Objektorientierung?!

Stellenwert und Konkretisierung im Informatikunterricht
von Marco Thomas

überarbeitet erschienen in LOGIN Heft 128/129

Einleitung

Mit der imperativen Programmiersprache SIMULA begann 1967 der Siegeszug der objektorientierten Programmierung in der Softwareentwicklung. Historisch gesehen ist SIMULA ein Nachfolger der Programmiersprache ALGOL 60. Diese wiederum gilt als Vorgänger vieler imperativer Sprachen, da erstmals Prozeduren, Blöcke und höhere Kontrollstrukturen zur Formulierung von Algorithmen zur Verfügung standen (Duden 2001). Mit SIMULA wurden Klassen und Objekte als Konzepte in die Informatik eingeführt. Klassen konnten hierarchisiert werden, Eigenschaften von Klasse zu Klasse vererbt werden und Objekte konnten selbständig miteinander agieren und kommunizieren. Das Selbstbild der Informatik wurde jedoch noch bis in die 80er Jahre vor allem durch strukturiert-prozedurales, funktionsorientiertes und logikorientiertes Denken bestimmt. Insbesondere die Entwicklung moderner Benutzeroberflächen bei Produkten von Xerox, Apple und letztlich auch Microsoft sowie die Versuche der KI, kognitive Leistungen auf einem Computer zu simulieren, haben Vorteile einer objektorientierten Sichtweise deutlich werden lassen.

Neue Entwicklungen in der Informatik wurden von engagierten Lehrern zumeist umgehend in den Informatikunterricht an den allgemeinbildenden Schulen integriert. Mit dem Einzug der Objektorientierung in die Softwareentwicklung wurden daher auch Programmiersprachen für den Informatikunterricht favorisiert, die objektorientiertes Entwickeln unterstützen. Erst anschließend folgten Ansätze zur Didaktik und Methodik, die eine Legitimation im Sinne des allgemeinbildenden und wissenschaftspropädeutischen Auftrags von Schule versuchten. Nachvollziehbare Forderungen aus der Fachwissenschaft wie „Objektorientierung von Anfang an“ wurden von einigen Lehrern und Fachdidaktikern auf die Schulinformatik mit der Begründung übertragen, objektorientierte Methoden entsprächen kognitionspsychologisch der natürlichen Herangehensweise von Menschen an Probleme und objektorientiertes Denken könne für Probleme im späteren Leben hilfreich sein. Ein nicht-spezifischer Transfer von Fertigkeiten aus dem Kontext der Softwareentwicklung auf andere Problembereiche konnte jedoch bisher nicht nachgewiesen werden (vgl. a. Stern 1996).

Zahlreiche Inhalte und Fragestellungen, die die Wissenschaft Informatik auszeichnen, mussten im Informatikunterricht dem neuen Inhalt Objektorientierung weichen, wurden ein Anhängsel oder gerieten in Vergessenheit (z.B. Software Life Cycle, algorithmische Entwurfsmethoden, Geschichte der Informationsverarbeitung). Die neuen Konzepte führten zu neuen Programmiersprachen und schürten den Programmiersprachenstreit zwischen den Lehrenden, insbesondere an den allgemeinbildenden Schulen. Hinzu kamen Entwicklungsumgebungen (IDEs) und Case-Tools, die zu weiteren Diskussionen führten, jedoch in der Unterrichtspraxis nur selten verwendet werden. Die geforderte Verlagerung des Schwerpunkts von der Implementierung zur Modellierung dürfte daher in der Schulpraxis selten stattfinden. Es stellt sich die Frage, welchen Stellenwert die Objektorientierung im Informatikunterricht haben sollte und wie die entsprechenden Konzepte im Informatikunterricht vermittelt werden können.

Objektorientierung

Objektorientierung ist eine Denkweise, die weder neu noch auf die Informatik beschränkt ist. Sie ähnelt kognitionspsychologisch gesehen der menschlichen Denkweise (Schwill 1993) und unterstützt eine natürliche Modellierung der „realen Welt“, in der „Dinge“ mit bestimmten Eigenschaften mehrfach genutzt werden ohne die „innere Struktur dieser Dinge“ zu kennen. Beispielsweise muss ein Blumenliebhaber keine Details zur Osmose bei Pflanzen kennen. Der Begriff „Objektorientierung“ ist eng mit der Wissenschaft Informatik verknüpft. Elemente einer objektorientierten Denkweise wurden zunächst durch spezielle Konzepte einiger Programmiersprachen eingeführt. Diese Konzepte bilden die Grundlage der objektorientierten Softwareentwicklung: Objekte, Klassen, Methoden, Objektinteraktion sowie Vererbung, Polymorphismus und Kapselung. Ziel jeder Softwareentwicklung ist die Abbildung von Komponenten eines Anwendungs- oder Problembereichs in ein computerfähiges Modell. Diese vollzieht sich über verschiedene Modellierungsschritte, die in Vorgehensmodellen beschrieben sind. Modellierungsnotationen und Modellierungswerkzeuge unterstützen den Prozess. Man spricht hier vom „Programmieren im Großen“, also der Entwicklung von Software zu relativ komplexen Problemen. Im Gegensatz dazu wird der Entwurf von speziellen Lösungsalgorithmen auch als „Programmieren im Kleinen“ bezeichnet (bis ca. 500 Programmzeilen). Der zielorientierte Entwurf beim „Programmieren im Kleinen“ orientiert sich eher an einzelnen Funktionalitäten eines Programms, während sich

der objektorientierte Entwurf beim „Programmieren im Großen“ (bis zu 10 Mio. Programmzeilen) an den Bestandteilen eines Problembereichs ausrichtet.

Objektorientierte Vorgehensmodelle gliedern sich vergleichbar mit denen der strukturierten, traditionellen Programmierung in Phasen der Analyse, des Entwurfs, der Implementierung und des Testen. Allerdings sind die Grenzen zwischen den einzelnen Phasen wesentlich fließender als ursprünglich bei der strukturierten Programmierung. Da beim objektorientiertem Vorgehen zudem die (gedachte) Realität mit ihren Elementen direkt modelliert wird und nicht unmittelbar funktionsorientiert in Prozeduren und Module zerlegt werden muss, lässt sich ein Strukturbruch zwischen Analyse und Entwurfsphase vermeiden.

Größere Programme oder Programmteile sind, auch bei guter Kommentierung und Dokumentation, nur schwer verständlich. Modellierungsnotationen unterstützen die Kommunikation zwischen den Entwicklern in einem Projekt bereits in der Entwurfsphase. Derzeit versucht sich die Unified Modeling Language (UML) als eine vereinheitlichte Modellierungsnotation durchzusetzen, die auch ein Bestandteil industrieller objektorientierter Entwicklungsmethoden ist (z.B. im Vorgehensmodell des Rational Unified Process). Die UML stellt zahlreiche Diagrammtypen zur Verfügung, die die Struktur eines Problems, das Verhalten der Systemkomponenten oder implementierungspezifische Aspekte darstellen. Struktogramme oder Programmablaufpläne der traditionellen Softwareentwicklung sind zur Modellierung von „Programmen im Großen“ eher ungeeignet.

Für die industrielle Entwicklung von Software existieren eine Reihe von Werkzeugen, von einfachen Entwicklungsumgebungen (Texteditor, GUI Builder) zu umfangreichen CASE-Tools. Ein Vergleich aktueller Werkzeuge zur objektorientierten Softwareentwicklung für die Schule befindet sich auf der Homepage des life3-Projektes (life.uni-paderborn.de). Eine Übersicht zu professionellen UML-CASE-Tools erhält man auf der Website von M. Jeckle (www.jeckle.de/umltools.htm).

Objektorientierung ist übrigens nicht auf imperative Sprachen beschränkt. Es existieren erste Ansätze für Sprachen, die dem logischen oder funktionalen Paradigma folgen (Brichau/Mens, Pizza-Compiler).

Stellenwert in der Informatik

Die Objektorientierung ist aktuell ein sehr populärer Ansatz in der Softwareentwicklung. Allerdings existieren auch sehr kritische Einschätzungen namhafter Wissenschaftler der Informatik, die zeigen, dass die Objektorientierung, so wie wir sie heute verstehen, möglicherweise nicht „der Weisheit letzter Schluss“ ist ‚da sie zahlreiche Forschungsergebnisse noch nicht berücksichtigt (Broy/Siedersleben 2002, Jähnichen/Hermann 2002). Die objektorientierte Denkweise wurde in die Informatik eingeführt, um große Systeme ingenieurmäßiger entwickeln zu können. Softwareengineering ist nach Balzert (1996)

> *„ein Fachgebiet der Informatik, das sich mit der zielorientierten Bereitstellung und systematischen Verwendung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwareprodukten befaßt“.*

Schwerpunkte bilden hier Ingenieurprinzipien, wie sie zum Teil auch in den klassischen Ingenieurwissenschaften wie dem Maschinenbau, der Elektrotechnik oder dem Bauwesen existieren. Grundlage des Softwareengineering sind neben der Informatik, die Betriebswirtschaft, die Psychologie u.a. (Broy/Rombach 2002). Mittlerweile hat der Bereich des Softwareengineering eine große Eigenständigkeit gewonnen, die sich beispielsweise in eigenen Studiengängen äußert, die gegenüber der grundlagenorientierten Wissenschaft Informatik stärker die praktische Anwendung hinsichtlich der Entwicklung umfangreicher Softwaresysteme in den Vordergrund stellt (z.B. am Hasso Plattner Institut an der Universität Potsdam). Die Informatik liefert ähnlich wie die Physik für den Maschinenbau „formale Modellierungstechniken“ oder „Architekturmodelle“. Zur Hochschullehre schreiben Jähnichen und Herrmann (2002*)*:

> *„Um Studierende für künftige Entwicklungen zu wappnen, muss die Informatikausbildung darauf hinweisen, dass Objektorientierung ein Ding mit vielen Gesichtern ist und dass bestimmte Konflikte schwierige Abwägungen erfordern. ... Nur Informatiker, die beide Seiten kennen, können sich ein eigenes Urteil bilden, in konkreten Projekten die geeignetsten Techniken auszuwählen und vielleicht an einer Verbesserung unserer Techniken und Methoden forschen. Aber vielleicht spricht dann schon keiner mehr von der Objektorientierung ...“.*

Diese Aussagen lassen sich auf einen allgemeinbildenden und wissenschaftspropädeutischen Informatikunterricht übertragen und gelten auch für andere Paradigma in der Informatik.

Informatische Modellbildung im Informatikunterricht

Der Informatikunterricht hat sich mehrfach in seiner inhaltlichen und methodischen Ausgestaltung gravierend gewandelt. Diese Änderungen waren zwar notwendig, haben jedoch aufgrund ihrer kurzen Abfolge eine

Etablierung des Faches im Schulkanon nicht gerade gefördert. Ursachen dürften zum einen der bis heute fehlende Konsens zum Verständnis der sehr dynamischen Wissenschaft Informatik sein und zum anderen eine unzureichende Konzentration auf langfristig in der Wissenschaft etablierte Inhalte für die Praxis des Informatikunterrichts sein. Für den Informatikunterricht an deutschen Schulen wurde zunächst der Schwerpunkt auf das Verständnis der Hardware gelegt. Rasch traten Algorithmen und ihre Implementierung in den Vordergrund. Mit diesen Inhalten der Informatik gelang dem Schulfach in den 70iger Jahren auch der Einzug in die gymnasiale Oberstufe. Als Reaktion auf Forderungen nach mehr lebensweltlicher Orientierung von Unterricht entstanden in den 80igern sogenannte anwendungsorientierte Ansätze, die ein Schulfach Informatik mit dem gesellschaftlichen Stellenwert informatischer Kenntnisse begründeten (Riedel 1979, Körber et al. 1981). Mit den anwendungsorientierten Ansätzen rückten in der Fachdidaktik Informatik die Begriffe des "Modells" und der "Modellbildung" in den Vordergrund, wobei je nach Autor unterschiedliche Begriffsverständnisse zu finden sind. Die für den Informatikunterricht postulierte Leitlinie "Informatische Modellierung" der GI-Empfehlung für ein "Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen" (2000) stellt die Vorbildfunktion informatischer Modelle für das zu erstellende Informatiksystem in den Vordergrund.

> *'Im Informatikunterricht bedeutet "Modellierung" im wesentlichen die Abgrenzung eines für den jeweiligen Zweck relevanten Ausschnittes der Erfahrungswelt, die Herausarbeitung seiner wichtigen Merkmale unter Vernachlässigung der unwichtigen sowie seine Beschreibung und Strukturierung mit Hilfe spezieller Techniken aus der Informatik. Informatische Modelle spielen bei der Konstruktion und Analyse von Informatiksystemen die Rolle von Bauplänen. [..] Die bei der Analyse von Informatiksystemen kennen gelernten Modellierungstechniken ermöglichen den Schülern dabei auch ganz allgemein die Strukturierung umfangreicher Datenbestände und die Orientierung in komplexen Informationsräumen.'*

Sicherlich arbeiten auch andere Wissenschaften und Unterrichtsfächer mit Modellen. Dies ergibt sich aus der allgegenwärtigen Verankerung von Modellen in unserer Kultur. Zur Präzisierung des informatischen Modellbegriffs wurde das Begriffsverständnis in der Wissenschaft Informatik analysiert (Thomas 2002). Es zeigte sich, dass informatische Modelle, unter anderem aufgrund ihrer Vielfalt an Modelltypen, stellvertretend das "Modellieren von Modellen" im Allgemeinen repräsentieren können. Mit Modellieren von Modellen wird ausgedrückt, dass Modelle nicht nur verwendet und erstellt werden, sondern meist in Modellketten ein Modell als Nachfolger eines anderen Modells konstruiert wird. Die produktorientierte Auseinandersetzung mit der Automatisierung von informationsverarbeitenden Prozessen bietet einen hervorragenden Kontext für den Schulunterricht, um eine Enkulturation des Modellierens von Modellen zu bewirken. Hierbei können mentale Modelle, die grundlegend für das Verständnis von Informatiksystemen sind, durch Exploration aufgebaut werden. Akzeptiert man die Erkenntnis, dass die Informatische Modellbildung ein durchgängiges Prinzip in der Informatik darstellt, sollte sie sich über alle Jahrgangsstufen spiralcurricular erstrecken. In diesem Sinne legitimiert sich eine Leitlinie „Informatische Modellbildung“ für einen allgemeinbildenden Informatikunterricht. Allerdings umfasst Informatische Modellbildung mehr als nur Objektorientierte Modellierung.

Objektorientierung im Informatikunterricht

Es ist weniger eine Frage, ob Objektorientierung im Informatikunterricht einen Platz hat, sondern mehr, wann, wie und in welchem Umfang Objektorientierung im Unterricht zu integrieren ist.

> *„Warum soll man nicht von Anfang an bessere Konzepte der Software-Entwicklung nutzen, statt den Schülern einen Programmierstil beizubringen, den man ihnen später bei größeren Projekten wieder austreiben muß!“*

schrieb Spolwig 1995 zum Einsatz von Objektorientierung im Anfangsunterricht. Auch Crutzen und Hein (1995) plädieren für eine Objektorientierung von Anfang an, da man erst durch Erfahrung lernt, Objekte zu sehen. Andererseits weist Füller (1999) auf seine schulpraktischen Erfahrungen hin, dass die Objektorientierung analytisch-planendes, an Strukturen orientiertes Vorgehen von den Schülern verlangt, das bei den Schülern zunächst ausgebildet werden muss und viele Schüler wohl auch überfordert. Zudem zahle sich häufig die objektorientierte Vorgehensweise erst so spät aus, dass die Schüler lange Zeit nur „auf Vorrat lernen“, unter Verlust der durch Spielen und Experimentieren begründeten Motivation bei den Schülern.
Im Chemieunterricht werden in begründeter Reihenfolge verschiedene Atommodelle den Schülern vermittelt. Hier hat man sich entschieden, nicht mit dem schwer verständlichen Orbitalmodell einzusteigen, sondern den Schülern ein dem jeweiligen Problemniveau angemessenes Atommodell an die Hand zu geben. Dabei dürfen die vorhergehenden Modelle nicht falsch in dem Sinne sein, dass sie die Einführung eines späteren Modells erschweren oder gar verhindern. Auf diese Weise erfahren die Schüler die historische Entwicklung der Modellvorstellungen in der Chemie und setzen sich mit den Fragestellungen der Wissenschaft intensiver auseinander. In ähnlicher Weise können verschiedene Modelle, Methoden und Prinzipien der Informatik historisch-genetisch in einem Informatikunterricht behandelt werden, um die Ziele und Forschungsfragen der Wissenschaft Informatik herauszustellen. Dabei muss ebenfalls sichergestellt werden, dass die curricular später liegende Einführung eines Inhalts nicht durch zuvor behandelte Inhalte erschwert wird.

Baumann (1996, 281) schreibt, dass der Übergang von der imperativen zur objektorientierten Programmierung zunächst nur eine Änderung der Terminologie zu beinhalten scheint. Wenn dem so ist, dann kann im Unterricht auch von Anfang an die Terminologie der Objektorientierung verwendet werden, um ein für Schüler unnötiges Umdefinieren zu vermeiden (vgl. auch die Ansätze zur Integration der objektorientierten Denkweise in der Sekundarstufe I: Hubwieser 2000 u.a.). Viele Konzepte der strukturierten Programmierung finden auch innerhalb von Klassen und Objekten ihre Anwendung. Ob ein Sortieralgorithmus in einer Methode einer Klasse implementiert wird oder in einer imperativen Sprache „prozedural" angelegt wird, hat keine grundlegenden Auswirkungen auf das Prinzip des Algorithmus. Es ist allerdings zu beachten, dass Standardalgorithmen häufig in – den Schülern gegebenenfalls bekannten – Bibliotheks-Funktionen zur Verfügung gestellt werden, so dass die Auseinandersetzung aus Schülersicht müßig erscheinen kann. Trotz dieser Argumente für „Objektorientierung von Anfang an" favorisiere ich, unter anderem aus historisch-genetischen Gründen, mindestens einen Exkurs zum zielorientierten Entwurf der Objektorientierung vorweg zu schicken. Dabei lassen sich beispielsweise Nachteile des Wasserfallmodells wie die sequentielle Abarbeitung der Phasen erkennen und diskutieren.

Anfangsunterricht – mit BlueJ

Ausgehend von der Überzeugung, dass die Objektorientierung ein wichtiges Konzept in der Informatik ist (aber eben nur eines von vielen), soll für einen Grundkurs Informatik ein Anfangsunterricht aufgestellt werden, der verschiedene Aspekte der Leitlinie „Informatische Modellbildung" berücksichtigt und damit einen Überblick zu Fragestellungen der Informatik gibt. Hierzu zählen beispielsweise der Software Life Cycle, Zustandsgraphen, der Algorithmusbegriff, die historische Entwicklung von Rechenhilfsmitteln, die Turingmaschine, eine Auseinandersetzung mit den Zielen der Künstlichen Intelligenz und Konzepte der objektorientierten Softwareentwicklung (Tab.1).
Hauptziel des Abschnitts zur objektorientierten Softwareentwicklung ist die Einführung des objektorientierten Denkens im Zusammenhang mit der Entwicklung von Software. Die fundamentalen Konzepte Objekte, Klassen, Attribute, Methoden und Interaktion sollen die Schüler kennen lernen und anwenden können. Vererbung und Polymorphismus werden zunächst nur ansatzweise und nicht tiefergehend thematisiert, da sie nach Goos (1995, Bd. 2) selbst im Studium eher „Hilfsmittel zum Zweck, aber nicht hauptsächliches Lernziel" sind. Um den Unterschied zur traditionellen Sichtweise deutlich zu machen, sind zunächst nur Objekt- und Klassendiagramme als Modellierungsnotationen erforderlich.
Die im Unterricht zu behandelnden Projektbeispiele müssen für die Schüler motivierend und überschaubar sein. Die Projekte sollen schrittweise in die grundlegenden(!) Konzepte der Objektorientierung einführen und objektorientiertes Denken nahe legen. Sie müssen nach einer geringen Anzahl von Unterrichtsstunden erfolgreich abgeschlossen werden, so dass die Motivation der Schüler durch neue Projekte wieder aufgefrischt werden kann. Ein im Unterricht einzusetzendes Modellierungs- und Entwicklungswerkzeug sollte intuitiv erschließbar sein, eine grafische Visualisierung anbieten und interaktives Analysieren von Projektbeispielen ermöglichen. Dabei sollte ein rascher Wechsel von Modellierungs- und Implementierungsebene möglich sein, um die jeweiligen Zusammenhänge erkennen zu können. Nur wenn beide Ebenen im Informatikunterricht frühzeitig integriert werden, können einerseits Erwartungen der Schüler hinsichtlich des Implementierens erfüllt werden und andererseits der Zweck informatischer Modellierung einsichtig gemacht werden.

Professionelle Entwicklungsumgebungen und UML-Werkzeuge zur objektorientierten Systementwicklung erfordern große Erfahrung, um ein erfolgreiches und effizientes Arbeiten zu ermöglichen. Die Philosophie der objektorientierten Programmierung, die auf Zuständen, Aktivitäten und Kommunikation von Objekten aufbaut, wird für den Anfänger durch diese Umgebungen eher verdeckt oder ist zumindest schwer erschließbar. Zwar existieren schülerorientierte, didaktisch-reduzierte Editoren zu objektorientierten Sprachen, doch konzentrieren sich diese auf die Implementierungsphase und vernachlässigen meist die Modellierungsphase. Die verfügbaren CASE-Tools, wie Fujaba, sind für Schüler im Anfangsunterricht zu komplex, die Einarbeitungszeit „steht in keiner Relation" zum Lerneffekt.
BlueJ ist ein Werkzeug, dass die angeführten Bedingungen zu erfüllen scheint und kostenlos verfügbar ist (www.bluej.org). Die Entwicklungsumgebung BlueJ ist das Produkt eines didaktischen Forschungsprojekts dreier Universitäten in Melbourne, Süd-Dänemark und Canterbury (UK). Ziel dieses Projektes ist die Entwicklung von Werkzeugen und Methoden, die Anfängern das Erlernen der Grundkonzepte objektorientierter Softwareentwicklung unter Verwendung der Sprache Java erleichtern. Ein weiteres Ergebnis dieser Forschungen ist ein didaktisch sehr gut aufgebautes Lehr- und Arbeitsbuch zur OOP mit Java und BlueJ, das auch m.E. in der allgemeinbildenden Schule einsetzbar ist. Dieses folgt konsequent dem „early bird"-Muster von J. Bergin (2000), d.h. wichtige Konzepte sollten möglichst früh geschult werden. Das Lehrkonzept wurde in mehreren Anfängervorlesungen erfolgreich eingesetzt. Es gibt eine Reihe von Argumentationen, weshalb die Sprache Java, auf der BlueJ aufsetzt, für den Anfangsunterricht ungeeignet erscheint (Böszörményi 2001). Andererseits ist Java eine recht populäre Sprache, die sich in der industriellen Softwareentwicklung zunehmend bewährt, sie bietet

eine saubere Umsetzung der wichtigsten objektorientierten Konzepte, es gibt zahlreiches Projekt-Material, und Werkzeuge wie BlueJ, Fujaba oder Poseidon, die mit Java arbeiten, sind kostenlos erhältlich.

Umsetzung und erste Erkenntnisse

Als 2003 das Buch zur Objektorientierten Programmierung mit Java und BlueJ erschien (zwar zunächst nur auf Englisch, aber dies war für die Schüler kein Hindernis), begeisterte der zugrundegelegte didaktische Ansatz, so dass er im Unterricht eines Grundkurses Klasse 12 erprobt wurde (in Brandenburg ist das erste Halbjahr eines Informatikkurses in der Klasse 12 als Anfangsunterricht zu bezeichnen, da der derzeitige Rahmenlehrplan im Wesentlichen eine Office-Schulung für die Klasse 11 intendiert). Der Unterrichtsabschnitt zur objektorientierten Softwareentwicklung folgte zunächst dem vorgeschlagenen Kurs im Buch, um den methodischen Weg (Tab. 2), der eigentlich für ein Grundstudium an einer Hochschule angelegt ist, im Oberstufenunterricht anzuwenden und zu testen. In dem Buch werden eine Fülle überschaubarer, zunehmend komplexer werdender Projekte behandelt (Tab. 3), die im Allgemeinen auf die Schüler motivierend wirken. Zu Beginn werden jeweils ein Ziel und eine Aufgabe formuliert. In der Regel wird ein Prototyp zu einer Problemstellung analysiert. Dabei werden neue Konzepte und damit verbundene Sprachkonstrukte eingeführt. Zahlreiche Übungen unterstützen den Lernprozess des Lernenden und führen zu Erweiterungen des Prototyps. Die Aufgaben in den Übungen werden allerdings von den Schülern teilweise als zu trivial empfunden.

Die Benutzungsschnittstelle von BlueJ (Abb. 1) ist innerhalb weniger Minuten von den Schülern erschließbar, so dass eine Konzentration auf die Konzepte möglich ist. Da BlueJ die Möglichkeit bietet, in einfacher Weise direkt Objekte von einzelnen Klassen zu erzeugen, deren Methoden aufgerufen werden können, interagieren die Schüler unmittelbar mit Objekten, ohne großen Aufwand durch die Entwicklungsumgebung. Eine Schülerin, nach ihrem Eindruck zum Kursverlauf und BlueJ gefragt, antwortete:

> *„Es dauert eine kleine Zeit, bis der Lernprozess in Gang gesetzt wird, aber da man in BlueJ alles gleich ausprobieren kann, versteht man z.B. den Aufbau von Methoden schnell. Auch die Unterscheidung von Klassen und Objekten ist durch die visuelle Hilfe leicht verständlich. Für Anfänger ist das Programm geeignet, da sie keine Vorkenntnisse benötigen und Schritt für Schritt vorgehen können."*

Die Visualisierung der Klassenstruktur ist in BlueJ ähnlich der UML-Notation, so dass an späterer Stelle im Unterricht die objektorientierte Entwicklung von größeren Systemen mit komplexeren Entwicklungsumgebungen fortgeführt werden kann, die diese Modellierungsnotation verwenden. Im Unterricht wurden die Klassen- und Objektdiagramme von BlueJ um Attribute und Methoden angereichert, die in BlueJ nicht visualisiert werden, sondern über ein Kontextmenü abgefragt werden können.

Obwohl die objektorientierte Sicht der menschlichen Denkweise ähnelt, scheint es für Schüler – ebenso wie für viele traditionelle Programmierer – sehr schwer zu akzeptieren, dass sie Objekte zunächst nur von außen betrachten sollen und nicht sofort deren Aufbau untersuchen dürfen. Schüler wollen erfahrungsgemäß implementieren. Daher muss der Modellierungsaufwand stets im angemessenen Verhältnis zum Ergebnis stehen, sonst wird das Modellieren nur als demotivierend empfunden. Mit der BlueJ-Umgebung scheint ein guter Kompromiss gefunden worden zu sein, um Erwartungen der Schüler und Intentionen des Informatikunterrichts in Einklang zu bringen. Eine wichtige Erfahrung für Schüler ist, dass bei der Analyse von Quelltexten nicht alle Details von Bedeutung sind, wenn zugehörige Modellierungsnotationen oder Kommentare vorhanden sind. Die Projekte in dem verwendeten Buch sind alle sehr gut kommentiert.

Ein Nachteil von BlueJ, Fujaba und anderer, auf Java basierender Software soll jedoch nicht verschwiegen werden: sie benötigen entsprechende Systemressourcen, die nicht auf allen Schulrechnern anzutreffen sind. Die Ausführungsgeschwindigkeit ist ansonsten sehr gewöhnungsbedürftig und kostet Unterrichtszeit.

In einer neuen Kursplanung soll die Einführung zur Objektorientierung mit BlueJ verbessert und auf einige Stunden des 2. Quartals ausgeweitet werden, wobei

- eine stärkere Variation der Arbeitsformen geplant ist (eine starke Orientierung am Buch bevorzugt die Einzelarbeit an Texten und am Rechner, andererseits wird die selbständige Erarbeitung von Texten gefördert),
- die Möglichkeit von Hausaufgaben unter Verwendung von BlueJ organisiert wird (hierbei ist insbesondere für Schüler ohne häuslichen Rechnerzugang eine Lösung zu finden),
- alternative, schülerorientierte Projektbeispiele gefunden werden, die unter anderem
    - Standardbeispiele zum Programmieren im Kleinen, wie Sortieren und Suchen, integrieren und
    - die Diskussion algorithmischer Aspekte stärker ermöglichen,
- der Übergang von BlueJ zu einem CASE-TOOL wie Fujaba erprobt wird,

- die Integration einer einfachen grafischen Benutzerschnittstelle in die Projekte geprüft wird, da die Schüler diese von anderen Programmen her gewohnt sind und sich unterschiedliche Sichtweisen (mentale Modelle) auf durch Software abzubildende Prozesse diskutieren lassen.

Als hilfreich dürften sich die zahlreichen Anregungen und Beispiele zum Informatikunterricht mit Java von Alfred Hermes auf der Webseite http://www.zitadelle.juel.nw.schule.de/if/java/java.shtml erweisen.

Die bisherigen Erfahrungen zur Einführung in die Objektorientierung mit BlueJ sind zusammengefasst recht positiv. Unterrichtsskizzen und –materialien zu dem skizzierten Anfangsunterricht mit BlueJ werden demnächst im HyFISCH (www.hyfisch.de) publiziert.

Dr. Marco Thomas

Didaktik der Informatik
Universität Potsdam
D-14482 Potsdam

Evangelisches Gymnasium Hermannswerder
D-14473 Potsdam

APPELRATH, Hans-Jürgen.; BOLES, Dietrich; CLAUS, Volker; WEGENER, Ingo: *Starthilfe Informatik.* Stuttgart, Leipzig (Teubner) 1998.

BALZERT, H.: *Lehrbuch der Software-Technik: Softwareentwicklung.* (Spektrum Akademischer Verlag) 1996.

BARNES, D. J.; KÖLLING, M.: *Objektorientierte Programmierung mit Java. Eine praxisnahe Einführung mit BlueJ.* München, Boston u.a. (Pearson Studium) 2003.

BAUMANN, R.: *Didaktik der Informatik. 2. neubearb. Aufl.* Stuttgart u.a. (Klett Verlag) 1996.

BERGIN, J.: *Fourteen pedagogical patterns. Proceedings of the Fifth European Conference on Pattern Languages of Programs.* 2000. http://csis.pace.edu/~bergin/PedPat1.3.html (15.10.03).

BÖSZÖRMÉNYI, L.: *JAVA für Anfänger? Warum Java nicht meine Lieblingssprache für einen Anfängerkurs ist.* In: *LOGIN*, 21. Jg. (2001), H. 1, S. 14.

BRICHAU, J.; MENS, K.: *Declarative Meta-programming.* http://prog.vub.ac.be/research/DMP/ (20.10.2003).

BROY, M.; SIEDERSLEBEN, J.: *Objektorientierte Programmierung und Softwareentwicklung. Eine kritische Einschätzung.* In: *Informatik Spektrum*, Jg. 2002, H. 2, S. 3-11.

BROY, M.; ROMBACH, D.: *Software Engineering. Wurzeln, Stand und Perspektiven.* In: *Informatik Spektrum*, Jg. 2002, H. 12, S. 438-451.

CRUTZEN, Cecile K. M.; HEIN, H.-W.: *Objektorientiertes Denken als didaktische Basis der Informatik.* Aus: Schubert, Sigrid (Hrsg.): *Innovative Konzepte für die Ausbildung. GI-Fachtagung "Informatik und Schule 1995"* Berlin (Springer) 1995. (=Informatik aktuell) S. 149-158.

CLAUS, Volker; SCHWILL, Andreas: *Duden Informatik. Ein Fachlexikon für Studium und Praxis.* Mannheim (Dudenverlag) 2001.

FÜLLER, K.: *Objektorientiertes Programmieren in der Schulpraxis.* Aus: Schwill, Andreas (Hrsg.): *Informatik und Schule Fachspezifische und fachübergreifende didaktische Konzepte* Berlin (Springer) 1999. (=Informatik aktuell) S. 190-201.

GESELLSCHAFT FÜR INFORMATIK E.V. - FACHAUSSCHUSS 7.3: *Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemeinbildenden Schulen.* In: *LOGIN*, 20. Jg. (2000), H. 2, S. Beilage.

GOOS, Gerhard: *Vorlesungen über Informatik* Berlin (Springer) 1995.

HUBWIESER, Peter: *Didaktik der Informatik. Grundlagen, Konzepte, Beispiele.* Berlin, Heidelberg, New York (Springer-Verlag) 2000.

JÄHNICHEN, S.; HERMANN, S.: *Was, bitte, bedeutet Objektorientierung?* In: *Informatik Spektrum*, Jg. 2002, H. 8, S. 266-275.

KOERBER, Bernhard; SACK, Lothar; SCHULZ-ZANDER, Renate: *Prinzipien des Informatikunterrichts* Aus: Arlt, Wolfgang (Hrsg.): *Informatik als Schulfach. Didaktische Handreichungen für das Schulfach Informatik.*

München, Wien (R. Oldenbourg Verlag) 1981. (=Datenverarbeitung, Informatik im Bildungsbereich. 4) S. 28-35.

The Pizza Compiler. http://pizzacompiler.sourceforge.net/ (20.10.2003).

RIEDEL, D.: *Grundsätze eines anwendungsorientierten Informatikunterrichts.* Berlin 1979.

SCHWILL, A.: *Objektorientierte Programmierung. Eine Rechtfertigung aus kognitionspsychologischer Sicht.* In: *LOGIN*, 13. Jg. (1993), H. 4, S. 44-45.

STERN, E.: *Grundlagen des erfolgreichen Lerntransfers.* Positionspapier, Universität Leipzig, Pädagogische Psychologie 1996.

SPOLWIG, S.: *Objektbasierte Programmierung im Anfangsunterricht.* In: *LOGIN*, 15. Jg. (1995), H. 3, S. 43-49.

THOMAS, Marco: *Informatische Modellbildung. Modellieren von Modellen als ein zentrales Element der Informatik für den allgemeinbildenden Schulunterricht.* Potsdam, Dissertation 2002.

Tab. 1: Übersicht zum 1. Quartal für einen Anfängerkurs in der gymnasialen Oberstufe (3-stündig)

Stunde	**Thema/Inhalte**	**Modelle**
1	Computereinsatz in der modernen Berufswelt	Software Life Cycle Mentale Nutzermodelle
2/3	Automaten und ihre Modellierung	Zustandsgraph, EVA Black-Box-Testen
4/5	Algorithmen und ihre Darstellung	Imperatives Rechenmodell Struktogramm
6	Rechenhilfsmittel bis Alan Turing	Technische Modelle
7-9	Alan Turing und die Turingmaschine - Berechenbarkeit, Entscheidbarkeit - Binärkodierung und Dualzahlensystem	Maschinenmodell Untersuchungsmodell Kodierungsmodell
10/11	Können Computer denken? Turing Test, Chinesisches Zimmer, Eliza	Mentale Modelle deklaratives Rechenmodell Patternmatching
12	Erfindung des Computers: Zuse, Neumann, u.a.	Technische Modelle Architekturmodell
13	Softwareentwicklung	Rechenmodelle, Programmiersprachen Vorgehensmodelle
14/15	Grundlagen der Objektorientierung - Objekt, Klasse, Methode - Parameter, Datentyp	OO-Vorgehensmodell Konzepte der Objektorientierung
16/17	Projekt Ticketautomat - Variablen, Datenübergabe - Konstruktoren und Methoden - bedingte Anweisung	Klassendiagramm Fundamentale Ideen
18/19	Objektinteraktion am Beispiel einer Digitaluhr - Abstraktion, Modularisierung - Teile und herrsche, Überladen - boolescher Ausdruck, modulo	Objektdiagramm Fundamentale Ideen
20	Objektinteraktion in einem Mail-System - Debugger	

Tab. 2: Kursabfolge und Konzepte im Buch „Objektorientierte Programmierung mit Java“ (Barnes/Kölling)

Kapitelnr. Überschrift	**Zentrale Konzepte**	**Weitere Konzepte und Begriffe**	**Java-Konstrukte**
1 Objekte und Klassen	Objekte, Klassen, Methoden, Parameter	Signatur, Typ, Instanz, Zustand, Methodenaufruf, Quelltext, Ergebnis (einer Methode), Compiler	-
2 Klassendefinitionen	Datenfelder, Konstruktoren, Parameter, sondierende und verändernde Methoden, Zuweisung, bedingte Anweisung	Kommentar, Sichtbarkeit und Lebensdauer einer Variablen, Boolescher Ausdruck, Lokale Variable Rumpf einer Methode, Deklaration, Initialisierung, Anweisung, Rückgabeanweisung, Ergebnistyp	Datenfeld, Konstruktor, Kommentar, Parameter, Zuweisung (=), Block, return, void, Zusammengesetzter Zuweisungsoperator (+=, -=), if
3 Objektinteraktion	Abstraktion, Modularisierung, Objekterzeugung, Objektdiagramme, Methodenaufrufe, Debugger	Lokale Variable, Klassen definieren Typen, Klassendiagramm, Objektreferenz, primitiver Typ, Überladen, interner und externer Methodenaufruf Teile und herrsche, Haltepunkt	Klassen als Typen, logischer Operatoren (&&,||), Verkettung von Zeichenketten, Modulo-Operator(%), Objekterzeugung (new), Methodenaufrufe (Punkt-Notation), this
4 Objektsammlungen	Sammlungen, Schleifen, Iteratoren, Arrays	null for-Schleife, Index, import-Anweisung, Bibliothek, Paket	ArrayList, Iterator, while-Schleife, null, Cast-Operator, Arrays, for-Schleife
5 Bibliotheksklassen	Benutzen von Bibliotheksklassen, Dokumentation lesen und schreiben	Java-Bibliothek, Schnittstelle, Implementierung, Map (Abbildung), Set (Menge), Zugriffsmodifikation, Geheimnisprinzip, unveränderliches Objekt, Klassenvariablen, statische Variablen Schnittstelle, javadoc, Kopplung, Konstante	String, ArrayList, Random, HashMap, HashSet, Iterator, StringTokenizer, static, final
6 Vermeiden von Fehlern	Testen, Tests automatisieren, Fehler beseitigen	Positives und negatives Testen, manuelle Ausführung Syntaxfehler, logischer Fehler, Modultest, Regressionstest, Aufrufsequenz	zusammengesetzter Zuweisungsoperator (&=)
7 Klassenentwurf	Entwurf nach Zuständigkeiten, Kohäsion, Kopplung, Refactoring	Code-Duplizierung, Kohäsion von Methoden und Klassen, Kapselung, Änderungen lokal halten Klassenmethode	static (für Methoden), Math
8 Bessere Struktur durch Vererbung	Vererbung, Subtyping, Ersetzbarkeit, polymorphe Variablen	Superklasse, Subklasse, Vererbungshierarchie, Subtyp, Variablen und Subtypen, Ersetzbarkeit ist-ein, polymorphe Sammlung, Verlust von Typinformation	extends, super (in Konstruktoren), Cast-Operator (erneut), Object, Wrapper-Klassen
9 Mehr über Vererbung	Methoden-Polymorphie, Überschreiben von Methoden, statischer und dynamischer Typ, dynamische Methodensuche	- Redefinition, Methodensuche, Methodenauswahl	super (für Methoden), toString, protected
10 Weitere Techniken zur Abstraktion	abstrakte Klassen, Interfaces	abstrakte Methode, abstrakte Subklassen, multiple Vererbung konkrete Klasse,	abstract, implements, interface, instanceof
11 Fehlerbehandlung	defensive Programmierung, Fehler melden, Exceptions auslösen und behandeln, einfache Datenverarbeitung	Exception (ungeprüft und geprüft), Exception-Handler, Serialisierung	TreeMap, TreeSet, SortedMap, Exception, throw, throws, try, catch
12 Entwurf von Anwendungen	Identifizierung von Klassen, Entwurf von Schnittstellen, CRC-Karten, Entwurfsmuster	Verb/Substantiv, Szenario, Prototyping Analyse und Entwurf, Geschäftsfall, Stub	-
13 Eine Fallstudie	Entwicklung einer kompletten Anwendung	- -	-

Tab. 3: Projekte im Buch „Objektorientierte Programmierung mit Java“ (Barnes/Kölling)

Kapitelnr.	**Projekte**
1	Interaktives Zeichnen einfacher grafischer Figuren. Anmelden von Studenten für Laborkurse.
2	Simulation eines Ticketautomaten für Zugtickets. Speicherung von Informationen über Bücher.
3	Implementierung einer Anzeige einer Digitaluhr. Simulation eines Mail-Systems zwischen zwei Clients.
4	Implementierung eines einfachen elektronischen Notizbuches. Auktionshaus. Programm zum Analysieren einer Logdatei für Zugriffe auf Webseiten.
5	Implementierung eines Eliza-ähnlichen Dialogsystems, das den technischen Kundendienst Kunden gegenüber simulieren soll. Grafische Animation springender Bälle.
6	Prototyp eines Terminkalenders. Implementierung eines Taschenrechners. Modellierung des Beladens von Paletten mit Ziegelsteinen.
7	Textbasiertes, interaktives Adventure-Spiel (1).
8	Datenbank für CDs und Videos (1)
9	Textbasiertes, interaktives Adventure-Spiel (2). Datenbank für CDs und Videos (2)
10	Jäger-Beute-Simulation.
11	Adressbuch mit einer optionalen grafischen Benutzungsschnittstelle. Das Nachschlagen im Adressbuch ist sehr flexibel: Adressen können auf Basis unvollständiger Angaben des Namens oder der Telefonnummer gesucht werden.
12	System zur Sitzplatzreservierung für ein Kino.
13	Taxi-Buchungs- und Verwaltungssystem.

Abb. 1: BlueJ-Interaktionsumgebung

Abb. 2: BlueJ-Editor