2008). Wesentlich größeren Einfluss als das Interview zum Abschluss der Erprobung hatten jedoch die Rückmeldungen und die konstruktive Kritik des Informatiklehrers nach jeder Unterrichtsstunde, durch die noch während der Erprobung Anpassungen des Konzeptes erfolgten. Ein weiterer Informatiklehrer der Schule, über den die Kooperation initiiert wurde und der bei der Planung und den Vorgesprächen zu beiden in der vorliegende Arbeit genannten Erprobungen aktiv mitwirkte, resümiert die Situation zu Beginn der ersten Erprobung folgendermaßen:

> „Trotz der intensiven wissenschaftlichen und didaktischen Vorbereitung [...] war zunächst eine Kluft zwischen universitärer Sprache und Vorgehensweise sowie schulischen Gewohnheiten zu überwinden" (Ganea und Koch 2008, S. 117).

Das hohe Abstraktionsniveau und der hohe theoretische Anteil wurden deshalb zugunsten einer Stärkung der Handlungsorientierung verringert. Insgesamt führte die

Bereitschaft, einen Kompromiss zwischen Schulpraxis und Wissenschaft zu erzielen, zu einem positiven Fazit der beteiligten Lehrer (Ganea und Koch 2008, S. 117), wenngleich einige der oben genannten Kritikpunkte im weiteren Verlauf der ersten Erprobung in geminderter Form weiter bestanden.

6.6 Zusammenfassung und Diskussion der Ergebnisse der ersten Unterrichtserprobung

6.6.1 Zusammenfassung der ersten Unterrichtserprobung

Die Beobachtungen im unterrichtlichen Geschehen und die Ergebnisse der selbständig gelösten schriftlichen und praktischen Aufgaben lassen den Rückschluss zu, dass es prinzipiell machbar ist, das theoretische Unterrichtsmodell in konkreten Unterricht zu transferieren. Auch die Akzeptanzbefragung der Schüler lieferte positive Rückmeldungen.

Untersuchungsschwerpunkte waren

1. die Tragfähigkeit und Auswirkung der Dreiteilung in Systemverhalten, innere Struktur und Implementierungsaspekte,
2. die Eignung ausgewählter Entwurfsmuster zur Repräsentation und Vernetzung fundamentaler Ideen der Informatik in kleinen Programmen,
3. die exemplarische Erprobung von Modulen der Lernsoftware Pattern Park,
4. die Lehr-Lernmethodik mit systematischen Erkundungen von Informatiksystemen und die Akzeptanz der Schüler.

(1) Die Dreiteilung der Basiskompetenzen in Aufgaben zu Systemverhalten, innerer Struktur und Implementierungsaspekten hat sich als tragfähig erwiesen. Eine konzeptionelle Schwierigkeit war jedoch, dass die Aufteilung zwar automatisch einen Sichtenwechsel, nicht aber die Kombination von Sichten impliziert. Deshalb ist eine Verfeinerung der Strukturierung der Basiskompetenzen notwendig (Abschnitt 7.2).

(2) Ein weiteres Ergebnis ist, dass ausgewählte Entwurfsmuster in der Sekundarstufe II als Hilfsmittel für Kompetenzentwicklung mit Informatiksystemen eingesetzt werden können und Schüler mit Vorkenntnissen in der objektorientierten Modellierung vernetzte fundamentale Ideen in ihnen und der entsprechenden Lernsoftware entdecken. Ein Programm setzte die fundamentale Idee der Zugriffskontrolle mit dem Proxymuster um, ein weiteres die Datenstruktur Liste mit dem Iteratormuster. Eine weitergehende theoretische Fundierung über die untersuchten Entwurfsmuster hinaus wird vor dem Hintergrund der positiven Resonanz angestrebt (Abschnitt 7.3).

(3) Außerdem sind allem Anschein nach die positiven Schülerrückmeldungen zu der eingesetzten Software und den Lernmaterialien im Zusammenhang mit den prototypischen Modulen der Lernsoftware Pattern Park zu sehen. Die Ergebnisse und Rückmeldungen aus der Erprobung der Module flossen in die Entwicklung von Pattern Park ein (Abschnitt 5.6).

(4) Die Unterrichtsmethodik ist selbstverständlich nicht unabhängig vom eingesetzten Untersuchungsgegenstand. In den Erprobungen sollten die Schüler die einsetzten Programme systematisch untersuchen. An dieser Stelle trat die Schwierigkeit auf, dass die Schüler keine Erfahrung mit der systematischen Erkundung und speziell mit dem Formulieren von Hypothesen zur Erklärung des Systemverhaltens hatten. Deshalb ist eine weitergehende Analyse der systematischen Erkundung notwendig (Abschnitt 7.4).

Auf der 12. GI-Fachtagung „Informatik und Schule – INFOS 2007" (Schubert 2007) wurden die Ergebnisse der Fallstudie präsentiert (Stechert 2007c). Direkt im Anschluss wurde in einem Workshop zur intervenierenden Fachdidaktik mit Informatiklehrern im Rahmen des 6. Informatiktages Nordrhein-Westfalen der GI über die Ergebnisse diskutiert (Schubert et al. 2007). Ausgangspunkt waren drei Hauptthesen zu den Möglichkeiten und Grenzen der Forschungsmethodik, die in diesem Kapitel bereits aufgegriffen wurden: (i) Das Interview der Lehrperson, die die Lerngruppe in der Kooperationsschule betreut, erlaubt eine Einschätzung der Angemessenheit bezüglich Stoffumfang und Schwierigkeitsgrad. Zudem kann die Lehrperson die Ergebnisse der Intervention zu anderen Lehr-Lern-Situationen in Bezug setzen und vergleichend bewerten. (ii) Die Befragung der Lernenden zum Unterricht ermöglicht eine Einschätzung zur Akzeptanz und Rückschlüsse auf den Lernerfolg. (iii) Die exemplarische Umsetzung von ausgewählten Teilen eines umfassenderen Forschungsprojekts erlaubt verallgemeinerbare Rückschlüsse hinsichtlich Lernschwierigkeiten auf das Forschungsprojekt insgesamt. Zu berücksichtigen sind dabei die in Abschnitt 1.2.2 beschriebenen Möglichkeiten und Grenzen der Forschungsmethodik, die Plausibilitätserklärungen hinsichtlich Machbarkeit und Akzeptanz sowie unterstützende Aussagen zu ähnlichen Untersuchungen implizieren.

6.6.2 Informatiksysteme und Kompetenzentwicklung in der ersten Unterrichtserprobung

Im Folgenden wird kurz resümiert, wie die erste Unterrichtserprobung die Kompetenzentwicklung mit Informatiksystemen unterstützt hat. Eine typische Aufgabe war die systematische Erkundung eines Informatiksystems zur Zugriffskontrolle, die Analyse und einfache Modifikation der Software umfasste. Es wurden mehrere Sichten auf bzw. Hauptfunktionen von Informatiksystemen thematisiert: „Design" wurde durch Betrachtung der Entwurfsmusterstrukturen von Iterator und Proxy diskutiert (Abschnitt 3.2.2; (Denning 2007)). Außerdem sind „Recollection", durch

Iteration der Datenstruktur Liste, und Grenzen der „Automation" betrachtet worden; letztere hinsichtlich des hypothetischen Einsatzes der analysierten Programme in komplexeren Szenarien. Zusätzlich wurden „Coordination" und „Communication" als Perspektiven auf Informatiksysteme durch den Unterrichtsinhalt Zugriffskontrolle genutzt.

Hinsichtlich der von der UNESCO definierten Literacys ist ein Beitrag zur ICT Literacy und speziell zur Software Literacy zu erwarten (UNESCO 2008), denn das Thema Zugriffskontrolle, das sowohl durch das Proxymuster als auch durch das Iteratormuster aufgegriffen wird, ist für viele Systeme im Alltag relevant. Dazu gehört die Zugriffskontrolle durch das Betriebssystem der Rechner im Schulnetz, aber auch bei Anwendungsprogrammen auf Mehrbenutzersystemen und auf Webseiten im Internet. Insbesondere unterschiedliche Zugriffsrechte wurden im Unterricht behandelt. Des Weiteren ist durch den Einsatz unterschiedlicher Arten von Lernsoftware, Pattern Park und kleine Programme zu einzelnen Entwurfsmustern, ein Beitrag zur Media Literacy geleistet worden. Durch oben genannte Betrachtung der Grenzen der eingesetzten Programme in hypothetischen, komplexeren Situationen, wurde das kritische Denken der Schüler unterstützt.

Durch die Zugriffskontrolle hat die Unterrichtserprobung eine Grundvoraussetzung für die interaktive Anwendung von Technologien (DeSeCo: 1c) thematisiert, ohne deren Kenntnis der Zugriff auf Informatiksysteme verwehrt bleibt, d h. eine Schlüsselkompetenz wurde gefördert (OECD 2005). Die Schüler mussten die Beobachtungen zu Zugriffsrechten dokumentieren, dann die Daten auswerten und fundamentale Ideen identifizieren, um im Alltag Systeme bewusst und zielgerichtet anzuwenden, z. B. für lebensbegleitendes Lernen. Außerdem ist Zugriffskontrolle wichtig, um Zugang zu Informationen zu erhalten, und betrifft somit die Schlüsselkompetenz „Fähigkeit zur interaktiven Nutzung von Wissen und Informationen" (DeSeCo: 1b) und gleichzeitig die Information Literacy gemäß UNESCO.

Zusätzlich fördern das Experimentieren mit und Erkunden von Informatiksystemen das Verknüpfen von Verhalten und Struktur. Die Schüler identifizieren eine Menge an Sonderfällen, die im Systemverhalten getestet werden. Dadurch entwickeln sie die Kompetenz, eigenständig mit Informatiksystemen zu handeln. Hypothesenbildung zu dem Einsatz eines Informatiksystems in der Lebenswelt bzw. komplexeren Szenarien adressieren die Fähigkeit zum Handeln im größeren Kontext (DeSeCo: 3) ebenso wie die Vermeidung der „Versuch-Irrtum"-Strategie. Die Schüler können sich durch die systematische Erkundung Informatiksysteme eigenständig erschließen und für ihre Ziele einsetzen. Die Antizipation von Systemfehlern im Lernprozess verringert Stress in individuellen Projekten der Schüler, so dass deren eigenständiges Handeln (DeSeCo: 3) unterstützt wird.

Interagieren in heterogenen Gruppen und vor allem Kooperationsfähigkeit (DeSeCo: 2b) werden durch die erste Unterrichtserprobung dahingehend gefördert, dass

die Schüler sich Kenntnisse über die innere Struktur und eine angemessene Fachsprache bezüglich des Systemverhaltens aneignen. Dadurch sind sie in der Lage, eigene Ideen zu präsentieren, anderen zuzuhören oder ein Problem mit der Technologie in eigenen Worten zu umreißen. Gleichzeitig wird das Informatiksystem entmystifiziert und die Motivation steigt (vgl. Kollee et al. 2009).

Die Einordnung des Unterrichtsmodells in den Europäischen Qualifikationsrahmen wurde in Abschnitt 5.7 vorgenommen. Hinsichtlich der erreichten Niveaustufen der Schüler ist keine fundierte Aussage möglich, da entsprechende qualitative und quantitative Messverfahren für den Informatikunterricht fehlen (Abschnitt 9.3).

7. Weiterentwicklung, Verfeinerung und Ergänzung des Unterrichtsmodells

7.1 Überblick

Die Durchführung der Unterrichtserprobung zur Kompetenzentwicklung mit Informatiksystemen und die Tragweite der Ergebnisse wurden im Rahmen eines Workshops (Schubert et al. 2007) mit Informatiklehrern auf der INFOS 2007 diskutiert. Die Ergebnisse der ersten Unterrichtserprobung resultierten in einer Verfeinerung der Strategie zur Strukturierung im Rahmen des Unterrichtsmodells (Abschnitt 7.2).

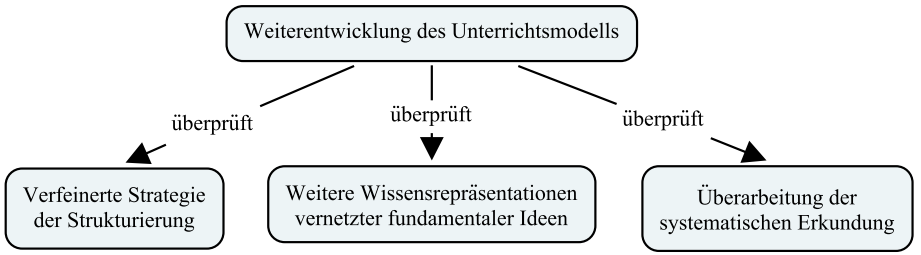


Abbildung 7.1: Struktur des Kapitels zu Weiterentwicklung, Verfeinerung und Ergänzung des Unterrichtsmodells

Die didaktischen Klassifizierungen von Entwurfs- und Architekturmustern als Wissensrepräsentationen werden weiterentwickelt: Betreut durch den Autor entstanden 2007 zwei Diplomarbeiten, die vernetzte fundamentale Ideen anhand von Softwaremustern präsentieren (Abschnitt 7.3). In der ersten wurde die fachdidaktische Klassifikation von Entwurfsmustern als Wissensrepräsentation für vernetzte fundamentale Ideen der Informatik auf Architekturmuster angewendet und mit Unterrichtbeispielen inklusive Lernmaterialien gestaltet (Ufer 2007). Die Ergebnisse der verfeinerten Strukturierung wurden bereits integriert. In der zweiten Diplomarbeit wurden Entwurfsmuster nach Gamma et al. (1995) klassifiziert und ausführlich auf ihren Beitrag zur Förderung der Kompetenzentwicklung mit Informatiksystemen hin untersucht sowie ein neues Konzept für eine Lernsoftware auf Basis von Entwurfsmustern entworfen (Weyer 2007b). In Abschnitt 7.4 wird die systematische Erkundung von Informatiksystemen durch die Methode des Laut-Denkens und anhand der neuen Strukturierung der Basiskompetenzen analysiert. Abbildung 7.1 zeigt die Struktur des Kapitels.

7.2 Verfeinerung der Strukturierung von Kompetenzen zu Informatiksystemen

7.2.1 Verfeinerung der Perspektiven auf Informatiksysteme

Kompetenzen zu Informatiksystemen wurden bisher nach den Perspektiven des nach außen sichtbaren Verhaltens (S_A), der inneren Struktur (S_B) und den Implementierungsaspekten (S_C) von Informatiksystemen strukturiert. Im Unterricht und in der benoteten schriftlichen Übung der ersten Erprobung wurden Problemstellen im Unterricht und kognitive Hürden ausgemacht (Kapitel 6). Bei der in Abbildung 6.2 dargestellten Vorgehensweise zur Erkundung des Informatiksystems wurde die für Unterrichtsexperimente wichtige Dokumentation der Schritte anfangs von vielen Schülern vernachlässigt. Geeignete und den Schülern bekannte formale informatische Modelle wurden genutzt, um Quelltext zu veranschaulichen, z. B. Klassendiagramm, aber nicht, um einen Sachverhalt oder eine Hypothese hinsichtlich des Systemverhaltens darzustellen. Die Auswertung ergab, dass eine Verfeinerung der Strukturierung der Basiskompetenzen zur Überwindung der gedanklichen Brüche zwischen den Perspektiven vorzunehmen ist. Damit wird es möglich, Überleitungen wie „... und nun kommen wir zur inneren Struktur des Informatiksystems!" zu vermeiden. Unter der Prämisse, dass ein Zugang zu Informatiksystemen über das nach außen sichtbare Verhalten zu wählen ist, ergibt sich folgende Verfeinerung der Strukturierung der Basiskompetenzen in $S_i, i \in \{A, B, C, AB, AC, BC\}$:

S_A: nach außen sichtbares Verhalten von Informatiksystemen,

S_{AB}: Wechselwirkungen zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur (z. B. anhand von Entwurfs- und Architekturmustern),

S_{AC}: Wechselwirkungen zwischen dem nach außen sichtbaren Verhalten und ausgewählten Implementierungsaspekten (durch systematisches Testen),

S_B: die innere Struktur von Informatiksystemen (z. B. anhand von Strukturmodellen: Entwurfsmuster, Schichtenmodelle etc.),

S_{BC}: Wechselwirkungen zwischen der inneren Struktur und ausgewählten Implementierungsaspekten (z. B. Idiome (Buschmann et al. 1996)),

S_C: ausgewählte Implementierungsaspekte zur Erstellung einer konkreten Realisierung.

Die Lernphasen zu den Basiskompetenzen werden spiralförmig wiederkehrend auf den größer werdenden Untersuchungsbereich des im Unterricht behandelten Informatiksystems angewendet (vgl. Stechert 2006c, S. 96). Im Folgenden werden die Perspektiven kurz skizziert (Stechert und Schubert 2007).

S_A: nach außen sichtbares Verhalten Das Verhalten von Informatiksystemen kann durch Beobachtungsaufgaben und einfache Experimente verstanden werden. Schüler und Lehrer sollten hierzu die notwendigerweise zugrunde liegende Hypothese gemeinsam erarbeiten. Die Schüler wenden ein konkretes Informatiksystem zielgerichtet an und entdecken beispielsweise eine fundamentale Idee der Informatik über das Systemverhalten. Neben Lehrerdemonstrationen sind auch Animationen geeignet, um auf unerwartetes Verhalten aufmerksam zu machen (vgl. Lernsoftware Pattern Park in Abschnitt 5.6). Anwendungsfalldiagramme, Aspekte der Anforderungsanalyse und Qualität von Informatiksystemen können thematisiert werden.

S_{AB}: Kombination von Systemverhalten und innerer Struktur Die Unterrichtsexperimente, in denen die Schüler selbständig Hypothesen formulieren, unterstützen die Kombination der Perspektiven Verhalten und Struktur. Die Schüler können beispielsweise funktionale Modelle nutzen, um das beobachtete Verhalten des Systems zu formalisieren. Fundamentale Ideen und Entwurfsstrukturen werden identifiziert. Ein Beispiel ist die Datenstruktur Schlange, die um die einen Iterator erweitert wurde, was die Schüler anhand des Systemverhaltens entdecken können (Kapitel 6). Ein weiteres Beispiel für ein funktionales Modell von Hardware liefert Hubwieser (Hubwieser 2007a, S. 152). Ziel der kombinierten Sichten ist, dass die Schüler in der Lage sind zu erklären, wie die innere Struktur das Systemverhalten beeinflusst.

S_{AC}: Kombination von Verhalten und Erstellung einer Realisierung Mittels systematischen Testens kann das Systemverhalten analysiert werden. Ist der Quelltext gegeben, so können Schüler im Sinne eines White-Box-Tests (Abschnitt 5.5.1) in einem Unterrichtsexperiment das Systemverhalten auf

Ursachen untersuchen. Das Systemverhalten kann klassifiziert werden, während im Quelltext besonders wichtige Stellen variiert werden. Das heißt, ein vorhandenes System wird modifiziert. Zum Beispiel kann bei dem Programm zur Zugriffskontrolle mit den zugewiesenen Zugriffsrechten experimentiert werden. Beim Iteratormuster kann der Algorithmus zur Iteration verändert werden.

S_B**: innere Struktur** Im Allgemeinen ist die innere Struktur eines Informatiksystems nur den Entwicklern, nicht aber den Anwendern bekannt (Claus und Schwill 2006). Systemkomponenten und deren Beziehungen müssen von den Schülern erkundet werden. Dazu sind statische und dynamische Aspekte zu betrachten. Die Anwendung unterschiedlicher Diagrammarten wie Klassen-, Objekt-, Sequenz- und Zustandsdiagramm kann die innere Struktur visualisieren (Karahasanovic und Holmboe 2006), (Hadar und Hazzan 2004). In Abgrenzung zu S_C werden insbesondere Strukturmodelle von Schülern zu identifizieren sein, die sie miteinander in Verbindung setzen müssen, z. B. Schichtenmodelle und ausgewählte Entwurfsmuster.

S_{BC}**: Kombination von Struktur und Erstellung einer Realisierung** Die Strukturierung von Implementierungsaspekten ist notwendig zur Erstellung einer konkreten Realisierung. Solche Strukturelemente liegen auf einer niedrigeren Abstraktionsebene als Entwurfsmuster. Ein Beispiel sind so genannte Idiome, d. h. programmiersprachenabhängige Strukturierungen. In Abschnitt 5.4.2 wurde argumentiert, dass Idiome in dieser Arbeit zur Förderung der Kompetenzentwicklung mit Informatiksystemen nicht betrachtet werden.

S_C**: Erstellung einer konkreten Realisierung** Zur Implementierung eines Informatiksystems sind Programmierkenntnisse unabdingbar. Studien zeigen jedoch, dass es Lernenden oftmals nicht gelingt, Programme zu entwerfen und zu erstellen, selbst wenn sie Programmierkonzepte kennen, weil ihnen der Überblick über das gesamte Programm fehlt (McCracken 2004). Deshalb wird der Schwerpunkt in der vorliegenden Arbeit auf Strukturmodelle gelegt, um Systemverhalten zu erklären.

Diese verfeinerte Strukturierung ermöglicht die Kombination unterschiedlicher Perspektiven für ein konsistentes Gesamtbild vom Informatiksystem und unterstützt explizit den lernförderlichen Sichtenwechsel. Handlungsorientierung und Transfer von einer Perspektive auf eine nächste werden durch die genannten Schülertätigkeiten gefördert. Wie bereits angedeutet sind S_{BC} und S_C für Basiskompetenzen nicht erforderlich bzw. in dieser Arbeit von nachgeordneter Relevanz. Diese Ansicht stützt sich ab auf der kontroversen fachdidaktischen Diskussion. Darin wird Programmierung oft entweder als ingenieurwissenschaftlicher Anteil und damit nicht allgemein bildend angesehen (vgl. Burkert 1995, S. 50). Oder Programmierung wird

als essentiell für Abstraktionsprozesse der Informatik angesehen. Wie Hubwieser und Schubert feststellen, ist die Kontroverse oft durch unterschiedliche Auslegung der Begriffe begründet (vgl. (Schubert 1991), (Hubwieser 2007a, S. 88)). Bei einer Aufspaltung des Begriffs Programmierung in Modellierung und Strukturierung sowie Codierung wird klar, dass Syntax- und Implementierungsdetails der Codierung zuzurechnen sind. Diese sind jedoch allenfalls in Verbindung mit Modellen von gewissem Bildungswert, wenn durch die Implementierung die abstrakten Modelle veranschaulicht und Versuch-Irrtum-Strategien ausgeschlossen werden.

7.2.2 Strukturierung der Unterrichtsinhalte

Exemplarische Kompetenzfacetten

Die Vorstrukturierung der Basiskompetenzen ermöglicht nun die Formulierung konkreter Ziele in Anlehnung an die Kompetenzfacetten nach Weinert (2001). Da dies in der vorliegenden Arbeit nicht empirisch fundiert geschehen kann, seien beispielhaft folgende motivationalen, kognitiven Fähigkeiten und Fertigkeiten sowie die Fähigkeit, Problemlösungen in variablen Situationen anzuwenden genannt (vgl. Kapitel 2). Auf die Zuordnung konkreter vernetzter fundamentaler Ideen wird an dieser Stelle der Übersichtlichkeit halber verzichtet. Beispiele finden sich bei der Erprobung des Unterrichtsmodells in den Kapiteln 6 und 8:

Die Schüler sind in der Lage ...

- Motivation
  - die Bereitschaft und Fähigkeit zu entwickeln, von Black-Box-Vorstellungen (Computergläubigkeit) zu rationalen Strukturmodellen zu wechseln (S_A).
- Können: Verstehen von Fachkonzepten und Anwenden von Fachmethoden (in „Entscheidungskonflikten" bzw. -situationen)
  - Strukturmodelle anzuwenden, um das nach außen sichtbare Verhalten von Informatiksystemen erklären zu können (S_{AB}).
- Fähigkeit: Transfer von Fachkonzepten
  - Erkenntnisse von Strukturmodellen für einen Sichtenwechsel auf andere Strukturmodelle zu übertragen (S_B).

Kompetenzfacetten nach Weinert sind Fähigkeit, Wissen, Verstehen, Können, Handeln, Erfahrung, Motivation (vgl. Klieme et al. 2007, S. 73). Diese werden um die Facetten des Transfers auf variable Anforderungssituationen im Sinne der Kompetenzdefinition nach Weinert ergänzt. Volitionale Aspekte werden unter der Facette der Motivation mitbetrachtet, auch wenn absichts- und willensbezogene Bereitschaften nicht identisch mit Motivation sind. Zur Begriffsklärung ist anzumerken,

dass Wissen nicht nur als reines Faktenwissen interpretiert wird, sondern Prozesse mit einschließt. Um den fachlichen Bezug herzustellen wird auf das Erfahrungswissen der Fachdidaktik zurückgegriffen, das in Kapitel 4 analysiert wurde. Neben der Zielorientierung für Informatikcurricula und der Definition von Standards dienen Kompetenzmodelle auch der Entwicklung von empirischen Messinstrumenten. Bezogen auf die Informatik kann damit die Wirksamkeit von Lernprozessen gemessen werden.

Bezug zu Wissensstrukturen des didaktischen Systems

Angemerkt sei an dieser Stelle, dass die Strukturierung der Basiskompetenzen Ähnlichkeiten zu Wissensstrukturen aus dem didaktischen System aufweist (vgl. Brinda und Schubert 2001). Denn sie sind wie Wissensstrukturen eine Strukturierungsempfehlung für Lehr-Lernprozesse, die eine Gesamtsicht auf das Themengebiet anbieten. Dargestellt werden Wissensstrukturen in einem Graph, dessen Knoten Unterrichtsinhalten und dessen Kanten Aufgabenklassen entsprechen. Aufgabenklassen sind dementsprechend notwendig, um einen neuen Unterrichtsinhalt zu erarbeiten. Wenn man die Basiskompetenzen nun einer Wissensstruktur zu Informatiksystemen zugrunde legt, bereichern sie das Konzept der Wissensstruktur gleichzeitig um Kompetenzclusterung, die dem Vorwurf entgegenwirkt, dass durch Wissensstrukturen hauptsächlich die Fachsystematik reproduziert werde. Bei der Unterscheidung zwischen Groblernzielen und Feinlernzielen ergibt sich folgende Lesart: Ein als Knoten repräsentiertes Groblernziel steht stellvertretend für einen neuen Teilgraph, der durch die zu dem Groblernziel gehörenden Feinlernziele und deren Verbindungen gebildet wird. Dies impliziert wiederum, dass der Übergang von einem Lernziel zum nächsten nicht durch nur eine Aufgabenklasse, sondern durch eine Menge von Aufgaben realisiert wird, die – bedingt durch die unterschiedlichen Abstraktionsebenen – nicht notwendiger Weise in derselben Aufgabenklasse liegen. Die Weiterentwicklung der Wissensstrukturen des Didaktischen Systems ist nicht Ziel der vorliegenden Arbeit. Deshalb sei an dieser Stelle auf Arbeiten von Freischlad verwiesen (Freischlad 2008).

Als weitere Verfeinerung der Lernziele zu den Basiskompetenzen bieten sich fundamentale Ideen der Informatik an. Durch die Kriterien für fundamentale Ideen wird der Bildungswert gesichert, während die Basiskompetenzen einen auf Informatiksysteme und Systemverständnis ausgerichteten Unterricht ermöglichen. Beispiele für die Zugriffskontrolle (Zk) sind:

- Wesentliche Aspekte des nach außen sichtbaren Verhaltens bei Zugriffskontrolle, z. B. den Unterschied zwischen aktivierter und deaktivierter Zugriffskontrolle ($S_{A,\ Zk}$).
- Statische und dynamische Aspekte der inneren Struktur eines Informatiksystems, um Zugriffskontrolle zu realisieren, z. B. über das Stellvertreterprinzip im Proxy-Entwurfsmuster mit Vererbung ($S_{B,\ Zk}$).

Für eine fachsystematische Strukturierung in Wissensstrukturen zum Thema Informatiksysteme bieten sich fundamentale Ideen der Informatik allein nur bedingt als Knoten an, da insbesondere die Vernetzung der Ideen nicht vorgenommen wird und der direkte Bezug zu den Perspektiven nicht deutlich wird.

Die fundamentale Idee der Iteration sei im Folgenden als „It" abgekürzt. Beispiele für Basiskompetenzen anhand von Kompetenzfacetten nach Weinert (2001) sind dann bezüglich der vernetzten fundamentalen Ideen:

Die Schülerinnen und Schüler sind in der Lage ...

- Motivation
  - die Bereitschaft und Fähigkeit zu entwickeln, von Black-Box-Vorstellungen (Computergläubigkeit) zum Iteratormuster als rationales Strukturmodell zu wechseln ($S_{A,It}$).
  - die Bereitschaft und Fähigkeit zu entwickeln, von Black-Box-Vorstellungen (Computergläubigkeit) zum Proxymuster als rationales Strukturmodell zu wechseln ($S_{A,Zk}$).

- Können: Verstehen von Fachkonzepten und Anwenden von Fachmethoden (in „Entscheidungskonflikten" bzw. -situationen)
  - Das Iteratormuster anzuwenden, um das nach außen sichtbare Verhalten von Informatiksystemen erklären zu können ($S_{AB,It}$).
  - Das Proxymuster anzuwenden, um das nach außen sichtbare Verhalten von Informatiksystemen erklären zu können ($S_{AB,Zk}$).

- Fähigkeit: Transfer von Fachkonzepten
  - Erkenntnisse von der Erkundung eines Programms mit Iteratormuster für einen Sichtenwechsel auf ein Programm mit Proxymuster zu übertragen ($S_{B,It}$).
  - Erkenntnisse von dem Zugriff, den ein Iterator auf Datenstrukturen ermöglicht, für einen Sichtenwechsel auf Zugriffskontrolle mit dem Proxymuster zu übertragen ($S_{B,Zk}$).

In praktischen Erprobungen müssen Erkenntnisse darüber gewonnen werden, in welchem Maße vernetzte fundamentale Ideen der Informatik Basiskompetenzen zu Informatiksystemen bei den Lernenden zu fördern vermögen.

7.3 Weiterentwicklung und Ergänzung zu Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik

7.3.1 Architekturmuster als Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik

Die Diplomarbeit „Architekturmuster als Beitrag zum Informatiksystemverständnis" wurde durch den Autor der vorliegenden Dissertationsschrift betreut und im Mai 2007 fertig gestellt (Ufer 2007). Im Rahmen der Arbeit sollten laut Themenbeschreibung objektorientierte Architekturmuster nach Buschmann et al. (1996) für Kompetenzentwicklung mit Informatiksystemen klassifiziert werden – auch unter Berücksichtigung der fundamentalen Ideen der Informatik bezüglich des nach außen sichtbaren Verhaltens S_A und dessen Kombination mit der inneren Struktur S_{AB} sowie den Implementierungsaspekten S_{AC}. Dazu gehörten die Strukturierung des Bildungsprozesses und deren Begründung sowie die Erstellung eines Entwurfs von Lernmitteln, die auf der Klassifikation der Architekturmuster beruhen.

Ufer würdigt darin Kompetenzentwicklung mit Informatiksystemen als wichtiges Bildungsziel des Informatikunterrichts. Die seiner Arbeit zugrunde liegende Annahme unter Berufung auf Stechert (2006b) ist, dass Architekturmuster eine Form der Wissensrepräsentation für vernetzte fundamentale Ideen der Informatik darstellen, als Strukturmodelle die Zusammenhänge zwischen Systemkomponenten beschreiben und bewährte Lösungen der Softwareentwicklung sind. Ufer geht auf fundamentale Ideen der Informatik nach Schwill (1993a) und deren Erweiterung nach Modrow (2002) ein. Darüber hinaus werden der Systembegriff sowie das vernetzte Denken betrachtet und deren Kombination im fachdidaktischen Ansatz zu Kompetenzentwicklung mit Informatiksystemen nach Stechert beschrieben. Nach einer Kompetenzdefinition gemäß Weinert leitet Ufer im Rahmen des fachdidaktischen Ansatzes acht Kompetenzen für Schüler in der Sekundarstufe II ab, die die Grundlage für operationalisierte Lernziele bilden. Trotz der Orientierung am Kompetenzbegriff nach Weinert (Abschnitt 2.1) ist das Vorgehen insofern kritisch zu sehen, dass sie nicht in ein Kompetenzmodell eingeordnet werden (können), so dass die Bezeichnung irreführend ist. Positiv ist jedoch die starke Einbeziehung motivationaler und volitionaler Bereitschaften zur bewussten Anwendung von Informatiksystemen, die einen Beitrag zur Erstellung eines Kompetenzmodells leisten kann.

Die zur Klassifikation der Architekturmuster notwendigen Leistungsparameter bestimmt Ufer in einer Anforderungsanalyse. Diese leiten sich aus dem fachdidaktischen Ansatz nach Stechert sowie der Arbeit von Harrer und Schneider zur Klassifikation von Entwurfsmustern für die Hochschulinformatik ab (Harrer und Schneider 2002). Er nutzt fünf Kriterien: die vernetzten fundamentalen Ideen der Informatik,

die Zusammenhänge mit weiteren Softwaremustern, die Komplexität, das Einsatzgebiet des Architekturmusters und ein Lebensweltbeispiel (vgl. Abschnitt 5.4.1).

Ufer untersucht die acht Architekturmuster Schichtenarchitekturmuster, Kanäle und Filter, Tafel, Vermittler, Model-View-Controller sowie Präsentation-Abstraktion-Steuerung, Mikrokern und Reflexion anhand der Kriterien einzeln (Tabellen 7.1 und 7.2). Die drei letztgenannten Muster werden als nicht geeignet für den Informatikunterricht der Sekundarstufe II eingestuft. Grund für die Ablehnung ist die starke inhaltliche Überschneidung des Präsentation-Abstraktion-Steuerung-Architekturmuster mit dem MVC-Architekturmuster bei erhöhter Komplexität. Bei den Architekturmustern Reflexion und Mikrokern ist das spezielle Anwendungsgebiet zusammen mit der geringen Anzahl fundamentaler Ideen und dem Schwierigkeitsgrad für die Aussortierung ausschlaggebend. In der Arbeit werden die den untersuchten Architekturmustern zugrunde liegenden vernetzten fundamentalen Ideen der Informatik in Wirkungsdiagrammen dargestellt bei gleichzeitig vorhandener Kritik der Wirkungsdiagramme (vgl. Abschnitt 5.4.5):

> „Die Wirkungsdiagramme hätten also auch in leicht veränderter Form erstellt werden können und erheben daher keinen Anspruch auf exklusive Korrektheit. Der Autor beschränkt deshalb die Diagramme auf die Veranschaulichung der wichtigsten Zusammenhänge zwischen den fundamentalen Ideen des jeweiligen Architekturmusters" (Ufer 2007, S. 28).

Jedes Muster wird nach dem gleichen Schema klassifiziert. Nach einer kurzen Einführung in Anlehnung an softwaretechnische Beschreibungen folgt die didaktische Analyse. Als fundamentale Ideen wird für das Schichtenarchitekturmuster beispielsweise die Masteridee Hierarchisierung beschrieben und im Kontext des – für Kompetenzentwicklung mit Informatiksystemen besonders wichtigen – Ebenenmodells des Rechners betrachtet. Weitere fundamentale Ideen in dem Beispiel sind die Master-Idee Sprache durch Bezug zu Compilern, Modularisierung durch Komponentenbildung im Gegensatz zum Abstraktionsgrad der Hierarchisierung sowie Top-down- bzw. Bottom-up-Vorgehensweisen zur Umsetzung einer hierarchischen Modularisierung beispielsweise bei Netzwerkprotokollen oder dem Ebenenmodell des Rechners. Darüber hinaus wird Delegation im Rahmen der Objektorientierung didaktisch begründet in Abgrenzung zur Vererbung beschrieben und das Geheimnisprinzip wird in Anlehnung an das Fassademuster angeführt. Zuletzt wird die Wichtigkeit der Schichtenarchitektur bei Wiederverwendung hervorgehoben.

Bezüglich des zweiten Kriteriums „Zusammenhänge mit weiteren Softwaremustern" werden korrespondierende Entwurfsmuster mit Hinweis auf fundamentale Ideen näher beschrieben. Beim Schichtenarchitekturmuster sind dies Fassade, Zuständigkeitskette, Befehl und Beobachter.

Das dritte Kriterium, Komplexität, wird in Anlehnung an die hochschuldidaktische Evaluation von Harrer und Schneider betrachtet, aber für die Sekundarstu-

fe II anhand erwarteter Lernschwierigkeiten bei den fundamentalen Ideen weiter begründet. Dazu werden Varianten des Musters im Lehr-Lernprozess betrachtet.

Beim vierten Kriterium, dem Einsatzgebiet des Musters, werden sowohl aus der Softwaretechnik heraus Einsatzbereiche des Musters genannt, als auch die Bezüge zu fundamentalen Ideen als Aspekte des Entwurfsmusters beschrieben, die per Definition in verschiedenen Bereichen der Informatik relevant sind.

Das letzte Analysekriterium, das Lebensweltbeispiel zur Funktionalität des Musters, wird in eine Problemstellung aus dem Erfahrungsbereich der Schüler und eine mögliche Lebenswelt-Lösung aufgeteilt. Darüber hinaus wird gegebenenfalls auf relevante Lebensweltbeispiele für einzelne fundamentale Ideen der Informatik hingewiesen wie das „Philosophenproblem" und den „schlafenden Frisör" bei der fundamentalen Idee Nebenläufigkeit beim Kanäle-und-Filter-Muster. Eine Übersicht der Klassifikation in Form einer Tabelle resümiert in einer vergleichenden Evaluation die untersuchten Architekturmuster bezüglich aller Klassifikationsparameter (Tabellen 7.1 und 7.2; (Ufer 2007, S. 95)).

Ufer konstruiert aus den Analyseergebnissen Lehr-Lernprozesse in Form von Unterrichtsbeispielen zu den durch die Klassifikation ausgewählten Architekturmustern. Dazu werden notwendige Vorkenntnisse der Lerngruppe und die operationalisierten Lernziele bezüglich der von ihm definierten Kompetenzen vorgestellt und der Lernprozess strukturiert. Nach der Vorstellung wichtiger Inhalte der Informatik werden durch die Klassifikation der Architekturmuster daraus die didaktischen Anforderungen an Lehr-Lern-Prozesse und technischen Konsequenzen für Lernsoftware abgeleitet. Dies geschieht unter Berücksichtigung des konstruktivistischen Unterrichtsmodells zu Informatiksystemen und Kompetenzentwicklung. Dabei bewährte sich die Weiterentwicklung im Bereich der Wissensstrukturen (S_i) nach Brinda, indem sie mit Aufgabe und Aufgabentyp sowie mit einer Einordnung in die Kompetenzstrukturierung, Lernziel, und Lernzielebene (Anderson und Krathwohl 2001) verknüpft wurden. Ein Aufgabentyp war z. B. „Beobachtung mit vorgegebener Vorgehensweise".

Bei der Entwicklung der Lernsoftware weist Ufer auf die Schwierigkeit hin, einerseits ein für die Schüler angemessenes und sinnvolles Informatiksystem bereitzustellen, andererseits die Architekturmuster möglichst im Kern unverändert zu integrieren (vgl. Ufer 2007, S. 99). Er bezieht sich dabei auf die von ihm umgesetzte Art der Lernsoftware, die auf einem Muster als Strukturgrundlage beruht (Abschnitt 5.6.2). Eine weitere Schwierigkeit ist die Umsetzung von Architekturmustern für nebenläufige Systeme, wenn sie auf einem Von-Neumann-Rechner in der Schule eingesetzt werden sollen. Im Fall des Vermittler-Architekturmusters wurde dieser Aspekt simuliert, da das Vorhandensein weiterer fundamentaler Ideen der Informatik den Nachteil kompensiert.

Tabelle 7.1: Geeignete Architekturmuster nach (Ufer 2007, S. 95)

Architekturmuster	**Fundamentale Ideen**	**Enthaltene Muster**	**Komplexität**	**Einsatzgebiet**	**Lebensweltbeispiel**
Schichten	Hierarchisierung, Sprache, Modularisierung, Top-down / Bottom-up, Delegation, Geheimnisprinzip, Umgekehrte Pyramide der Wiederverwendung	Fassade, Zuständigkeitskette, Befehl, Beobachter	einfach	allgemein	Entfernte Kommunikation in verschiedenen Sprachen
Kanäle und Filter	Modularisierung, Geheimnisprinzip, Wiederverwendbarkeit, Wartbarkeit, Teamarbeit, Eingabe-Verarbeitung-Ausgabe, Schnittstelle, FIFO, Nebenläufigkeit, Thread, Zustand, Fairness	Singleton	einfach-mittel	nicht ereignisgesteuerte Systeme	Wasch-Firma
Tafel	Teile-und-Herrsche, Geheimnisprinzip, Nichtdeterminismus, Backtracking, Nebenläufigkeit, Thread, Zustand, Grenzen der Berechenbarkeit, Alternative, Schleife, Schlange	Beobachter, Singleton, Strategie, Adapter	einfach-mittel	schwierige Probleme, künstliche Intelligenz	Aufklärung eines Verbrechens
Vermittler	Client-Server, Vernetzung, Graphen, Parallelverarbeitung, Delegation, Schnittstelle, Geheimnisprinzip, Wiederverwendung, gemeinsame Ressourcennutzung, Verfügbarkeit, Zuverlässigkeit, Zugriffskontrolle	Proxy, Zusteller-Empfänger, Adapter, Vermittler	hoch	verteilte Systeme	Reisebüro
Model-View-Controller	Eingabe-Verarbeitung-Ausgabe, Ereignisse, Benutzungsoberfläche, Daten / Darstellung, Geheimnisprinzip, Wiederverwendung, Assoziation, Vererbung	Beobachter, Kompositum, Strategie, Dekorierer, Fabrikmethode	mittel	interaktive Systeme	Ergebnisse und Diagramme einer politischen Wahl

Als Unterrichtsbeispiele werden folgende Lehr-Lernprozesse beschrieben: ein Informatiksystem für politische Wahlen mit dem Model-View-Controller-Architektur-

Tabelle 7.2: Weniger geeignete Architekturmuster nach (Ufer 2007, S. 95)

Architekturmuster	**Fundamentale Ideen**	**Enthaltene Muster**	**Komplexität**	**Einsatzgebiet**	**Lebensweltbeispiel**
Präsentation-Abstraktion-Steuerung	Hierarchische Modularisierung, Baum, Schnittstelle, Daten / Darstellung, Prozesse	Adapter, Kompositum, Fassade, Vermittler, Beobachter, Zuständigkeitskette	hoch	interaktive Systeme	Ergebnisse und Diagramme einer politischen Wahl auf Verwaltungsebenen
Mikrokern	Emulation, Prozesse, Nebenläufigkeit, Ressourcenverwaltung, Fairness	Adapter, Fassade, Proxy	hoch	adaptierbare Systeme, Betriebssysteme	Spielekonsole, digitale Fotokamera
Reflexion	Wartbarkeit, Schnittstelle, Metadaten	Strategie	einfachmittel	adaptierbare Systeme	Gesetze und Gerichte

muster, ein Labyrinthsystem in Prolog anhand des Tafelarchitekturmusters, ein Übersetzungsprogramm mit dem Klartext verschlüsselt und kodiert wird (Schichtenarchitektur), ein Informatiksystem zur Auswertung von Temperaturdaten (Kanäle-und-Filter-Muster) und ein Reisebuchungssystem anhand des Vermittlerarchitekturmusters. Für jeden Lehr-Lernprozess werden Lernziele zu den Basiskompetenzbereichen nach außen sichtbares Verhalten von Informatiksystemen (S_A), Kombination von nach außen sichtbarem Verhalten mit innerer Struktur (S_{AB}) sowie Kombination von nach außen sichtbarem Verhalten und Implementierung (S_{AC}) angegeben und mit konkreten Aufgaben unterlegt. Es werden viele weitere Materialien wie UML-Diagramme und Arbeitsblätter mit Lösungsvorschlägen angegeben. Für jedes Architekturmuster schließt die unterrichtspraktische Beschreibung mit einer Übersicht über alle Lernziele mit Zuordnung des Kompetenzbereichs, der Lernzielebene nach Blooms überarbeiteter Taxonomie, der konkreten Aufgabe, des Aufgabentyps. Hervorzuheben ist, dass Ufer durchaus die Schwierigkeit benennt, dass die Lernzielebene in Abhängigkeit zum Vorwissen der Schüler steht.

Ufer macht deutlich, wie die Einbindung von für Lernende angemessener Lernsoftware zu Architekturmustern mit Schwerpunkt auf vernetzten fundamentalen Ideen der Informatik in den Informatikunterricht der Sekundarstufe II gelingen kann, wenngleich eine Unterrichtserprobung noch aussteht. Es werden Verbindungen zu der vom Autor durchgeführten Unterrichtserprobung mit Entwurfsmustern angedeutet, beispielsweise mit Hinweis auf die Lernsoftware Pattern Park (vgl. Ufer 2007, S. 154).

7.3.2 Entwurfsmuster als Wissensrepräsentationen: Einfluss von Musterparametern auf das Systemverhalten und Entwurf einer Lernsoftware

Weyer (2007b) klassifiziert in seiner vom Autor betreuten Diplomarbeit objektorientierte Entwurfsmuster nach Gamma et al. (1995) exemplarisch unter Berücksichtigung fundamentaler Ideen der Informatik bezüglich des nach außen sichtbaren Verhaltens von Informatiksystemen. Er betrachtet die Entwurfsmuster Beobachter, Fassade, Proxy, Dekorierer, Iterator, Schablonenmethode und Zustand. Dazu nutzt er Kriterien, die auch Ufer (2007) einsetzte, passte sie aber für Entwurfsmuster an (vgl. Abschnitt 5.4; S. 184). Bei dem Kriterium der Zusammenhänge mit anderen Mustern geht er vornehmlich auf strukturelle Ähnlichkeiten von Mustern ein. Der Lebensweltbezug der Muster wird konsequent in den Kontext einer Arztpraxis gesetzt. Anschließend wird der Entwurf einer Lernsoftware beschrieben, die eine Produktsicht (Arztpraxissoftware), eine Parametersicht (Entwurfsmusterparameter), eine Animationssicht und eine Drag-and-Drop-Sicht anbietet. Dem vorangestellt sind die die Interaktivitätstaxonomie nach Schulmeister und Repräsentationsebenen nach Bruner, in die später eine Einordnung der Sichten erfolgt.

Für die Entwurfsmuster Beobachter, Fassade, Proxy, Iterator und Zustand wird anschließend je ein Szenario skizziert, in dem vornehmlich durch Veränderung der Parameter des Musters in der Parametersicht das Verhalten des Informatiksystems in der Produktsicht verändert wird. Drag-and-Drop-Sicht sowie Animationssicht ergänzen die Lernsoftware um weitere Repräsentationsformen und Interaktivitätsstufen für einen gestuften Zugang zu den Unterrichtsinhalten anhand der kognitiven Prozesse aus der Lernzieltaxonomie nach Anderson und Krathwohl (2001). Nach den Modulen zu den einzelnen Entwurfsmustern wird ein Modul zur Kombination der Muster Proxy und Beobachter entworfen. Insgesamt sind die Ideen für die Lernsoftware ansprechend hinsichtlich der Förderung der Kompetenzentwicklung mit Informatiksystemen, aber über das Stadium eines Grobentwurfs noch nicht hinaus.

Zum Abschluss seiner Arbeit nutzt Weyer die Erkenntnisse aus dem Entwurf der Module, um ein neues fachdidaktisches Kriterium für die Evaluation von Entwurfsmustern zu beschreiben:

Kriterium 7 (Auswirkung von Parametern auf das Systemverhalten)
Die Korrelation zwischen der Anzahl der Parameter und dem Einfluss von Änderungen der Parameter auf das nach außen sichtbare Verhalten ist in Entwurfsmustern zu untersuchen. Einerseits ist das Kriterium dadurch gegenläufig zum Kriterium der Komplexität, denn eine hohe Anzahl an veränderbaren Parametern ist Indiz für erhöhte Komplexität des Musters. Andererseits können Parameter, die das Verhal-

ten des Systems beeinflussen, jedoch helfen, das System zu analysieren, wodurch ein didaktischer Mehrwert geschaffen wird.

Damit wird die durch die überarbeitete Strukturierung der Basiskompetenzen geforderte Kombination von Verhalten und Struktur (S_{AB} und S_{AC}) auf die Kriterien für Entwurfsmuster abgebildet. Tabelle 7.3 zeigt die Analyse der fünf von Weyer diskutierten Muster bezüglich des neuen Kriteriums (→ Kriterium 7: Auswirkung von Parametern auf das Systemverhalten).

Tabelle 7.3: Exemplarische Analyse von Entwurfsmustern anhand des neuen Kriteriums (→ Kriterium 7: Auswirkung von Parametern auf das Systemverhalten; (Weyer 2007b, S. 74))

Muster	**Parameter**	**Auswirkung auf das Systemverhalten**
Beobachter	Anpassen an neuen Dokumenttyp (Neuer Beobachter)	Animationssicht zeigt verschiedene Dokumente mit individuellen Daten
	Melde An / Melde Ab	Entscheidet darüber, ob ein Dokument mit Daten bestückt werden kann.
	Anpassen an neuen Beobachter	Stellt zusätzliche Dokumenttypen zur Verfügung
Fassade	Typ der Anfrage	Entscheidend dafür welches Subsystem angesprochen wird
	Anpassen an neues Subsystem (Neues Subsystem)	Weitere Verarbeitungsfälle entstehen
Proxy	Benutzungsrechte	Ermöglicht grafische Darstellung von zugelassenen oder blockierten Zugriffen
	Aktivieren des Platzhalters (Proxy aktivieren)	Entscheidet darüber, ob ein Eingriff des Schutzproxys überhaupt stattfinden kann
	Anzahl Zugriffe zählen	Ein Zähler für die Anzahl der vorgenommenen Zugriffe
Iterator	Sequentiell, Polymorph, Mehrfach	Entscheiden über den Typ der Iteration
	Art der Iteration (nach Datenstruktur)	Passt den Iterator an die zu traversierende Datenstruktur an
	Anpassen an neuen Iterator (Neuer Iterator)	Ohne diese Funktion wäre es nicht möglich mehrfach zu iterieren
Zustand	Zustände definieren	Essentiell für den Ablauf des Algorithmus
	Zustandsübergänge definieren	Demnach kann in der Animationssicht ein Film abgespielt werden
Kombination von Proxy und Beobachter	Analog zu den einzelnen Entwurfsmustern	Gegenseitige Beeinflussung der bereits aufgeführten Parameter

In einem Fazit weist Weyer darauf hin, dass weitere Entwurfsmuster anhand des neu entwickelten Kriteriums zu untersuchen sind und dass eine Implementierung des Entwurfs der Lernsoftware aussteht (Weyer 2007b, S. 80).

7.4 Analyse der systematischen Erkundung von Informatiksystemen

7.4.1 Auswirkungen der verfeinerten Strukturierung auf die systematische Erkundung des Systemverhaltens

Unterrichtsinhalte sind nicht unabhängig von Unterrichtsmethoden. Erfahrungen und Beobachtungen aus der ersten Erprobung des Unterrichtsmodells haben offen gelegt, dass Schüler Schwierigkeiten haben, ihr Wissen um Informatikkonzepte mit deren Realisierung in typischen Informatiksystemen zu verknüpfen und die Struktur der Informatiksysteme zu beschreiben (vgl. Kapitel 6). Durch die erste empirische Erprobung des Unterrichtsmodells und die anschließende Strukturierung der Basiskompetenzen ist weiterhin offensichtlich geworden, dass eine Analyse der systematischen Erkundungen stattfinden muss, die eine höhere interne Validität hat (Abschnitt 1.2.2). Mit Hilfe der neuen Strukturierung ist es möglich, die in der ersten Unterrichtserprobung eingesetzten Aufgaben genauer zu analysieren. Bei der Erstellung der Aufgaben wurde bereits Wert auf die Kombination des nach außen sichtbaren Verhaltens mit der inneren Struktur und die Vernetzung fundamentaler Ideen der Informatik gelegt. Dabei stand der Lebensweltbezug im Vordergrund, um die Schüler zusätzlich zu motivieren.

Zur Evaluation ordnen Stechert und Schubert (2007) die Schritte der systematischen Erkundungen des Systemverhaltens und der inneren Struktur (Abschnitt 5.5) den Aspekten (I) Anwendung eines konkreten Informatiksystems, (II) Strukturmodelle von Informatiksystemen und (III) fundamentale Ideen zu. Gleichzeitig werden sie in die Kategorien S_A, S_{AB}, S_{AC} und S_B exemplarisch eingeordnet (Abschnitt 7.2). Die Einordnung ist nicht immer eindeutig, da einige Erkundungsschritte unterschiedliche Aspekte kombinieren, z. B. die Aufgabe, ein Klassendiagramm (S_B) zur Zugriffskontrolle (III) zu erstellen, resultiert in der Struktur des Proxymusters (II). In diesem Fall wurde die Aufgabe der Kombination S_B-II zugeordnet (statt S_B-III), da das Entwurfsmuster als Wissensrepräsentation mehrere vernetzte fundamentale Ideen der Informatik enthält.

Betrachtet man die Schritte der systematischen Erkundung der inneren Struktur von Informatiksystemen (Abschnitt 5.5) und deren Umsetzung im Unterricht, dann fällt auf, dass sie sich größtenteils auf ein konkretes Informatiksystem (I) und konstruktionsgemäß auf das Systemverhalten S_A beziehen. Die Kategorie S_{AB} wird nur bei der Analyse der Systemkomponenten hinsichtlich ihrer Auswirkungen auf

das Systemverhalten angesprochen und bei der Identifikation fundamentaler Ideen, die sich auf das Systemverhalten auswirken. Um diese Kategorie zu stärken, werden in der zweiten Unterrichtserprobung Systemzustände thematisiert (Kapitel 8), in denen das Potential gesehen wird, die aufeinander abfolgenden Zustände eines Systems zur Beschreibung des Verhaltens zu nutzen (Goos und Zimmermann 2006, S. 18). S_{AC} war in der ersten Unterrichtserprobung durch mehrere Aufgaben zur Modifikation eines Programms vertreten, es fehlte jedoch an kleineren Variationen, wie sie beim systematischen Testen eingesetzt werden können. Insgesamt ist festzustellen, dass die Dimensionen I, II, III sowie S_A, S_{AB}, S_{AC} und S_B in der ersten Erkundung angesprochen wurden (Stechert und Schubert 2007, S. 8f). Deren Verteilung ist jedoch sehr unterschiedlich.

Die erste Erprobung im unterrichtlichen Geschehen ergab, dass die Schüler keine Erfahrung mit systematischen Erkundungen des Systemverhaltens haben. Zusammen mit dem Ergebnis der Analyse des Standes der Forschung, dass Beobachtungsaufgaben im Informatikunterricht selten sind, wurde vom Autor die Entscheidung getroffen, das Beobachten im Rahmen der Unterrichtsexperimente weiter zu stärken. Dafür ist Transparenz für die Schüler erforderlich. Gemäß der neuen Strukturierung der Basiskompetenzen kann die systematische Erkundung unterschiedlichen Bereichen zugeordnet werden. Der erste Teil stellt die Beobachtung in den Vordergrund (S_A), die in Hypothesen über vernetzte fundamentale Ideen resultiert (S_{AB}). Die Beobachtungen müssen dokumentiert werden, um die Versuch-und-Irrtum-Strategie zu vermeiden. Bereits hier sind Sonderfälle und Variationen zu notieren, damit auf die dahinter stehenden Konzepte geschlossen werden kann. Der zweite Teil umfasst dann die Formulierung der Hypothese hinsichtlich der inneren Struktur des Systems (S_{AB} und S_B) und die Planung des Experiments. Am Ende der systematischen Erkundung der inneren Struktur stehen kleine Modifikationen (S_{AC}). Eine Klassifikation der in der ersten Unterrichtserprobung eingesetzten Aufgaben zeigt, dass zwei Drittel von ihnen den Kompetenzen S_A und S_{AB} zuzuordnen sind und damit (potentiell oder tatsächlich) durch die Vorgehensweise der systematischen Erkundung von Informatiksystemen gelöst werden können (Stechert und Schubert 2007, S. 9). Dabei ist jedoch zu relativieren, dass die Aufgaben nicht gleichwertig bzw. gleich aufwändig sind. Speziell die Aufgaben, die nur die innere Struktur (S_B) und systematisches Testen S_{AC} betreffen, erfordern in der Regel mehr Bearbeitungszeit. Schlussfolgerung ist, dass die einzelnen Schritte der systematischen Erkundungen zu Beginn im Unterricht sehr klar definiert und klein zu wählen sind, um sie der Strukturierung zuordnen zu können. Darauf aufbauend kann dann eine für den jeweiligen Lehr-Lernprozess geeignete Schwerpunktsetzung durch die Lehrperson erfolgen.

7.4.2 Analyse des Erkundens von Informatiksystemen mittels Laut-Denken

Für die Analyse der systematischen Erkundung mittels Laut-Denken gab es drei wesentliche Gründe:

- den Widerspruch zwischen
  (a) den weniger positiven Ergebnissen der Schüler in der Lernerfolgskontrolle bei Aufgaben zur systematischen Erkundung und
  (b) der Einschätzung der Schüler in der Akzeptanzbefragung, dass die Erkundung von Informatiksystemen einfach sei,
- die Frage, ob Fehlvorstellungen zu Objektorientierung die kognitiven Barrieren zu Informatiksystemen und Kompetenzentwicklung überlagern,
- die Frage, wie die systematische Erkundung anhand der neuen Strukturierung der Basiskompetenzen einzuordnen ist.

Die Akzeptanzbefragung nach der ersten Erprobung ergab: Die Mehrheit von 19 (82%) Schüler gibt an, die systematische Analyse des Systemverhaltens verstanden zu haben. Ebenfalls 19 (82%) Schüler stimmten der Aussage zu, dass die Inhalte des Unterrichtsprojekts für den Umgang mit Informatiksystemen sehr nützlich sind (Abschnitt A.1). Dabei nahm die systematische Erkundung von Informatiksystemen einen hohen Anteil der Unterrichtszeit ein.

Die Lernerfolgskontrolle ergab jedoch folgende, auf den ersten Blick überraschenden Resultate: Schwierigkeiten traten auf in Bezug auf eine systematische Erkundung des nach außen sichtbaren Verhaltens. Hieraus könnte der Schluss gezogen werden, dass es kaum Vorkenntnisse bezüglich eines solchen Vorgehens gab, denn bei Aufgaben zur inneren Struktur, die den Aufgaben aus dem vorherigen Unterricht ähnlicher waren, schnitten die Schüler besser ab. Dennoch haben in einer Aufgabe alle Schüler angegeben, dass zum Verstehen des nach außen sichtbaren Verhaltens eines Informatiksystems ein systematisches Experimentieren notwendig sei und dass unerwartetes Verhalten eines Informatiksystems Rückschlüsse auf dessen innere Struktur liefere. Es zeigte sich eine teilweise Überforderung der Lernenden bei der Verbindung des nach außen sichtbaren Verhaltens mit der inneren Struktur (Abschnitt 6.5). Angebracht gewesen wäre hier eine Aufteilung in kleinere Teilaufgaben. Diese Aufteilung in kleine Aufgaben ist durch die Schrittfolge zur systematischen Erkundung implizit gegeben.

Insbesondere ist in der Feldstudie nicht exakt auszumachen gewesen, worauf die Schwierigkeiten zurückzuführen sind, z. B. Fehlvorstellungen der Objektorientierung. Wegen der dennoch hohen Akzeptanz des systematischen Erkundens bei den

Schülern, wurden einerseits der hohe Abstraktionsgrad der Schrittfolge und inhaltliche Schwierigkeiten mit eben jenen konkreten informatischen Konzepten vermutet. Zum Beispiel gab es zahlreiche Nachfragen zur Bedeutung des Begriffs „informatische Konzepte" und der Bezug zur bekannten Listenstruktur war für die Schüler nicht offensichtlich. Darüber hinaus zeigte es sich, dass typische Fehlvorstellungen der Objektorientierung Schwierigkeiten zu Informatiksystemen in dieser Konstellation überlagern könnten.

Weigend (2006) erklärt kognitive Barrieren und Fehlvorstellungen damit, dass sich Intuitionen, also Vorstellungen von der Welt, nicht mehr mit der Realität in Einklang bringen lassen. Weigend überprüft solche kognitiven Barrieren im Bereich der Programmierung. Karahasanovic und Holmboe (2006) nennen als typische Lernschwierigkeiten bei der objektorientierten Modellierung beispielsweise die Erzeugung geeigneter Klassen sowie die Integration deklarativer Aspekte (Klassenhierarchie) und prozeduraler Aspekte (Main-Methode), Fehlvorstellung von fundamentalen Konzepten wie Klassen und Vererbung sowie den Transfer von Vorerfahrungen mit traditionellem Entwurf in der prozeduralen Modellierung zur Objektorientierung (vgl. Karahasanovic und Holmboe 2006, S. 50). Beim Entwurf von Klassen sind Kapselung, innere Komplexität, Platzierung und Benennung von Methoden und Attributen fehlerträchtig. Bei der Struktur die Beziehungen zwischen Klassen, Objekten und Exemplaren sowie die statische Systemstruktur und Assoziationsformen. Außerdem ist die Beschreibung der Problemdomäne schwierig, da sie auf Einschränkungen hin interpretiert werden muss und die Frage aufkommt, ob man das Problem zur Gänze erfasst hat (vgl. Karahasanovic und Holmboe 2006, S. 56).

Deshalb wurde zur weiteren Evaluation die systematische Erkundung in einer exemplarischen Laborstudie mit der Methode des Laut-Denkens analysiert (Stechert 2008c). Die Methode fällt in die Kategorie Laborbeobachtung, weil die Beobachtungsbedingungen, unter denen die Untersuchung stattfindet, vom Versuchsleiter künstlich geschaffen werden:

> „Grundsätzliche Unterschiede hinsichtlich des Beobachtungsfeldes lassen sich zwischen Feldbeobachtungen und Laborbeobachtungen feststellen" (Atteslander et al. 2006, S. 76).

Laut-Denken zielt als qualitative Untersuchung darauf ab,

> „in erster Linie komplexere Situationen und Interaktionen in ihrer Ganzheit [zu] erfassen" (Atteslander et al. 2006, S. 77).

In ihrer vom Autor betreuten Seminararbeit ließen Stupperich und Warkentin (2007) vier Studierende Aufgaben zur systematischen Erkundung lösen. Neben der systematischen Erkundung des Systemverhaltens wurden zwei weitere Aufgaben zur Erkundung der inneren Struktur des Informatiksystems gestellt. Da anschließend die Schwerpunktverlagerung zum Systemverhalten vorgenommen wurde, werden letztere im Folgenden nicht explizit betrachtet. Einige Aspekte der inneren

Struktur sind jedoch in die Erkundung des nach außen sichtbaren Verhaltens zu integrieren, wenn es um die Kombination mit der inneren Struktur geht.

Die Probanden gehören nicht zur Zielgruppe, Schüler der Sekundarstufe II, sondern waren Studierende. Für die Untersuchung mittels Laut-Denken standen Schüler der Sekundarstufe II nicht zur Verfügung, denn es bestand keine Möglichkeit, Schülern Anreize zu bieten, trotz ihres überfüllten Schullalltags an der Studie teilzunehmen. Die Informatikvorkenntnisse der Studierenden waren unterschiedlich: Versuchsperson 1 studiert Lehramt Informatik im ersten Semester, Versuchsperson 2 und 3 ebenfalls, aber im siebten bzw. fünften Semester, und Versuchsperson 4 hatte Informatik nach zwei Semestern abgewählt. Insbesondere die Versuchspersonen 1 und 4 wurden ausgewählt, da sie ähnliches Vorwissen wie die Schüler der Sekundarstufe II aufwiesen. Zusätzlich waren die Versuchsleiterinnen unerfahren hinsichtlich der Durchführung des Laut-Denkens, was sie in ihrer eigenen Kritik und im Rahmen von Verbesserungsvorschlägen für weitere Überprüfungen mittels Laut-Denken offen diskutieren (Stupperich und Warkentin 2007, S. 37). Deshalb ist klar, dass Rückschlüsse vorsichtig zu ziehen sind und gefundene Problemstellen bei den Studierenden nur als Hinweise für die Sekundarstufe II dienen können.

Konkrete Aufgabe war, das nach außen sichtbare Verhalten eines Arztpraxissystems nach der vorgegebenen Systematik zu erkunden (Stupperich und Warkentin 2007, S. 39).

Die Ergebnisse lassen sich in zwei Kategorien unterteilen. Einerseits sind typische Verständnisprobleme bezüglich der Aufgabenstellung und andererseits Erkenntnisse über die Vorgehensweisen bei der Bearbeitung der Aufgabenstellungen zu nennen. Die Verständnisprobleme bezüglich Aufgabenstellung umfassten vor allem unklare, unbekannte Begriffe wie den allgemein gehaltenen Terminus „informatische Konzepte", die von den Schülern identifiziert werden sollten. Und unerwartete Schwerpunktsetzungen, wie sie das systematische Erkunden des nach außen sichtbaren Verhaltens offenbar aufgrund mangelnder Erfahrung mit einer solchen Vorgehensweise darstellte. Erwartet wurden vermutlich Programmieraufgaben. Außerdem wurden Hinweise zu den Aufgaben nicht beachtet.

Stupperich und Warkentin (2007) fiel bei allen Versuchspersonen auf, dass sie Schwierigkeiten mit der Umsetzung der Vorgehensweise hatten. Sie hätten keinen richtigen Zugang zur Umsetzung der Vorgehensweise gefunden (Stupperich und Warkentin 2007, S. 25). Für die Überprüfung der Aufgabe in weiteren Laborstudien empfehlen sie daher, eine Übungsaufgabe des gleichen Schemas im Vorfeld zu stellen, damit die Probanden sie einüben können. Für den Unterricht in der Sekundarstufe II ist diese Schlussfolgerung insofern nachvollziehbar, dass auch bei Schülern anzunehmen ist, dass diese Vorgehensweise nicht zu den Vorkenntnissen gehört und sie deswegen von der Lehrperson dafür sensibilisiert werden müssen. Die Bearbeitungszeit für die Aufgabe konnte durch die Studie zum Laut-Denken

sehr viel genauer bestimmt werden, als es in der Unterrichtserprobung möglich war, so dass die Zeitplanung angepasst werden konnte. Die detaillierte Bearbeitung der Aufgabe durch Versuchsperson 3 dauerte beispielsweise 14 Minuten statt der erwarteten 5 Minuten (Stupperich und Warkentin 2007, S. 21), die durchschnittliche Bearbeitungszeit betrug 11 Minuten. Dennoch ist gerade bei der Einführung der systematischen Erkundung auf eine detaillierte Bearbeitung durch die Schüler wert zu legen, so dass die Bearbeitungszeit heraufzusetzen ist.

Darüber hinaus wurden unterschiedliche Vorgehensweisen bei der Bearbeitung der Schritte zur systematischen Erkundung festgestellt. So arbeitete Versuchsperson 1 vom Konkreten zum Abstrakten, versuchte erst, jeden Erkundungsschritt grob abzuarbeiten und sich ein Bild vom Verhalten des Systems zu machen. Später wurden die Ergebnisse einzelner Schritte näher spezifiziert (Stupperich und Warkentin 2007, S. 10). Dabei ging sie dennoch strukturiert vor und formulierte Hypothesen über das Verhalten, bevor es erkundet wurde. Auch Versuchsperson 2 stellte Hypothesen auf und revidierte sie gegebenenfalls (Stupperich und Warkentin 2007, S. 16), arbeitete die Schritte der systematischen Erkundung jedoch sukzessive ab.

Interessant war, dass bei den späteren Aufgaben zur Erkundung der inneren Struktur oft Bezug genommen wurde auf die Erkenntnisse aus der systematischen Erkundung des Verhaltens. Es wurde also ein Perspektivwechsel zwischen Erkenntnissen aus beobachtetem Verhalten und Quellcodeanalyse sowie zwischen Klassendiagrammen und Quellcode vorgenommen:

> „Positiv aufgefallen ist die Tatsache, dass diese Versuchperson beim Bearbeiten des zweiten und dritten Schrittes [der Erkundung der inneren Struktur; Anm. d. V.] immer einen Perspektivenwechsel gemacht hat. Sie hat bestimmte Operationen im laufenden Programm mit den entsprechenden Funktionen im Quellcode verglichen. ‚Wenn ich im Programm den Button 'neuer Patient' klicke, wird ein Patient erstellt mit den Parametern aus meinen Eingabefeldern'." (Stupperich und Warkentin 2007, S. 26).

Ziel muss es sein, diesen Perspektivwechsel weiter zu unterstützen.

7.4.3 Zusammenfassung der Analyse des Erkundens von Informatiksystemen

Die Schwerpunktverlagerung auf die Betrachtung des Systemverhaltens resultiert darin, dass in der zweiten Erprobung (Kapitel 8) keine explizite Erkundung der inneren Struktur vorgenommen wird. Einige Aspekte der inneren Struktur sind jedoch in die Erkundung des nach außen sichtbaren Verhaltens zu integrieren, wenn es um die Kombination mit der inneren Struktur geht. Die Resultate der Laboruntersuchung führten in erster Linie nur zu einer minimalen Überarbeitung in der Formulierung der Schritte zur systematischen Erkundung von Informatiksystemen, aber nicht zu einer Reorganisation. Stattdessen ist die Konsequenz eine sehr viel

detailliertere Einführung der Vorgehensweise in den Unterricht. Die erste Unterrichtserprobung und Analyse mittels Laut-Denken legen nahe, dass den Schülern Erfahrungen mit einer systematischen Erkundung des Systemverhaltens fehlen und die Schrittfolge zur systematischen Erkundung nicht selbsterklärend ist. Somit ist das systematische Erkunden im Unterricht stärker zu thematisieren und beispielsweise von der Lehrperson zu demonstrieren. Der hohe Abstraktionsgrad der systematischen Erkundung kann durch eine Kontextualisierung verringert werden. Die Forderung nach Kontextualisierung impliziert eine Umformulierung in Anlehnung an die Lebenswelt der Schüler und hinsichtlich des konkreten Informatiksystems (vgl. Abbildung 8.3). Damit kann auch explizit auf den Zweck des jeweiligen Informatiksystems eingegangen werden, wie Karahasanovic und Holmboe es fordern (Karahasanovic und Holmboe 2006, S. 58), und die Verwendung von Begriffen vermieden werden, die den Schülern unbekannt sind.

7.5 Zusammenfassung

In diesem Kapitel wurde eine verfeinerte Strukturierung der Kompetenzen zu Informatiksystemen angegeben, die den Schwerpunkt auf die Analyse des nach außen sichtbaren Verhaltens und dessen Kombination mit der inneren Struktur und Implementierungsaspekten legt (Abschnitt 7.2). Die Analyse von Architekturmustern liefert eine Erweiterung der einsetzbaren Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik im Rahmen des Unterrichtsmodells. Die Anwendung der neuen Strukturierung und die Gestaltung von Aufgaben zur Architektur- und Entwurfsmustern sowie entsprechender Lernsoftware erweitern das Spektrum des Unterrichtsmodells (Abschnitt 7.3). Durch die erste Unterrichtserprobung (Kapitel 6), die neue Strukturierung und die Analyse mittels Laut-Denken werden die Schülertätigkeiten zur systematischen Erkundung des Systemverhaltens im Rahmen des Unterrichtsmodells neu bewertet. Wichtigste Schlussfolgerung ist, dass die Erkundung im Unterricht explizit thematisiert werden muss, was beispielsweise eine Lehrerdemonstration mit einschließt, da den Schülern Erfahrungen mit einer solchen Vorgehensweise fehlen.

8. Zweite exemplarische Erprobung des Unterrichtsmodells

8.1 Überblick

In diesem Kapitel wird die zweite Erprobung des Unterrichtsmodells beschrieben und evaluiert. Grundlage ist das überarbeitete Unterrichtsmodell, dessen Schwerpunkt auf der Verbindung unterschiedlicher Sichten auf Informatiksysteme liegt. Die Rahmenbedingungen einschließlich inhaltlicher Konzeption, Lerngruppe, zeitlichen und technischen Restriktionen sowie Unterrichtsmethodik werden in Abschnitt 8.2 angegeben.

Die Beschreibung und Durchführung der Erprobung geschieht in Abschnitt 8.3. In Abschnitt 8.4 werden die erhobenen Daten zur Evaluation hinsichtlich Machbarkeit und Akzeptanz durch Schüler und Lehrer herangezogen. Abschließend wird eine kurze Diskussion der Ergebnisse vorgenommen (Abschnitt 8.5). Abbildung 8.1 zeigt die Einordnung des Kapitels in den Forschungsverlauf.

8.2 Rahmenbedingungen und Untersuchungsmethodik

8.2.1 Inhaltliche Konzeption und Einordnung in den Forschungsverlauf

Untersuchungsschwerpunkte sind ...

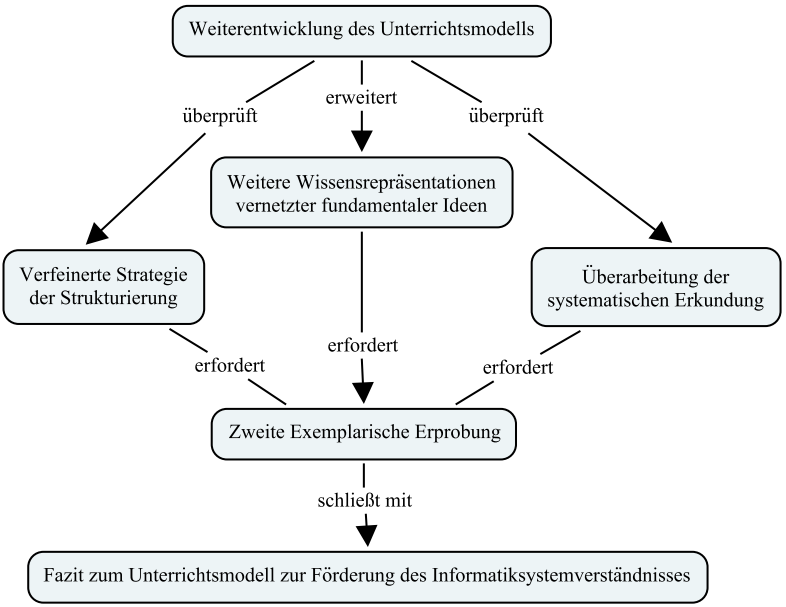


Abbildung 8.1: Einordnung des achten Kapitels in den Forschungsverlauf

1. die Brücke zwischen dem Verhalten und der inneren Struktur von Systemen und speziell die Eignung ausgewählter Entwurfsmuster zur Beschreibung von kombinierten Sichten auf Informatiksysteme (Abschnitt 7.2),
2. die lehr-lernmethodische Herausforderung durch die fehlende Erfahrung der Schüler mit systematischen Erkundungen von Informatiksystemen und speziell die Akzeptanz der Schüler bezüglich der überarbeiteten Vorgehensweise zur systematischen Erkundung von Informatiksystemen (Abschnitt 7.4),
3. mehrfacher Einsatz (von Variationen) eines Entwurfmusters in unterschiedlichen Programmen und Kontexten zur Förderung des Transfers (vgl. Abschnitt 6.5.1),
4. die zunehmende Vernetzung fundamentaler Ideen durch Vernetzung von Entwurfsmustern im Unterricht (Stechert 2008a), (Stechert 2008b).

Wie in der ersten Erprobung ist damit die Objektorientierung grundlegend. Insgesamt ist die zweite Unterrichtserprobung auf vier Wochen à drei Unterrichtsstunden pro Woche ausgelegt. Es wurden Zugriffskontrolle und Zustände als Inhalte gewählt. Das Thema Zugriffskontrolle wurde bereits in der ersten Unterrichtserprobung umgesetzt. Damit eignet es sich, um die Weiterentwicklung des Unterrichtsmodells in

der Unterrichtspraxis zu überprüfen. Das Thema Zustände wiederum ist aufgrund der Fokussierung auf das Systemverhalten in die Unterrichtserprobung aufgenommen worden. Systemzustände beschreiben die innere Struktur von Informatiksystemen im Sinne von Konfigurationen der Komponenten (Abschnitt 3.2.1; Abschnitt 7.4). Die Folge der Systemzustände wiederum kennzeichnet das Systemverhalten, das ggf. nach außen sichtbar wird (Goos und Zimmermann 2006, S. 18).

Weitere Änderung gegenüber der ersten Unterrichtserprobung ist, dass Implementierungsaspekte (S_C) einen wiederum verringerten Stellenwert haben. Die Lernziele hinsichtlich des nach außen sichtbaren Verhaltens (S_A) müssen im Rahmen systematischer Erkundungen von Informatiksystemen stattfinden, um unerwartetes Verhalten zu entdecken, z. B. Fehler. Zu erarbeitende Lösungen können auf der Ebene des nach außen sichtbaren Verhaltens durch Hypothesenbildung umgesetzt werden. Zur Kombination des Verhaltens mit der inneren Struktur (S_{AB}) sollen eine vorliegende Dokumentation eines Programms, unterschiedliche Modellierungen und Diagrammarten genutzt werden, und zwar Klassen-, Sequenz- und Zustandsdiagramme. Zur Erstellung von Zustandsdiagrammen wurden die Zustände aus dem Verhalten eines Informatiksystems abgeleitet, um sich von der Modellierung abzugrenzen, d. h. sie wurden nicht aus vorhandenen oder erarbeiteten objektorientierten Modellen wie Klassendiagrammen erstellt, bei denen Attribute Hinweise auf Zustände und Operationen auf Zustandsübergänge liefern.

Zusammenfassend bilden die fundamentalen Ideen Zugriffskontrolle und Zustände in den Entwurfsmustern Proxy und Zustand die Leitmotive, da die Entwurfsmuster eine konsistente Repräsentation für verschiedene Perspektiven auf das System bieten. Mit Blick auf den Zusammenhang zwischen Systemzuständen und dem Verhalten von Informatiksystemen wird implizit das Zustandsmuster im Systemverhalten nachgewiesen. Tabelle 8.1 zeigt exemplarische Unterrichtsinhalte anhand der Strukturierung der Basiskompetenzen.

8.2.2 Lerngruppe und zeitlicher Rahmen

Im November und Dezember 2007 wurde die zweite Unterrichtserprobung über vier Wochen durchgeführt. Um ähnliche Rahmenbedingungen wie in der ersten Unterrichtserprobung zu schaffen, wurde wieder ein Grundkurs Informatik am gleichen Gymnasium in der gleichen Jahrgangsstufe der Sekundarstufe II, nämlich 12. Klassenstufe, zur gleichen Zeit im Schulhalbjahr ausgewählt. An der Erprobung beteiligt waren 13 Schüler im Alter von etwa 17 Jahren, davon drei Schülerinnen. Vorkenntnisse der Schüler umfassten wie in der ersten Erprobung objektorientiertes Modellieren, Klassendiagramme und die Warteschlange als Datenstruktur. Programmiert wurde bis dahin in Delphi 6. Abgeschlossen wurde das Unterrichtsprojekt mit einer schriftlichen Übung als Lernerfolgskontrolle, einer schriftlichen Akzeptanzbefragung der Schüler und einem Interview mit der betreuenden Informatiklehrper-

Tabelle 8.1: Exemplarische Übersicht der inhaltlichen Konzeption anhand der Strukturierung der Basiskompetenzen

	exemplarische Unterrichtsinhalte
$S_{A,1}$	Die fundamentale Idee der Zugriffskontrolle und systematische Erkundung des Verhaltens von Informatiksystemen bezüglich unterschiedlicher Zugriffsrechte (Proxymuster).
$S_{A,2}$	Die Relevanz von Zuständen anhand der Ergebnisse einer systematischen Erkundung des Verhaltens von Informatiksystemen (Zustandsmuster).
$S_{AB,1}$	Die Beziehungen zwischen Aspekten der Zugriffskontrolle im Verhalten und dem objektorientierten Entwurf, z. B. dass Vererbung es einem Stellvertreterobjekt erlaubt, wie ein anderes Objekt, auf das es den Zugriff schützt, zu erscheinen.
$S_{AB,2}$	Zustandsdiagramme aus dem Verhalten von Informatiksystemen ableiten, wie es in $S_{A,2}$ erkundet wurde.
$S_{AC,1}$	Repräsentation des Programms durch Quellcode und Darstellung des Verhaltens des Informatiksystems in einem Sequenzdiagramm zur Zugriffskontrolle.
$S_{AC,2}$	Ableiten von Zustandsübergängen aus dem Verhalten des Informatiksystems und Identifikation von Ursachen Quelltext.
$S_{B,1}$	Zugriffskontrolle durch Vererbung und Ableitung eines objektorientierten Modells des Proxymusters ($S_{A,1}$).
$S_{B,2}$	Zustände und Zustandsübergänge sowie Zustandsdiagramme.

son der Schule. Tabelle 8.2 zeigt die Übersicht über die Unterrichtsstunden und -themen.

Eine wichtige Änderung zur ersten Unterrichtserprobung war, dass zwei Lehramtsstudierende im Rahmen ihres fachdidaktischen Praktikums unterrichteten. Unterrichtsentwürfe und Lernsoftware hatten sie anhand der Vorlagen aus der ersten Unterrichtserprobung zusammen mit dem Autor ausgearbeitet. Dies hatte zur Folge, dass der Autor die praktische Umsetzung des Unterrichtsmodells durch andere Lehrpersonen beobachten konnte. Der Umstand, dass in dieser zweiten Unterrichtserprobung zwei Lehramtsstudierende die unterrichtenden Personen waren und eine neue Informatiklehrperson den Kurs betreute, kann als Indiz dafür gewertet werden, dass Wiederholbarkeit und Anwendbarkeit des Unterrichtsmodells nicht allein durch die Forschungsperson gelingt. Annahme war, dass die Schüler bisher keine systematischen Vorgehensweisen zur Erkundung von Informatiksystemen kennen gelernt haben. Die Informatiklehrperson der Schule, beide Informatiklehramtsstudierende und der Autor waren an der Stundenvorbereitung beteiligt und werteten die Unterrichtsverläufe anschließend aus.

8.2.3 Unterrichtsmethodik und technischer Rahmen

In der zweiten Erprobung wurde verstärkt auf Motivierung der Schüler eingegangen, da der Bedarf durch die Akzeptanzbefragung nach der ersten Unterrichtserprobung erkannt wurde. Mit diesem Ziel wurden die ersten Unterrichtsentwürfe nach

Tabelle 8.2: Themenliste der zweiten Unterrichtserprobung mit Einzel- (ES) und Doppelstunden (DS)

Termin	**Stundenthema**
12.11.2007 (DS)	1. Sensibilisierung für Informatiksysteme; 2. Sensibilisierung für Zugriffskontrolle (Proxy, Stellvertreter)
15.11.2007 (ES)	Entwurfsmuster Proxy identifizieren
19.11.2007 (DS)	Modellierung dynamischer Abläufe bei der Zugriffskontrolle mit Sequenzdiagrammen
22.11.2007 (ES)	Wiederholung und Festigung vom Entwurfsmuster Proxy und von Sequenzdiagrammen
26.11.2007 (DS)	1. Sensibilisierung für Systemzustände; 2. Zustandsmodellierung
29.11.2007 (ES)	Beschreibung dynamischer Abläufe durch Zustandswechsel
03.12.2007 (DS)	Das Entwurfsmuster Zustand identifizieren und Modellierung mit Zustandsdiagrammen
06.12.2007 (ES)	schriftliche Übung: Test Akzeptanzbefragung

dem ARCS-Modell (Attention – Relevance – Confidence – Satisfaction) nach (Keller 1987) ausgearbeitet (vgl. Hubwieser 2007a, S. 17): Ein Informatiksystem kann durch Fehlermeldungen Schüler vor mentale Herausforderungen stellen (Attention) und die Notwendigkeit von Kompetenzen wird schnell ersichtlich (Relevance). Mit Hinweis auf geeignete Vorgehensweisen wie der systematischen Erkundung und späterem Einsatz von Modellierungsmethoden wie Sequenz- und Zustandsdiagrammen wurde ein angemessener Grad an Zuversicht erzeugt, das Ziel zu erreichen, wobei auch Grenzen der Methoden offen gelegt werden müssen (Confidence). Durch das erfolgreiche Anwenden der genannten Methoden und Vorgehensweisen wird der Lernerfolg (Satisfaction) sichtbar.

Grundlage der Unterrichtsplanung bildeten des Weiteren Überlegungen zu den Schüleraktivitäten. Die Lehrervorträge und Einführungen sollten nur wenige Minuten pro Stunde in Anspruch nehmen. Dies führte zu einem Zielkonflikt, denn in der ersten Erprobung wurde der Bedarf an Lehrerdemonstrationen zu den Unterrichtsexperimenten und Beobachtungsaufgaben der Schüler festgestellt. Durch die Vermeidung von Programmieraufgaben gelang es im Vergleich zur ersten Unterrichtserprobung, mehr Zeit auf Beobachtungen und Gruppenarbeit statt auf Einzel- bzw. Partnerarbeit zu verwenden. Partnerarbeit nahm jedoch weiterhin den Hauptteil der Aufgabenlösung ein.

In dieser zweiten Erprobung fand der Unterricht in einem anderen Informatiklabor der Schule statt. Dennoch war auch hier ein zentraler Kommunikationsbereich, dezentrale Rechnerarbeitsplätze an zwei Seiten und ein Demonstrationsbereich mit Tafel, Projektor und Präsentationsrechner an der Raumvorderseite. Damit unterstützte der Raum den unkomplizierten Wechsel zwischen geistig planenden Tätigkeiten im Kommunikationsbereich und praktischen Tätigkeiten an den Rech-

nerarbeitsplätzen. Das Schulintranet erlaubte Verbindung zum Internet, so dass Rechercheaufgaben gestellt werden konnten, wie z. B. zu objektorientierten Entwurfsmustern. Eine gemeinsam nutzbare Partition ermöglichte das Austauschen aller eingesetzten Programme.

8.3 Beschreibung und Durchführung der Erprobung

8.3.1 Lernphasen und Problemstellen im Unterrichtsprojekt

Ziel der Unterrichtserprobung war es, kognitive Barrieren und Fehlvorstellungen bezüglich der Sichten auf Informatiksysteme und deren systematischer Erkundung qualitativ zu analysieren. Schülerlösungen wurden dementsprechend eingesammelt und vor der Erprobung wurde angekündigt, dass die Note der abschließenden Lernerfolgskontrolle zur Gesamtnote zählen würde, um zuverlässigere Daten zu bekommen. Gleichzeitig bedeutete es, dass die Testergebnisse nicht anonymisiert erhoben wurden.

Begonnen wurde mit einem Programm zur Zugriffskontrolle, dass auch in der ersten Unterrichtserprobung im Einsatz war, da diese eine fundamentale Idee der Informatik ist (Abschnitt 5.4.4). In dem Programm waren unterschiedliche Benutzerrollen mit verschiedenen Zugriffsrechten umgesetzt: Administrator, Benutzer und Gast. Grundlage des Programms war das Proxymuster, das in der inneren Struktur eine von Softwareentwicklern erprobte Lösung darstellt. Das Verstehen der Benutzungsoberfläche und der darauf befindlichen Funktionen ermöglicht die angemessene Bedienung des Programms. Für eine über die Bedienung hinausgehende Analyse des Informatiksystems müssen die Schüler dessen Funktionsweise verstehen lernen und abstrahieren können. Außerdem ist das System mit ähnlichen Systemen zu vergleichen. Ein zweites, vor der Erprobung neu entwickeltes Programm auf Basis des Proxy realisierte Zugriffskontrolle in einer Arztpraxis. Darüber hinaus kam die mittlerweile fertig gestellte Lernsoftware Pattern Park zum Einsatz, um dynamische Aspekte der Zugriffskontrolle in einer Animation und zwei Übungen mit Sequenzdiagrammen zu ermöglichen. Für den zweiten Schwerpunkt, die Zustände, wurde das Modul zum Zustandsmuster aus der Lernsoftware Pattern Park zur Einführung genutzt. Anschließend konnte ein vorbereitetes Programm eingesetzt werden, dass eine Erweiterung des Programms der Zugriffskontrolle in der Arztpraxis um das Zustandsmuster darstellte, d. h. Zustandsmuster und Proxymuster wurden kombiniert. Bei der Arztpraxissoftware sind unterschiedliche Zustände zu erkennen, die durch bestimmte Eingaben verändert werden und zu Systemausgaben und geändertem Systemverhalten führen.

Eine Woche vor Beginn der Unterrichtserprobung hospitierten die Lehramtsstudierenden und der Autor in einer Doppelstunde des entsprechenden Kurses. Darin

wurde bestätigt, dass die Schüler Klassendiagramme nutzen konnten, um die innere Struktur von Informatiksystemen objektorientiert darzustellen. Schwierigkeit schien jedoch die Unterscheidung zwischen der Manipulation von Daten und der Manipulation einer Darstellung der Daten in der GUI zu sein.

8.3.2 Brücke zwischen Verhalten und innerer Struktur durch Erkundung von Informatiksystemen

Unter der Prämisse, dass ein Zugang zu Informatiksystemen über das nach außen sichtbare Verhalten zu wählen ist, wurde die Strukturierung der Basiskompetenzen verfeinert (Abschnitt 7.2). Die systematische Erkundung des nach außen sichtbaren Verhaltens der lernförderlichen Software erhält hinsichtlich der Strukturierung der Basiskompetenzen die Einordnung S_A, wobei das Informatiksystem zunächst als Black-Box systematisch analysiert wird. Anschließend werden Bezüge zur inneren Struktur und zu ausgewählten Implementierungsaspekten betrachtet, um das zugrunde liegende Strukturmodell zu verstehen. Zur Kombination des nach außen sichtbaren Verhaltens mit der inneren Struktur (S_{AB}) wurden Sequenzdiagramme als formale Darstellung genutzt, die z. B. durch die Lernsoftware Pattern Park (Stechert 2006c) im Kontext von vernetzten fundamentalen Ideen und Entwurfsmustern erlernt werden können. Mit ihnen können Objekte, die aus dem Verhalten des Systems abgeleitet und auf der Benutzungsoberfläche identifiziert wurden, beschrieben werden. Das Analysieren der Systemkomponenten kann beispielsweise zum Testen und Modifizieren (S_{AC}) einzelner Komponenten führen.

Im Zusammenhang mit der Kombination der Perspektiven des nach außen sichtbaren Verhaltens und der inneren Struktur steht, dass den Schülern in der ersten Unterrichtserprobung Erfahrung mit systematischen Erkundungen des Verhaltens von Informatiksystemen fehlte. Die systematische Erkundung von Informatiksystemen wurde im Vorfeld des zweiten Unterrichtsprojektes analysiert, um kognitive Barrieren aufzudecken, z. B. bezüglich der gewählten Fachsprache (Abschnitt 7.4). Die handlungsorientierte Vorgehensweise zur systematischen Erkundung von Informatiksystemen bildete in dieser Unterrichtserprobung den ersten Schwerpunkt, um den vorher festgestellten Mangel an Vorerfahrungen mit systematischen Erkundungen von Informatiksystemen zu beheben. Dafür wurde zu Beginn die Wichtigkeit der Hypothesenbildung der Schüler zur Analyse des nach außen sichtbaren Verhaltens thematisiert. Es wurde die Schrittfolge der systematischen Erkundung von Schülern und Lehrperson gemeinsam im Plenum durchgeführt. Anschließend erkundeten die Schüler wie im ersten Unterrichtsprojekt die lernförderliche Software zur Zugriffskontrolle ($S_{A,1}$) mit den Rollen Administrator, Benutzer und Gast (Abbildung 6.2).

Bei der Beschreibung der Benutzungsoberfläche sind zwei Vorgehensweisen der Schüler offensichtlich geworden: Eine Schülergruppe hat die grafische Benutzungs-

oberfläche in Blöcke mit Sinnzusammenhang hinsichtlich unterschiedlicher Teilnehmer (-objekte) mit unterschiedlichen Zugriffsrechten unterteilt und diese in Leserichtung von links nach rechts beschrieben. Eine zweite Gruppe hat die Oberflächenelemente nach bekannten Programmiersprachenkonstrukten sortiert, nämlich nach Labels, Buttons und Textfeldern. Beim Experimentieren mit Sonderfällen vermutete eine Schülergruppe allein durch das nach außen sichtbare Verhalten des Programms eine Realisierung der Zugriffskontrolle über Case-Anweisungen. Diese nicht mit der objektorientierten Denkweise konforme Vorstellung konnte analog zur ersten Unterrichtserprobung durch Identifizieren von Klassen überwunden werden, die zeigen, wie ein Stellvertreter durch Vererbung das gleiche Erscheinungsbild bzw. die gleiche Schnittstelle wie das Original haben kann. Durch die aufeinander aufbauenden Schritte zur systematischen Erkundung des Informatiksystems stellte die begleitende Anfertigung eines Klassendiagramms zur Dokumentation und Formalisierung keine unüberwindbare Hürde dar. Interessant ist, dass eine Schülerin bei der Beschreibung der Beziehungen zwischen den Oberflächenelementen nachfragte: „Ab wann dürfen wir ausprobieren?". Wenngleich die Frage darauf abzielte, eine Versuch-Irrtum-Vorgehensweise anzuwenden, so zeigte der Vergleich der Schülerlösungen durch Nennung unterschiedlicher, auch verworfener Hypothesen, dass Hypothesenbildung erfolgte. Die Wiederholbarkeit als Kriterium für Experimente zeigte sich dadurch, dass in beiden Erprobungen ähnliche Ergebnisse zu beobachten waren, die in der zweiten durch gezielte Vorbereitung verständiger diskutiert wurden.

Um der fehlenden Erfahrung der Schüler mit der systematischen Erkundung von Informatiksystemen zu begegnen, konnte eine weitere, auf dem Entwurfsmuster Proxy als Wissensrepräsentation basierende lernförderliche Software eingesetzt werden. Dabei unterstützte der Ansatz der auf dem Proxymuster basierenden Software die Erstellung einer in der Struktur unveränderten Software mit einem anderen Kontext. Es handelte sich um eine Arztpraxissoftware, die von den Schülern systematisch erkundet, und in der Zugriffskontrolle auf Patienten (-daten) durch eine Arzthelferin realisiert wurde (→ Kriterium 2: Zweck und Einsatzgebiet; S. 172). Das in der Klassifikation der Entwurfsmuster genannte Kriterium der Komplexität (→ Kriterium 5: Komplexität; S. 173) liefert dabei Hinweise auf Variationen des Proxymusters, so dass z. B. Schwerpunktverlagerungen auf weitere fundamentale Ideen im Proxymuster (→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik; S. 172) vorgenommen werden können (Stechert 2006c) oder die Anzahl der Zugriffe durch den Stellvertreter gezählt werden kann. Das wieder kehrende Schema des Proxy unterstützte den Lernprozess, da ein Transfer zu einer weiteren Situation vorgenommen wurde. Dadurch bietet sich die Gelegenheit, andere Schwerpunkte zu setzen, denn die Betrachtung des Schnittstellenverhaltens lässt die weiteren in den Entwurfsmustern vorkommenden vernetzten fundamentalen Ideen hervortreten, z. B. Schnittstelle, Parametrisierung und dynamisches Binden. Den Bedarf

an Aufgaben, die den Transfer unterstützen, zeigte die Lernerfolgskontrolle in der ersten Unterrichtserprobung (Abschnitt 6.5.1).

Bezüglich eines Spiralcurriculums ist nun zu fragen, wie sich die systematische Erkundung in einem vertikalen, also mit der Zeit vertiefenden Charakter wiederkehrend ins unterrichtliche Geschehen integrieren lässt. Neben der Vielzahl möglicher Fehlersituationen und unerwarteten Systemreaktionen liegt der Schlüssel in komplexeren Anwendungskontexten und speziell dem Erkennen stärker werdender Beziehungen zwischen vernetzten fundamentalen Ideen.

Da dynamische Aspekte essentiell für Informatiksysteme sind, wurden Sequenzdiagramme eingeführt. In einem ersten Schritt mussten die Schüler alle an der Situation beteiligten Objekte identifizieren. Danach konnten sie vorgegebenen Quelltext identifizieren, der den entsprechenden Objekten entsprach. Dabei wurde besonders auf Hypothesen über die notwendige Objektkommunikation geachtet, d. h. den Austausch von Nachrichten ($S_{AC,1}$). Mit Hilfe der ersten einfachen Sequenzdiagramme waren die Schüler in der Lage, dynamisches Verhalten in der inneren Struktur und ggf. mit Einfluss auf das nach außen sichtbare Verhalten zu modellieren. Dabei unterstützte die Lernsoftware Pattern Park die ersten Schritte zu Sequenzdiagrammen ($S_{AB,1}$ und $S_{B,1}$)

8.3.3 Vernetzung mit dem Zustandskonzept

Ein wichtiger Vorteil von Entwurfsmustern wurde bislang nicht aufgegriffen: ihre Kombinierbarkeit mit weiteren Entwurfsmustern (→ Kriterium 4: Zusammenhänge mit anderen Strukturmodellen; S. 173). Da dies kein Selbstzweck ist, wurde für eine Fortsetzung im Unterricht ein Beitrag zur Erklärung des nach außen sichtbaren Verhaltens durch das Zustandskonzept analysiert, denn die Folge der Zustände beschreibt das Verhalten des Systems (Goos und Zimmermann 2006, S. 18). Zustände lassen sich in der Klassifikation der Unterrichtsinhalte nach Hubwieser und Broy als charakteristisch für alle Informatiksysteme klassifizieren und ihre Anwendung ist auch außerhalb von Informatiksystemen möglich (Hubwieser 2007a, S. 83).

Automatenmodelle betonen die Zustände und Zustandsübergänge, also dynamische Aspekte. Dies ermöglicht eine Beschreibung von Systemzuständen und Ereignisketten. Dabei ist jedoch darauf hinzuweisen, dass es sich um eine theoretische Betrachtung des Systemverhaltens handelt, die nicht ohne Schwierigkeiten mit dem nach außen sichtbaren Verhalten eines Systems gleichzusetzen ist. Dadurch ist eine Bildungsherausforderung beschrieben, die für Schüler das Potential bietet, mittels Zustandsautomaten eine Brücke vom nach außen sichtbaren Verhalten zur inneren Struktur des Informatiksystems zu schlagen. Zustandsautomaten stellen im Unterricht neben den Sequenzdiagrammen eine weitere Sicht auf Abläufe dar, um die kognitive Hürde der Formalisierung zur Dokumentation zu überwinden. Deshalb

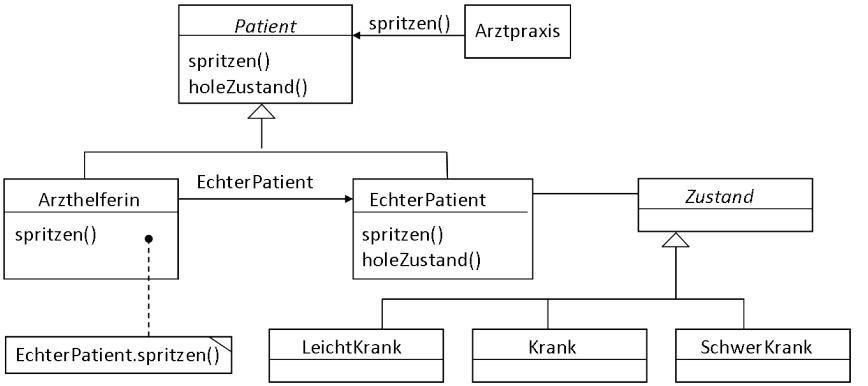
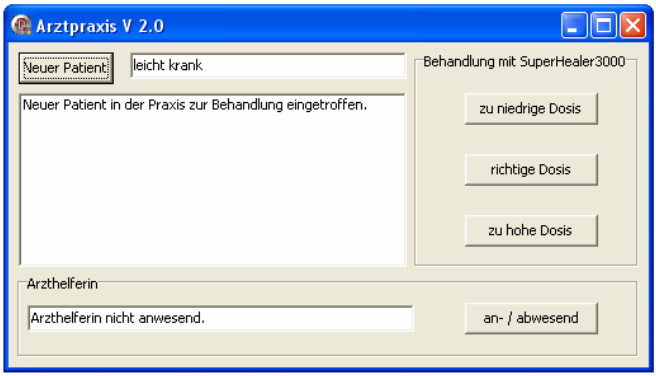


Abbildung 8.2: Klassendiagramm der Arztpraxis mit den Entwurfsmustern Proxy und Zustand

wurde eine Erweiterung der Arztpraxissoftware um das Zustandsmuster vorgenommen und im Unterricht eingesetzt. Abbildung 8.2 zeigt die Kombination von Proxy- und Zustandsmuster in der Arztpraxis.

Es wurde eine stark kontextualisierte Variante der systematischen Erkundung verwendet ($S_{A,2}$; Abbildung 8.3). Zustände des Programms sind abhängig von Gesundheitszuständen der Patienten. Streng genommen muss zwischen Systemzuständen und Zuständen der Patienten unterschieden werden. Für Schüler ist die im Informatiksystem vorhandene Zugriffskontrolle dahingehend ein Unterscheidungsmerkmal. Die Schüler erstellten anhand des Verhaltens des Programms Zustandsdiagramme in Abhängigkeit der Zugriffskontrolle (→ Kriterium 3: Vernetzte fundamentale Ideen der Informatik; S. 172). Somit erfolgte eine weitere Vernetzung. Bei der systematischen Erkundung erleichtert der Lebensweltbezug die Hypothesenbildung (→ Kriterium 6: Lebensweltbezug; S. 173). Die Schüler mussten in der Rolle des Arztes dem Patienten die richtige Medikamentendosis in Abhängigkeit von deren Gesundheitszustand geben.

Interessant für das Thema Informatiksysteme und Gesellschaft war die Diskussion der Schüler, in der sie hinterfragten, inwieweit eine durch ein Programm vorgegebene Vereinfachung der Lebenswelt hinzunehmen sei. Damit wurde ein wichtiges Ziel bezüglich des Einsatzes von Informatiksystemen erreicht. Direkt zusammen hängt damit die Erkenntnis, dass ein Informatiksystem nur einen Teil der Realität abbildet, also ein Modell darstellt. Informatiksysteme sind blind bezüglich solcher Probleme, die bei ihrer Entwicklung nicht bedacht wurden, d. h. ein Ereignis wie eine Eingabe resultiert in einem nicht angemessenen Systemzustand und ggf. in einem Fehlerzustand ((Winograd und Flores 1988, S. 166f), (Hubwieser 2007a, S. 47)).



1. Wie lautet der Name des Systems? (Erinnern; Faktenwissen)
2. Beschreiben Sie die Benutzungsoberfläche. Gehen Sie dabei nur auf die Veränderungen zu der Arztpraxis V1.0 ein. (Verstehen; Faktenwissen)
3. Was sind die wesentlichen Unterschiede zu dem Programm Arztpraxis V1.0? (Verstehen; Faktenwissen)
4. Heilen Sie einen Patienten, der im Zustand „krank" ist. Hinweis: Eine Krankenschwester ist nicht anwesend. Dokumentieren Sie Ihre Schritte und die Zustände des Patienten. (Anwenden; Verfahrensorientiertes Wissen)
5. In welchen Zuständen kann ein Patient zum Arzt kommen? (Analysieren; Begriffliches Wissen)
6. Ist es möglich einen Patienten im Zustand „krank" sofort zu heilen? Wie viele Schritte sind mindestens bis zur Genesung des Patienten nötig? (Analysieren; Begriffliches Wissen)
7. Ein Patient, der „schwer krank" ist, kommt in Ihre Praxis. Was müssen Sie als Arzt tun, damit er schnellstens wieder gesund wird? (Analysieren; Begriffliches Wissen)
8. In welchem Zustand befindet sich ein „kranker" Patient, wenn Sie ihm eine zu hohe Dosis geben? Hat sich ein Zustand verbessert oder verschlechtert? Retten Sie Ihren Ruf als Arzt und heilen den Patienten anschließend vollständig. Dokumentieren Sie Ihre gesamten Aktionen. (Analysieren; Begriffliches Wissen)
9. Versuchen Sie die folgende Darstellung nachzumachen. Dokumentieren Sie Ihre Schritte. Hinweis: Eine Krankenschwester ist nicht anwesend.
10. Erstellen Sie eine eigene solche Zustandsdarstellung (Zustandsdiagramm) für eine Krankheitsgeschichte eines Patienten. Hinweis: Eine Krankenschwester ist nicht anwesend. (Erstellen; Begriffliches Wissen)

Abbildung 8.3: Benutzungsoberfläche und Aufgabe zur Erkundung des Programms Arztpraxis V2.0 zu Zugriffskontrolle und Systemzuständen mit kognitivem Prozess und Wissensart gemäß Lernzieltaxonomie (Anderson und Krathwohl 2001)

In Vertiefungen in weiteren Unterrichtserprobungen könnten ähnliche Programme freiere Eingaben ermöglichen, deren Interpretation durch das Programm weitere nicht bedachte Aspekte offen legt. Nächster Schritt war eine Modellierung der in-

neren Struktur durch Zustandsdiagramme ($S_{B,2}$), um Quelltext mit dem Zustandsdiagramm in Verbindung zu bringen ($S_{AC,2}$). Bewusst wurde dazu ein Zugang über die Analyse des Systemverhaltens gewählt, statt anhand der Klassen, Operationen und Attribute.

8.3.4 Rolle der Lernsoftware

In dieser Unterrichtserprobung wurden drei unterschiedliche Arten von Software eingesetzt. Hauptanteil nahmen die kleinen Programme ein, die durch ein bis zwei Entwurfsmuster mit einer grafischen Benutzungsoberfläche erstellt wurden. Diese waren die Zugriffskontrolle anhand unterschiedlicher Zugriffsrechte, Arztpraxis V1.0 zur Zugriffskontrolle in einer Arztpraxis und die Arztpraxis V2.0, in der das Proxymuster mit dem Zustandsmuster kombiniert wurde. Vorteil dieser kleinen Programme ist, dass sie nur aus wenigen Klassen aufgebaut sind und ihre innere Struktur sowie ihr nach außen sichtbares Verhalten fast ausschließlich durch die Entwurfsmuster und die in ihnen vorhandenen fundamentalen Ideen der Informatik bestimmt ist.

Die zweite eingesetzte Software war die Lernsoftware Pattern Park. Diese will Kompetenzentwicklung mit Informatiksystemen dadurch fördern, dass sie fundamentale Ideen anhand von Entwurfsmustern erklärt und dabei insbesondere unterschiedliche Sichten anbietet, die an die Sichten auf Informatiksysteme angelehnt sind. Die beiden eingesetzten Module aus Pattern Park zur Zugriffskontrolle und zu den Zuständen bestehen aus Lebensweltdarstellungen (Geldentnahme an einem Geldautomaten im Freizeitpark bzw. Steuerung eines Achterbahnverkehrssystems), die durch Animationen und einfache Übungen mit Objekten der Lebenswelt durchgeführt werden können. Darüber hinaus gibt es Aufgaben mit der UML, die sowohl Klassen- und Sequenzdiagramme zur Zugriffskontrolle als auch Zustandsdiagramme umfassen. Die Schüler nahmen es positiv auf, dass Zugriffskontrolle und Zustände im Pattern Park über das gemeinsame Szenario eines Freizeitparks erklärt wurde.

Dritte eingesetzte Software war ein Werkzeug zur Erstellung und Simulation von Zustandsdiagrammen. Der Einsatz dieser Software war für die Schüler nicht verpflichtend, aber dennoch nutzen fast alle sie, da mit ihr die Ergebnisse über den Projektor präsentiert werden konnten. Die Simulationsfunktion verdeutlichte die Dynamik des Modells.

8.4 Evaluation

8.4.1 Auswertung der Lernerfolgskontrolle

Die Entwicklung der abschließenden schriftlichen Übung verdeutlichte wiederum das Dilemma zwischen möglichst objektiven Ergebnissen, die durch Multiple-Choice-Fragen erreicht werden können (Abschnitt 1.2.2) und dem Bedarf an komplexen

Anforderungssituationen, die für Kompetenzen kennzeichnend sind. Da schriftliche Übungen in dem Gymnasium nicht mehr als 30 Minuten umfassen durften, wurde der überwiegende Anteil der Lernerfolgskontrolle in Form von Multiple-Choice-Aufgaben gestaltet. Als Stimulus wurde beispielsweise ein Sequenzdiagramm verwendet. Abschließende Aufgaben waren das Modifizieren des Klassendiagramms des Proxymusters und die Erstellung eines Zustandsdiagramms bei gegebener textueller Beschreibung der Situation.

Inhaltlich umfasst die Lernerfolgskontrolle die Schritte der systematischen Erkundung, statische und dynamische Aspekte der Modellierung sowie Zusammenhänge zwischen den unterschiedlichen Sichten auf Informatiksysteme. Außerdem wurden für die Lernerfolgskontrolle zwei Varianten A und B der schriftlichen Übung angefertigt. Insgesamt nahmen zehn Schüler an der Lernerfolgskontrolle teil. Ergebnisse lagen zwischen 90% und 62% der maximal erreichbaren Punktzahl. Mittelwert lag bei etwa 79% und der Median bei etwa 80%. Wie vorab erwartet lagen die Schwierigkeiten der Schüler vor allem beim Modifizieren und Erstellen des Klassen- und Zustandsdiagramms. Offenbar gelingt es vielen Schülern nicht, von einer informellen Situationsbeschreibung zu einer formalen Darstellung in einem Diagramm zu kommen.

Nach Auswertung der Lernerfolgskontrolle bleibt es wie auch nach der ersten Unterrichtserprobung schwierig, Fehlern der Schüler eindeutig Fehlvorstellungen zuzuordnen. Beispielsweise ist wieder die Vererbung zur Realisierung der Zugriffskontrolle im Proxy ein Fehlerschwerpunkt, Vererbung selbst aber auch eine bekannte Lernschwierigkeit bei der Objektorientierung. Zuletzt ist anzumerken, dass fast alle Schüler die Aufgaben zur systematischen Erkundung korrekt lösten. Dennoch muss sie in ein komplexes Szenario integriert werden, um zu sehen, ob die Schüler die systematische Erkundung selbst anwenden können, und ob sie dadurch Vorteile bei einer nachfolgenden intensiveren Betrachtung des Verhaltens oder aber der inneren Struktur eines Informatiksystems haben. Von den beiden Lehramtsstudierenden, die unterrichteten, wurden im Rahmen ihres fachdidaktischen Praktikums in Zusammenarbeit mit dem Autor Vorschläge für zwei komplexere Aufgaben erarbeitet, die beispielsweise in einer Klausur gestellt werden können. Schwierigkeit bei Lernerfolgskontrolle und Klausur hinsichtlich einer Kompetenzüberprüfung war, dass sie gemäß der schulischen Rahmenbedingungen ohne Rechnereinsatz stattfinden mussten, wodurch die beschriebenen Anforderungssituationen von den Schülern vorstellungsmäßig nachzuvollziehen, aber nicht erlebbar waren.

8.4.2 Schriftliche Akzeptanzbefragung der Schüler

Auswertung der Akzeptanzbefragung

Zur Akzeptanzbefragung kam der Fragebogen zum Einsatz, der bereits zur ersten Unterrichtserprobung entworfen wurde (Abschnitt 6.5.1). Allein die wenigen Fragen, die die Selbsteinschätzung des Lernfortschritts zum Thema Iteration mit dem

Iteratormuster betrafen, wurden analog für das Thema Zustände gestellt. Nach der schriftlichen Übung wurde die Akzeptanzbefragung ergänzend vorgenommen, vor allem um motivationale Aspekte, die für die Kompetenzentwicklung notwendig sind, zu erkennen. Außerdem können insbesondere durch die Akzeptanzbefragung Rückschlüsse darauf gezogen werden, inwieweit die Entwurfsmuster geeignet sind, konsistent unterschiedliche Sichten auf Informatiksysteme zur ermöglichen. Die Befragung erlaubt Schlussfolgerungen, ob Schüler den Zusammenhang zwischen der fundamentalen Idee Zugriffskontrolle und dem Proxymuster sehen bzw. ob der Zusammenhang nicht wahrgenommen wurde. So müssen die Schüler ihren Lernfortschritt bezüglich Proxymuster und Zugriffskontrolle jeweils getrennt einschätzen. Wieder wurde der Akzeptanzfragebogen anonym ausgefüllt. Durch Angabe der Initialen der Eltern kann jedoch in etwaigen späteren Befragungen eine Zuordnung zur gleichen Person vorgenommen werden. Die Abstufung der Zustimmung zu den Aussagen auf den Fragebögen reicht von „trifft völlig zu", über „trifft eher zu" und „trifft eher nicht zu" bis „trifft nicht zu". Im nachfolgenden Text werden die Stufen zu „Zustimmung" und „Ablehnung" vereinfachend subsumiert und mit dem Ergebnis der ersten Erprobung verglichen (Ergebnis der zweiten Erprobung in Prozent / Ergebnis der ersten Erprobung in Prozent). Die Fragen und Ergebnisse befinden sich im Anhang A.2.

Neun Schüler gaben an, dass der Schwierigkeitsgrad angemessen war (69% / 70%) und drei Schüler (23% / 0%) meinten, er sei zu gering. Im Gegensatz dazu wurde bei der ersten Erprobung von 17% der Schüler angegeben, dass der Schwierigkeitsgrad zu hoch war. Der Stoffumfang wurde von neun Schülern (69% / 70%) als angemessen und von einem Schüler (8% / 26%) als hoch eingeschätzt. In der ersten Erprobung wurde jedoch von zehn Schülern aus der 23 Schüler umfassenden Klasse in schriftlichen Kommentaren angegeben, dass sie nicht ausreichend Zeit gehabt hätten, die gestellten Aufgaben zu lösen. Die Rückmeldungen aus der zweiten Erprobung sind daher bezüglich des Ziels, Stoffumfang und Schwierigkeitsgrad im Vergleich zur ersten Erprobung zu verringern, als positiv einzuschätzen.

In der Selbsteinschätzung des eigenen Lernfortschritts denken zwölf Schüler (92% / 70%), dass sie Fortschritte bezüglich des Analysierens von Informatiksystemen, elf (85% / 70%) gaben an, dass sie Fortschritte bezüglich der Prozesse in der inneren Struktur von Informatiksystemen und zehn (77% / 70%), dass sie Fortschritte bezüglich des Aufbaus von Informatiksystemen gemacht hätten.

Zwölf Schüler (92% / –) stimmten zu, dass sie nun um Zustände von Informatiksystemen wissen, und die gleiche Anzahl sagte, sie hätte das Proxymuster verstanden (92% / 100%). Elf Schüler (85% / 70%) sind der Meinung, den Zusammenhang zwischen den beiden Sichten, nach außen sichtbares Verhalten und innere Struktur, zu verstehen. Da dieses Resultat sehr wichtig für die Frage ist, ob Entwurfsmuster geeignet sind, unterschiedliche Perspektiven auf Informatiksysteme zu kombinieren, seien an dieser Stelle auch die ablehnenden Stimmen betrachtet: Ein Schüler (8%

/ 22%) stimmt der Aussage eher nicht zu und ein Schüler gibt an wegen sechs versäumter Stunden die Aussage nicht beurteilen zu können (8% / 0%). Kein Schüler lehnt die Aussage völlig ab.

Der Aussage, dass Lernmaterialen und Lernsoftware ausreichend und in guter Qualität vorliegen, wurde von elf Schülern unterstützt (85% / 74%). Dies liegt sicherlich auch an den überarbeiteten Programmen zur Zugriffskontrolle und zu Zuständen, die auf Entwurfsmuster basieren, aber auch an der Lernsoftware Pattern Park. Letztere wurde in der ersten Unterrichtserprobung noch anhand von prototypischen Modulen im Unterricht erprobt, während bei der zweiten Unterrichtsintervention bereits die vollständige Software vorlag, in deren weitere Entwicklung die Ergebnisse der ersten Erprobung eingeflossen sind.

In Übereinstimmung damit, dass es kein Lernziel war, gaben zehn Schüler (77% / 69%) an, dass sie nichts oder nicht viel zur Delphi-Programmierung hinzugelernt hätten. Neun Schüler (69% / 70%) sind der Meinung, für sich etwas dazugelernt zu haben, und zehn Schüler (77% / 69%) vermuten, dass die gelernten Inhalte für den Umgang mit Informatiksystemen nützlich sind. Im Gegensatz dazu denken nur fünf Schüler (38% / 43%), dass die bearbeiteten Aufgaben in der Unterrichtssequenz ihnen auch im täglichen Leben helfen werden. Neun Schüler (69% / 65%) stimmten der Aussage zu, dass ihnen die Konzentration auf die Aufgaben leicht fiel. Dieses Ergebnis ist insofern positiv zu bewerten, als dass die Angaben der Schüler aus der ersten Unterrichtserprobung nicht wesentlich davon abweichen, so dass die kleinere Schülerzahl während der zweiten Unterrichtserprobung nicht zu großen Einfluss auf das Lernverhalten hatte. Zehn Schüler (77% / 39%) sagten darüber hinaus, dass sie Spaß daran fanden, ihr Wissen über Informatiksysteme zu vertiefen. Somit ist es offenbar gelungen, die motivationalen Aspekte stärker zu fördern, was für die Kompetenzentwicklung unabdingbar ist. Dennoch bleibt es eine weitere Herausforderung, Anforderungssituationen mit stärkerem Bezug zur Lebenswelt der Schüler zu integrieren.

8.4.3 Auswertung des Interviews mit der Informatiklehrperson

Die verantwortliche Informatiklehrperson, die in allen Unterrichtsstunden der Erprobung anwesend war, stand abschließend für ein Leitfrageninterview zur Verfügung. Ihrer Meinung nach war der Schwierigkeitsgrad der Sequenz angemessen.

Die Notwendigkeit wiederholter Sichtenwechsel zwischen systematischer Erkundung des Informatiksystems und der Modellierung der inneren Struktur wurde von ihr unterstützt. Insbesondere merkte sie an, dass die Schüler davon profitierten, mit der Erkundung der Informatiksysteme zu starten, da diese die enaktive Repräsentationsebene adressiert. Als Verbesserungen schlug sie vor, komplexere Aufgaben zu formulieren, die Schüleraktivitäten weiter in den Vordergrund zu stellen und

den Theorieanteil etwas zu reduzieren. Die Unterrichtsmethodik bezüglich der oft eingesetzten Partner- und Gruppenarbeit wurde befürwortet.

Abschließend verglich sie ihren Kurs mit dem des Parallelkurses in dem keine Unterrichtsintervention stattfand. Sie zeigte sich erfreut darüber, dass die Schüler durch die Intervention zur Kompetenzentwicklung mit Informatiksystemen die Informatik als ein sehr facettenreiches Fach kennen gelernt haben, das mehr ist als nur programmieren. Gerade die Analyse bestehender Informatiksysteme gekoppelt mit Fragen der Modellierung seien wichtig, denn Informatik sei nicht nur Modellierung und Programmerstellung.

In einem Beitrag für das Kolloquium des Teilprojektes A8 – „Informatikunterricht und E-Learning zur aktiven Mitwirkung am digitalen Medienumbruch" im DFG Sonderforschungsbereich / Forschungskolleg 615 „Medienumbrüche" reflektiert die Informatiklehrperson die Unterrichtserprobung aus ihrer Sicht. Der Artikel ist gemeinsam verfasst mit einem zweiten Informatiklehrer der Kooperationsschule, der sowohl die Vorbereitungen für die erste als auch für die zweite Erprobung mitgestaltete. Darin werden folgende Kritikpunkte genannt:

> - „Das Abstraktionsniveau war bei der Aufbereitung der beiden Themen zu hoch und der theoretische Anteil zu groß.
> - Am Anfang der Unterrichtsreihe stand das selbständige Handeln der Schülerinnen und Schüler zu wenig im Mittelpunkt. Das verhinderte anfangs ein wirkliches Verstehen des neuen Begriffs Entwurfsmuster.
>
> Die kritischen Aspekte wurden in zahlreichen Nachbesprechungen mit den Lehramtsstudierenden und dem Koordinator erläutert. Die Kritik wurde positiv aufgenommen, so dass sich der frontal gerichtete Unterricht zugunsten eines handlungsorientierten Unterrichts entwickelte" (Ganea und Koch 2008, S. 119).

Die beiden Informatiklehrpersonen schließen damit, dass sich durch die drei an der Schule durchgeführten Unterrichtsreihen, zwei zum Unterrichtsmodell „Kompetenzentwicklung mit Informatiksystemen" und eine zu Internetworking (Freischlad 2007), nach ihren Beobachtungen eine veränderte Wahrnehmung der Informatik bei den Schülern feststellen lässt:

> In den ersten drei Halbjahren des Informatikstudiums [Informatikunterricht an der Schule; Anm. d. V] verengte sich das Bild über die Informatik sehr auf Programmiertechniken und Modellierung.
> Die im Rahmen des Forschungsprojektes durchgeführten Unterrichtsreihen veränderten dieses Bild. Die Schülerinnen und Schüler haben die Erfahrung gemacht, dass Informatik mehr ist als nur programmieren. Das systematische Erkunden von Informatiksystemen, das im zweiten Halbjahr der Jgst. 12 und in der Jgst. 13 weiter vertieft wird, ermöglicht ein besseres Verständnis für die Vernetzung der zahlreichen Bereiche der Informatik und somit der Wahrnehmung der Informatik als ‚Wissenschaft von Entwurf und Gestaltung von Informatiksystemen' (Claus und Schwill 2006, S. 314)" (Ganea und Koch 2008, S. 119).

Die Kooperation zwischen Schule und Universität zur Förderung der Kompetenzentwicklung mit Informatiksystemen wurde im Herbst 2008 fortgesetzt (Abschnitt 9.3).

8.5 Zusammenfassung und Diskussion der Ergebnisse der zweiten Unterrichtserprobung

8.5.1 Zusammenfassung der zweiten Unterrichtserprobung

In dem vorgestellten Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen werden Verhalten, innere Struktur und Implementierungsaspekte als Perspektiven auf Informatiksysteme genutzt, um die Systeme in Form lernförderlicher Software planvoll und handlungsorientiert zu erkunden. Eine Schlüsselrolle nehmen Strukturmodelle, z. B. Entwurfsmuster, ein (Stechert und Schubert 2007). Die lernförderliche Software zur Zugriffskontrolle hat diesbezüglich exemplarischen Charakter, da sie auf dem Proxymuster basiert, und die Dualität von Verhalten und Struktur eine systematische Erkundung unterstützt. In den Unterrichtsprojekten wurde die lernförderliche Software genutzt, um Problemstellen zu bewältigen. Diese waren (1) die Brücke zwischen der Perspektive des nach außen sichtbaren Verhaltens und der inneren Struktur, (2) fehlende Erfahrung der Schüler mit systematischen Erkundungen des Systemverhaltens, (3) mehrfacher Einsatz (von Variationen) eines Entwurfmusters in unterschiedlichen Programmen und Kontexten zur Förderung des Transfers und (4) die Vernetzung fundamentaler Ideen und Entwurfsmuster.

Zu (1) erwies sich die verfeinerte Strukturierung der Basiskompetenzen zu Informatiksystemen als geeignet, um bei der systematischen Erkundung der Software das Systemverhalten mit Aspekten ihrer inneren Struktur zu verbinden. Die gezielte Auswahl und der Einsatz von Aufgaben, die die Sichten zu Systemverhalten und innerer Struktur bzw. Systemverhalten und Implementierungsaspekten kombinierten, erforderte den Sichtenwechsel der Schüler bei gleichbleibendem Untersuchungsgegenstand, d. h. die Sichten waren aufeinander abgestimmt. Der fehlenden Erfahrung der Schüler mit einer systematischen Vorgehensweise zur Erkundung des Systemverhaltens (2) konnte durch explizite Thematisierung der Hypothesenbildung und -prüfung im Unterricht begegnet werden. Dabei unterstützte der Ansatz der auf dem Entwurfsmuster Proxy aufbauenden lernförderlichen Software den Wissenstransfer der Schüler: Eine strukturell unveränderte Software mit einem anderen Kontext, z. B. die Arztpraxis, konnte im Unterricht ohne großen Aufwand für die Lehrperson eingesetzt werden (3). Die Vernetzung fundamentaler Ideen (4) konnte einerseits durch den Einsatz solcher Variationen der im Kern unveränderten Software mit anderer Schwerpunktsetzung im Unterricht erfolgen. Andererseits ist es möglich, durch Kombination mit weiteren Entwurfsmustern, z. B. Zustandsmuster, eine für Kompetenzentwicklung mit Informatiksystemen förderliche Software zu gestalten. Bei einer Erkundung des Verhaltens des erweiterten Systems können die Schüler auf vorherige Erkenntnisse aufbauen.

Darüber hinaus wurde versucht, auf die motivationalen Kompetenzaspekte weiter einzugehen. Die Akzeptanzbefragung lässt im Vergleich mit der ersten Unterrichts-

erprobung Fortschritte bezüglich der Schülermotivation erkennen, dennoch scheint es noch nicht gelungen zu sein, den Bezug zur Lebenswelt der Schüler stärker herzustellen. Möglicherweise müssen dafür insbesondere mehr typische Repräsentanten von Informatiksystemen thematisiert werden.

Fazit ist, dass sich die Software auf Grundlage von Entwurfsmustern, speziell das Proxymuster, in Kombination mit einer systematischen Vorgehensweise zur Erkundung von Informatiksystemen im Informatikunterricht bewährt hat. Über das Proxymuster kann die lernförderliche Software sowohl auf ihr nach außen sichtbares Verhalten als auch auf ihre für die informatische Bildung angemessene innere Struktur überprüft werden. Die Unterrichtsprojekte zeigten, dass Schüler einen experimentierenden Zugang über die lernförderliche Software annehmen und die darin enthaltene Vernetzung fundamentaler Ideen der Informatik entdecken.

8.5.2 Informatiksysteme und Kompetenzentwicklung in der zweiten Unterrichtserprobung

Im Folgenden wird kurz resümiert, wie die zweite Unterrichtserprobung die Kompetenzentwicklung mit Informatiksystemen unterstützt hat. Analog zur ersten Erprobung war die systematische Erkundung eines Informatiksystems eine typische Schülertätigkeit. Neben der Zugriffskontrolle wurden Systemzustände besonders betont. Letztere sind für die Hauptfunktion „Coordination" nach (Denning 2007) hilfreich, denn mit Ihnen lassen sich das Eingabe- und Ausgabeverhalten von Systemen hinsichtlich der Arbeitsprozesse beschreiben, z. B. in der Arztpraxis. Wie in der ersten Erprobung war die Sicht „Design" durch Analyse der Entwurfsmusterstrukturen ein Schwerpunkt. Das gleiche gilt für „Automation" durch Fragen zu den Grenzen der Automatisierung während der systematischen Erkundung von Informatiksystemen. Zugriffskontrolle ist besonders für die Perspektiven „Coordination" und „Communication" relevant.

Hinsichtlich der von der UNESCO definierten Literacys ist wiederum ein Beitrag zur ICT Literacy und speziell zur Software Literacy zu erwarten (UNESCO 2008), denn Systemzustände lassen sich im Verhalten aller Informatiksysteme identifizieren. Die Verknüpfung von Zugriffskontrolle durch das Proxymuster mit dem Zustandsmuster lässt die Schüler unterschiedliche Systemzustände entsprechend der Zugriffsrechte von Anwendern erkennen. Insbesondere durch Klassifikation der Eingaben- und Ausgaben ist ein Zustandsmodell hilfreich, das Verhalten des Systems zu analysieren. Als Lernsoftware wurde neben Pattern Park und den Programmen, die auf Proxy- und Zustandsmuster bzw. deren Kombination basieren, eine Simulationssoftware genutzt, mit der Zustandsdiagramme erstellt und Zustandswechsel simuliert werden konnten. Damit ist wiederum ein Beitrag zur Media Literacy zu erwarten. Durch oben genannte Betrachtung der Grenzen der eingesetzten Programme in hypothetischen, komplexeren Situationen, wurde das kritische Denken der Schüler unterstützt.

Das Klassifizieren des Systemverhaltens durch Zustände fördert die Schlüsselkompetenz „Interaktive Anwendung von Technologien" (DeSeCo: 1c). Die Schüler mussten in der systematischen Erkundung unterschiedliche Zustände, die durch den Kontext des Programms bestimmt sind, z. B. Gesundheitszustände von Patienten, auf ihre Umsetzung im Informatiksystem analysieren. Dies ist eine wichtige Voraussetzung, um Information im System zu lokalisieren (DeSeCo: 1c). Neben der Förderung des eigenständigen Agierens (DeSeCo: 3) durch die systematische Erkundung – wie bereits bei der ersten Erprobung –, ermöglicht ein Zustandsmodell die Kommunikation über das System mit anderen und damit das Interagieren in heterogenen Gruppen (DeSeCo: 2b). Die Antizipation von Systemzuständen verringert Stress in individuellen Projekten der Schüler, so dass deren eigenständiges Handeln (DeSeCo: 3b) unterstützt wird.

Die Einordnung des Unterrichtsmodells in den Europäischen Qualifikationsrahmen wurde in Abschnitt 5.7 vorgenommen. Hinsichtlich der erreichten Niveaustufen der Schüler ist, wie in Abschnitt 6.6.2 begründet, keine fundierte Aussage möglich, da entsprechende qualitative und quantitative Messverfahren für den Informatikunterricht fehlen (Abschnitt 9.3).

9. Zusammenfassung, Fazit und Ausblick

9.1 Zusammenfassung

In der vorliegenden Arbeit wurde ein Unterrichtsmodell zur Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II vorgestellt. Der Bedarf wurde u. a. damit begründet, dass Informatiksysteme zu Beginn des 21. Jahrhunderts allgegenwärtig sind (Kapitel 1). Für die Kompetenzentwicklung sind Informatiksysteme in ihrer Einheit aus Hardware, Software und Vernetzung anhand ihres nach außen sichtbaren Verhaltens, der inneren Struktur und Implementierungsaspekten zu analysieren. Ausgehend vom Kompetenzbegriff (Kapitel 2) und dem Informatiksystembegriff (Kapitel 3) erfolgte eine Analyse des fachdidaktischen Forschungsstandes zur Kompetenzentwicklung mit Informatiksystemen. Die Ergebnisse lassen sich in die Bereiche (1) Bildungsziele zu Informatiksystemen, (2) Unterrichtsinhalte, (3) Lehr-Lernmethodik und (4) Lehr-Lernmedien aufteilen (Kapitel 4).

In Kapitel 5 wurde die Unterrichtsmodellentwicklung beschrieben. Ziel war es, ein fachdidaktisch begründetes Konzept zu konstruieren, das Erkenntnisse aus der Lehr-Lerntheorie, der Fachwissenschaft, der Erziehungswissenschaft und anderen Fachdidaktiken aufgreift und mit informatikdidaktischen Konzepten

- zur Vernetzung der Grundbegriffe und Wirkprinzipien sowie zur fachdidaktischen Kommunikation (Wissensrepräsentation für vernetzte fundamentale Ideen),
- zur Kategorisierung von Bildungszielen und -inhalten (Perspektiven auf Informatiksysteme; Strategie zur Strukturierung) und

- zur Umsetzung im unterrichtlichen Geschehen (Gestaltung von lernförderlicher Software und Schülervorgehensweisen mit Handreichungen zu Motivation und Durchführung von Experimenten)

verknüpft.

Den Zugang zu Informatiksystemen bildet in der vorliegenden Dissertationsschrift das nach außen sichtbare Verhalten. Es erfolgte eine Fokussierung auf vernetzte fundamentale Ideen der Informatik und Strukturmodelle von Informatiksystemen als Unterrichtsinhalte. Es wurde begründet, dass ausgewählte objektorientierte Entwurfsmuster vernetzte fundamentale Ideen repräsentieren. In Abschnitt 5.4 wurden dementsprechend Entwurfsmuster als Wissensrepräsentation für vernetzte fundamentale Ideen klassifiziert. Das systematische Erkunden des Verhaltens von Informatiksystemen wird im Informatikunterricht bisher kaum thematisiert. Deshalb wurden Schülertätigkeiten in Anlehnung an Unterrichtsexperimente hergeleitet, die Schüler unterstützen, Informatiksysteme bewusst anzuwenden (Abschnitt 5.5). Durch diese Lehr-Lernmethodik werden das nach außen sichtbare Verhalten von Informatiksystemen, im Sinne einer Black-Box, und das Wechselspiel von Verhalten und Struktur bei vorliegender Implementierung des Systems als White-Box analysiert. Die Adressierung schrittweise höherer kognitiver Niveaustufen wurde in die Konzeption einbezogen. Unterstützend wurde für das Unterrichtsmodell lernförderliche Software gestaltet, die vernetzte fundamentale Ideen in Entwurfsmustern und das Experimentieren aufgreift (Abschnitt 5.6). Schwerpunkte bilden im Unterrichtsmodell zwei Arten von lernförderlicher Software: (1) Die Lernsoftware Pattern Park wurde von einer studentischen Projektgruppe entwickelt. In ihr können in Entwurfsmustern enthaltene fundamentale Ideen der Informatik über ihren Lebensweltbezug im Szenario eines Freizeitparks erschlossen werden. (2) Als weitere Art Lernsoftware wurden kleine Programme eingesetzt, deren innere Struktur durch ausgewählte Entwurfsmuster gebildet und deren Verhalten direkt durch die darin enthaltenen fundamentalen Ideen bestimmt wird. Diese Programme können durch die Experimente im Unterricht systematisch untersucht werden.

Mit dem Ziel, die normative Perspektive um Rückkopplung mit der Praxis zu ergänzen, wurden zwei Erprobungen im Informatikunterricht vorgenommen. Diese lieferten Erkenntnisse zur Machbarkeit des Unterrichtsmodells und dessen Akzeptanz durch die Schüler (Kapitel 6 und 8). Exemplarisch umgesetzt wurden die Themen Zugriffskontrolle mit dem Proxymuster, Iteration mit dem Iteratormuster und Systemzustände mit dem Zustandsmuster. Der intensive Austausch mit Informatiklehrpersonen in der Kooperationsschule über Informatiksysteme und Kompetenzentwicklung sowie die Durchführung von zwei Lehrerfortbildungen ergänzen die Beobachtungen im unterrichtlichen Geschehen (vgl. (Schubert et al. 2007), (Freischlad und Stechert 2008)). Die erste Unterrichtserprobung resultierte in einer Weiterentwicklung des Unterrichtsmodells zu Informatiksystemen und Kompetenzent-

wicklung (Kapitel 7). Darin erfolgte eine Fokussierung auf das nach außen sichtbare Verhalten von Informatiksystemen und eine Verfeinerung der Perspektiven auf innere Struktur und ausgewählte Implementierungsaspekte. Anschließend wurde die zweite Unterrichtserprobung durchgeführt und evaluiert (Kapitel 8). Am Schluss der Forschungsarbeit steht ein in empirischen Phasen erprobtes Unterrichtsmodell. Abbildung 9.1 fasst das Unterrichtsmodell schematisch zusammen.

9.2 Fazit

Das Erreichte ist mit den Forschungszielen und wissenschaftlichen Fragestellungen zu vergleichen. Die Fragestellungen lauteten (Abschnitt 4.4.6):

1. Wie ist ein Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen gemäß der Ergebnisse der Analyse des fachdidaktischen Forschungsstandes zu gestalten?
2. Wie kann das Unterrichtsmodell im Informatikunterricht der Sekundarstufe II exemplarisch erprobt werden?

Zu (1): Ausgehend von den in der Literatur dokumentierten fachlichen Aspekten zu Informatiksystemen und den fachdidaktisch formulierten Bildungszielen, -inhalten, -methoden und -medien zur Kompetenzentwicklung mit Informatiksystemen wurde ein Unterrichtsmodell entwickelt. Zur Fokussierung des Unterrichts auf Informatiksysteme wurden der Blick auf das System und der Blick auf die innere Struktur des Systems als Perspektiven gewählt. Der besondere Beitrag der Arbeit wird in der Vernetzung fundamentaler Ideen der Informatik gesehen. Die Begründung ausgewählter Entwurfsmuster als Wissensrepräsentation vernetzter fundamentaler Ideen liefert zum einen eine Vorgehensweise zur Analyse potentieller weiterer Wissensrepräsentationen. Zum anderen werden als Nebeneffekt ausgewählte Entwurfsmuster für die allgemein bildende Schulinformatik einsetzbar ohne die problematische Begründung über deren Nützlichkeit im Softwareentwicklungsprozess. Ein weiterer Beitrag der Arbeit zur Fachdidaktikforschung wird in der Verknüpfung der Unterrichtsexperimente als zentrale Schülertätigkeit mit lernförderlicher Software gesehen, die Entwurfsmuster als strukturelle Grundlage enthalten und vernetzte fundamentale Ideen repräsentieren. Deren Beitrag zur Förderung der Kompetenzentwicklung mit Informatiksystemen wurde dargestellt. Bei der Entwicklung des Unterrichtsmodells wurde auf Stimmigkeit der Inhalte, Realisierbarkeit im unterrichtlichen Geschehen und fortwährend auf den Beitrag zur Kompetenzentwicklung (Kapitel 2) und den in Kapitel 3 analysierten Eigenschaften von Informatiksystemen geachtet. Andere vorhandene Unterrichtsreihen, z. B. zu Internet und Datenbanksystemen, können anhand der in dem vorliegenden Unterrichtsmodell

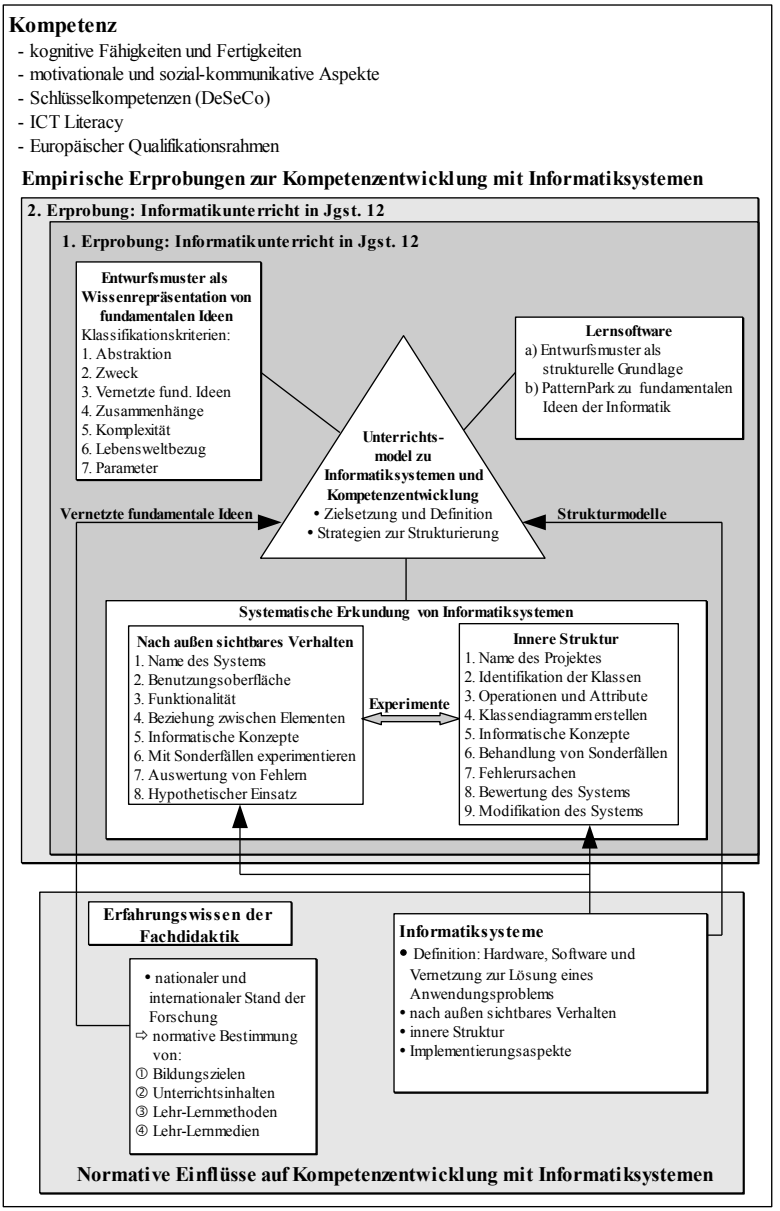


Abbildung 9.1: Schematische Darstellung der normativen und empirischen Einflüsse auf das Unterrichtsmodell zur Kompetenzentwicklung mit Informatiksystemen

entwickelten Aspekte analysiert werden. Gleichzeitig sind von diesen Spezialisierungen mögliche Ergänzungen und Verfeinerungen des Unterrichtsmodells zu erwarten, z. B. zum Thema Internetworking (vgl. Freischlad und Schubert 2007).

Zu (2): Für die exemplarische Erprobung des Unterrichtsmodells wurden die Themen Zugriffskontrolle mit dem Proxymuster, Iteration mit dem Iteratormuster und Systemzustände mit dem Zustandsmuster ausgewählt. Die Kombination der Sicht auf das System mit der Sicht in das System wurde betrachtet. Damit konnte die Verbindung der Unterrichtsexperimente mit lernförderlicher Software exemplarisch umgesetzt werden. Die vernetzten fundamentalen Ideen wiesen einen starken Bezug zu Informatiksystemen auf. Dazu war es notwendig, sie in Bezug zu den Erkenntnissen zum Informatiksystembegriff zu setzen: Für Zugriffskontrolle konnte mit ihrer Relevanz in mehreren Kategorien der Informatik bzw. im Sinne von Hauptfunktionen von Informatiksystemen argumentiert werden (Kapitel 3), z. B. Zugriff auf Daten (Recollection). Hinsichtlich der Klassifikation von Unterrichtsinhalten nach ihrer Allgemeingültigkeit (Hubwieser 2007a, S. 83) kann Zugriffskontrolle als charakteristisch für viele Informatiksysteme eingestuft werden, z. B. Mehrbenutzersysteme. Die Ergebnisse der Erprobungen wurden dokumentiert. Rückmeldungen von Informatiklehrern in zwei Workshops ergänzten die Resultate.

Die umfangreiche Analyse des fachdidaktischen Forschungsstandes zur Kompetenzentwicklung mit Informatiksystemen (Kapitel 4) dient als eine Grundlage bei der Entwicklung eines Kompetenzmodells für die Sekundarstufe II (Abschnitt 9.3). Die Forderung nach empirisch überprüften Kompetenzmodellen (Abschnitt 2.1) offenbart, dass die Entwicklung eines Unterrichtsmodells nicht bei einer theoretischen Fundierung durch Erfahrungswissen der Didaktik der Informatik stehen bleiben kann, um Schülerleistungen realistisch einschätzen zu können. Die vorliegende Arbeit bereitet den Weg für ein Kompetenzmodell zu Informatiksystemen durch Angabe eines theoretisch begründeten, durch normative Analyse des Forschungsstands praxisrelevanten Unterrichtsmodells, das exemplarisch gestaltet und im Unterricht erprobt wurde.

9.3 Ausblick

Das Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II ist entstanden, während Bildungsstandards für die Sekundarstufe I entwickelt wurden. Da Bildungsstandards für die Sekundarstufe II sich erst in Planung befinden, erhofft der Autor deren positive Beeinflussung durch die vorliegende Arbeit. In ihr ist die Förderung der Kompetenzentwicklung mit Informatiksystemen als eine zentrale Aufgabe des Informatikunterrichts der Sekundarstufe dargestellt. Es fehlen jedoch Instrumentarien zur Kompetenzmessung in der Informatik und ein entsprechendes Kompetenzmodell, das an repräsentativen Stichproben evaluiert wurde. Dazu bedarf es der Kooperation von Psychologen

und Informatikern. Die Deutsche Forschungsgemeinschaft fördert von 2008 bis 2010 an den Universitäten Paderborn und Siegen das Projekt „Entwicklung von qualitativen und quantitativen Messverfahren zu Lehr-Lern-Prozessen für Modellierung und Systemverständnis in der Informatik", an dem der Autor mitwirkt ((Schubert 2008), (Kollee et al. 2009)). Neben den Professuren für Didaktik der Informatik an beiden Hochschulen ist die Professur für Arbeits- und Organisationspsychologie der Universität Paderborn beteiligt. Das Projekt wurde vom Gutachtergremium der Ingenieurwissenschaften, einschließlich Informatik, befürwortet. Partnerschule der Universität Siegen ist das Gymnasium, mit dem bereits eine erfolgreiche Kooperationsbeziehung besteht, die durch die in der vorliegenden Dissertationsschrift beschriebenen Unterrichtserprobungen belegt ist.

Im Siegener Teilprojekt werden Unterrichtsprojekte zur Kompetenzentwicklung mit Informatiksystemen in der Sekundarstufe II angestrebt, die acht Wochen mit je drei Wochenstunden dauern (Stechert et al. 2009). Neben fundamentalen Ideen der Informatik sind weitere Strukturmodelle Schwerpunkte der Untersuchung, denn die vorliegende Dissertationsschrift zeigt, dass der Einsatz von Strukturmodellen einen wesentlichen Beitrag zu einer Kompetenzentwicklung mit Informatiksystemen liefern kann (Stechert und Schubert 2007). Ziel ist, das Zusammenwirken von Hardware, System- und Anwendungssoftware mit Netzverbindungen über Strukturmodelle miteinander in Beziehung zu setzen. Eingesetzt wird das Blockmodell des Von-Neumann-Rechners, das eine Synthese aus mathematisch-logischer Beschreibung und der technischen Umsetzung von Informatiksystemen repräsentiert. Es beschreibt jedoch nur die Vernetzung von Komponenten eines isolierten Einzelplatzrechners. Zur Ergänzung wird ein Ebenenmodell des Rechners genutzt, um die hierarchische Strukturierung von Systemkomponenten darzustellen und Komponenten zu gruppieren (Tanenbaum und Goodman 2001). Die zusätzlich eingesetzte 4-Schichtenarchitektur des Internets beschreibt die Kommunikationsaufgaben in Netzen anhand funktionaler Ebenen, denen die Netzwerkprotokolle zugeordnet sind (Peterson und Davie 2003). Inhaltliche Bruchstellen werden vermieden, da das Ebenenmodell als Begründung eines Bottom-up-Zugangs, von der Hardware zu den Anwendungen, oder eines Top-down-Zugangs genutzt werden kann. Dabei können die Schüler Erkenntnisse vom Von-Neumann-Blockmodell für einen Sichtenwechsel auf Schichtenmodelle nutzen und Strukturmodelle anwenden, um das nach außen sichtbare Verhalten von Informatiksystemen zu erklären.

Ein aktuelles Forschungsfeld bilden eingebettete Mikrosysteme, so dass sich die Frage ergibt, wie das Unterrichtsmodell zur Förderung der Kompetenzentwicklung mit Informatiksystemen erweitert werden kann und muss, um diesen hybriden Systemen (Claus und Schwill 2006, S. 677) gerecht zu werden: Mobile Kommunikationssysteme verfügen über einen zunehmenden Funktionsumfang, der durch das Zusammenspiel von Berechnungseinheiten mit Sensoren und Aktoren bereitgestellt wird. Das heißt, die Wandlung von digitalen Signalen in analoge Signale und umge-

kehrt beeinflusst das Systemverhalten maßgeblich ((Schubert und Stechert 2008), (Schwidrowski et al. 2009)). Ein entsprechendes Forschungsvorhaben wurde im Februar 2009 bei der Begehung des DFG Sonderforschungsbereich / Forschungskolleg 615 „Medienumbrüche" im Teilprojekt A8 „Informatikunterricht und E-Learning zur aktiven Mitwirkung am digitalen Medienumbruch" positiv begutachtet.

A. Anhang

A.1 Akzeptanzfragebogen der ersten Unterrichtserprobung

Im Folgenden sind die Ergebnisse der schriftlichen Befragung nach der zweiten Unterrichtserprobung zusammen gefasst (Kapitel 6). Befragt wurden 23 Schüler. Anmerkung: Halbe Punkte sind dann vergeben worden, wenn ein Schüler einen nicht vorhandenen Zwischenwert angekreuzt hat, gleichzeitig aber die Zuordnung der Aussage eindeutig war.

Tabelle A.1: Befragung zum Informatikunterricht allgemein

Befragung zum Informatikunterricht allgemein						
		trifft völlig zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	kann ich nicht beurteilen
1	Informatikunterricht macht mir Spaß.	2	14	2	5	0
2	Ich verstehe den Unterrichtsstoff im Fach Informatik.	2	11	9	1	0
3	Im Informatikunterricht geht es darum, Programme zu erstellen.	3	16,5	2,5	0	1
4	Im Informatikunterricht geht es darum, Problemstellungen zu modellieren.	8	9,5	3,5	0	2
5	Ich finde die Inhalte im Informatikunterricht interessant.	0	8	12	3	0
6	Ich beteilige mich am Unterricht.	2	8	9	2	2
7	Ich bin im Unterricht abgelenkt.	2	7	8,5	4,5	1

Tabelle A.2: Befragung zu Schwierigkeit, Stoffumfang und eigenem Lernen

Wie schätzen Sie den Schwierigkeitsgrad des vorgestellten Lernstoffs ein?					
		hoch	angemessen	niedrig	fehlende Angabe
8	Der Schwierigkeitsgrad war ...	4	17	2	0
		viel	angemessen	wenig	
9	Der Stoffumfang war ...	6	16	1	0
10	Ich habe im Unterricht gelernt ...	0	20	3	0

Tabelle A.3: Befragung zum konkreten Informatikunterricht

	Geben Sie bitte an, in welchem Maße Sie den Aussagen zustimmen.					
		ich stimme überhaupt nicht zu	ich stimme nicht zu	ich stimme etwas zu	ich stimme voll zu	kann ich nicht beurteilen
11	Darstellung und Veranschaulichung der Lerninhalte waren verständlich.	0	3	15	5	0
12	Das praktische Üben war ausreichend	1	3	10,5	8,5	0
13	Die gestellten Aufgaben waren lösbar.	0	1	12	9	1
14	Die Inhalte sind vermutlich für den Umgang mit Informatiksystemen sehr nützlich.	0	3	12	7	1
15	Die Verwendbarkeit und der Nutzen des behandelten Stoffes wurden deutlich.	1	3	10,5	8,5	0
16	Die Hilfsmittel zur Unterstützung des Lernens (z. B. Arbeitsblätter, Software) sind ausreichend und in guter Qualität vorhanden.	1	5	7	10	0

Tabelle A.4: Befragung zum Einfluss der Unterrichtsthemen auf motivationale und volitionale Bereitschaften sowie Einstellungen (eine fehlende Angabe in Zeile 22)

		Befragung zum Unterrichtsthema				
		trifft völlig zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	kann ich nicht beurteilen
17	Der Unterricht beschäftigte sich mit Aufgaben, die mir im täglichen Leben begegnen.	0	10	8	5	0
18	Ich konnte im Unterricht etwas Neues entdecken.	5	8	7	3	0
19	Einige Themen haben mich besonders interessiert.	2	5	12	4	0
20	Ich habe auch außerhalb des Unterrichts über die Aufgaben nachgedacht.	1	0	10	12	0
21	Ich konnte mich leicht auf die Sache konzentrieren.	4	11	5	3	0
22	Ich habe das Gefühl, für mich etwas dazugelernt zu haben.	1	15	4	2	0
23	Ich kann mir vorstellen, dass ich das erworbene Wissen in Zukunft gebrauchen kann.	0	11	6	5	1
24	Es hat mir Spaß gemacht, mein Verständnis für dieses Thema zu vertiefen.	1	8	11	3	0

Tabelle A.5: Befragung zur Einschätzung des eigenen Lernfortschritts

	Wie schätzen Sie selbst Ihren Lernfortschritt ein?				
		Ich konnte viel dazulernen	Ich konnte einiges dazulernen	Ich konnte nicht viel dazulernen	Die Inhalte brachten mir keine neuen Erkenntnisse
25	(a) Analyse von Informatiksystemen	5	11	6	1
26	(b) Beschreibung von Abläufen in einem Informatiksystem	2	17	4	0
27	(c) Beschreibung des Aufbaus eines Informatiksystems	3	13	7	0
28	(d) Bausteine der Softwareentwicklung (Entwurfsmuster)	5	12	5	1
29	(e) Zugriffskontrolle	5	13	3	2
30	(f) Durchlaufen einer Schlange	5	12	6	0
31	(g) Delphi-Programmierung	1	6	11	5

Tabelle A.6: Befragung zur Einschätzung des eigenen Lernens

	Geben Sie bitte an, in welchem Maße Sie den Aussagen zustimmen.					
		trifft völlig zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	kann ich nicht beurteilen
32	Ich habe verstanden, wie man das nach außen sichtbare Verhalten von Informatiksystemen systematisch erkundet.	11	8	4	0	0
33	Ich habe verstanden, wie man die innere Struktur von Informatiksystemen.	6	11	5	1	0
34	Ich weiß, wie innere Struktur und Verhalten eines Informatiksystems zusammenhängen.	4	14	5	0	0
35	Ich habe verstanden, was ein Iterator ist.	14	8	1	0	0
36	Ich habe verstanden, was ein Proxy ist.	15,5	7,5	0	0	0

A.2 Akzeptanzfragebogen der zweiten Unterrichtserprobung

Im Folgenden werden die Ergebnisse der schriftlichen Befragung nach der zweiten Unterrichtserprobung offen gelegt (Kapitel 8). Befragt wurden 13 Schüler.

Tabelle A.7: Befragung zum Informatikunterricht allgemein

Befragung zum Informatikunterricht allgemein						
		trifft völlig zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	kann ich nicht beurteilen
1	Informatikunterricht macht mir Spaß.	1	8	4	0	0
2	Ich verstehe den Unterrichtsstoff im Fach Informatik.	4	7	2	0	0
3	Im Informatikunterricht geht es darum, Programme zu erstellen.	4	4	5	0	0
4	Im Informatikunterricht geht es darum, Problemstellungen zu modellieren.	6	4	2	0	1
5	Ich finde die Inhalte im Informatikunterricht interessant.	2	5	4	2	0
6	Ich beteilige mich am Unterricht.	4	4	5	0	0
7	Ich bin im Unterricht abgelenkt.	0	2	4	7	0

Tabelle A.8: Befragung zu Schwierigkeit, Stoffumfang und eigenem Lernen

Wie schätzen Sie den Schwierigkeitsgrad des vorgestellten Lernstoffs ein?					
		hoch	angemessen	niedrig	fehlende Angabe
8	Der Schwierigkeitsgrad war ...	0	9	3	1
		viel	angemessen	wenig	0
9	Der Stoffumfang war ...	1	9	2	1
10	Ich habe im Unterricht gelernt ...	4	8	0	1

Tabelle A.9: Befragung zum konkreten Informatikunterricht

	Geben Sie bitte an, in welchem Maße Sie den Aussagen zustimmen.					
		ich stimme überhaupt nicht zu	ich stimme nicht zu	ich stimme etwas zu	ich stimme voll zu	kann ich nicht beurteilen
11	Darstellung und Veranschaulichung der Lerninhalte waren verständlich.	0	1	3	8	1
12	Das praktische Üben war ausreichend	0	1	11	0	1
13	Die gestellten Aufgaben waren lösbar.	0	1	3	8	1
14	Die Inhalte sind vermutlich für den Umgang mit Informatiksystemen sehr nützlich.	0	2	1	9	1
15	Die Verwendbarkeit und der Nutzen des behandelten Stoffes wurden deutlich.	0	1	7	4	1
16	Die Hilfsmittel zur Unterstützung des Lernens (z. B. Arbeitsblätter, Software) sind ausreichend und in guter Qualität vorhanden.	0	2	5	6	0

Tabelle A.10: Befragung zum Einfluss der Unterrichtsthemen auf motivationale und volitionale Bereitschaften sowie Einstellungen

		Befragung zum Unterrichtsthema				
		trifft völlig zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	kann ich nicht beurteilen
17	Der Unterricht beschäftigte sich mit Aufgaben, die mir im täglichen Leben begegnen.	1	4	3	4	1
18	Ich konnte im Unterricht etwas Neues entdecken.	5	6	2	0	0
19	Einige Themen haben mich besonders interessiert.	3	5	4	0	1
20	Ich habe auch außerhalb des Unterrichts über die Aufgaben nachgedacht.	1	4	3	4	1
21	Ich konnte mich leicht auf die Sache konzentrieren.	2	7	0	1	3
22	Ich habe das Gefühl, für mich etwas dazugelernt zu haben.	2	7	3	0	1
23	Ich kann mir vorstellen, dass ich das erworbene Wissen in Zukunft gebrauchen kann.	2	4	4	2	1
24	Es hat mir Spaß gemacht, mein Verständnis für dieses Thema zu vertiefen.	2	8	2	0	1

Tabelle A.11: Befragung zur Einschätzung des eigenen Lernfortschritts

	Wie schätzen Sie selbst Ihren Lernfortschritt ein?				
		Ich konnte viel dazulernen	Ich konnte einiges dazulernen	Ich konnte nicht viel dazulernen	Die Inhalte brachten mir keine neuen Erkenntnisse
25	(a) Analyse von Informatiksystemen	4	8	1	0
26	(b) Beschreibung von Abläufen in einem Informatiksystem	4	7	2	0
27	(c) Beschreibung des Aufbaus eines Informatiksystems	5	5	3	0
28	(d) Bausteine der Softwareentwicklung (Entwurfsmuster)	3	8	2	0
29	(e) Zugriffskontrolle	3	6	3	1
30	(f) Zustände von Informatiksystemen	4	8	1	0
31	(g) Delphi-Programmierung	0	3	6	4

Tabelle A.12: Befragung zur Einschätzung des eigenen Lernens

	Geben Sie bitte an, in welchem Maße Sie den Aussagen zustimmen.					
		trifft völlig zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	kann ich nicht beurteilen
32	Ich habe verstanden, wie man das nach außen sichtbare Verhalten von Informatiksystemen systematisch erkundet.	8	4	0	0	1
33	Ich habe verstanden, wie man die innere Struktur von Informatiksystemen.	2	8	2	0	1
34	Ich weiß, wie innere Struktur und Verhalten eines Informatiksystems zusammenhängen.	2	9	1	0	1
35	Ich habe verstanden, was ein Zustand eines Informatiksystems ist.	6	6	0	0	1
36	Ich habe verstanden, was ein Proxy ist.	7	5	0	0	1

A.3 Unterrichtsmaterialien und studentische Arbeiten

Um den Umfang der vorliegenden Dissertationsschrift einzuschränken, existiert zu dem Dissertationsprojekt eine Webseite: http://www.die.informatik.uni-siegen.de/informatics-systems/index.html. Die folgenden Materialien und Dokumente sind über die Webseite erreichbar:

- Materialien der ersten und zweiten exemplarischen Unterrichtserprobung mit jeweils acht Unterrichtsbeschreibungen inklusive Aufgabenstellungen;
- Lernsoftware Pattern Park;
- ausgewählte schriftliche Arbeiten von Studierenden:
  - Diplomarbeit von Ufer inklusive Programme zu Architekturmustern (Ufer 2007),
  - Diplomarbeit von Weyer (Weyer 2007b),
  - Hauptseminararbeiten (Weyer 2007a), (Sülz 2007), (Stupperich und Warkentin 2007), (Graf 2008), (Gerding 2008), (Dittich 2008);
- Lernförderliche Software zu den Entwurfsmustern Proxy, Iterator und Zustand;
- Publikationen des Autors.

Des Weiteren ist die Dissertationsschrift als PDF-Dokument verfügbar.

Literatur

ACM 1993
ACM: *Model High School Computer Science Curriculum*. New York : ACM (Association for Computing Machinery), 1993. – ISBN 0–89791–607–7

ACM 2006
ACM ; Tucker, Allen (Hrsg.): *A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee*. 2nd Edition. New York : Association for Computing Machinery (ACM), 2006. – ISBN 59593–596–7. – http://www.csta.acm.org/Curriculum/sub/CurrFiles/K-12ModelCurr2ndEd.pdf – geprüft: 22. Februar 2009

Aebli 1980
Aebli, Hans (Hrsg.): *Denken: Das Ordnen des Tuns. Band 1. Kognitive Aspekte der Handlungstheorie*. 1. Aufl. Klett-Cotta, Stuttgart, 1980. – ISBN 3–12–930120–8

Aiken und Sandås 2001
Aiken, Robert M. ; Sandås, Cheryl: A Lifelong Learning Courses Based on Learning Objectives. In: **(Watson und Andersen 2002)**, S. 533–540

Alexander et al. 1977
Alexander, Christopher ; Ishikawa, Sara ; Silverstein, Murray: *A Pattern Language: Towns, Buildings, Construction*. First Edition. New York : Oxford University Press, 1977

Amdahl et al. 1964
Amdahl, Gene M. ; Blaauw, Gerrit A. ; Jr., Frederick P. B.: Architecture of the IBM System/360. In: *IBM Journal of Research and Development* 8 (1964), Nr. 2. – Reprinted in Vol. 44, No. 1/2, 2000

Anderson 1987
Anderson, John R.: Skill acquisition: Compilation of weak-method problem solutions. In: *Psychological Review* 94 (1987), Nr. 2, S. 192–210

Anderson 2001
Anderson, John R.: *Kognitive Psychologie*. 3. Aufl. Heidelberg, Berlin : Spektrum Akademischer Verlag, 2001. – ISBN 3–8274–1024–X

Anderson und Krathwohl 2001
Anderson, Lorin W. (Hrsg.) ; Krathwohl, David A. (Hrsg.): *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. New York : Longman, 2001

Antonitsch 2005
Antonitsch, Peter K.: Standard Software as Microworld? In: **(Mittermeir 2005)**, S. 189–197

Antonitsch 2007
Antonitsch, Peter K.: Datenbanken – (etwas) anders gesehen. In: **(Schubert 2007)**, S. 229–240

Appelrath et al. 2002
Appelrath, Hans-Jürgen ; Boles, Dietrich ; Claus, Volker: *Starthilfe Informatik*. B.G. Teubner Verlag, 2002. – ISBN 3519102412

Appelrath und Ludewig 2000
Appelrath, Hans-Jürgen ; Ludewig, Jochen: *Skriptum Informatik – eine konventionelle Einführung.* 5. Aufll. Stuttgart. Leipzig : B. G. Teubner, 2000. – ISBN 3–519–42153–4

Arestova et al. 2000
Arestova, Olga N. ; Babanin, Leonid N. ; Voiskounsky, Alexander E.: The Internet User Motivation. In: *Alexander E. Voiskounsky (Hrsg.): Research on the Internet: Humanitarian and Social Aspects. Moscow* (2000), S. 55–76

Arnold 2007
Arnold, Ruedi: *Interactive Learning Environments for Mathematical Topics*, ETH Zürich, Dissertation, November 2007. – http://people.inf.ethz.ch/rarnold/publications/2007_Dissertation-Ruedi-Arnold.pdf – geprüft: 22. Februar 2009

Arnold und Hartmann 2007
Arnold, Ruedi ; Hartmann, Werner: Pragmatische Empfehlungen zur Entwicklung von interaktiven Lernumgebungen. In: **(Schubert 2007)**, S. 171–182

Arnold et al. 2005
Arnold, Ruedi ; Hartmann, Werner ; Reichert, Raimond: Entdeckendes Lernen im Informatik-Unterricht. In: **(Friedrich 2005)**, S. 197–205

Astrachan 2001
Astrachan, Owen: OO Overkill: When Simple is Better than Not. In: *SIGCSE ACM* 2001 (2001), S. 302–306

Atteslander et al. 2006
Atteslander, Peter ; Cromm, Jürgen ; Grabow, Busso: *Methoden der empirischen Sozialforschung.* 11. neu bearbeitete und erweiterte Aufl. Berlin : de Gruyter, 2006. – ISBN 978–3503097401

Balzert 1999
Balzert, Heide: *Lehrbuch der Objektmodellierung: Analyse und Entwurf.* Heidelberg : Spektrum Akademischer Verlag, 1999 (Lehrbücher der Informatik). – ISBN 3–8274–0285–9

Balzert 2005
Balzert, Heide: *UML 2 in 5 Tagen. Der schnelle Einstieg in die Objektorientierung.* Herdecke, Bochum : W3L-Verlag, 2005. – ISBN 3–937137–61–0

Balzert 2000
Balzert, Helmut: *Lehrbuch der Software-Technik. Software-Entwicklung.* 2. Aufl. Heidelberg : Spektrum Akademischer Verlag, 2000 (Lehrbücher der Informatik). – ISBN 3–8274–0480–0

Barron et al. 2005
Barron, Brigid J. ; Martin, Caitlin K. ; Robert, Eric S.: Designing a Computer Science Curriculum for Bermuda's Public Schools. In: **(Samways 2005)**. – 076.pdf

Bartke und Maurer 2000
Bartke, Peter ; Maurer, Christian: Thesen zum Informatikunterricht der Oberstufe. (2000). – http://lwb.mi.fu-berlin.de/inf/mix/thesen.html – geprüft: 22. Februar 2009

Bassey 1999
Bassey, Michael (Hrsg.): *Case study research in educational settings.* UK : Open University Press, 1999. – ISBN 978–0335199846

Bauer und Goos 2004
Bauer, Friedrich L. ; Goos, Gerhard: *Informatik 1. Eine einführende Übersicht.* 4. Aufl. Berlin. Heidelberg : Springer, 2004

Baumann 1990
Baumann, Rüdeger: *Didaktik der Informatik.* 1. Aufl. Stuttgart : Klett-Schulbuchverlag, 1990

Baumann 1993
Baumann, Rüdeger: Ziele und Inhalte des Informatikunterrichts. In: *Zentralblatt für Didaktik der Mathematik* (1993), S. 9–19

Baumann 1996
Baumann, Rüdeger: *Didaktik der Informatik.* 2. vollständig neu bearbeitete Aufl. Stuttgart : Ernst Klett Verlag, 1996. – ISBN 3–12–985010–4

Baumann 1998
Baumann, Rüdeger: Fundamentale Ideen der Informatik – gibt es das? In: Koerber, Bernhard (Hrsg.) ; Peters, Ingo-Rüdiger (Hrsg.): *Informatische Bildung in Deutschland. Perspektiven für das 21. Jahrhundert.* Berlin : LOG IN Verlag, 1998, S. 89–107

Büdding 2007
Büdding, Hendrik: Mobiles Lernen unter Verwendung von Handheld Computern im Bereich der Schulinformatik. In: **(Stechert 2007a)**, S. 7–15

Bell et al. 2002
Bell, Tim (Hrsg.) ; Witten, Ian H. (Hrsg.) ; Fellows, Mike (Hrsg.): *Computer Science Unplugged. An enrichment and extension programme for primary-aged children.* NZ : Computer Science Unplugged Project, 2002. – http://csunplugged.org/index.php/de/06-searching-algorithms-activitiesmenu-112 – geprüft 22. Februar 2009

Benzie und Iding 2007
Benzie, David (Hrsg.) ; Iding, Marie (Hrsg.): *Informatics, Mathematics and ICT: A golden triangle (IMICT2007).* Northeastern University. Boston. MA, 2007 . – ISBN 978–0–615–14623–2. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Biundo et al. 2006
Biundo, Susanne ; Claus, Volker ; Mayr, Heinrich C. ; Gesellschaft für Informatik e. V. (GI) (Hrsg.): *Was ist Informatik?* Langfassung vom Mai 2006. Bonn : Gesellschaft für Informatik e.V., 2006. – http://www.gi-ev.de/fileadmin/redaktion/Download/was-ist-informatik-lang.pdf – geprüft: 22. Februar 2009

Bluhm et al. 2004
Bluhm, Hartmut ; Fricke, Uwe ; Otto, Torsten ; Ricklefs, Tammo ; Seiffert, Monika ; Siegel, Christian ; Renz, Werner (Hrsg.) ; Seiffert, Monika (Hrsg.): *Rahmenplan Informatik. Bildungsplan gymnasiale Oberstufe.* Hamburg : Freie und Hansestadt Hamburg – Behörde für Bildung und Sport – Amt für Bildung – B 22, 2004. – http://www.mint-hamburg.de/rahmenplaene/RPGyOInf.pdf – geprüft: 22. Februar 2009

Borchel et al. 2005
Borchel, Christiane ; Humbert, Ludger ; Reinertz, Martin: Design of an Informatics System to Bridge the Gap Between Using and Understanding in Informatics. In: **(Micheuz et al. 2005)**, S. 53–63

Bortz und Döring 2002
Bortz, Jürgen ; Döring, Nicola: *Forschungsmethoden und Evaluation für Sozialwissenschaftler*. 3. Aufl. Berlin : Springer, 2002. – ISBN 3–540–41940–3

Bosse et al. 1986
Bosse, Johanna ; Fleischhut, Jens ; Arlt, Wolfgang: *Informatik. Unterrichtsmaterialien für die 10. Jahrgangsstufe. Datenerhebung. Datenverarbeitung. Datenschutz.* Berlin : Senator für Schulwesen, Berufsausbildung und Sport, 1986

Bransford et al. 1990
Bransford, John D. ; Sherwood, Robert D. ; Hasselbring, Ted S. ; Kinzer, Charles K. ; Williams, Susan M.: Anchored Instructions: Why we need it and how technology can help. In: Nix, Don (Hrsg.) ; Spiro, Rand J. (Hrsg.): *Cognition, Education and Multimedia: Exploring ideas in high technology.* Hillsdale, NJ : Erlbaum, 1990, S. 163–205

Brauer 1990
Brauer, Wilfried: Trends der Informatik-Ausbildung. In: Reuter, Andreas (Hrsg.): *GI Jahrestagung (1)* Bd. 257, Springer, 1990 (Informatik-Fachberichte). – ISBN 3–540–53212–9, S. 456–464

Brauer 2001
Brauer, Wilfried: Informatikbetrachtungen. In: Desel, Jörg (Hrsg.): *Das ist Informatik.* Springer, 2001. – ISBN 3–540–41091–0, S. 23–32

Brauer und Brauer 1992
Brauer, Wilfried ; Brauer, Ute: Wissenschaftliche Herausforderungen für die Informatik: Änderungen von Forschungszielen und Denkgewohnheiten. In: Langenheder, Werner (Hrsg.) ; Müller, Günter (Hrsg.) ; Schinzel, Britta (Hrsg.): *Informatik cui bono?*, Springer, 1992 (Informatik Aktuell). – ISBN 3–540–55957–4, S. 11–19

Brauer und Brauer 1995
Brauer, Wilfried ; Brauer, Ute: Informatik - das neue Paradigma. Änderungen von Forschungszielen und Denkgewohnheiten der Informatik. In: *LOG IN* (1995), Nr. 4, S. 25–29

Brauer et al. 1976
Brauer, Wilfried ; Claus, Volker ; Deussen, Peter ; Jürgen Eickel ; Haacke, Wolfhart ; Hosseus, Winfried ; Koster, Cornelis H. A. ; Ollesky, Dieter ; Weinhart, Karl ; Gesellschaft für Informatik e. V.: Zielsetzungen und Inhalte des Informatikunterrichts. In: *ZDM* 8 (1976), Nr. 1, S. 35–43. – ISSN 0044–4103. – ZDM – Zentralblatt für Didaktik der Mathematik

Brauer und Münch 1996
Brauer, Wilfried ; Münch, Siegfried: *Studien- und Forschungsführer Informatik.* 3. völlig neu bearbeitete Aufl. Berlin : Springer-Verlag GmbH, 1996. – ISBN 3–540–60417–0

Breier und Hubwieser 2002
Breier, Norbert ; Hubwieser, Peter: An information-oriented approach to informatical education. In: *Informatics in education* 1 (2002), Nr. 1, S. 31–42. – ISSN 1648–5831

Brügge 2000
Brügge, Bernd: *Einführung in die Informatik I. Informatik-Systeme. Skriptum zur Vorlesung im Wintersemester 2000/2001.* Oktober 2000. – http://wwwbruegge.in.tum.de/teaching/ws00/Info1/vorlesung/folien/02_Informatiksysteme.pdf – geprüft: 22. Februar 2009

Brinda 2004a
Brinda, Torsten: *Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II*, Universität Siegen, Didaktik der Informatik und E-Learning, Dissertation, März 2004. – http://www.ub.uni-siegen.de/pub/diss/fb12/2004/brinda/brinda.pdf – geprüft: 22. Februar 2009

Brinda 2004b
Brinda, Torsten: Integration of new exercise classes into the Informatics education in the field of object-oriented modelling. In: *Education and Information Technologies* 9 (2004), Nr. 2, S. 117–130

Brinda 2007
Brinda, Torsten: Development of the exercise culture in informatics. In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Brinda und Schubert 2001
Brinda, Torsten ; Schubert, Sigrid: Didactic System for Object-oriented Modelling. In: **(Watson und Andersen 2002)**, S. 473–482

Brooks und Danserau 1983
Brooks, Larry W. ; Danserau, Donald F.: Effects of structural schema training and text organisation on expository prose processing. In: *Journal of Educational Psychology* 75 (1983), Nr. 6, S. 811–820

Broy und Rumpe 2007
Broy, Manfred ; Rumpe, Bernhard: Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. In: *Informatik Spektrum* 30 (2007), Nr. 1, S. 3–18

Broy und Steinbrüggen 2004
Broy, Manfred ; Steinbrüggen, Ralf: *Modellbildung in der Informatik*. 1. Aufl. Berlin. Heidelberg : Springer. Xpert.press, 2004

Börstler 2007
Börstler, Jürgen: Objektorientiertes Programmieren – Machen wir irgendwas falsch? In: **(Schubert 2007)**, S. 9–20

Bruidegom und Koolen-Wijkstra 2007
Bruidegom, Ben ; Koolen-Wijkstra, Wouter: SIM-PL: Software for teaching computer hardware at secondary schools in the Netherlands. In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Bruner 1960
Bruner, Jerome S.: *The Process of Education*. Harvard University Press, Cambridge, 1960

Bruner 1966
Bruner, Jerome S.: *Toward a theory of instruction*. Harvard University Press, Cambridge, 1966

Brunnstein 2001
Brunnstein, Klaus: Mit IT-Risiken umgehen lernen: Über Probleme der Beherrschbarkeit komplexer Informatiksysteme. In: **(Keil-Slawik und Magenheim 2001)**, S. 9–12

Böszörmenyi 2001
Böszörmenyi, Laszlo: Java für Anfänger? In: *LOG IN* 21 (2001), Nr. 1, S. 14–19

Bunge 1967
Bunge, Mario A.: *Scientific Research, Vol. II.* Berlin : Springer, 1967

Burkert 1994a
Burkert, Jürgen: Umorientierung des Informatikunterrichts. In: *LOG IN* 14 (1994), Nr. 4, S. 55–58

Burkert 1994b
Burkert, Jürgen: Umorientierung des Informatikunterrichts (Teil 2). In: *LOG IN* 14 (1994), Nr. 5/6, S. 86–89

Burkert 1995
Burkert, Jürgen: Informatikunterricht – Quo vadis? In: **(Schubert 1995)**, S. 49–52

Burkhardt und Schoenfeld 2003
Burkhardt, Hugh ; Schoenfeld, Alan H.: Improving Educational Research: Toward a More Useful, More Influential, and Better-Funded Enterprise. In: *Educational Researcher* 32 (2003), Nr. 9, S. 3–14. – http://www-gse.berkeley.edu/faculty/AHSchoenfeld/Schoenfeld_BurkhardtEdResearch.pdf – geprüft: 22. Februar 2009

Buschmann et al. 1996
Buschmann, Frank ; Meunier, Regine ; Rohnert, Hans ; Sommerlad, Peter ; Stal, Michael: *Pattern-Oriented Software Architecture. A System of Patterns. Volume 1.* Wiley & Sons, 1996. – ISBN 978–0471958697

Bussmann und Heymann 1987
Bussmann, Hans ; Heymann, Hans-Werner: Computer und Allgemeinbildung. In: *Neue Sammlung 1* (1987), S. 2–39

Callegarin und Cortesi 2001
Callegarin, Giuseppe ; Cortesi, Agostino: An Italian National Curriculum on ICT for Schools. In: **(Watson und Andersen 2002)**, S. 767–776

Carpenter und Gorg 2000
Carpenter, Jean (Hrsg.) ; Gorg, Sheila (Hrsg.): *Principles and Standards for School Mathematics.* Reston : National Council of Teachers of Mathematics (NCTM), 2000. – ISBN 0–87353–480–8. – http://standards.nctm.org/ – geprüft: 22. Februar 2009

Cassel et al. 1995
Cassel, Lillian N. ; Beck, Robert E. ; Hardt, Daniel: Computing and the understanding of text. In: **(Tinsley und Weert 1995)**, S. 669–677

Chikofsky und Cross 1990
Chikofsky, Elliot J. ; Cross, James H.: Reverse Engineering and Design Recovery: A Taxonomy. In: *IEEE Software* 7(1) (1990), S. 13–18

Clancy und Linn 1999
Clancy, Michael J. ; Linn, Marcia C.: Patterns and Pedagogy. In: *SIGCSE ACM* 1999 (1999), S. 37–42

Claus und Schwill 1997
Claus, Volker ; Schwill, Andreas: *Schülerduden Informatik. Ein Lexikon zum Informatikunterricht.* Mannheim : Dudenverlag, 1997

Claus und Schwill 2006
Claus, Volker ; Schwill, Andreas: *Duden Informatik – A-Z. Fachlexikon für Studium, Ausbildung und Beruf.* Mannheim : Dudenverlag, 2006

Cyranek 1990
Cyranek, Günther: Lehrerausbildung Informatik. In: Cyranek, Günther (Hrsg.) ; Forneck, Hermann J. (Hrsg.) ; Goorhuis, Henk (Hrsg.): *Beiträge zur Didaktik der Informatik.* Frankfurt, Aarau : Verlag Moritz Diesterweg. Verlag Sauerländer, 1990. – ISBN 3–425–05309–4, S. 1–9

Czischke et al. 1999
Czischke, Jürgen ; Dick, Georg ; Hildebrecht, Horst ; Humbert, Ludger ; Ueding, Werner ; Wallos, Klaus ; Landesinstitut für Schule und Weiterbildung (Hrsg.): *Von Stiften und Mäusen.* 1. Aufl. Bönen : DruckVerlag Kettler GmbH, 1999. – ISBN 3–8165–4165–8

Dagiene 1999
Dagiene, Valentina: Programming-Based Solutions of Problems In Informatics Curricula. In: **(Downes und Watson 1999)**, S. 88–94

Denning 2003
Denning, Peter J.: Great principles of computing. In: *Commun. ACM* 46 (2003), Nr. 11, S. 15–20. – ISSN 0001–0782. – http://doi.acm.org/10.1145/948383.948400 – geprüft: 22. Februar 2009

Denning 2007
Denning, Peter J.: Computing is a natural science. In: *Commun. ACM* 50 (2007), Nr. 7, S. 13–18. – ISSN 0001–0782. – http://doi.acm.org/10.1145/1272516.1272529 – geprüft: 22. Februar 2009

DeRemer und Kron 1975
DeRemer, Frank ; Kron, Hans: Programming-in-the large versus programming-in-the-small. In: *Proceedings of the international conference on Reliable software.* New York, NY, USA : ACM Press, 1975, S. 114–121

Diethelm 2007
Diethelm, Ira: *„Strictly models and objects first" – Unterrichtskonzept und -methodik für objektorientierte Modellierung im Informatikunterricht*, Universität Kassel, Dissertation, 2007. – http://kobra.bibliothek.uni-kassel.de/bitstream/urn:nbn:de:hebis:34-2007101119340/1/DissIraDruckfassungA5.1.pdf – geprüft: 22. Februar 2009

Dittich 2008
Dittich, Carsten ; Stechert, Peer (Hrsg.) ; Schubert, Sigrid (Hrsg.): *Einsatz von Datenbanksystemen als typische Repräsentanten von Informatiksystemen im Rahmen des Unterrichtsmodells für das Verstehen von Informatiksystemen.* Hauptseminararbeit. Lehrstuhl für Didaktik der Informatik und E-Learning. Universität Siegen, 2008. – http://www.die.informatik.uni-siegen.de/lehrstuhl/stechert/informatics-systems/seminararbeiten/2008_repraesentanten.pdf – geprüft: 22. Februar 2009

Doebeli Honegger 2007
Doebeli Honegger, Beat: Wiki und die fundamentalen Ideen der Informatik. In: **(Schubert 2007)**, S. 207–216

Downes und Watson 1999
Downes, Toni (Hrsg.) ; Watson, Deryn (Hrsg.): *Communications and Networking in Education: Learning in a Networked Society.* IFIP WG 3.1 and 3.5 (in co-operation with 3.6) Open Conference. Aulanko-Hämeenlinna. Finland, 1999 . – ISBN 951–45–8627–1

Dreier und Hartmann 2005
Dreier, Matthias ; Hartmann, Werner: Suchmaschinen verstehen. Einsatz einer didaktischen Suchmaschine im Unterricht. In: **(Micheuz et al. 2005)**, S. 151–152

Dörner und Schaub 1994
Dörner, Dietrich ; Schaub, Harald: Errors in Planning und Decision Making and the Nature of Human Information Processing. In: *Applied Psychology. Volume 43 Issue 4. An International Review*, 1994, S. 433–453

Drosdowski et al. 2001
Drosdowski, Günther (Hrsg.) ; Scholze-Stubenrecht, Werner (Hrsg.) ; Wermke, Matthias (Hrsg.): *Duden 5. Das Fremdwörterbuch.* 7. Aufl. Dudenverlag. Mannheim, 2001

Duell et al. 1998
Duell, Michael ; Rising, Linda ; Sommerlad, Peter ; Stall, Michael: Examples to Accompany: Pattern-Oriented Software Architecture. In: *Conference on Object Oriented Programming Systems Languages and Applications.* (1998). – http://www.cs.uni.edu/~wallingf/teaching/062/sessions/support/pattern-examples.pdf – geprüft: 22. Februar 2009

Dutke 1994
Dutke, Stephan: *Mentale Modelle: Konstrukte des Wissens und Verstehens. Kognitionspsychologische Grundlagen für die Software-Ergonomie.* Göttingen : Verlag für Angewandte Psychologie, 1994 (Reihe: Arbeit und Technik – Band 4). – ISBN 978–3–87844–111–3

Eberle 1996
Eberle, Franz ; Wettstein, Emil (Hrsg.) ; Weibel, Walter (Hrsg.) ; Gonon, Philipp (Hrsg.): *Didaktik der Informatik bzw. einer informations- und kommunikationstechnologischen Bildung auf der Sekundarstufe II – Ziele und Inhalte, Bezug zu anderen Fächern sowie unterrichtspraktische Handlungsempfehlungen.* 1. Aufl. Aarau : Verlag Sauerländer, 1996 (Pädagogik bei Sauerländer: Dokumentation und Materialien 24). – ISBN 3–7941–4157–1

Eckerdal et al. 2006
Eckerdal, Anna ; McCartney, Robert ; Moström, Jan E. ; Ratcliffe, Mark ; Sanders, Kate ; Zander, Carol: Putting threshold concepts into context in computer science education. In: *ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education.* New York, NY, USA : ACM, 2006. – ISBN 1–59593–055–8, S. 103–107

Edelmann 1986
Edelmann, Walter (Hrsg.): *Lernpsychologie – Eine Einführung.* München, Weinheim : Psychologie-Verlags-Union, 1986

Engbring 1995
Engbring, Dieter: Kultur- und technikgeschichtlich begründete Bildungswerte der Informatik. In: **(Schubert 1995)**, S. 68–77

Engbring 2004
Engbring, Dieter: *Informatik im Herstellungs- und Nutzungskontext. Ein technikbezogener Zugang zur fachübergreifenden Lehre*, Universität Paderborn – Fakultät für Elektrotechnik, Informatik und Mathematik – Arbeitsgruppe Informatik und Gesellschaft, Dissertation, November 2004. – http://ubdata.uni-paderborn.de/ediss/17/2004/engbring/disserta.pdf – geprüft: 22. Februar 2009

Ershov 1981
Ershov, Andrei: Programming, the second literacy. In: *3. IFIP World Conference on Computers in Education (WCCE), Lausanne* (1981), S. 146–161

EU 2008
EU ; Europäische Gemeinschaften (Hrsg.): *Der Europäische Qualifikationsrahmen für lebenslanges Lernen (EQR)*. Belgium : EU, 2008. – http://ec.europa.eu/education/policies/educ/eqf/eqf08_de.pdf – geprüft 22. Februar 2009

Fincher und Petre 2004
Fincher, Sally (Hrsg.) ; Petre, Marian (Hrsg.): *Computer Science Education Research*. Routledge Falmer, 2004. – 239 S. – ISBN 90 265 1969 9

Foegen 1996
Foegen, Malte: *Entwurf eines didaktischen Konzepts der Informatik*. Diplomarbeit. Technische Hochschule Darmstadt : FB Informatik. FG Praktische Informatik, 1996

Forneck 1990
Forneck, Hermann J.: Entwicklungstendenzen und Problemlinien der Didaktik der Informatik. In: Cyranek, Günther (Hrsg.) ; Forneck, Hermann J. (Hrsg.) ; Goorhuis, Henk (Hrsg.): *Beiträge zur Didaktik der Informatik*. Frankfurt, Aarau : Verlag Moritz Diesterweg. Verlag Sauerländer, 1990. – ISBN 3–425–05309–4

Forneck 1997
Forneck, Hermann J.: Eine neue Konzeption informationstechnischer Allgemeinbildung. In: *LOG IN* 17 (1997), Nr. 6, S. 24–28

Franke et al. 2007
Franke, Demian ; Freischlad, Stefan ; Friedrich, Lars ; Haug, Florian ; Klein, Benjamin ; Koslowski, Rudolf ; Stechert, Peer ; Ufer, Jonathan ; Stechert, Peer (Hrsg.) ; Freischlad, Stefan (Hrsg.): *Abschlussbericht der Projektgruppe Entwurfsmuster im Informatikunterricht des Fachbereichs Elektrotechnik und Informatik der Universität Siegen – Lernsoftware Pattern Park*. Siegen : Lehrstuhl für Didaktik der Informatik und E-Learning, 2007. – http://www.die.informatik.uni-siegen.de/pgpatternpark/ – geprüft: 22. Februar 2009

Freeman et al. 2005
Freeman, Eric ; Freeman, Elisabeth ; Sierra, Kathy: *Entwurfsmuster von Kopf bis Fuß*. First edition. O'Reilly, 2005. – ISBN 3897214210

Freischlad 2006
Freischlad, Stefan: Beitrag des Informatikunterrichts zur Entwicklung von Medienkompetenzen. In: **(Schwill et al. 2006)**, S. 29–38

Freischlad 2007
Freischlad, Stefan: Anwenden und Verstehen des Internets – eine Erprobung im Informatikunterricht. In: **(Schubert 2007)**, S. 195–206

Freischlad 2008
Freischlad, Stefan: Zur theoretischen Fundierung von Wissensstrukturen am Beispiel „Internetworking". In: Brinda, Torsten (Hrsg.) ; Fothe, Michael (Hrsg.) ; Hubwieser, Peter (Hrsg.) ; Schlüter, Kirsten (Hrsg.): *Didaktik der Informatik – Aktuelle Forschungsergebnisse. 5. Workshop der GI-Fachgruppe „Didaktik der Informatik"*. Bonn : Köllen, 2008 (Lecture Notes in Informatics (LNI) 135). – ISBN 978–3–88579–229–1, S. 45–54

Freischlad und Schubert 2006
FREISCHLAD, Stefan ; SCHUBERT, Sigrid: Media Upheaval and Standards of Informatics. In: **(Watson und Benzie 2006)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Alesund-CD.zip – geprüft: 22. Februar 2009

Freischlad und Schubert 2007
FREISCHLAD, Stefan ; SCHUBERT, Sigrid: Towards high quality exercise classes for Internetworking. In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Freischlad und Stechert 2008
FREISCHLAD, Stefan ; STECHERT, Peer: Discovery Learning about Informatics Systems. In: **(Wheeler et al. 2008)**. – (CD-Publikation; 2 Seiten)

Frey 2003
FREY, Elke: Informatik in der Jahrgangsstufe 6 – ein Bericht aus der Praxis. In: **(Hubwieser 2003)**, S. 33

Frühauf et al. 1997
FRÜHAUF, Karol ; LUDEWIG, Jochen ; SANDMAYR, Helmut: *Software-Prüfung: eine Anleitung zum Test und zur Inspektion*. 3., durchgesehene Aufl. Zürich : VDF, Hochschulverlag an der ETH Zürich, Lehrbuch Informatik, 1997. – ISBN 3–7281–2353–6

Friedrich 1995a
FRIEDRICH, Steffen: Grundpositionen eines Schulfaches. In: *LOG IN* 15 (1995), Nr. 5/6, S. 30–34

Friedrich 1995b
FRIEDRICH, Steffen: Informatik-Didaktik – ein Fachgebiet im Aufbruch. In: **(Schubert 1995)**, S. 33–39

Friedrich 2003
FRIEDRICH, Steffen: Informatik und PISA – vom Wehe zum Wohl der Schulinformatik. In: **(Hubwieser 2003)**, S. 133–144

Friedrich 2005
FRIEDRICH, Steffen (Hrsg.): *Unterrichtskonzepte für informatische Bildung, INFOS 2005, 11. GI-Fachtagung Informatik und Schule, 28.-30. September 2005 an der TU Dresden*. Bd. *60*. GI, 2005 (LNI). – ISBN 3–88579–389–X

Friedrich et al. 1996
FRIEDRICH, Steffen ; SCHUBERT, Sigrid E. ; SCHWILL, Andreas: Informatik in der Schule – ein Fach im Wandel. In: *LOG IN* 16 (1996), Nr. 2, S. 29–33

Fuller et al. 2007
FULLER, Ursula ; JOHNSON, Colin G. ; AHONIEMI, Tuukka ; CUKIERMAN, Diana ; HERNÁN-LOSADA, Isidoro ; JACKOVA, Jana ; LAHTINEN, Essi ; LEWIS, Tracy L. ; THOMPSON, Donna M. ; RIEDESEL, Charles ; THOMPSON, Errol: Developing a computer science-specific learning taxonomy. In: *ITiCSE-WGR '07: Working group reports on ITiCSE on Innovation and technology in computer science education*. New York, NY, USA : ACM, 2007, S. 152–170

Gamma et al. 1995
GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: Design Patterns: Elements of Resusable Object-Oriented Software. In: *Addison-Wesley, Reading, MA* 1995 (1995)

Ganea und Koch 2008
Ganea, Milena ; Koch, Hartmud: Kooperation zwischen dem Fürst-Johann-Moritz-Gymnasium und dem Institut „Didaktik der Informatik und E-Learning" an der Universität Siegen. In: **(Schubert und Stechert 2008)**, S. 115–120

Gasper et al. 1992
Gasper, Friedrich ; Leiss, Ina ; Spengler, Mario ; Stimm, Hermann: *Technische und theoretische Informatik*. München : Bayerischer Schulbuch-Verlag, 1992. – ISBN 3–7627–3701–0

Gerding 2008
Gerding, Thomas ; Stechert, Peer (Hrsg.) ; Schubert, Sigrid (Hrsg.): *Unterrichtsvarianten zur systematischen Erkundung von Informatiksystemen mit dem Schwerpunkt Beobachtungsaufgaben, Lehrer- und Schülerexperiment.* Hauptseminararbeit. Lehrstuhl für Didaktik der Informatik und E-Learning. Universität Siegen, 2008. – http://www.die.informatik.uni-siegen.de/lehrstuhl/stechert/informatics-systems/seminararbeiten/unterrichtsvarianten.pdf – geprüft: 22. Februar 2009

GI 2000
GI: Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen der Gesellschaft für Informatik e. V. In: *Informatik Spektrum* 23 (2000), Nr. 6, S. 378–382

GI 2008
GI: Grundsätze und Standards für die Informatik in der Schule – Bildungsstandards Informatik für die Sekundarstufe 1. Beschluss des Präsidiums der Gesellschaft für Informatik e. V. vom 24. Januar 2008. In: *LOG IN* 28 (2008), Nr. 150/151. – (Beilage zum Heft; 72 Seiten)

Goos und Zimmermann 2006
Goos, Gerhard ; Zimmermann, Wolf: *Vorlesungen über Informatik. Band 1: Grundlagen und funktionales Programmieren.* Springer, 2006. – ISBN 978–3–540–24405–9

Graf 2008
Graf, Daniel ; Stechert, Peer (Hrsg.) ; Schubert, Sigrid (Hrsg.): *Konzeption von Aufgaben für ein Unterrichtsprojekt mit dem Schwerpunkt Verstehen von Informatiksystemen. Unterrichtsaufgaben zu den Themenbereichen Zustand und Beobachter.* Hauptseminararbeit. Lehrstuhl für Didaktik der Informatik und E-Learning. Universität Siegen, 2008

Guzdial 1995
Guzdial, Mark: Centralized Mindset: A Student Problem with Object-Oriented Programming. In: *SIGCSE ACM* 1995 (1995), S. 182–185

Hacker und Skell 1993
Hacker, Winfried ; Skell, Wolfgang: *Lernen in der Arbeit.* Berlin, Bonn : Bundesinstitut für Berufsbildung, 1993. – ISBN 3–88555–525–5

Hadar und Hazzan 2004
Hadar, Irit ; Hazzan, Orit: On the Contribution of UML Diagrams to Software System Comprehension. In: *Journal of Object Technology* 3 (2004), Nr. 1, S. 143–156. – http://www.jot.fm/issues/issue_2004_01/article3.pdf – geprüft: 22. Februar 2009

Hampel et al. 1999
Hampel, Thorsten ; Magenheim, Johannes ; Schulte, Carsten: Dekonstruktion von Informatiksystemen als Unterrichtsmethode – Zugang zu objektorientierten Sichtweisen im Informatikunterricht. In: **(Schwill 1999)**, S. 149–164

Harrer und Schneider 2002
HARRER, Andreas ; SCHNEIDER, Markus: Didaktische Betrachtung zur Unterrichtung von Software-Mustern im Hochschulbereich. In: **(Schubert et al. 2002)**, S. 67–76

Hartmann et al. 2006
HARTMANN, Werner ; NÄF, Michael ; REICHERT, Raimond: *Informatikunterricht planen und durchführen.* 1. Aufl. Berlin : Springer, 2006. – ISBN 978–3540344841

Hartmann und Nievergelt 2002
HARTMANN, Werner ; NIEVERGELT, Jürg: Informatik und Bildung zwischen Wandel und Beständigkeit. In: *Informatik Spektrum* 25 (2002), Nr. 6, S. 465–476

Heimann 1976
HEIMANN, Paul: Didaktik als Theorie und Lehre. In: REICH, Kersten (Hrsg.) ; THOMAS, Helga (Hrsg.): *Didaktik als Unterrichtswissenschaft*, Klett: Stuttgart, 1976, S. 142–167

Hermes 2003
HERMES, Alfred: Werkstatt: Entwurfsmuster für grafische Benutzeroberflächen. Modell – Darstellung – Steuerung in JAVA. In: *LOG IN* 23 (2003), Nr. 124, S. 57–63

Herper und Hinz 2005
HERPER, Henry ; HINZ, Volkmar: Analyse eines Informatiksystems durch unterschiedliche Modellierungsansätze. In: **(Friedrich 2005)**, S. 253–262

Hertwig und Brück 2000
HERTWIG, André ; BRÜCK, Rainer: *Entwurf digitaler Systeme.* München : Carl Hanser Verlag, 2000

Hesse et al. 1994
HESSE, Wolfgang ; BARKOW, Georg ; BRAUN, Hubert von ; KITTLAUS, Hans-Bernd ; SCHESCHONK, Gert: Terminologie der Softwaretechnik, Ein Begriffssystem für die Analyse und Modellierung von Anwendungssystemen, Teil 2: Tätigkeits- und ergebnisbezogene Elemente. In: *Informatik Spektrum* 17 (1994), Nr. 2, S. 96–105

Heymann 2004
HEYMANN, Hans W.: Besserer Unterricht durch Sicherung von „Standards"? In: *Pädagogik. Beltz-Verlag* 56 (2004), Nr. 6, S. 6–9. – http://www.beltz.de/paedagogik/heft200406/n_02_02.html – geprüft 22. Februar 2009

Hinkelmann et al. 2007
HINKELMANN, Markus ; JAKOBY, Andreas ; STECHERT, Peer: *t*-Private and Secure Auctions. In: CAI, Jin-yi (Hrsg.) ; COOPER, S. B. (Hrsg.) ; ZHU, Hong (Hrsg.): *TAMC* Bd. 4484, Springer, 2007 (Lecture Notes in Computer Science). – ISBN 978–3–540–72503–9, S. 486–498

Hinkelmann et al. 2008
HINKELMANN, Markus ; JAKOBY, Andreas ; STECHERT, Peer: *t*-Private and *t*-Secure Auctions. In: *Journal of Computer Science and Technology* 23 (2008), Nr. 5, S. 694–710. – http://jcst.ict.ac.cn/paper/8501.pdf – geprüft: 22. Februar 2009

Hirsch et al. 2000
HIRSCH, Michael ; MAGENHEIM, Johannes ; REINSCH, Thorsten: Zugänge zur Informatik mit Mindstorms (Teil 1). In: *LOG IN* 20 (2000), Nr. 2, S. 34–46

Hodnigg 2005
HODNIGG, Karin: A Pragmatic Approach to Spreadsheet Training Based Upon the "Projection-Screen" Model. In: **(Mittermeir 2005)**, S. 116–129

Holl 2003
HOLL, Berit: *Entwicklung und Evaluation eines Unterrichtskonzeptes für computergestütztes kooperatives Lernen. Computer Supported Cooperative Learning (CSCL) am beruflichen Gymnasium für Informations- und Kommunikationstechnologie*, Technische Universität Chemnitz, Philosophische Fakultät, Dissertation, Dezember 2003. – http://archiv.tu-chemnitz.de/pub/2004/0021/data/Dissertation_Holl.pdf – geprüft: 22. Februar 2009

Hoppe und Luther 1997
HOPPE, Heinz U. (Hrsg.) ; LUTHER, Wolfram (Hrsg.): *Informatik und Lernen in der Informationsgesellschaft, 7. GI-Fachtagung Informatik und Schule, INFOS'97, Duisburg, 15.-18. September 1997*. Springer, 1997 (Informatik Aktuell). – ISBN 3–540–63432–0

Hubwieser 1999
HUBWIESER, Peter: Informatik als Pflichtfach an bayerischen Gymnasien. In: **(Schwill 1999)**, S. 165–174

Hubwieser 2003
HUBWIESER, Peter (Hrsg.): *Informatische Fachkonzepte im Unterricht, INFOS 2003, 10. GI-Fachtagung Informatik und Schule, 17.-19. September 2003 in Garching bei München*. Bd. *32*. GI, 2003 (LNI). – ISBN 3–88579–361–X

Hubwieser 2005
HUBWIESER, Peter: Von der Funktion zum Objekt – Informatik für die Sekundarstufe I. In: **(Friedrich 2005)**, S. 27–41

Hubwieser 2007a
HUBWIESER, Peter: *Didaktik der Informatik*. 3. Aufl. Berlin : Springer examen.press, 2007. – ISBN 13–978–3–540–72477–3

Hubwieser 2007b
HUBWIESER, Peter: A smooth way towards object oriented programming in secondary schools. In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Hubwieser und Broy 1997a
HUBWIESER, Peter ; BROY, Manfred: Grundlegende Konzepte von Informations- und Kommunikationssystemen für den Informatikunterricht. In: **(Hoppe und Luther 1997)**, S. 40–50

Hubwieser und Broy 1997b
HUBWIESER, Peter ; BROY, Manfred: Ein neuer Ansatz für den Informatikunterricht am Gymnasium. In: *LOG IN* 17 (1997), Nr. 3/4, S. 42–47

Hubwieser und Broy 1999
HUBWIESER, Peter ; BROY, Manfred: Educating Surfers Or Craftsmen: Introducing An ICT Curriculum For The 21st Century. In: **(Downes und Watson 1999)**, S. 163–177

Hubwieser et al. 2001
HUBWIESER, Peter ; HUMBERT, Ludger ; SCHUBERT, Sigrid: Evaluation von Informatikunterricht. In: **(Keil-Slawik und Magenheim 2001)**, S. 213–215

Humbert 2001
HUMBERT, Ludger: Informatik lehren – zeitgemäße Ansätze zur nachhaltigen Qualifikation aller Schülerinnen. In: **(Keil-Slawik und Magenheim 2001)**, S. 121–132

Humbert 2003
HUMBERT, Ludger: *Zur wissenschaftlichen Fundierung der Schulinformatik*. Witten : pad-Verlag, 2003. – ISBN 3–88515–214–2. – Dissertation an der Universität Siegen http://www.ub.uni-siegen.de/pub/diss/fb12/2003/humbert/humbert.pdf – geprüft: 22. Februar 2009

Humbert et al. 2005
HUMBERT, Ludger ; EICKHOFF, Patrick ; FIGGEN, Bernd ; HAMMERSEN, Thomas ; POMMERENKE, Dirk ; RICHTER, Detlef ; STRIEWE, Jörg: Informatik - innovative Konzepte zur Gestaltung einer offenen Anfangssequenz mit vielfältigen Erweiterungen. In: **(Friedrich 2005)**, S. 263–274

Humbert und Puhlmann 2004
HUMBERT, Ludger ; PUHLMANN, Hermann: Essential Ingredients of Literacy in Informatics. In: **(Magenheim und Schubert 2004)**, S. 65–76

Iding 2007
IDING, Marie: Critical information literacy: students' determinations of web site credibility. In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

IEEE Standards Department 1990
Norm ANSI/IEEE Std 610.12 1990. *IEEE Standard Glossary of Software Engineering Terminology*

ISO 1989
Norm ISO 7498-2. 1989. *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture. International Organization for Standardization.*

ISO 1996
Norm ISO 9241-10. 1996. *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten Teil 10: Grundsätze der Dialoggestaltung. International Organization for Standardization.*

ISTE 1998
ISTE: *National Educational Technology Standards (NETS) Project: Technology Foundation Standards for all Students*. International Society for Technology in Education (ISTE), 1998. – http://cnets.iste.org – geprüft: 22. Februar 2009

ISTE 2007
ISTE: *The ISTE National Educational Technology Standards (NETS-S) and Performance Indicators for Students*. International Society for Technology in Education (ISTE), 2007. – http://www.iste.org/Content/NavigationMenu/NETS/ForStudents/2007Standards/NETS_for_Students_2007_Standards.pdf – geprüft: 22. Februar 2009

Jank und Meyer 2002
JANK, Werner ; MEYER, Hilbert: *Didaktische Modelle*. 5. vollst. überarb. Aufl. Berlin : Cornelsen Scriptor, 2002. – ISBN 3–589–21566–6. – erste Aufl. 1991

Jih und Reeves 1992
Jih, Hueyching J. ; Reeves, Thomas C.: Mental models: a research focus for interactive learning systems. In: *Educational Technology Research and Development* 40 (1992), Nr. 3, S. 39–53

Kalkbrenner 2007
Kalkbrenner, Gerrit: Gibt es einen mobilkommunikationszentrierten Ansatz für die Schulinformatik? In: **(Schubert 2007)**, S. 265–271

Karahasanovic und Holmboe 2006
Karahasanovic, Amela ; Holmboe, Christian: In: Fjuk, A. (Hrsg.); Karahasanovic, A. (Hrsg.); Kaasboell, Jens (Hrsg.): Challenges of learning object-oriented analysis and design through a modeling-first approach. In: *Comprehensive Object-Oriented Learning: The Learners Perspective*, 2006, S. 49–66

Keil-Slawik 1992
Keil-Slawik, Reinhard: Konstruktives Design. Ein ökologischer Ansatz. In: *LOG IN* 12 (1992), Nr. 5/6, S. 18–27

Keil-Slawik und Magenheim 2001
Keil-Slawik, Reinhard (Hrsg.) ; Magenheim, Johannes (Hrsg.): *Informatikunterricht und Medienbildung, INFOS 2001, 9. GI-Fachtagung Informatik und Schule, 17.-20. September 2001 in Paderborn.* Bd. *8.* GI, 2001 (LNI). – ISBN 3–88579–334–2

Keller 1987
Keller, John M.: Strategies for Stimulating the Motivation to Learn. In: *Performance and Instruction* 26 (1987), Nr. 8, S. 1–7

Kennewell 1995
Kennewell, Steve: Information technology capability – how does it develop? In: **(Tinsley und Weert 1995)**, S. 419–427

Klafki 1996
Klafki, Wolfgang: *Neue Studien zur Bildungstheorie und Didaktik, zeitgemäße Allgemeinbildung und kritisch-konstruktive Didaktik.* 5. Aufll. Beltz Verlag. Weinheim, 1996

Klieme 2004
Klieme, Eckhard: Was sind Kompetenzen und wie lassen sie sich messen? In: *Pädagogik* 56 (2004), Nr. 6, S. 10–13

Klieme et al. 2007
Klieme, Eckhard ; Avenarius, Hermann ; Blum, Werner ; Döbrich, Peter ; Gruber, Hans ; Prenzel, Manfred ; Reiss, Kristina ; Riquarts, Kurt ; Rost, Jürgen ; Tenorth, Heinz-Elmar ; Vollmer, Helmut J.: *Zur Entwicklung nationaler Bildungsstandards. Bildungsreform Band 1. Expertise.* unveränderte Aufl. Bundesministerium für Bildung und Forschung (BMBF) Referat Publikationen; Internetredaktion. Bonn. Berlin, 2007. – http://www.bmbf.de/pub/zur_entwicklung_nationaler_bildungsstandards.pdf – geprüft: 22. Februar 2009

KMK 2004
KMK (Hrsg.): *Einheitliche Prüfungsanforderungen in der Abiturprüfung „Informatik“.* Bonn : KMK, 2004. – KMK – Ständige Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/1989/1989_12_01_EPA_Informatik.pdf – geprüft: 22. Februar 2009

Koerber 1978
KOERBER, Bernhard: Informatik im Unterricht. In: ARLT, Wolfgang (Hrsg.): *EDV-Einsatz in Schule und Ausbildung. Modelle und Erfahrungen.* München : Oldenbourg Verlag, 1978. – ISBN 3–486–22021–7, S. 6–12

Koerber et al. 1989
KOERBER, Bernhard ; PETERS, Ingo-Rüdiger ; ARLT, Wolfgang: *Informatik. Unterrichtsmaterialien für die 9. Jahrgangsstufe. Einführung in die Informatik.* 2. Aufl. Berlin : Senator für Schule, Berufsbildung und Sport, 1989

Koerber und Peters 1993
KOERBER, Bernhard ; PETERS, Ingo-Rüdiger: Informatikunterricht und informationstechnische Grundbildung – ausgrenzen, abgrenzen oder integrieren? In: **(Troitzsch 1993)**, S. 108–115

Koerber et al. 1981
KOERBER, Bernhard ; SACK, Lothar ; SCHULZ-ZANDER, Renate: Prinzipien des Informatikunterrichts. In: ARLT, Wolfgang (Hrsg.): *Informatik als Schulfach. Didaktische Handreichungen für das Schulfach Informatik.* München : Oldenbourg Verlag, 1981. – ISBN 3–486–24301–2, S. 28–35

Kollee et al. 2009
KOLLEE, Christian ; MAGENHEIM, Johannes ; NELLES, Wolfgang ; RHODE, Thomas ; SCHAPER, Niclas ; SCHUBERT, Sigrid ; STECHERT, Peer: Computer Science Education and Key Competencies. In: **(Santos et al. 2009)**. – (CD-Publikation; 11 Seiten)

Kopp und Mandl 2006
KOPP, Birgitta ; MANDL, Heinz: Wissensschemata. In: MANDL, Heinz (Hrsg.) ; FRIEDRICH, Helmut F. (Hrsg.): *Handbuch Lernstrategien. Analyse und Intervention.* Göttingen : Hogrefe-Verlag, 2006. – ISBN 3–8017–1813–1, S. 127–134

Koubek 2005
KOUBEK, Jochen: Informatische Allgemeinbildung. In: **(Friedrich 2005)**, S. 57–66

Krämer 1997
KRÄMER, Sybille: Werkzeug-Denkzeug-Spielzeug. Zehn Thesen über unseren Umgang mit Computern. In: **(Hoppe und Luther 1997)**, S. 7–13

Kuhn 1969
KUHN, Thomas S.: *Die Struktur wissenschaftlicher Revolutionen.* 15. Aufl. Reinbek : Rowohlt, 1969

Lehmann 1993
LEHMANN, Eberhard: Software-Wartung. Ein neuartiger Einstieg in den Informatik-Anfangsunterricht. In: **(Troitzsch 1993)**, S. 134–140

Lehmann 1995
LEHMANN, Eberhard: Komplexe Systeme – Eine fundamentale Idee im Informatikunterricht. In: *LOG IN* 15 (1995), Nr. 1, S. 29–37

Löthe et al. 1986
LÖTHE, Herbert ; CLAUS, Volker ; GUNZENHÄUSER, Rul ; HOSSEUS ; KEIDEL ; LOOS ; LÜBBERS ; PETERS ; PULVER ; SCHRUFF ; SPENGLER, Mario: Rahmenempfehlungen für die Informatik im Unterricht der Sekundarstufe I. In: *Informatik Spektrum* Bd. 9. Berlin : Springer, 1986, S. 143–146. – Erarbeitet vom GI-Arbeitskreis 7.3.4 „Informatik in der Sekundarstufe I" der Gesellschaft für Informatik e.V. Ergänzt um Zusammenfassung durch K. Haefner.

Luft und Kötter 1994
Luft, Alfred L. (Hrsg.) ; Kötter, Rudolf (Hrsg.): *Informatik – eine moderne Wissenstechnik*. Mannheim : BI-Wissenschaftsverlag, 1994

Magenheim 2001
Magenheim, Johannes: Informatiksystem und Dekonstruktion als didaktische Kategorien – Theoretische Aspekte und unterrichtspraktische Implikationen einer systemorientierten Didaktik der Informatik. (2001)

Magenheim 2003
Magenheim, Johannes: Informatik Lernlabor - Systemorientierte Didaktik in der Praxis. In: **(Hubwieser 2003)**, S. 13–31

Magenheim 2005
Magenheim, Johannes: Towards a Competence Model for Educational Standards of Informatics. In: **(Samways 2005)**. – 452.pdf

Magenheim und Schubert 2004
Magenheim, Johannes (Hrsg.) ; Schubert, Sigrid (Hrsg.): *Informatics and Student Assessment*. Bd. *1*. GI, 2004 (LNI Seminars). – ISBN 3–88579–435–7

Magenheim und Schulte 2006
Magenheim, Johannes ; Schulte, Carsten: Social, ethical and technical issues in informatics – An integrated approach. In: *Education and Information Technologies* 11 (2006), Nr. 3-4, S. 319–339. – ISSN 1360–2357. – http://dx.doi.org/10.1007/s10639-006-9012-6 – geprüft: 22. Februar 2009

Mandl und Fischer 2000
Mandl, Heinz ; Fischer, Frank: Mapping-Techniken und Begriffsnetze in Lern- und Kooperationsprozessen. In: Mandl, Heinz (Hrsg.) ; Fischer, Frank (Hrsg.): *Wissen sichtbar machen. Wissensmanagement mit Mapping-Techniken*. Göttingen : Hogrefe-Verlag, 2000. – ISBN 3–8017–1337–7, S. 3–10

McCracken 2004
McCracken, W. M.: Research on Learning to Design Software. In: **(Fincher und Petre 2004)**, S. 155–174

Meier 1990
Meier, Markus W.: Anforderungen an die Informatikausbildung in den neunziger Jahren aus Sicht der Wirtschaft. In: Cyranek, Günther (Hrsg.) ; Forneck, Hermann J. (Hrsg.) ; Goorhuis, Henk (Hrsg.): *Beiträge zur Didaktik der Informatik*. Frankfurt, Aarau : Verlag Moritz Diesterweg u. Verlag Sauerländer, 1990. – ISBN 3–425–05309–4

Meißner 1971
Meissner, Hartwig: *Datenverarbeitung und Informatik*. München : Ehrenwirth, 1971 (Unterrichtswerk der Mathematik). – ISBN 3–431–01441–0

Merrienboer und Sweller 2005
Merrienboer, Jeroen ; Sweller, John: Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions. In: *Educational Psychology Review* 17 (2005), June, Nr. 2, S. 147-177. http://dx.doi.org/10.1007/s10648-005-3951-0. – ISSN 1040–726X

Meyer 2005
Meyer, Hilbert: *Unterrichtsmethoden*. Bd. I: Theorieband. 12. Aufl. Berlin : Cornelsen Scriptor, 2005. – ISBN 3–589–20850–3. – erste Aufl. 1987

Meyer 2006
MEYER, Hilbert: *Unterrichtsmethoden.* Bd. II: Praxisband. 13. Aufl. Berlin : Cornelsen Scriptor, 2006. – ISBN 3–589–20851–1. – erste Aufl. 1987

Micheuz et al. 2005
MICHEUZ, Peter (Hrsg.) ; ANTONITSCH, Peter (Hrsg.) ; MITTERMEIR, Roland (Hrsg.): *Innovative Concepts for Teaching Informatics. Informatics in Secondary Schools: Evolution and Perspectives – Klagenfurt, 30th March to 1st April 2005.* Wien : Ueberreuter Verlag, 2005 . – ISBN 3–8000–5167–2

Micheuz et al. 2007
MICHEUZ, Peter ; FUCHS, Karl J. ; LANDERER, Claudio: Mission possible – computers in "Anyschool". In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Mittermeir 2005
MITTERMEIR, Roland (Hrsg.): *From Computer Literacy to Informatics Fundamentals, International Conference on Informatics in Secondary Schools - Evolution and Perspectives, ISSEP 2005, Klagenfurt, Austria, March 30 - April 1, 2005, Proceedings.* Bd. *3422.* Springer, 2005 (Lecture Notes in Computer Science). – ISBN 3–540–25336–X

Mittermeir 2006
MITTERMEIR, Roland (Hrsg.): *Informatics Education - The Bridge between Using and Understanding Computers, International Conference in Informatics in Secondary Schools - Evolution and Perspectives, ISSEP 2006, Vilnius, Lithuania, November 7-11, 2006, Proceedings.* Bd. *4226.* Springer, 2006 (Lecture Notes in Computer Science). – ISBN 3–540–48218–0

Modrow 1991
MODROW, Eckart: *Zur Didaktik des Informatik-Unterrichts, Band 1: Ziele und Inhalte. Anfangsunterricht. Beispiele und Anwendungen.* Bonn : Dümmler, 1991. – ISBN 3–427–46791–0

Modrow 1992
MODROW, Eckart: *Zur Didaktik des Informatik-Unterrichts, Band 2: Gesellschaftliche Auswirkungen. Fachunterricht. Abitur.* Bonn : Dümmler, 1992. – ISBN 3–427–46801–1

Modrow 2002
MODROW, Eckart: *Pragmatischer Konstruktivismus und fundamentale Ideen als Leitlinien der Curriculumentwicklung – am Beispiel der theoretischen und technischen Informatik*, Martin-Luther-Universität Halle-Wittenberg – Mathematisch-Naturwissenschaftlich-Technische Fakultät, Dissertation, Oktober 2002. – http://ddi.cs.uni-potsdam.de/Examensarbeiten/Modrow2003.pdf – geprüft: 22. Februar 2009

Modrow 2004
MODROW, Eckart: The Contribution of Computer Science to Technical Literacy. In: **(Magenheim und Schubert 2004)**, S. 103–110

MSWWF 1999
MSWWF (Hrsg.): *Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasium/Gesamtschule in Nordrhein-Westfalen – Informatik.* 1. Aufl. Frechen : Ritterbach Verlag, 1999 (Schriftenreihe Schule in NRW 4725). – MSWWF (Ministerium für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen)

Mühlhäuser 2002
Mühlhäuser, Max: Verteilte Systeme (Distributed Systems). In: *Informatik-Handbuch, 3. völlig neu bearbeitete Aufl.* (2002), S. 673–708. – Hanser, München

Nake 1998
Nake, Frieder: Was heißt und zu welchem Ende studiert man Informatik? Ein akademischer Diskursbeitrag nebst Anwendung. In: Claus, Volker (Hrsg.): *Informatik und Ausbildung*, Springer, 1998 (Informatik Aktuell). – ISBN 3–540–64178–5, S. 1–13

Neupert und Friedrich 1997
Neupert, Heiko ; Friedrich, Steffen: Lernen mit Netzen – Lernen über Netze. In: *LOG IN* 17 (1997), Nr. 6, S. 18–23

Nievergelt 1995
Nievergelt, Jürg: Welchen Wert haben theoretische Grundlagen für die Berufspraxis? Gedanken zum Fundament des Informatikturms. In: *Informatik Spektrum* 18 (1995), Nr. 6, S. 342–344

Nievergelt und Ventura 1983
Nievergelt, Jürg (Hrsg.) ; Ventura, Andrea (Hrsg.): *Die Gestaltung interaktiver Programme. Mit Anwendungsbeispielen für den Unterricht.* Stuttgart : Teubner B.G. GmbH, 1983. – ISBN 3519025094

NRCCITL 1999
NRCCITL ; National Research Council Committee on Information Technology Literacy (Hrsg.): *Being Fluent with Information Technology.* Washington, D.C : National Academy Press, 1999. – ISBN 978–0309063999. – http://www.nap.edu/openbook.php?record_id=6482 – geprüft: 22. Februar 2009

OECD 2000
OECD ; Deutsches PISA-Konsortium (Hrsg.): *Schülerleistungen im internationalen Vergleich. Eine neue Rahmenkonzeption für die Erfassung von Wissen und Fähigkeiten.* Berlin : OECD PISA Deutschland, 2000. – ISBN 3–87985–078–X. – http://www.mpib-berlin.mpg.de/en/Pisa/pdfs/Rahmenkonzeptiondt.pdf – geprüft 22. Februar 2009

OECD 2005
OECD ; OECD. Directorate for Education (Hrsg.): *Definition und Auswahl von Schlüsselkompetenzen. Zusammenfassung.* Paris : OECD, 2005. – http://www.oecd.org/dataoecd/36/56/35693281.pdf – geprüft 22. Februar 2009

Ossimitz 2000
Ossimitz, Günther: Zur Entwicklung systemischen Denkens. In: *Habilitationsschrift. Universität Klagenfurt.* (2000). – http://wwwu.uni-klu.ac.at/gossimit/pap/kap1_2.PDF – geprüft: 22. Februar 2009

Ossimitz 2002
Ossimitz, Günther: Systemisches Denken braucht systemische Darstellungsformen. In: Milling, Peter (Hrsg.): *Entscheiden in komplexen Systemen. Wissenschaftliche Jahrestagung der Gesellschaft für Wirtschafts- und Sozialkybernetik vom 29. und 30. September 2000 in Mannheim. Wirtschaftskybernetik und Systemanalyse. Band 20.* Berlin : Duncker und Humblot, 2002. – ISBN 3–428–10638–0, S. 161–174. – http://wwwu.uni-klu.ac.at/gossimit/pap/sysdd.pdf – geprüft: 22. Februar 2009

Paul 1994
Paul, Hansjürgen: *Exploratives Agieren.* Berlin : Peter Lang GmbH, 1994. – ISBN 3–631–48060–1

Peschke 1990
Peschke, Rudolf: Grundideen des Informatikunterrichts. In: *LOG IN* 10 (1990), Nr. 6, S. 25–33

Peschke 1991
Peschke, Rudolf: Neue Allgemeinbildung – Bietet die Informationstechnik eine Chance? In: Cyranek, Günther (Hrsg.): *Computerkultur im Umbruch? Neue Technologien und die Zukunft für Schule und berufliche Bildung*, Diesterweg. Sauerländer. Frankfurt am Main, Aarau, 1991. – ISBN 3–425–05330–2, S. 131–169

Peterson und Davie 2003
Peterson, Larry ; Davie, Bruce: *Computernetze. Eine systemorientierte Einführung.* Heidelberg : dpunkt- Verlag, 2003. – 3. Aufl.

Picot und Rohrbach 1995
Picot, Arnold ; Rohrbach, Peter: Organisatorische Einsatzmöglichkeiten von Workflow-Management-Systemen. In: *DIN-Mitteilungen + elektronorm* 74 (1995), Nr. 4, S. 230–236

Porter und Calder 2003
Porter, Ron ; Calder, Paul: A Pattern-Based Problem-Solving Process for Novice Programmers. In: *ACE* 2003 (2003), S. 231–238

Proulx 1995
Proulx, Viera K.: Computer science / informatics: the study of the Information World. In: **(Tinsley und Weert 1995)**, S. 495–503

Proulx 2000
Proulx, Viera K.: Programming Patterns and Design Patterns in the Introductory Computer Science Course. In: *SIGCSE ACM* 2000 (2000), Nr. 3, S. 80–84

Puhlmann 2005
Puhlmann, Hermann: Bildungsstandards Informatik - zwischen Vision und Leistungstests. In: **(Friedrich 2005)**, S. 79–89

Ramsky und Rezina 2005
Ramsky, Yuri ; Rezina, Olga: Study of Information Search Systems of the Internet. In: **(Mittermeir 2005)**, S. 84–91

Rehm
Rehm, Markus: Das empirische Forschungsinstrument „Phänomenprotokoll". In: Höttecke, Dietmar (Hrsg.): *Jahrestagung der Gesellschaft für Didaktik der Chemie und Physik, Bern*

Rehm 2006
Rehm, Markus: Allgemeine naturwissenschaftliche Bildung – Entwicklung eines vom Begriff „Verstehen" ausgehenden Kompetenzmodells, 2006, S. 45–66

Reinmann 2006
Reinmann, Gabi: Nur „Forschung danach"? Vom faktischen und potentiellen Beitrag der Forschung zu alltagstauglichen Innovationen beim E-Learning. Arbeitsbericht Universität Augsburg, Nr. 14, 2006. – http://www.imb-uni-augsburg.de/files/Arbeitsbericht14.pdf – geprüft: 22. Februar 2009

Reischuk 1999
Reischuk, Karl R.: *Komplexitätstheorie Bd. 1: Grundlagen* . 2. Aufl. Stuttgart, Leipzig : Teubner Verlag, 1999 (Leitfäden u. Monographien der Informatik). – ISBN 978–3519122753

Renkl und Nückles 2006
Renkl, Alexander ; Nückles, Matthias: Lernstrategien der externen Visualisierung. In: Mandl, Heinz (Hrsg.) ; Friedrich, Helmut F. (Hrsg.): *Handbuch Lernstrategien. Analyse und Intervention.* Göttingen : Hogrefe-Verlag, 2006. – ISBN 3–8017–1813–1, S. 135–147

Resnick 1992
Resnick, Mitchel: *Beyond the centralized mindset: explorations in massively-parallel microworlds.* Cambridge, MA, USA, Dissertation, 1992

Rincon et al. 2005
Rincon, Fernando ; Moya, Francisco ; Barba, Jesus ; Lopez, Juan C.: Model Reuse through Hardware Design Patterns. In: *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe.* Washington, DC, USA : IEEE Computer Society, 2005. – ISBN 0–7695–2288–2, S. 324–329

Romeike 2007
Romeike, Ralf: Three drivers for creativity in computer science education. In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Ropohl 1991
Ropohl, Günter: *Technologische Aufklärung. Beiträge zur Technikphilosophie.* 1. Aufl. Frankfurt am Main : Suhrkamp, 1991

Rumbaugh et al. 2005
Rumbaugh, James ; Jacobson, Ivar ; Booch, Grady: *The Unified Modeling Language Reference Manual.* Boston : Addison-Wesley, 2005. – ISBN 3–431–02616–8

Samways 2005
Samways, Brian (Hrsg.): *35 Years of Computers in Education: What Works?, Proceedings of IFIP 8th World Conference on Computers in Education - WCCE 2005, July 4-7, 2005, University of Stellenbosch, Western Cape, South Africa.* Document Transformation Technologies cc, 2005

Santos et al. 2009
Santos, Elder R. (Hrsg.) ; Miletto, Evandro M. (Hrsg.) ; Turcsanyi-Szabo, Marta (Hrsg.): *WCCE 2009 – Education and Technology for a Better World. 9th IFIP World Conference on Computers in Education. July 27th - 31th, 2009. Bento Gonçalves. Brasilien (CD Publication).* WCCE 2009, 2009 . – ISBN 978–3–901882–35–7

Schefe 1999
Schefe, Peter: Softwaretechnik und Erkenntnistheorie. In: *Informatik Spektrum* 22 (1999), Nr. 2, S. 122–135

Schneider 2003
Schneider, Markus: Design Pattern, a topic of the new mandatory subject informatics? In: Weert, Tom J. (Hrsg.) ; Munro, Robert K. (Hrsg.) ; IFIP TC 3 (Veranst.): *Informatics and the Digital Society – Social, Ethical and Cognitive Issues.* Norwell, Massachusetts : Kluwer Academic Publishers, April 2003. – ISBN 1–4020–7363–1, S. 157–171

Schneider 2005
Schneider, Markus: A Strategy to Introduce Functional Data Modeling at School Informatics. In: **(Mittermeir 2005)**, S. 130–144

Schobel und Holdt 2004
Schobel, K. (Hrsg.) ; Holdt, U. (Hrsg.): *Entwicklung und Erprobung eines integrierten Leistungspunktsystems in der Weiterentwicklung modularisierter Studienangebote am Beispiel der Ingenieurwissenschaften. Verifikation von Qualifikation in einem Leistungspunktsystem. Abschlussbericht.* Hannover : Universität Hannover, 2004. – http://www4.tu-ilmenau.de/lps/hannover/Abschlussbericht_Hannover.pdf – geprüft 22. Februar 2009

Schreiber 1983
Schreiber, Alfred: Bemerkungen zur Rolle universeller Ideen im mathematischen Denken. In: *mathematica didactica* 6 (1983), S. 65–76

Schubert 1991
Schubert, Sigrid: Fachdidaktische Fragen der Schulinformatik und (un)mögliche Antworten. In: Gorny, Peter (Hrsg.): *INFOS* Bd. 292, Springer, 1991 (Informatik-Fachberichte). – ISBN 3–540–54619–7, S. 27–33

Schubert 1995
Schubert, Sigrid (Hrsg.): *Innovative Konzepte für die Ausbildung, 7. GI-Fachtagung Informatik und Schule, INFOS'95, Chemnitz, 25.-28. September 1995.* Springer, 1995 (Informatik Aktuell). – ISBN 3–540–60245–3

Schubert 2005
Schubert, Sigrid: From Didactic Systems to Educational Standards. **(Samways 2005)**. – 397.pdf

Schubert 2007
Schubert, Sigrid (Hrsg.): *Didaktik der Informatik in Theorie und Praxis, INFOS 2007, 12. GI-Fachtagung Informatik und Schule, 19.-21. September 2007 an der Universität Siegen.* Bd. *112.* GI, 2007 (LNI). – ISBN 978–3–88579–206–2

Schubert 2008
Schubert, Sigrid: Forschungsbeispiele zur Didaktik der Informatik. In: **(Schubert und Stechert 2008)**, S. 7–32

Schubert et al. 2007
Schubert, Sigrid ; Freischlad, Stefan ; Stechert, Peer ; Kempf, Wolfgang ; Koch, Hartmud: Internetworking und Verstehen von Informatiksystemen. In: **(Stechert 2007a)**, S. 65–74

Schubert et al. 2002
Schubert, Sigrid (Hrsg.) ; Magenheim, Johannes (Hrsg.) ; Hubwieser, Peter (Hrsg.) ; Brinda, Torsten (Hrsg.): *Forschungsbeiträge zur Didaktik der Informatik – Theorie, Praxis, Evaluation, Tagungsband des 1. Workshops der GI-Fachgruppe „Didaktik der Informatik" (DDI'02), Schwerpunkt: Modellierung in der informatischen Bildung, 10.-11. Oktober 2002, Witten-Bommerholz.* Bd. *22.* GI, 2002 (LNI). – ISBN 3–88579–351–2

Schubert und Schwill 2004
Schubert, Sigrid ; Schwill, Andreas: *Didaktik der Informatik.* 1. Aufl. Heidelberg, Berlin : Spektrum Akademischer Verlag, 2004. – ISBN 978–3–8274–1382–6

Schubert und Stechert 2008
SCHUBERT, Sigrid (Hrsg.) ; STECHERT, Peer (Hrsg.): *Bildungskonzepte für Internetworking und eingebettete Mikrosysteme. Didaktik der Informatik für E-Learning, Schule und Hochschule. Kolloquium des Teilprojektes A8 „Informatikunterricht und E-Learning zur aktiven Mitwirkung am digitalen Medienumbruch" im DFG SFB/FK 615 „Medienumbrüche"*. Bd. *8*. Universitätsverlag Siegen – universi, 2008 (Reihe Medienwissenschaften). – ISBN 978–3–936533–28–6

Schubert et al. 2005a
SCHUBERT, Sigrid ; STECHERT, Peer ; FREISCHLAD, Stefan: Digitaler Medienumbruch. In: *LOG IN* 25 (2005), Nr. 135, S. 7–8

Schubert et al. 2005b
SCHUBERT, Sigrid ; STECHERT, Peer ; FREISCHLAD, Stefan: Die Phisher im Internet – Ein Beitrag zu Standards der informatischen Bildung. In: *LOG IN* 25 (2005), Nr. 135, S. 66–68

Schubert et al. 2009
SCHUBERT, Sigrid ; STECHERT, Peer ; FREISCHLAD, Stefan: Information Technology Learning Aids for Informatics. In: MCDOUGALL, Anne (Hrsg.): *Researching IT in Education: Theory, Practice and Future Directions*, Routledge, 2009. – (im Druck)

Schulmeister 2002
SCHULMEISTER, Rolf: Taxonomie der Interaktivität von Multimedia – Ein Beitrag zur aktuellen Metadaten-Diskussion. In: *it + ti – Informationstechnik und Technische Informatik* 44 (2002), S. 193–199

Schulte 2001
SCHULTE, Carsten: Vom Modellieren zum Gestalten – Objektorientierung als Impuls für einen neuen Informatikunterricht? In: *informatica didactica* (2001), Juli, Nr. 3. – Ausgewählte Beiträge der Tagung „IAB2000 – Informatik und Ausbildung"; http://didaktik.cs.uni-potsdam.de/InformaticaDidactica/Issue3 – geprüft 22. Februar 2009

Schulte 2004
SCHULTE, Carsten: *Lehr- Lernprozesse im Informatik-Anfangsunterricht: theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II*, Universität Paderborn, Didaktik der Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Dissertation, März 2004. – http://ubdata.uni-paderborn.de/ediss/17/2003/schulte/disserta.pdf – geprüft: 22. Februar 2009

Schulte 2008
SCHULTE, Carsten: Duality Reconstruction – Teaching Digital Artifacts from a Sociotechnical Perspective. In: MITTERMEIR, Roland T. (Hrsg.) ; SYSLO, Maciej M. (Hrsg.): *ISSEP* Bd. 5090, Springer, 2008 (Lecture Notes in Computer Science). – ISBN 978–3–540–69923–1, S. 110–121

Schulte und Block 2002
SCHULTE, Carsten ; BLOCK, Ulrich: Das Sieben-Schritte-Schema zur Dekonstruktion objektorientierter Software. In: **(Schubert et al. 2002)**, S. 3–12

Schulz-Zander et al. 1993
SCHULZ-ZANDER, Renate ; 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: Empfehlungen für das Fach Informatik in der Sekundarstufe II allgemeinbildender Schulen der Gesellschaft für Informatik e. V. In: **(Troitzsch 1993)**, S. 205–218

Schweiger 1982
Schweiger, Fritz: Fundamentale Ideen der Analysis und handlungsorientierter Unterricht. In: *Beiträge zum Mathematikunterricht* (1982), S. 103–111

Schwidrowski 2007
Schwidrowski, Kirstin: Introducing Internetworking in Vocational Training. In: Abbott, Chris (Hrsg.) ; Lustigova, Zdena (Hrsg.): *Information Technologies for Education and Training: iTET 2007. IFIP WG 3.6 and 3.4 Joint Working Conference. Prag, Czech Republic, 26th-28th September 2007*, ETIC Prague, 2007. – ISBN 978–80–254–0391–4, S. 154–161

Schwidrowski et al. 2009
Schwidrowski, Kirstin ; Schmidt, Thilo ; Brück, Rainer ; Freischlad, Stefan ; Schubert, Sigrid ; Stechert, Peer: Mikrosystemverständnis im Hochschulstudium – Ein praktikumsorientierter Ansatz. In: Schwill, A. (Hrsg.): *3. GI-Fachtagung „Hochschuldidaktik Informatik" (HDI). 4.-5. Dezember 2008*, Universität Potsdam, Reihe „Commentarii informaticae didacticae", Bd. 1, 2009, S. 131–142

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

Schwill 1993b
Schwill, Andreas: Verifikation – zu schwierig für die Schule? – Drei Gegenbeispiele! Teil 1. In: *LOG IN* 13 (1993), Nr. 6, S. 45–48

Schwill 1994
Schwill, Andreas: Verifikation – zu schwierig für die Schule? – Drei Gegenbeispiele! Teil 2. In: *LOG IN* 14 (1994), Nr. 1, S. 37–43

Schwill 1999
Schwill, Andreas (Hrsg.): *Informatik und Schule, Fachspezifische und fachübergreifende didaktische Konzepte, 8. GI-Fachtagung Informatik und Schule, INFOS99, Potsdam, 22.-25. September 1999*. Springer, 1999 (Informatik Aktuell). – ISBN 3–540–66300–2

Schwill et al. 2006
Schwill, Andreas (Hrsg.) ; Schulte, Carsten (Hrsg.) ; Thomas, Marco (Hrsg.): *Didaktik der Informatik, Tagungsband des 3. Workshops der GI-Fachgruppe „Didaktik der Informatik" (DDI'06), 19.-20. Juni 2006, Potsdam*. Bd. *99*. GI, 2006 (LNI). – ISBN 978–3–88579–193–5

Seiffert 1989
Seiffert, Helmut ; Seiffert, Helmut (Hrsg.) ; Radnitzky, Gerard (Hrsg.): *Handlexikon zur Wissenschaftstheorie*. München : Ehrenwirth Verlag, 1989. – ISBN 3–431–02616–8

Senkbeil und Drechsel 2004
Senkbeil, Martin ; Drechsel, Barbara: Vertrautheit mit dem Computer. In: Prenzel, Manfred (Hrsg.) ; Baumert, Jürgen (Hrsg.) ; Blum, Werner (Hrsg.) ; Lehmann, Rainer (Hrsg.) ; Leutner, Detlev (Hrsg.) ; Neubrand, Michael (Hrsg.) ; Pekrun, Reinhard (Hrsg.) ; Rost, Jürgen (Hrsg.) ; Schiefele, Ulrich (Hrsg.): *PISA 2003 – Der Bildungsstand der Jugendlichen in Deutschland – Ergebnisse des zweiten internationalen Vergleichs*, 2004, S. 177–190

Shalloway und Trott 2002
Shalloway, Alan ; Trott, James R.: *Design patterns explained: a new perspective on object-oriented design*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2002. – ISBN 0–201–71594–5

Sülz 2007
Sülz, Daniel ; Stechert, Peer (Hrsg.) ; Schubert, Sigrid (Hrsg.): *Zuverlässigkeit und Kompatibilität von Informatiksystemen: Unterrichtsbeispiele zur Förderung des Informatiksystemverständnisses*. Hauptseminararbeit. Lehrstuhl für Didaktik der Informatik und E-Learning. Universität Siegen, 2007

Stachowiak 1973
Stachowiak, Herbert: *Allgemeine Modelltheorie*. Springer, Wien, 1973

Stechert 2006a
Stechert, Peer: Informatics System Comprehension – A learner-centred cognitive approach to networked thinking. In: **(Watson und Benzie 2006)**. – (CD-Publikation; 13 Seiten)

Stechert 2006b
Stechert, Peer: Informatics system comprehension: A learner-centred cognitive approach to networked thinking. In: *Education and Information Technologies* 11 (2006), Nr. 3-4, S. 305–318. – ISSN 1360–2357. – http://dx.doi.org/10.1007/s10639-006-9014-4 – geprüft: 22. Februar 2009

Stechert 2006c
Stechert, Peer: Unterrichtsmodellentwicklung zur Förderung des Informatiksystemverständnisses mit Entwurfsmustern. In: **(Schwill et al. 2006)**, S. 89–98

Stechert 2007a
Stechert, Peer (Hrsg.): *Informatische Bildung in der Wissensgesellschaft. Praxisband der 12. GI-Fachtagung Informatik und Schule, 19.-21. September 2007 an der Universität Siegen*. Bd. *6*. Universitätsverlag Siegen – universi, 2007 (Reihe Medienwissenschaften). – ISBN 978–3–936533–23–1

Stechert 2007b
Stechert, Peer: Understanding of Informatics Systems – A theoretical framework implying levels of competence. In: Berglund, Anders (Hrsg.) ; Wiggberg, Mattias (Hrsg.): *6th Baltic Sea Conference on Computing Education Koli Calling 2006*. New York, NY, USA : ACM, 2007, S. 128–131. – http://doi.acm.org/10.1145/1315803.1315827 – geprüft: 22. Februar 2009

Stechert 2007c
Stechert, Peer: Von vernetzten fundamentalen Ideen zum Verstehen von Informatiksystemen – Eine Unterrichtserprobung in der Sekundarstufe II. In: **(Schubert 2007)**, S. 183–194

Stechert 2008a
Stechert, Peer: Combining Different Perspectives on Informatics Systems – A Case Study at Upper Secondary Level. In: **(Wheeler et al. 2008)**. – (CD-Publikation; 10 Seiten)

Stechert 2008b
Stechert, Peer: Exemplarische Betrachtungen zu lernförderlicher Software mit Entwurfsmustern für Informatiksystemverständnis. In: Brinda, Torsten (Hrsg.) ; Fothe, Michael (Hrsg.) ; Hubwieser, Peter (Hrsg.) ; Schlüter, Kirsten (Hrsg.): *Didaktik der Informatik – Aktuelle Forschungsergebnisse. 5. Workshop der GI-Fachgruppe „Didaktik der Informatik"*. Bonn : Köllen, 2008 (Lecture Notes in Informatics (LNI) 135). – ISBN 978–3–88579–229–1, S. 55–64

Stechert 2008c
Stechert, Peer: Systematic Exploration of Informatics Systems. In: Kendall, Mike (Hrsg.) ; Samways, Brian (Hrsg.): *Learning to Live in the Knowledge Society* Bd. 281, Springer, 2008. – ISBN 978–0–387–09728–2, S. 359–360

Stechert et al. 2009
Stechert, Peer ; Schubert, Sigrid ; Kollee, Christian: Classroom Practice Project "UNIS – Understanding of Informatics Systems". In: **(Santos et al. 2009)**. – (CD-Publikation; 10 Seiten)

Stechert und Schubert 2007
Stechert, Peer ; Schubert, Sigrid E.: A Strategy to Structure the Learning Process Towards Understanding of Informatics Systems. In: **(Benzie und Iding 2007)**. – (CD-Publikation; 11 Seiten)

Steinbock 1993
Steinbock, Hans-Joachim: *Unternehmerische Potentiale der Informationstechnik in den neunziger Jahren*, Universität St. Gallen, Dissertation, 1993

Steinert 2007
Steinert, Markus: Lernzielgraphen und Lernzielerfolgsanalyse. In: **(Schubert 2007)**, S. 147–158

Steinkamp 1999
Steinkamp, Dirk: *Informatikexperimente im Schullabor*. Diplomarbeit. Universität Dortmund : Fachbereich Informatik, 1999. – http://www.die.informatik.uni-siegen.de/forschung/steinkamp – geprüft 22. Februar 2009

Stone, Harold S. 1975
Stone, Harold S. (Hrsg.): *Introduction to Computer Architecture*. . Chicago : Science Research Associates Inc., 1975

Stupperich und Warkentin 2007
Stupperich, Pamina ; Warkentin, Swetlana ; Stechert, Peer (Hrsg.) ; Schubert, Sigrid (Hrsg.): *„Laut-Denken" für die Didaktik der Informatik am Beispiel von Vorgehensweisen für das Verstehen von Informatiksystemen*. Hauptseminararbeit. Lehrstuhl für Didaktik der Informatik und E-Learning. Universität Siegen, 2007

Tanenbaum und Goodman 2001
Tanenbaum, Andrew ; Goodman, James: *Computerachitektur*. Prentice Hall, 2001

Tao 2000
Tao, Yongiel: Teaching Software Tools via Design Patterns. In: *Australasian Conference on Computing Education* 2000 (2000), Nr. 262, S. 248–252

Tedre et al. 2006
Tedre, Matti ; Sutinen, Erkki ; Kähkönen, Esko ; Kommers, Piet: Ethnocomputing: ICT in cultural and social context. New York, NY, USA : ACM, 2006. – ISSN 0001–0782, S. 126–130

Tergan und Keller 2005
Tergan, Sigmar-Olaf ; Keller, Tanja: Digital Concept Mapping in Learning Contexts: Integrating Knowledge, Arguments and Information Resources. In: *Proceedings of the Ninth International Conference on Information Visualisation (IV'05). IEEE.* (2005). – http://ieeexplore.ieee.org/iel5/10086/32319/01509103.pdf – geprüft: 22. Februar 2009

Thaller 1994
Thaller, Georg E.: *Verifikation und Validation: Software-Test für Studenten und Praktiker*. Braunschweig; Wiesbaden : Vieweg, 1994. – ISBN 3–528–05442–5

Thomas 2001
THOMAS, Marco: Die Vielfalt der Modelle in der Informatik. In: **(Keil-Slawik und Magenheim 2001)**, S. 173–186

Thomas 2002
THOMAS, Marco: *Informatische Modellbildung – Modellieren von Modellen als ein zentrales Element der Informatik für den allgemeinbildenden Schulunterricht*, Universität Potsdam Didaktik der Informatik, Dissertation, Juli 2002. – http://ddi.cs.uni-potsdam.de/Personen/marco/Informatische_Modellbildung_Thomas_2002.pdf – geprüft: 22. Februar 2009

Thomas 2003
THOMAS, Marco: Informatische Modelle zur Strukturierung von Anfangsunterricht. In: **(Hubwieser 2003)**, S. 155–164

Thurber und Stratton 1995
THURBER, Barton D. ; STRATTON, Jerry: Computers, telecommunications and Western culture. In: **(Tinsley und Weert 1995)**, S. 871–878

Tichy 1997
TICHY, Walter F.: A Catalogue of General-Purpose Software Design Patterns. In: *TOOLS '97: Proceedings of the Tools-23 „Technology of Object-Oriented Languages and Systems".* Washington, DC, USA : IEEE Computer Society, 1997. – ISBN 0–8186–8383–X, S. 330–339

Tinsley und Weert 1995
TINSLEY, David (Hrsg.) ; WEERT, Tom J. v. (Hrsg.): *Liberating the Learner, IFIP TC3 Sixth IFIP World Conference on Computers in Education, WCCE 1995, 1995, Birmingham, United Kingdom.* Chapman & Hall, 1995 (IFIP Conference Proceedings). – ISBN 0–412–62670–5

Tort und Blondel 2007
TORT, Françoise ; BLONDEL, François-Marie: Uses of spreadsheets and assessment of competencies of high school students. In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Troitzsch 1993
TROITZSCH, Klaus G. (Hrsg.): *Informatik als Schlüssel zur Qualifikation, GI-Fachtagung Informatik und Schule 1993, Koblenz, 11.-13. Oktober 1993.* Springer, 1993 (Informatik Aktuell). – ISBN 3–540–57256–2

Tulodziecki und Herzig 1998
TULODZIECKI, Gerhard ; HERZIG, Bardo: Praxis- und theorieorientierte Entwicklung und Evaluation von Konzepten für pädagogisches Handeln, 1998. – Universität Paderborn, Institut für Erziehungswissenschaft Ruhr-Universität Bochum, Institut für Pädagogik. http://www.bardo-herzig.de/blog/media/theorie.pdf – geprüft: 22. Februar 2009

Ufer 2007
UFER, Jonathan: *Architekturmuster als Beitrag zum Informatiksystemverständnis.* Diplomarbeit. Lehrstuhl für Didaktik der Informatik und E-Learning. Universität Siegen, 2007. – http://www.die.informatik.uni-siegen.de/forschung/Ufer/Architekturmuster.pdf – geprüft: 22. Februar 2009

Ullenboom 2005
ULLENBOOM, Christian: *Vermittlung von Entwurfsmustern in der informatischen Ausbildung am Beispiel eines Media Players. Diplomarbeit. Arbeitsgruppe Didaktik der Informatik der Universität Paderborn.* 2005

Ulrich und Probst 1988
Ulrich, Hans ; Probst, Gilbert J.: *Anleitung zum ganzheitlichen Denken und Handeln. Ein Brevier für Führungskräfte.* Bern : Paul Haupt Verlag, 1988. – ISBN 978–3258051826

UNESCO 1994
UNESCO ; Weert, Tom van (Hrsg.) ; Tinsley, David (Hrsg.): *INFORMATICS FOR SECONDARY EDUCATION – A Curriculum for Schools.* Original 1994. Paris : UNESCO. Produced by working party of the IFIP under auspices of UNESCO. Paris, 1994. – http://wwwedu.ge.ch/cptic/prospective/projets/unesco/1994/en/welcome.html – geprüft: 22. Februar 2009

UNESCO 2002
UNESCO ; Weert, Tom J. v. (Hrsg.): *Information and Communication Technology in Education – A Curriculum for Schools and Programme of Teacher Development.* Paris : UNESCO, 2002. – http://unesdoc.unesco.org/images/0012/001295/129538e.pdf – geprüft: 22. Februar 2009

UNESCO 2005
UNESCO (Hrsg.): *Towards knowledge societies: UNESCO world report.* UNESCO Publishing. Paris, 2005. – ISBN 92–3–104000–6. – http://unesdoc.unesco.org/images/0014/001418/141843e.pdf – geprüft 22. Februar 2009

UNESCO 2008
UNESCO ; Forest Woody Horton, Jr. (Hrsg.): *Understanding Information Literacy: A Primer.* Paris : UNESCO, 2008. – http://unesdoc.unesco.org/images/0015/001570/157020e.pdf – geprüft 22. Februar 2009

Unger-Lamprecht 2001
Unger-Lamprecht, Barbara: *Experimentelle Bewertung der Auswirkungen von Entwurfsmustern*, Universität Karlsruhe, Dissertation, Januar 2001. – ISBN 3935363117

Verhoeff 2006
Verhoeff, Tom: A Master Class Software Engineering for Secondary Education. In: **(Mittermeir 2006)**, S. 150–158

Vester 1988
Vester, Frederic: *Leitmotiv vernetztes Denken.* München : Heyne Verlag, 1988. – ISBN 3–453–02865–1

Voß 2006
Voss, Siglinde: *Modellierung von Standardsoftwaresystemen aus didaktischer Sicht*, Technische Universität München, Institut für Informatik, Dissertation, 2006

Voß 2005a
Voss, Siglinde: Informatic Models in Vocational Training for Teaching Standard Software. In: **(Mittermeir 2005)**, S. 145–155

Voß 2005b
Voss, Siglinde: Informatische Bildung in Anwenderschulungen. In: **(Friedrich 2005)**, S. 285–296

Wagenschein 1991
Wagenschein, Martin: *Verstehen lehren. genetisch – sokratisch – exemplarisch.* Weinheim, Basel : Beltz, 1991. – ISBN 3–407–29001–2. – 9. Aufl.

Wallingford 1996
WALLINGFORD, Eugene: Toward a First Course Based on Object-Oriented Patterns. In: *SIGCSE ACM* 1996 (1996), Nr. 2, S. 27–31

Watson und Andersen 2002
WATSON, Deryn (Hrsg.) ; ANDERSEN, Jane (Hrsg.): *Networking the Learner: Computers in Education, IFIP TC3 Seventh IFIP World Conference on Computers in Education, WCCE 2001, July 29 - August 3, 2001, Copenhagen, Denmark*. Bd. *217*. Kluwer, 2002 (IFIP Conference Proceedings). – ISBN 1–4020–7133–7

Watson und Benzie 2006
WATSON, Deryn (Hrsg.) ; BENZIE, David (Hrsg.): *Imagining the future for ICT and Education*. Hogskolen Ålesund, 2006 . – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Alesund-CD.zip – geprüft: 22. Februar 2009

Wedekind et al. 1998
WEDEKIND, Hartmut ; GÖRZ, Günther ; KÖTTER, Rudolf ; INHETVEEN, Rüdiger: Modellierung, Simulation, Visualisierung: Zu aktuellen Aufgaben der Informatik (Zur Diskussion gestellt). In: *Informatik Spektrum* 21 (1998), Nr. 5, S. 265–272

Wedekind et al. 2004
WEDEKIND, Hartmut ; ORTNER, Erich ; INHETVEEN, Rüdiger: Informatik als Grundbildung. In: *Informatik Spektrum* 27 (2004), Nr. 2, S. 172–180

Weert 1984
WEERT, Tom J. v.: *A model syllabus for literacy in information technology for all teachers*. Bruxelles : Association for Teacher Education in Europe, 1984. – ISBN 2–87125–014–6

Weert 1993
WEERT, Tom J. v.: Informatik als Teil der Allgemeinbildung. In: **(Troitzsch 1993)**, S. 11–19

Weert 1995
WEERT, Tom J. v.: IFIP Working Group 3.1: towards integration of computers into education. In: **(Tinsley und Weert 1995)**, S. 3–12

Weert und Kendall 2005
WEERT, Tom v. ; KENDALL, Mike: Growing Importance of Lifelong Learning with ICT. In: **(Samways 2005)**. – 408.pdf

Wegner 1997
WEGNER, Peter: Why interaction is more powerful than algorithms. In: *Commun. ACM* 40 (1997), Nr. 5, S. 80–91. – ISSN 0001–0782. – http://doi.acm.org/10.1145/253769.253801 – geprüft 22. Februar 2009

Weigend 2006
WEIGEND, Michael: Experimental Programming. In: **(Watson und Benzie 2006)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Alesund-CD.zip – geprüft: 22. Februar 2009

Weigend 2007
WEIGEND, Michael: Origins of action – protagonists in drama-like interpretations of computer programmes. In: **(Benzie und Iding 2007)**. – http://www.die.informatik.uni-siegen.de/ifip-wg31/Proceedings/Boston-CD.zip – geprüft: 22. Februar 2009

Weinert 2001
Weinert, Franz E.: Vergleichende Leistungsmessung in Schulen – eine umstrittene Selbstverständlichkeit. In: Weinert, Franz E. (Hrsg.): *Leistungsmessungen in Schulen*. Weinheim : Beltz, 2001. – ISBN 3–407–25256–0, S. 17–31. – 2. Aufl.

Wende 2002
Wende, Ingo: Normen und Spezifikationen der Informationstechnik. In: Rechenberg, Peter (Hrsg.) ; Pomberger, Gustav (Hrsg.): *Informatikhandbuch*. München : Hanser Fachbuch, 2002, S. 1113–1132

Weyer 2007a
Weyer, Michell ; Stechert, Peer (Hrsg.) ; Schubert, Sigrid (Hrsg.): *Konzeption und Evaluation des Unterrichtsprojekts an einem Siegener Gymnasium mit dem Schwerpunkt Verstehen von Informatiksystemen*. Hauptseminararbeit. Lehrstuhl für Didaktik der Informatik und E-Learning. Universität Siegen, 2007. – http://www.die.informatik.uni-siegen.de/gruppe/stechert/informatics-systems/seminararbeiten/unterrichtsprojekt.pdf – geprüft: 22. Februar 2009

Weyer 2007b
Weyer, Michell: *Systemorientierte Klassifikation von Entwurfsmustern und Erarbeitung des Entwurfs einer Lernsoftware*. Diplomarbeit. Lehrstuhl für Didaktik der Informatik und E-Learning. Universität Siegen, 2007. – http://www.die.informatik.uni-siegen.de/forschung/Weyer/Entwurfsmusterklassifikation_weyer.pdf – geprüft: 22. Februar 2009

Wheeler et al. 2008
Wheeler, Steve (Hrsg.) ; Brown, Doug (Hrsg.) ; Kassam, Alnaaz (Hrsg.): *ICT and Learning for the Net Generation*. Open University Malaysia, 2008 . – ISBN 978–3–901882–29–6. – http://cs.anu.edu.au/iojs/index.php/ifip/issue/view/41 – geprüft: 22. Februar 2009

Wiesner und Brinda 2007
Wiesner, Bernhard ; Brinda, Torsten: Erfahrungen bei der Vermittlung algorithmischer Grundstrukturen im Informatikunterricht der Realschule mit einem Robotersystem. In: **(Schubert 2007)**, S. 113–124

Winograd und Flores 1988
Winograd, Terry (Hrsg.) ; Flores, Fernando (Hrsg.): *Understanding Computers and Cognition. A New Foundation for Design*. Reading, MA. : Addison Wesley, 1988. – ISBN 0–201–11297–3. – 3. Aufl.

Witten und Penon 1997
Witten, Helmut ; Penon, Johann: Internet und Informatik – „Runderneuerung" für den Unterricht? In: *LOG IN* 17 (1997), Nr. 6, S. 10–17

Wittmann 1981
Wittmann, Erich: *Grundfragen des Mathematikunterrichts*. 6. Aufl. Braunschweig : Vieweg Verlag, 1981

Wursthorn 2006
Wursthorn, Birgit: *Informatische Grundkonzepte in Klasse 5 der Realschule – Entwurf und Evaluation von fächerübergreifendem Unterricht*. Ludwigsburg, Pädagogische Hochschule Ludwigsburg, Dissertation, 2006

Zammit und Downes 2001
Zammit, Katina ; Downes, Toni: Tracking Technology and Literacy Practices. In: **(Watson und Andersen 2002)**, S. 189–198

Zimmer 1995
ZIMMER, Walter: Relationships Between Design Patterns. In: COPLIEN, James O. (Hrsg.) ; SCHMIDT, Douglas C. (Hrsg.): *Pattern Languages of Program Design.* Amsterdam : Addison Wesley, 1995, S. 345–364

Bisher in dieser Reihe erschienene Bände:

1 Andreas Schwill (Hrsg.): Hochschuldidaktik der Informatik. HDI2008 – 3. Workshop des GI-Fachbereichs Ausbildung und Beruf / Didaktik der Informatik 2008
2009 | ISBN 978-3-940793-75-1

2 Stechert, Peer: Fachdidaktische Diskussion von Informatiksystemen und der Kompetenzentwicklung im Informatikunterricht
2009 | ISBN 978-3-86956-024-3

In dieser Reihe erscheinen Tagungsbände und ausgewählte Forschungsberichte zu Themen aus der Didaktik der Informatik in Schule und Hochschule.

In dieser Dissertationsschrift wird ein Unterrichtsmodell zur Kompetenzentwicklung mit Informatiksystemen für die Sekundarstufe II vorgestellt. Ausgehend vom Kompetenz- und Informatiksystembegriff wird der fachdidaktische Forschungsstand historisch analysiert.

In dem bildungstheoretisch hergeleiteten Unterrichtsmodell erfolgt der Zugang zu Informatiksystemen in ihrer Einheit aus Hardware, Software und Vernetzung über ihr nach außen sichtbares Verhalten, ihre innere Struktur und ausgewählte Implementierungsaspekte. Fokussiert wird auf

- vernetzte fundamentale Ideen der Informatik in objektorientierten Entwurfsmustern,
- Experimente mit Informatiksystemen in Anlehnung an systematisches (Software-) Testen,
- die Gestaltung von Lernsoftware zur Förderung der Kompetenzentwicklung mit Informatiksystemen.

Die normative Perspektive wird um Rückkopplung mit der Praxis ergänzt. Dafür werden zwei Erprobungen im Informatikunterricht und ihre Evaluationsergebnisse fachdidaktisch diskutiert.

ISSN 1868-0844
ISBN 978-3-86956-024-3