

# **Dekonstruktion von Informatiksystemen als Unterrichtsmethode**

## **- Zugang zu objektorientierten Sichtweisen im Informatikunterricht**

Thorsten Hampel  
Heinz Nixdorf Institut  
Informatik und Gesellschaft  
Fürstenallee 11  
33102 Paderborn  
E-Mail: [hampel@uni-paderborn.de](mailto:hampel@uni-paderborn.de)  
Tel. 05251 60 6651

Johannes Magenheim  
Universität GH Paderborn  
Didaktik der Informatik  
Fürstenallee 11  
33102 Paderborn  
E-Mail: [jsm@uni-paderborn.de](mailto:jsm@uni-paderborn.de)  
Tel. 05251 60 6341

Carsten Schulte  
Universität GH Paderborn  
Didaktik der Informatik  
Fürstenallee 11  
33102 Paderborn  
E-Mail: [carsten@uni-paderborn.de](mailto:carsten@uni-paderborn.de)  
Tel. 05251 60 6340

Im didaktischen Forschungsprojekt OMI (objektorientiertes Modellieren im Informatikunterricht) am Fachbereich Mathematik / Informatik der Universität GH – Paderborn soll er- kundet werden, ob Schülerinnen und Schülern Einsicht in Verfahren der objektorientierten Modellierung sowie in relevante Methoden der Gestaltung und Bewertung von Informatik- systemen gegeben werden kann. Dazu wird eine didaktische Software eingesetzt, die ein kleines Warenwirtschaftssystem in Form eines Schulkiosks simuliert und ein - auf schulis- ches Niveau bezogen - relativ ‚komplexes Informatiksystem‘ darstellt. Dieses wird von den Lernenden dekonstruiert. Für konstruierende Unterrichtsphasen werden didaktisch aufberei- tete Modellierungstechniken und -notationen sowie angepasste Entwicklungswerkzeuge ver- wendet. In diesem Artikel sollen dem Forschungsprojekt zugrundeliegende fachdidaktische Positionen kurz skizziert, Methoden, Möglichkeiten und Grenzen objektorientierten Model- lierens im Informatikunterricht erörtert, sowie weiterführende Forschungsfragen und Phasen des Projekts dargestellt werden.

## **Projektidee aus fachdidaktischer Perspektive**

Allzu oft präsentiert sich Informatikunterricht in der Praxis bei näherem Hinsehen als Programmierkurs, der orientiert am Konzept einer Programmiersprache, systematisch neue Sprachelemente einzuführen versucht. Aufgaben und Beispiele weisen häufig einen geringen Komplexitätsgrad auf oder beziehen sich auf Standardalgorithmen mit engem Bezug zur Mathematik. Problemlösestrategien, Schulung analyti- schen Denkens, Abstraktionsvermögen, Zergliedern von Problemen in Teilprobleme und andere Prob- lemlösestrategien können hierbei nur eingeschränkt eingeübt werden. Der für ein Unterrichtsfach an all- gemeinbildenden Schulen übliche Anspruch auf Vermittlung von Allgemeinbildung ist so kaum zu reali- sieren und wissenschaftspropädeutische Funktionen im Hinblick auf eine sich dynamisch entwickelnde Bezugswissenschaft Informatik sind ebenfalls in Frage gestellt.

In Abgrenzung von derartigen Konzeptionen des Informatikunterrichts kann der Ausgangspunkt einer systemorientierten Didaktik der Informatik die Modellierung und Analyse oder besser die Dekonstruktion von Informatiksystemen sein. Anhand einer in Java codierten Software, mit hinreichender jedoch unter didaktischen Gesichtspunkten reduzierten Komplexität, werden Systemfunktionen erkundet und das der Software implizite Modell eines Realitätsausschnitts so weit als möglich expliziert und mit einem realen System verglichen. Verfahren des objektorientierten Modellierens können erprobt, Entwurfsentscheidun- gen in ihrer Konsequenz für die Gestaltung des Informatiksystems nachvollzogen werden. Später werden einzelne Klassen und Objekte des Produkts analysiert, Datenstrukturen, Methoden und Ereignisse in ihrer Wechselwirkung offengelegt. Auf diese Weise sollen am Beispiel der didaktischen Software sukzessive objektorientierte Sichtweisen und Prinzipien sowie ihre programmiertechnische Codierung in Java er- schlossen werden. Beispiele und Übungsaufgaben ergänzen jeden Dekonstruktionsschritt, verbreitern und verallgemeinern die Wahrnehmungsperspektive durch Transfer in einen anderen Anwendungskontext und fördern so die Einsicht in objektorientierte Sichtweisen und Modellbildung sowie die Syntax der Pro- grammiersprache Java. Später können einzelne Zusatz-Module der Software von den Lernenden selbst

entwickelt und in die Software integriert werden. Dies setzt entsprechende Offenheit der didaktischen Software voraus. In einer fortgeschrittenen Dekonstruktionsphase können Prinzipien der objektorientierten Modellierung auf elementarer Ebene in Bezug auf eine andere Themenstellung angewandt und vertieft werden. Hier sind vereinfachte und in der Schule handhabbare Formen von CRC-, UML- und *use case*-Techniken und Notationen angesprochen. Die Hoffnung ist, das Lernende am Ende des Dekonstruktionsprozesses in der Lage sind, kleinere Softwareprojekte zunächst zu modellieren und nach objektorientierten Prinzipien selbst zu entwickeln. Mit dem Versuch einer eigenen Modellierung sollten den Lernenden auch die Grenzen dieses Konzepts im Hinblick auf die Abbildung eines realen sozio-technischen Informatiksystems und den mit ihm interagierenden Menschen bewußt werden.

Dieses dem Forschungsprojekt OMI zugrundeliegende unterrichtsmethodische Konzept und seine fachdidaktischen Implikationen sollen nun am Beispiel der Modellierung eines Informatiksystems im Umfeld eines Schulkiosks und der Software ‚Klasse Kasse‘ erläutert werden.

## **Fachdidaktische und lerntheoretische Postulate des Projekts**

Zunächst einige Bemerkungen zu fachdidaktischen Positionen, die dem Konzept des Forschungsprojekts OMI zugrunde liegen. Sie können an dieser Stelle nur skizzenhaft und thesenartig vorgetragen werden und bedürfen weiterer fachdidaktischer Diskussion. Mit ihrer Hilfe sollte es aber möglich sein, sich der Bedeutung der bei der Projektbeschreibung verwendeten Begriffe wie Informatiksystem, Modellieren, Objektorientierung, systemorientierte Didaktik oder Dekonstruktion anzunähern.

### **Objektorientierung im Informatikunterricht**

Zahlreiche Diskussionsbeiträge zur Didaktik der Informatik betonen, dass es bei der Inhaltsbestimmung und Zielsetzung für den Informatikunterricht notwendig sei, sich auf elementare und fundamentale Prinzipien der zentralen Bezugswissenschaft des Faches, die Informatik, zu konzentrieren und nicht zu versuchen, kurzlebigen Modetrends nachzulaufen. Bei dem Versuch, dieser Forderung gerecht zu werden, bleiben sie jedoch oft der Sphäre der Algorithmik verhaftet und blenden wesentliche Themenbereiche der Fachwissenschaft, wie beispielsweise Kommunikation in vernetzten Systemen, Prinzipien der Systemgestaltung und Modellierung oder der Bewertung des Einsatzes von Informatiksysteme in deren sozialem Kontext aus. Nicht zuletzt diese Gegenstandsbereiche der Informatik sind es jedoch, die die Legitimation für die Existenz des Schulfaches Informatik an allgemeinbildenden Schulen mit begründen können.

Der Versuch, objektorientiertes Modellieren und die Implementation eines Informatiksystems mittels einer objektorientierten Programmiersprache wie Java als Thema des Informatikunterrichts einzuführen, setzt sich leicht dem Verdacht aus, Modetrends zu folgen, ohne dem Informatikunterricht neue Impulse geben zu können. Würde man Java, wie oben im Zusammenhang mit der Sprache Pascal beschrieben, lediglich als weitere Programmiersprache sukzessive und der Systematik der Sprachsyntax folgend einführen, wäre dieser Verdacht sicher gerechtfertigt. Wesentliche Aspekte von Objektorientierung wären so im Unterricht nicht oder kaum vermittelbar.

In der fachwissenschaftlichen Diskussion spielen Argumente wie Sicherheit und Stabilität von Software, leichte Wartbarkeit, Wiederverwendbarkeit und Qualitätsicherung eine gewichtige Rolle, wenn es um die Begründung von objektorientierter Softwareentwicklung geht. Komplexe Systemarchitekturen und die Nutzung mächtiger Softwarebibliotheken sind oft die Charakteristika objektorientierter Softwareprojekte in der Praxis. Gerade bei komplexen Softwareprojekten entfalten objektorientierte Maximen wie Kapselung, Abstraktion, Vererbung oder Polymorphie ihre entscheidenden Vorteile gegenüber imperativen Konzepten. Eine derartige Komplexität bei der Softwareentwicklung, die das Verwenden objektorientierter Methoden dringend erforderlich machte, ist im schulischen Unterricht kaum zu realisieren. Hinzu kommt, dass die Werkzeuge, die objektorientierte Analyse-, Design- und Spezifikationsprozesse unterstützen, wie z. B. UML-Notationseditoren, CASE-tools oder visuelle Programmierumgebungen zur Gestaltung von user-interfaces, für den schulischen Gebrauch meist viel zu mächtig und damit nicht verwendbar sind.

Kann Objektorientierung trotz dieser Einwände ein Thema des Informatikunterrichts sein, und wenn ja, wie ist es realisierbar? Die Antwort in Form einer These lautet: Objektorientierte Softwareentwicklung und Objektorientierung wird in manchen Publikationen zum Paradigma erhoben und ist eine fundamen-

tale Idee der Fachwissenschaft, die es im wissenschaftspropädeutischen und allgemeinbildenden Sinn zu thematisieren gilt. Hierzu müßte es gelingen, die Komplexität einer Software, eines Informatiksystems für den unterrichtlichen Einsatz didaktisch zu reduzieren, ohne dass die Einführung objektorientierter Sichtweisen beim verwendeten Beispiel zur Trivialität verkommt. Objektorientierung sollte auch im Informatikunterricht immer im Zusammenhang mit Modellierungsprozessen und Systemgestaltung gesehen werden und den Gedanken der Wiederverwendbarkeit etwa durch das Nutzen von Systembibliotheken verdeutlichen. Ferner gilt es, geeignete interaktive Entwicklungsumgebungen bereitzustellen, die den Modellierungs- und Implementationsprozeß unterstützen. Unter diesen Prämissen könnte das Thematisieren von objektorientierter Softwareentwicklung im Informatikunterricht gelingen und ihm qualitativ neue Perspektiven erschließen.

### **Modellieren eines Informatiksystems als Unterrichtsprinzip**

Der Begriff des Informatiksystems, auf den wir uns im folgenden beziehen und der für Modellierungsprozesse und ihre unterrichtsmethodische Umsetzung von großer Bedeutung ist, wird in der didaktischen Diskussion recht unterschiedlich interpretiert (vgl. SCHULZ-ZANDER U. A. 1993, BAUMANN 1996). Es soll deshalb an dieser Stelle eine kurze Begriffsklärung erfolgen.

Informatiksysteme beinhalten soft- und hardwaretechnische Komponenten, die ihrerseits fundamentale Methoden und Ideen der Informatik und zugleich in digitaler Form materialisierte Modelle eines Realitätsausschnitts repräsentieren. Diese Modelle werden in Modellierungsprozessen entwickelt und bilden ein wesentliches Element der Systemgestaltung.

Die technische Seite eines Informatiksystems ist insofern mit seiner sozialen Seite verbunden, als sie durch *HCI (Human Computer Interaction)* und weitere direkte oder indirekte technische Funktionalitäten des Informatiksystems auf die Interaktionen der mit dem System und miteinander interagierenden Personen einwirkt. Die soziale Interaktion im Kontext eines existierenden oder zu konstruierenden technischen Informatiksystems erweitert dieses zu einem sozio-technischen System.

Informatiksysteme sind in dieser erweiterten Perspektive sozio-technische Systeme mit stark differierendem Komplexitätsgrad und unterschiedlich ausgeprägten Erscheinungsformen: Ein Fahrkartenautomat; ein PC mit einer aktuell im Arbeitsspeicher befindlichen Software inklusive der ansteuerbaren Peripheriegeräte; ein innerbetriebliches Netzwerk mit Server und Workstations; eine Fertigungsstrasse oder ein CNC-Arbeitsplatz in der Produktion; das Warenwirtschaftssystem mit Datenbanken, angeschlossenen Kassensystemen und teilautomatisierter Lagerhaltung einer Handelskette; ein Verkehrsleitsystem, etc..

Informatiksysteme maschinisieren sowohl Kopf- als auch Handarbeit des Menschen.

Informatik, die ihre gestaltenden Funktionen realisiert, ist mit dem Wandel der Industriegesellschaft verknüpft und hat sich von daher sowohl mit den theoretischen Grundlagen des Gestaltungsprozesses (Systemanalyse, Sprache, Information, Kognition) als auch mit dessen Auswirkungen zu beschäftigen. Sprache als symbolvermittelte Kommunikation bildet in diesem Zusammenhang sowohl die Grundlage der Systemgestaltung (Modellierung, formale Spezifikation, Algorithmik) als auch das Medium, in dem sich Sinnverständigung und Interaktion im sozialen Kontext des Informatiksystems vollziehen (*HCI*, sprachliche, textuelle und visuelle Kommunikation). Der kommunikative Aspekt von Informatiksystemen gewinnt bei der Systemgestaltung und in den fachwissenschaftlichen Bemühungen zur Beschreibung einer allgemeinen Theorie von Informatik eine immer größere Bedeutung. (vgl. WEGNER 1997, SCHELHOWE 1997)

Systeme zu modellieren und zu erkennen, dass ihre Gestaltung auf sorgfältiger Analyse sozialer Kommunikations- und Interaktionsstrukturen der mit dem System interagierenden Menschen und der technologischen Rahmenbedingungen ihres Handlungsfeldes beruht, dass Systemgestaltung infolgedessen nicht nur ein technischer sondern auch ein kooperativer Kommunikationsprozess ist, erscheint dann als eine Hauptaufgabe informatischer Bildung und einer systemorientierten Didaktik der Informatik. Programmierung als Teil des Konstruktionsprozesses eines Informatiksystems steht nicht mehr allein im Mittelpunkt des Informatikunterrichts, sondern ggf. eher am Ende eines Modellierungs-, Formalisierungs-, Abstraktions- und ggf. Dekonstruktionsprozesses. Objektorientierte Modellierung bewegt sich dann zwischen den Ebenen von Systemanalyse (reales Objekt), Formalisierung bzw. Abstraktion von Aufgaben und Kooperationen (abstraktes Objekt) sowie Klassendefinitionen (implementierte Klasse) (vgl. SPOLWIG 1996).

Beim Modellieren im Unterricht ist sorgfältig zwischen den sozialen Rollen der mit dem System interagierenden Menschen, den Gegenständen und Formen der Handlungsorganisation, die in den sozio-technischen Kontext des Informatiksystems einbezogen sind und den zu konstruierten Komponenten des Informatiksystems, als von Menschen zu schaffende Artefakte zu unterscheiden. Es sollte wesentliche Erkenntnis unterrichtlichen Modellierens sein, dass soziale Rollen von Personen nicht in ihrer Komplexität darstellbar sind, funktionale Handlungen innerhalb einer Organisation nicht notwendigerweise vollständig in ein Informatiksystem abgebildet werden müssen, Designpräferenzen auch von der verwendeten Entwicklungsumgebung abhängen. Die technische Realisierung eines Informatiksystems erweist sich so als eine Folge von Entscheidungen im Rahmen des Modellierungsprozesses, der die technische Seite eines Informatiksystems gegenüber seinem sozialen Kontext abgrenzt. Systemgestaltung sollte auch im Unterricht als Prozess erkannt werden, der keinesfalls eindeutige Lösungen zu vorher definierten Systemanforderungen liefert und somit den scheinbar zweckrationalen Charakter von Informatiksystemen infrage stellt.

Handlungsorientierter Informatikunterricht kann mit Modellierungstechniken, wie wir sie später beschreiben werden, Schülerinnen und Schülern anhand der Gestaltung von technischen Komponenten eines Informatiksystems, Modellierung als schrittweisen Abstraktions- Formalisierungs- und Entscheidungsprozeß erfahrbar machen. Damit werden nicht nur fundamentale Konzepte der Informatik sondern auch wichtige Einsichten zur Orientierung in einer technisierten Welt und damit ein wesentlicher Aspekt von Allgemeinbildung vermittelt (vgl. KLAFKI 1995).

### **Konstruierendes Lernen und Dekonstruktion im Informatikunterricht**

Seit geraumer Zeit werden gerade im Zusammenhang mit Lernen in vernetzten Lernumgebungen und mit interaktiven computerbasierten Medien Diskussionen um den Lernbegriff geführt, die für die Organisation schulischer Lernprozesse auch im Informatikunterricht von Bedeutung sein können. Hierbei findet vor allem eine kritische Auseinandersetzung mit konstruktivistischen Positionen statt. Nach konstruktivistischer Auffassung entstehen Weltbilder im Kopf, und die Wirklichkeit als zu erkennendes Objekt ist stets kognitiv konstruierte Wirklichkeit des erkennenden Individuums. Erkennen ist somit ein Prozeß zwischen Subjekt und Objekt, der vor allem durch die subjektiven Rekonstruktionsbedingungen des Individuums geprägt ist.

Es kann an dieser Stelle keine ausführliche Darstellung und kritische Würdigung konstruktivistischer Positionen des Lernens erfolgen. Für handlungsorientierte Lernprozesse mit Informatiksystemen im Unterricht verdient als vorläufiges Resümee der Diskussion festgehalten zu werden, dass Lernprozesse als Balanceakt zwischen Instruktion und Konstruktion organisiert werden sollten. Lernen erfordert zum einen Anleitung, Orientierung, Zielvorgaben und Hilfen - als Elemente traditioneller Unterrichtsorganisation. Zum anderen aber auch Motivation, Interesse und entdeckende Eigenaktivität des Lernenden - Elemente eines selbstorganisierten, lernergesteuerten Prozesses kognitiver Strukturbildung, die vor allem auch in individuellen Lernsituationen mit computerunterstützten Lernumgebungen zu erreichen sind.

Die Arbeit mit Modellen von Wirklichkeit im Informatikunterricht darf darüber hinaus nicht dazu führen, den Bezug zur realen Welt zu verlieren und Realbegegnungen weitgehend durch medial vermittelte zu ersetzen. Lernen bleibt zudem ein situativer Prozeß, bei dem Kenntnisse und Fertigkeiten in Situationen erworben werden sollten, die den späteren Anwendungszusammenhängen strukturell sehr ähnlich sind. Ein methodischer Wechsel zwischen Realbegegnung und medial vermittelter Abbildung von Wirklichkeit, die Variabilität von Lernorten und das Einbeziehen ausserschulischer Lernsituationen in den Unterricht sind Forderungen, die sich als Konsequenz aus der Diskussion um das Lernen in computergestützten Lernumgebungen ergeben. (vgl. MÜLLER 1996)

Für die Realisierung der Projektidee ergaben sich aus dieser Diskussion eine Reihe von unterrichtsmethodischen Konsequenzen:

Der Gegenstandsbereich der Modellierung sollte dem Erfahrungsbereich der Schülerinnen und Schüler entstammen (Schulkiosk). Es sollten Transfermöglichkeiten zu einem realen Informatiksystem (Warenwirtschaftssystem) bestehen, das auch unter Einbeziehung seines sozialen Umfeldes erkundet werden kann. Es sollte eine hinreichende Komplexität des zu modellierenden Systems bezogen auf schulisches

Niveau bestehen, um die Bedeutung objektorientierte Prinzipien der Informatik veranschaulichen zu können.

Der Einstieg in das Unterrichtsprojekt sollte über das Modellieren des Systems ‚Schulkiosk‘ erfolgen, um den Schülern/innen eine eigenständige Tätigkeit zu ermöglichen. Mittels der eingesetzten Modellierungsverfahren (CRC-Technik) entwerfen sie kooperativ im Team und selbständig konstruierend ein Modell des Schulkiosks, bis hin zur Gestaltung von möglichen Benutzungsoberflächen der Software. Sie werden in der anschließenden Unterrichtsphase anhand der vorliegenden didaktischen Software ‚Schulkiosk‘ vermutlich feststellen, dass andere Personen aufgrund divergierender Entscheidungen zu anderen Entwurfsergebnissen gelangen.

Ferner ist ein Bruch zwischen den modellierten abstrakten Objekten (CRC-Karten) und den später offengelegten Implementationen der Klassen in der Software ‚Schulkiosk‘ zu entdecken. Die semantische Lücke manifestiert auf drei Ebenen:

Gefundene abstrakte Objekte decken nicht die Ereignisse und Eigenschaften der realen Objektwelt ab.

Verschiedene Entwicklergruppen ‚entdecken‘ verschiedene abstrakte Objekte.

Abstrakte Objekte lassen sich nicht linear und vollständig auf Klassen der implementierten Software abbilden.

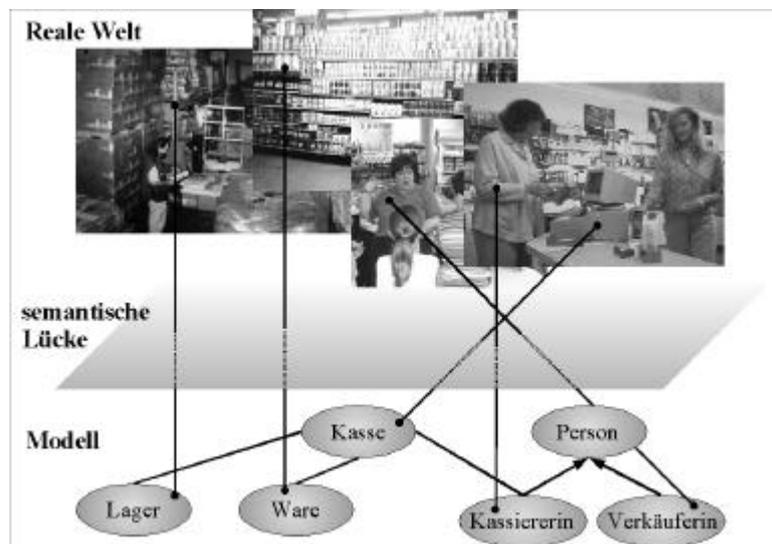


Abbildung 1: Reale Welt und abstraktes Modell (vgl. JAKOBSEN 1995)

Dieser Einstiegsphase schließt sich die Dekonstruktionsphase der Software ‚Schulkiosk‘ an. Dieser längere Abschnitt der Unterrichtseinheit besteht aus einem permanenten Wechsel zwischen Dekonstruktion der didaktischen Software und der konstruktiven Verbreiterung und Vertiefung der Kenntnisse beim Transfer in andere Problembereiche mittels geeigneter Übungsaufgaben und anderer Materialien. Gleichzeitig findet damit ein Wechsel zwischen dem analytischen Zugang zum Programmieren im Großen und dem konstruktiven produktorientierten Programmieren im Kleinen statt. Nach dem Erkunden der Systemfunktionalität werden sukzessive weitere objektorientierte Konzepte und deren Implementation in Java offengelegt, wobei die Schüler/innen anhand von geeigneten Erkundungsaufträgen die Konzepte weitgehend selbständig herausarbeiten sollen. Hilfestellung sollen bei diesem Prozeß geeignete Notationseditoren und Entwicklungsumgebungen zu Java leisten. Sie erweitern die Palette der nach diesem Konzept erforderlichen didaktischen Software. Durch Transfer des gewonnenen Wissens und der methodischen Kompetenzen in einen anderen Problemkontext können die Kenntnisse schrittweise vertieft und verbreitert werden.

Gegen Ende dieser Projektphase sollten die Schüler/innen nicht nur in der Lage sein, eigenständig fehlende Module der Software ‚Schulkiosk‘ zu ergänzen, sondern auch ein anderes Unterrichtsprojekt von der Modellierung bis zur Implementation konstruktiv zu entwickeln. Mit diesem Konzept kann keine systematische Vermittlung des Sprachumfangs einer objektorientierten Programmiersprache, etwa Java, erfolgen. Schüler/innen sollten aber in der Lage sein, Systemressourcen wie (online) Handbücher und System-

bibliotheken, zumindest mit Hilfestellung seitens des Lehrers, für ihre erweiterte Aufgabenstellung sinnvoll zu nutzen.

## Objektorientierte Modellierung am Beispiel Schulkiosk

Im folgenden wird der Prozeß des Modellierens für die Arbeitsweise im Projekt OMI beispielhaft erläutert.

Ausgangspunkt ist die Wirklichkeit, beim Beispiel Schulkiosk die Erfahrungswirklichkeit der Schülerinnen und Schüler, welche zunächst mit Hilfe der Begrifflichkeit und des Vokabulars dieses realen Anwendungskontextes in einem Analysemodell beschrieben werden soll. Wichtig ist in diesem Prozeß der Erstellung des Analysemodells, dass die Schüler möglicherweise mehr über den konkreten Wirklichkeitsausschnitt Schulkiosk lernen und auf jeden Fall das sog. ‚objektorientierte Denken‘ üben. Einer der Vordenker objektorientierter Softwaremethoden, Ivar Jacobsen (JACOBSEN 1995), bezeichnet dieses objektorientierte Denken als eine Methode, Systeme zu beschreiben in Form von einzelnen, geschlossenen Einheiten, die aber mit anderen Einheiten in Beziehung stehen. Diese Einheiten finden sich meist direkt in den Begriffen des Anwendungsbereichs wieder. Hier könnten, auf den Anwendungsfall des Schulkiosk bezogen, ein Brötchen, ein Verkäufer, eine Kasse typische Kandidaten für diese Einheiten oder in dem objektorientierten Sprachgebrauch gesprochen, Klassen oder Objekte sein. Auch die Zusammenhänge zwischen diesen Einheiten oder späteren Objekten lassen sich durch das Vokabular des Anwendungsbereiches erschließen. "Der Verkäufer bucht oder bont das Brötchen in der Kasse ein." Die einzelnen Einheiten stehen in einer bestimmten Art und Weise miteinander in Beziehung, die das Systemverhalten wie z. B. das Verkaufen eines Brötchens ermöglichen<sup>1</sup>. In einem objektorientierten Sprachgebrauch gedacht handelt es sich bei den Einheiten um Objekte, die in Form von Methoden ein bestimmtes Verhalten zeigen, und Aufgaben (wie die obige) die sich durch Versenden von Nachrichten an andere Objekte kennzeichnen. Es bleibt in einigen Fällen der Perspektive der analysierenden Gruppe überlassen, ob nun das Einbuchten, ‚bonnen‘ als eine Methode, d.h. Eigenschaft oder Tätigkeit der Kasse oder des Verkäufers gesehen wird. Dieses Problem läßt sich als eine für die Objektorientierung typische Designentscheidung sehen. Es stellt sich im Verlauf eines OO-Designprozesses immer wieder die Frage nach dem Zusammenspiel zwischen Objekten und der Zugehörigkeit bestimmter Eigenschaften zu verschiedenen Objekten. Die Schüler/innen werden sich in ihrer Auseinandersetzung mit einer OO-Analyse eines realen Systems wie z. B. dem Schulkiosk immer wieder fragen: "Welche realweltlichen Einheiten stellen Objekte des Modelles dar?", "Wie spielen diese Objekte und Einheiten zusammen?"

In ihrem Aufsatz ‚A Laboratory For Teaching Object-Oriented Thinking‘ zeigen die Autoren Kent Beck und Ward Cunningham, dass diese Fragen im Kernbereich des Lernens und Verstehens der objektorientierten Programmierung liegen: „teaching objects reduces to teaching the design of objects“ (BECK 1989). Mit möglichst einfachen Beschreibungen können die Lernenden nun Objekte konstruieren und so eine Analyse des Anwendungsbereichs vornehmen. Dabei kann und sollte an dieser Stelle darauf verzichtet werden, die Unterscheidung von Objekt und Klasse einzuführen. Es sei betont, dass es in diesem frühen Stadium der OO-Analyse um eine objektorientierte Beschreibung eines realen Schulkiosks – nicht um die Beschreibung einer Implementation geht. Unsere Erfahrungen zeigen, dass die Unterscheidung Objekt – Klasse oftmals schwer zu lernen und zu verinnerlichen ist. Daher sollte sie gründlich und nicht beiläufig, also im Zuge einer realen Auseinandersetzung eingeführt werden. Mit größerer Erfahrung mit Objekten dürfte dieser Aspekt zudem leichter zu begreifen sein.

---

<sup>1</sup> Diese Art der textuellen Beschreibungen werden in einem Ansatz tatsächlich direkt zur objektorientierten Modellierung des Systems benutzt: Die Hauptwörter sind Kandidaten für Objekte, die Verben beschreiben die Methoden der Objekte. Wir bevorzugen jedoch eine Technik, die u. E. besser für die Schulen geeignet ist, da das entstehende Modell leichter korrigiert, besser in Gruppen bearbeitet und nahtloser in formalere Modelle (Design, Implementation) übersetzbar ist.

## CRC Karten: der Ausgangspunkt eines Objektorientierten Denkens

Wir schlagen daher vor, die CRC-Karten von Cunningham/Beck (BECK 1989) zu benutzen. Diese beschreiben *Classes*, *responsibilities* und *collaborators*<sup>2</sup>. Auf das zu modellierende Beispiel bezogen, beschreibt eine Karte eine Einheit des Schulkiosks mit ihren *responsibilities* (das, was diese Einheit weiß und das, was sie kann - also das, wofür sie verantwortlich ist) und *collaborators* (andere CRC-Karten, die herangezogen werden, damit die Karte ihre Verantwortlichkeiten erfüllen kann). Wichtig ist in diesem Prozeß des Findes von Kerneinheiten, später Klassen, sich einer strikten Beschränkung auf drei bis fünf sogenannter Kerneinheiten, 'Kernclasses', zu unterwerfen. In einem zweiten Schritt werden diese Kernklassen räumlich angeordnet. Die so gewonnene räumliche Strukturierung sollte die Form der in *collaborator*-Listen angegebenen Verzahnung widerspiegeln. Konkret bedeutet dies, zentrale Klassen werden auch räumlich zentral angeordnet, und eher randständige, die nur für einzelne Verantwortlichkeiten herangezogen werden, werden an den Rande der Ordnung verschoben. Als weiteres wichtiges Grundmaxim der CRC-Methode ist zu beachten, dass insbesondere bei den aktiven, zentralen Klassen die Funktionalität bewußt zu beschränken ist. Hierbei hilft die durch das Format und Größe der Karten vorgegebene Platzbeschränkung - es darf auf keinen Fall eine Karte mit zu vielen Eigenschaften oder *collaborators* versehen werden, ist dies der Fall muß über ein Aufteilen oder Splitten der Karte in zwei Karten nachgedacht werden (vgl. AMBLER 1995, 1998). Denn für die OO-Idee ist es nicht wie für das imperative Paradigma üblich, eine zentrale Kontrollinstanz, sondern eine Struktur von abgegrenzten und in sich zusammenhängenden Einheiten (oder: *classes*) zu schaffen. Nur dieser Ansatz führt zu einer aus objektorientierter Sicht angemessenen Systemanalyse. Dabei wird sich ebenfalls herausstellen, dass die Beschreibung vom Zweck des Modells abhängt. Im Kontext Schulkiosk ist z. B. die Freundlichkeit des Verkäufers oder der Verkäuferin nicht von Relevanz und gehört daher nicht zu den Verantwortlichkeiten dieser Einheit in der CRC-Terminologie "class". Im Gegensatz dazu ist aber z. B. die Anschrift interessant und wird als *responsibility* vermerkt. Ob das so entstandene Analysemodell funktional ist, läßt sich meist durch schlichtes Ausprobieren testen.

## Das wäre wenn Spiel - use cases lebendig gestalten

Unsere Erfahrungen haben gezeigt, dass sich, insbesondere für die in diesem Artikel skizzierte Adaption der OO-Methoden in den schulischen Anwendungskontext, die Methode von Cunningham/Beck besonders eignet. Cunningham/Beck beschreiben ihre Methode zur Überprüfung gefundener CRC Karten als das ‚Was wäre, wenn..‘-Spiel. Lapidar gesprochen bedeutet dies für den Schulkiosk: Jeder Anwendungsfall wird konkret in einer Art Rollenspiel durchexerziert - "Was passiert und wer ist beteiligt, wenn ein Brötchen verkauft wird". Voraussetzung für dieses Spiel ist eine Sammlung einzelner Vorgänge, wie z. B. "ein Brötchen verkaufen." Diese sogenannten Anwendungsfälle (engl. use cases, vgl. JACOBSEN 1995) haben zudem weitere Funktionen. Zusätzlich kann diese einfache Rollenspiel zwischen Schülern, d.h. OO-modellierenden Personen durch einen kleinen Ball und geschickte Verteilung der CRC-Karten weiter veranschaulicht werden: Der konkrete Ablauf des *use-case*-Spieles gliedert sich wie folgt (vgl. AMBLER 1995):

- Die CRC-Karten werden so auf die Schülerinnen und Schüler verteilt, dass möglichst niemand ‚benachbarte‘ Karten, also solche, die als *collaborators* auf anderen Karten aufgeführt sind, besitzt. Nun sollte sich z. B. herausstellen, dass die einzelnen Anwendungsfälle einen definierten Startpunkt haben, falls nicht, fehlt eine Verantwortlichkeit auf Seite der Karten oder im Anwendungsfall.
- Ein Schüler mit der Karte, die den Startpunkt des durchzuspielenden Falls definiert, erhält den Ball als Markierung und versucht die Aufgabe zu lösen. Entweder gelingt dies mit Hilfe der aufgelisteten Verantwortlichkeiten direkt, oder der Ball wird zu demjenigen geworfen, der die unterstützende *collaborator*-Klasse in Händen hält. Auf diese Weise wird allen Beteiligten der Zusammenhang der einzelnen *classes* deutlicher, das Modell wird überprüft und verfeinert: Fehlende *responsibilities* oder *collaborators*, ja sogar ganze *classes* können schnell eingefügt werden. Gleichzeitig ist dieses Vorgehen eine

---

<sup>2</sup> In Ermangelung einer geeigneten Übersetzung bleiben wir bei den englischen Begriffen. Insbesondere class soll andeuten, dass nicht genau das Konzept Klasse gemeint ist. Das folgende Konzept bezieht die Anregungen Amblers und darüber z. T. auch das Konzept der use cases von Jacobsen mit ein und entspricht insofern nicht ganz dem ursprünglichen Vorschlag.

Unterrichtsmethode, um die prinzipiell bedingten internen Vorgänge beim Ablauf eines objektorientierten Programms zu veranschaulichen.

- Die benötigten Startpunkte für die Anwendungsfälle werden sich später als Teile der Benutzungsschnittstelle wiederfinden, die zum besseren Verständnis als Papier-Prototyp gezeichnet werden sollte. Diese Zeichnungen sollten für alle Teilnehmer der Gruppe sichtbar aufgehängt und ebenso wie CRC-Karten zur späteren Diskussion benutzt werden.

Im folgenden seien einige Hinweise zum Hintergrund dieser Sichtweise auf den Softwareentwicklungsprozeß gegeben, um schließlich zur Beschreibung der ‚Design-Modellierung‘ überzugehen: Anwendungsfälle sind in der Methode von Jacobsen als Teil einer Anforderungsermittlung gedacht. Aus diesem Grunde sind die Anforderungen der Benutzer (sprich: Auftraggeber bzw. Kunden der Softwareentwickler) Start- und Angelpunkt der Systementwicklung. Im ‚*use-case-driven approach*‘ steht das Anforderungsmodell vor der Analyse; ein Teil davon ist das sog. Anwendungsfall-Modell, das aus *use cases* besteht, die von *actors* aktiviert, d.h. angestoßen werden. *Actors* existieren nur außerhalb der zu entwickelnden Software. Allerdings sind sie nicht mit den Anwendern zu verwechseln: in einem konkreten Fall kann ein Benutzer die (funktionale) Rolle eines *actors* annehmen. Eine Anforderung eines externen Informatiksystems kann daher ebenfalls als *actor* dargestellt werden. Aus Sicht der zu entwickelnden Software sind diese Anforderungen nicht vorhersehbar, die Software sollte jedoch angemessen mit diesen Anforderungen umgehen können. Im Beispiel Schulkiosk besteht also kein Zusammenhang zwischen dem Verkäufer, der innerhalb des Systems für verkaufsstatische Zwecke enthalten ist und in gewisser Weise ein Modell eines realen Verkäufers darstellt und einem *actor*, den wir ebenfalls Verkäufer nennen, der den Anwendungsfall ‚Brötchen verkaufen‘ anstößt.

Zusammenfassend: Unser Analysemodell besteht prinzipiell aus den Teilen:

- Ein Modell der Anwendungsfälle in Form einer Auflistung. In der Standardobjektmodellierungssprache (UML) gibt es einpassenden Diagrammtyp (siehe z. B. FOWLER 1998), der aber u. E. nicht unbedingt in der Schule benötigt wird,
- ein Benutzungsschnittstellenmodell in Form grober Skizzen, und
- ein Einheitenmodell (vgl. JACOBSEN 1995) in Form von CRC-Karten, die die Struktur, den groben Zusammenhang und Eigenschaften der Einheiten des analysierten Systems darstellen.

Als zweites wichtiges Element eines OO-Modellierungsprozesses kann von den Schülern eine Skizze der späteren Benutzungsschnittstellen des zu spezifizierenden Informatiksystems angefertigt werden. Diese Skizzen der Benutzungsschnittstelle deuten in ihrer Form schon an, dass der Übergang von der Analyse zum Design her fließend zu charakterisieren ist – was von der Methode her beabsichtigt ist (vgl. JACOBSEN 1995). Zumindest abstrakt kann aber der genaue Unterscheidungspunkt zwischen diesen beiden Phasen in der Softwareentwicklung angegeben werden:



Abbildung 2: Übergang zwischen Analyse- und Designphase (vgl. JACOBSON 1995)

Definiert wird der Übergang zwischen Analyse und Design als der Punkt, wo Aspekte der gewählten Entwicklungsumgebung bedacht werden müssen. In unserem Falle wurde im Design auf Eigenschaften der Sprache Java und der Entwicklungsumgebung Visual Age von IBM eingegangen.

### **OO-Design, der Blick wird auf die spätere Implementierung gerichtet.**

Die Designphase ist durch eine weitere Verfeinerung des CRC-Karten-Modelles charakterisiert. Die *responsibilities* werden in Methoden und Attributen umformuliert, Datentypen werden festgelegt. Es wird unterschieden zwischen privaten Methoden zur Erfüllung der eigenen Verantwortlichkeit und öffentlichen, welche helfen die Verantwortlichkeiten der ‚benachbarten‘ ‚kollaborierenden‘ Klassen zu erfüllen. Der Zusammenhang zwischen den Klassen wird genauer dargestellt: Klassen können sich gegenseitig

,benutzen', im Fachterminus sind sie assoziiert oder sie enthalten sich, in der OO-Notation wird von Aggregation gesprochen. Der Aufbau der Klassen wird in die Java-Klassenhierarchie eingebunden, bestimmte Eigenschaften können Bibliotheken entnommen oder aus ihnen abgewandelt werden. Die gegenseitigen Methoden-Aufrufe, der oft so genannte Botschaften-, Nachrichten- oder Stimuli-Austausch wird durch das Spiel mit dem Ball sinnvoll repräsentiert, aber möglicherweise ist es zusätzlich sinnvoll, den genauen Ablauf in kleinen Zeichnungen festzuhalten: UML bietet dazu die Interaktionsdiagramme.

### Statt Implementation in Java - 'Dekonstruktion des Schulkiosks'

Ab einem gewissen Feinheitsgrad des Designs werden Kenntnisse der Sprache, der Entwicklungsumgebung und vor allem der Bibliotheken unverzichtbar.

Die Software Schulkiosk wurde in Java implementiert, da Entwicklungswerkzeuge und Bibliotheken frei verfügbar sind. Neben einer fortschreitenden Verbreitung der Sprache in der universitären und industriellen Anwendung vereinigt Java die wichtigsten objektorientierten Konzepte bei einer gleichzeitigen Reduktion auf wesentliche Sprachelemente. Doch so elegant und minimalistisch der Java Sprachstandard angelegt ist, so vielfältig sind die zur Verfügung stehenden Klassenbibliotheken. Bibliotheken für 2D und 3D Visualisierung (Java2D, Java3D API), Kommunikation, Oberflächengestaltung usw. bieten vielfältigste Möglichkeiten. Für Lehrer und Schüler, aber auch fortgeschrittenen Java Programmierer eröffnet die Vielzahl der Bibliotheken die Chance, dem eigenen Design entsprechende Klassen zu extrahieren oder gegebenenfalls durch Überschreiben vordefinierter Methoden an das Design anzupassen.

In der Implementationsphase sind solche Kenntnissen und Fähigkeiten gefordert, die die Schülerinnen und Schüler in dieser Unterrichtsphase erst noch erwerben müssen. Daher wird die Dekonstruktion von Informatiksystemen als Unterrichtsmethode vorgeschlagen, um so einen Zugang zu objektorientierten Sichtweisen im Informatikunterricht zu schaffen. Anstelle der Neu-Implementation der eigenen Modellierung dekonstruieren die Lernenden eine Implementation, die auf den Analyse-, Design- und Implementationsentscheidungen eines anderen 'Entwicklerteams' beruht. Mit ihrem bereits vorhandenen Wissen und Fähigkeiten, objektorientiert zu modellieren, fällt es leichter, einzelne Entwurfsentscheidungen in ihrem Zusammenhang zum gesamten System zu sehen und später Elemente des 'Programmieren im Kleinen' selbständig umzusetzen.

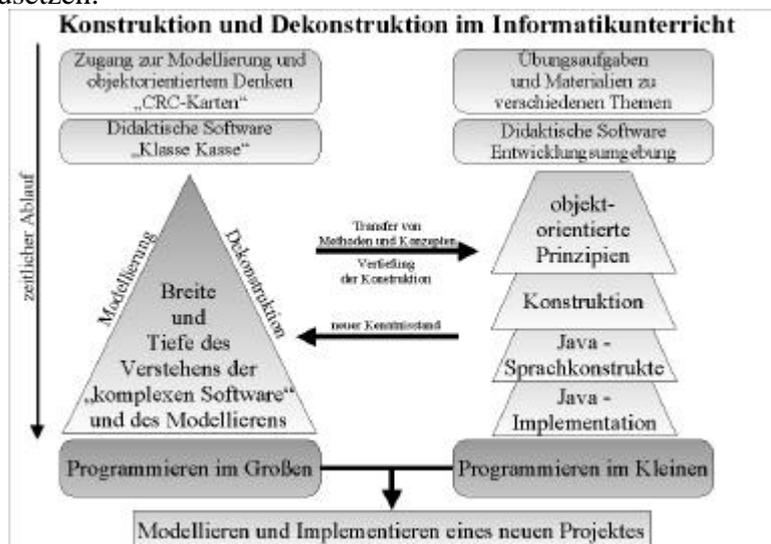


Abbildung 3: Konstruktion und Dekonstruktion als didaktisches Konzept

Ergänzend zum eher isolierten Lernens einzelner Sprachkonstrukte lernen sie einige konzeptionelle Möglichkeiten der Objektorientierung und deren Realisierung in Java mitsamt der benutzten Bibliotheken eingebettet in einen Anwendungszusammenhang, den Schulkiosk, kennen. Eine geeignete Implementation des Schulkiosk wurde analog zu dem in diesem Abschnitt beschriebenen Vorgehen (Analyse mit CRC, Design der Klassen) mit der Entwicklungsumgebung Visual Age von Studierenden an der Universität Paderborn entwickelt.

## Ausblick auf die weiteren Projektphasen und Forschungsfragen

Für die Dekonstruktion ist es außerordentlich hilfreich, dass sich Begriffe und Strukturen auch aus dem ersten Analyseschritt in dem Implementationsmodell - wenn auch verändert - wiederfinden lassen. Diese Eigenschaft der objektorientierten Softwareentwicklung läßt es erfolgversprechend erscheinen, durch die Analyse von Quelltexten und dem Verhalten einer Software während der Ausführung Dekonstruktionsprozesse auszulösen, die mehr freilegen als einzelne Sprachkonstrukte. Dazu ist jedoch eine Abstimmung der Unterrichtsphasen Modellierung / Konstruktion und Dekonstruktion erforderlich, die durch geeignete Materialien und Unterrichtsmethoden erreicht werden soll.

### Die Projektphasen:

1. Entwicklung der Unterrichtssoftware Schulkiosk.
2. Material für das Lernen von Java-Sprachkonstrukten und OO-Konzepten als Schulbuch entwickeln.
3. Geeignete Entwicklungsumgebung für Schulen testen und ggf. modifizieren.
4. Weitere Werkzeuge zur Dekonstruktion testen und ggf. modifizieren.
5. Das Konzept unterrichtspraktisch erproben und evaluieren.

### Integration und Abstimmung der OO-Analyse-, Design- und Implementierungswerkzeuge

Als Perspektive einer integrativen Anwendung der objektorientierten Methoden in der Schule ist eine Eingliederung der schon beschriebenen OO-Notationen (UML, CRC-Karten) in die Entwicklungsumgebungen ein zentraler Punkt. Auf diese Weise ließen sich schon im Desingprozeß erschlossene Objektbeziehungen und Klassendiagramme einfacher und ohne Wechsel des verwendeten Mediums mit Darstellungstechniken verschmelzen, z. B. den Ausgaben des Debuggers oder Classbrowsers. In einer solchen Umgebung stellen sich die aufeinanderfolgenden Modellierungsphasen (Analyse, Design, Implementation) als textuelle und visuelle Kommunikationen dar, die einen zunehmenden Grad an Formalisierung und formalisierter Semantik erfahren. Diese Konzeption befindet sich in Einklang mit fachwissenschaftlichen Sichtweisen, die Softwareentwicklung als kooperativen und gegenseitigen Lernprozeß begreifen (z. B. ZÜLINGHOVEN 1998, JACOBSEN 1995, FLOYD 1997). Zum einen gelingt so ein schülerzentrierter Zugang zur Informatik, indem der Unterricht auf Alltagserfahrungen der Schüler/innen aufbauen kann. Zum anderen wird eine einbettende Sichtweise auf informationstechnische Produkte angeregt, die deren Herstellungs- und Verwendungskontext nicht außer acht läßt.

Dekonstruktionsprozesse untersuchter Informatiksysteme finden ebenfalls in dieser Perspektive statt. Die Behandlung von Sprachkonstrukten und objektorientierten Konzepten kann so - zum Teil möglicherweise auch eigenständig von den Schülerinnen und Schülern - in einen sinnvollen Kontext gestellt werden. Die Breite und Tiefe dieser Dekonstruktionen eines Informatiksystems ist dabei in einem recht hohen Maße skalierbar.

### Werkzeuge zur Dekonstruktion

Durch die werkzeugunterstützte Generierung von Klassen und Objektdiagrammen zur Laufzeit ist die Dekonstruktion direkt an die Notationen der Modellierung anknüpfbar. Eine Zusammenführung des gesamten OO-Entwicklungsprozesses, deren Methoden und Werkzeuge zu einem universellen Lernkonzept und didaktischen Ansatz wäre dann eine umsetzbare Idee und Perspektive im Sinne einer systemorientierten Didaktik der Informatik. Es wäre wünschenswert, wie im Spiel mit dem Ball und den CRC-Karten, während der Analysephase auch den 'virtuellen Ball' im Ablaufen der Implementation zu verfolgen. Es gibt dazu einen ersten Ansatz in Form eines sogenannten 'dynamic object browsing system', das die interne Objektstruktur der Java-Virtuellen Maschine grafisch in einer Art Objektdiagramm abbildet. Eine weitere Anwendung des Werkzeugs liegt im Testen bzw. im Bereich der Fehlersuche. Die Liste der Anwendungsfälle stellt die Grundlage für das Testen bzw. für die Dekonstruktion bereit. Das Verfolgen eines Anwendungsfalles erfordert das schrittweise, animierte Anzeigen der Veränderungen in der Objektstruktur - daran arbeitet ein Projekt der Arbeitsgruppe Softwaretechnik an der Universität Paderborn.

In der zweiten Projektphase wird ein Schulbuch mit Materialien entwickelt, mit denen anhand der Dekonstruktion des Schulkiosks in Sprachkonstrukte, Konzepte und in Bibliotheken der Sprache Java sowie in objektorientierte Sichtweisen eingeführt wird. Neben der Analyse wird durch eine Reihe von Aufgaben die Eigentätigkeit der Schülerinnen und Schüler angeregt.

In einem weiteren Schritt ist an die Erstellung von multimedialem Material gedacht, mit dessen Hilfe weitere Aspekte der Systemeinkbettung einer Warenwirtschafts-Software in unterrichtlichem Zusammenhang erschließbar wären.

### **Werkzeuge für die Implementation**

Ähnlich wie Werkzeuge zur Dekonstruktion - die z. T. auch zur Implementation verwendet werden können - sollte auch die Entwicklungsumgebung unseren Ansatz durch Integration der Notationen und Techniken aus der Design- und Analysephase unterstützen.

Die Werkzeuge sollten folgenden Anforderungen genügen:

- Einfach und ohne große Einarbeitungszeit grafische Oberflächen erstellen können.
- Das Analysemodell - bestehend aus einer UML-artigen Notation in Form von Klassendiagrammen, die Vererbungs- Aggregations- und Assoziationsbeziehungen enthalten - möglichst direkt in die Entwicklungsumgebung übernehmen können.
- Während der Implementationsphase automatisiert stets ein aktuelles Klassendiagramm aus dem Quelltext erzeugen können, das das Design und Abweichungen zum Ergebnis der Designphase - aber auch den Zusammenhang der beiden Phasen erkennen läßt.
- Durch geeignete Zugriffs- und Suchmechanismen sowie erklärende Hilfen das Benutzen der Bibliotheken vereinfachen.
- In Bezug auf Preis und erforderlichen Systemressourcen in Schulen einsetzbar sein.

Insgesamt soll so die Genese des Systems bis zu seinen Ursprüngen verfolgbar sein. Java-Entwicklungsumgebungen zeigen in bezug auf obige Forderungen verschiedene sehr interessante und überzeugende Eigenschaften. Ein Werkzeug, das alle diese Forderungen erfüllt, ist uns noch nicht bekannt. Es werden daher weitere Werkzeuge untersucht und später ggf. im Hinblick auf den Einsatz im Unterricht modifiziert.

### **Empirische Evaluation anhand qualitativer Fallstudien**

Ob das vorgestellte didaktische Konzept die mit ihm verknüpften Erwartungen erfüllt, soll in schulischen Feldversuchen mit einer qualitativen empirischen Studie überprüft werden. Dazu wird ein Forschungsdesign entwickelt, das auch den Vergleich mit Kontrollgruppen einschließt, die mit traditionellen Unterrichtsmethoden arbeiten.

Mit dem Projekt ist die Erwartung verbunden, dass Schüler/innen Einsichten in wesentliche Konzepte der objektorientierten Modellierung und der anschließenden Implementation in Java gewinnen. Darüberhinaus sollten sie lernen, in systemischen Zusammenhängen zu denken, die Entwicklung von Informatiksystemen als kooperativen Gestaltungsprozeß zu begreifen und deren Bewertung in Abhängigkeit vom umgebenden sozialen Kontext vorzunehmen. Im Rahmen der empirischen Überprüfung dieser Thesen soll auch geklärt werden, ob ein derartig anwendungsbezogener didaktischer Zugang zu Informatiksystemen geschlechtsspezifische Differenzen im Unterrichtsverhalten und bei den Ergebnissen des Unterrichts aufweist.

## **Literatur**

Ambler, S.W.: Building Object Applications That Work, New York , SIGS Books, NY 1998

Ambler, S.W.: The object primer. The application developer's guide to object-orientation. SIGS Books, 1995

Baumann, R.: Didaktik der Informatik, Stuttgart; Klett, <sup>2</sup>1996

Beck, K. und Cunningham, W.: A Laboratory For Teaching Object-Oriented Thinking. in: OOPSLA'89 Conference Proceedings 1989, New Orleans, Louisiana. SIGPLAN Notices, Volume 24, Number 10, October 1989.

Bishop, J.: Java Gently, Programming Principles Explained, Harlow (GB) u.a., AddisonWesley <sup>2</sup>1998

- Booch, Grady: Objektorientierte Analyse und Design. Bonn u. a., Addison-Wesley, 1996
- Flanagan, D.: Java in a nutshell, Köln, O'Reilly, <sup>2</sup>1998
- Floyd, C.: Einführung in die Softwaretechnik. Skriptum zur gleichnamigen Vorlesung, Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, Hamburg 1997
- Fowler, M.; Scott, K.: UML - konzentriert. Bonn u. a., AddisonWesley, 1998
- Jacobson, I., u.a.: Object-Oriented Software Engineering. Wokingham u. a. ,Addison-Wesley, 1995
- Klafki, W. : „Schlüsselprobleme“ als thematische Dimension einer zukunftsbezogenen „Allgemeinbildung“ – Zwölf Thesen, in: Die Deutsche Schule, 3. Beiheft 1995, Schlüsselprobleme im Unterricht, Thematische Dimensionen einer zukunftsorientierten Allgemeinbildung, S. 9ff
- Lehmann, E.: Komplexe Systeme, Eine fundamentale Idee im Informatikunterricht, in Login 15. Jg. 1995, H.1, S. 29ff
- Meyer, B.: Objektorientierte Softwareentwicklung. München, Hanser, 1990
- Müller, K. (Hg.): Konstruktivismus, Lehren – Lernen – Ästhetische Prozesse, Berlin, Luchterhand 1996
- Ropohl, G.: Technologische Aufklärung : Beiträge zur Technikphilosophie, Frankfurt/M., Suhrkamp, 1991
- Schelhowe, H.: Das Medium aus der Maschine : zur Metarmorphose des Computers , Frankfurt/M. [u.a.] Campus, 1997
- Schulz – Zander, R. u. a.: GI –Empfehlungen für das Fach Informatik in der Sekundarstufe II allgemeinbildender Schulen, Beilage zu Login 13 (1993) H. 3
- Schwill, A.: Fundamentale Ideen der Informatik, in: Zentralblatt für Didaktik der Mathematik 93 /1 S. 30ff
- Spolwig, S.: Objektorientierung im Informatikunterricht, Bonn, Dümmler, 1997
- Wegner, P.: Why Interaction Is More Powerful Than Algorithms, in: CACM, May 1997, Vol. 40, No. 5, p. 81ff
- Züllighoven, H.: Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz, Heidelberg, dpunkt 1998