

Strictly models and objects first – Unterrichtskonzept für objektorientierte Modellierung

Ira Diethelm

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

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

1 Einleitung

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

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

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

2 Entwicklung der Leitideen

2.1 Ausgangslage

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

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

1. Welche Kriterien muss eine Unterrichtskonzept und dessen Methodik zur objektorientierten Modellierung erfüllen, um die lerntheoretischen Vorteile der OOM zu unterstützen und die genannten Probleme von vornherein zu reduzieren?
2. Welche methodischen Hilfen können aus den genannten Erfahrungen, der Lerntheorie, der allgemeinen Didaktik und auch aus den Erkenntnissen der Softwaretechnik hergeleitet werden, um bestimmte Lernschwierigkeiten beim Unterricht in OOM gezielt zu vermindern?

2.2 Models first

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

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

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

2.3 Strictly objects first

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

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

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

2.4 Nachvollziehbarkeit

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

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

2.5 Ausführbarkeit

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

3 Auswahlkriterien für Werkzeuge und Beispiele

3.1 Die Wahl der Werkzeuge

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

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

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

3.2 Die Wahl der Beispiele

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

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

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

4 Entwicklung der Unterrichtsmethoden

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

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

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

4.1 Einführung der Objekte

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

4.1.1 Vom Objekt zur Klasse

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

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

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

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

4.1.2 Vererbung

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

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

4.2 Methoden entwerfen

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

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

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

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

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

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

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

4.3 Den Ablauf von Methoden verstehen

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

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

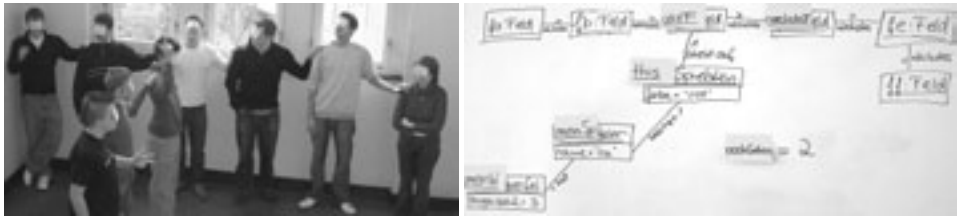


Abbildung 1: Objektspiel und Zetteltest

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

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

4.4 Ein Projekt durchführen

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

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

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

4.5 Eine textuelle Programmiersprache erlernen

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

Dieses Vorgehen forderte bereits Wilhelm Viëtor 1882 bei der Einführung von fremdsprachlichen Pflichtunterricht an deutschen Gymnasien bzw. deren Vorgängern (den Realgymnasien): „Und wenn es euch gelänge, [dem Lerner] die beste Grammatik und das umfassendste Wörterbuch in den Kopf zu schaffen, so hätte er noch immer keine Sprache gelernt!“, vgl. [Vi84]. Er kritisiert hier die bis dahin im Fremdsprachenunterricht vorherrschende und mühsame erklärungsbasierte synthetisch-deduktive Unterrichtsmethode und fordert ein von der Beobachtung und dem Beispiel ausgehendes, induktives Lernen. In Bezug auf die Verarbeitungsrichtung stellte z.B. Butzkamm in [Bu93] fest, dass beim natürlichen Spracherwerb „datengeleitete bottom-up-Prozesse¹“, (von der Wahrnehmung zur kognitiven Repräsentation) vorherrschen.

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

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

5 Fazit und Ausblick

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

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

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

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

Literaturverzeichnis

- [BB04] Helmut Balzert und Heide Balzert. Modellieren oder Programmieren oder beides? LOG IN, 128/129, Seiten 20–25, 2004.
- [Be00] Joseph Bergin. Why Procedural is the Wrong First Paradigm if OOP is the Goal, 2000.
- [BS05] Jürgen Börstler und Carsten Schulte. Teaching Object Oriented Modelling with CRC-Cards and Roleplaying Games. In 8th IFIP World Conference on Computers in Education, Cape Town, South Africa, July 2005.
- [Bu93] Wolfgang Butzkamm. Psycholinguistik des Fremdsprachenunterrichts. Francke, Tübingen, 1993.
- [DGS06] Ira Diethelm, Leif Geiger und Carsten Schulte. Einführung in die Objektorientierung im Informatik-Anfangsunterricht. www.se.eecs.uni-kassel.de/se/fileadmin/se/LehrerFortbildung/OOM-Skript061108.pdf, 2006.

- [DGZ02] Ira Diethelm, Leif Geiger und Albert Zündorf. UML im Unterricht: Systematische objektorientierte Problemlösung mit Hilfe von Szenarien am Beispiel der Türme von Hanoi. In Sigrid Schubert, Johannes Magenheimer, Peter Hubwieser und Torsten Brinda, Hrsg., Erster Workshop der GI-Fachgruppe Didaktik der Informatik, Bommerholz, Germany, 2002.
- [DGZ03] Ira Diethelm, Leif Geiger und Albert Zündorf. Fujaba goes Mindstorms: Objektorientierte Modellierung zum Anfassen. In Peter Hubwieser, Hrsg., Informatische Fachkonzepte im Unterricht, Informatik und Schule INFOS 03, München, Germany, 2003.
- [DGZ05a] Ira Diethelm, Leif Geiger und Albert Zündorf. Mit Klebezettel und Augenbinde durch die Objektwelt. In Steffen Friedrich, Hrsg., Unterrichtskonzepte für informatische Bildung, Informatik und Schule INFOS 05, Dresden, 2005.
- [DGZ05b] Ira Diethelm, Leif Geiger und Albert Zündorf. Teaching Modeling with Objects First. In 8th IFIP World Conference on Computers in Education, Cape Town, South Africa, 2005.
- [Di07] Ira Diethelm. „Strictly models and objects first“ – Unterrichtskonzept und -methodik für objektorientierte Modellierung im Informatikunterricht. Dissertation, Universität Kassel, Fachbereich Elektrotechnik / Informatik, Mai 2007. im Druck.
- [Fü99] Klaus Füller. Objektorientiertes Programmieren in der Schulpraxis. In Andreas Schwill, Hrsg., Informatik und Schule, INFOS 99, Seiten 190–201. Springer Verlag, 1999.
- [HGW97] Simon Holland, Robert Griffiths und Mark Woodman. Avoiding object misconceptions. SIGCSE Bull., 29(1):131–134, 1997.
- [Sc95] Andreas Schwill. Programmierstile im Anfangsunterricht. In Sigrid Schubert, Hrsg., Innovative Konzepte für die Ausbildung, 6. GI-Fachtagung Informatik und Schule INFOS 95, Seiten 178–187. Springer Verlag, 1995.
- [Sc04] Carsten Schulte. Lehr- Lernprozesse im Informatik-Anfangsunterricht: theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II. Dissertation, Universität Paderborn, Didaktik der Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, März 2004.
- [Sp01] Siegfried Spolwig. Methodische Probleme mit OOP im Anfangsunterricht, 2001.
- [Vi84] Wilhelm Viëtor. Der Sprachunterricht muss umkehren : ein Pamphlet aus dem 19. Jahrhundert neu gelesen. Hueber, München, 1984. erstmals in Forum Sprache, 1882.