Objektorientierte Modellierung unter Einsatz eines CASE-Tools im Informatikunterricht der Jahrgangsstufe 11

Stefan Moll

Teletta-Groß-Gymnasium
Gaswerkstr. 17
D-26789 Leer
stmoll@web.de

Abstract: Für einen objektorientierten Informatikunterricht in der Jahrgangsstufe 11 wird ein Unterrichtskonzept mit dem Schwerpunkt auf Modellierung begründet, entwickelt und die Erfahrungen bei der Umsetzung skizziert. Dabei kommt ein CASE-Tool gewinnbringend zum Einsatz.

1 Einleitung

Im Informatikunterricht der Jahrgangsstufe 11 wird zunehmend ein objektorientierter Ansatz verfolgt. Aus pragmatischen Gründen wird dabei oft auf Objekt-Pascal (Delphi) als Programmiersprache zurückgegriffen, worauf ich mich im Folgenden bei der Planung und Werkzeugauswahl beziehen werde. Dabei gelten die meisten meiner Überlegungen unabhängig von der Programmiersprache.

Unterrichtserfahrungen im Bereich des objektorientierten Ansatzes haben nach meinen Erfahrungen Schwierigkeiten der Schülerinnen und Schüler offen gelegt, die insbesondere dann auftreten, wenn der Anspruch der eigenständigen Modellierung und Entwicklung, auch von relativ größeren Vorhaben, vertreten wird. Diese Schwierigkeiten möchte ich auf die folgenden Ausgangsfragen zuspitzen, die als Orientierungspunkte für die weiteren Überlegungen dienen:

1. Wie kann die Vorstellung von einzelnen Objekten und deren Zusammenwirken sowie das Zusammenspiel der verschiedenen Klassen besser gefördert werden?
2. Wie können grafische Methoden der Modellierung von den Schülerinnen und Schülern besser genutzt werden bei der Erstellung und Weiterentwicklung von Modellen?
3. Wie können die unterschiedlichen Ebenen der sachbezogenen Modellierung (Modellierung des Anwendungskontextes) und der darstellungsbezogenen Modellierung (Darstellung auf dem Bildschirm) den Lernenden deutlicher gemacht werden?
4. Wie können die Schwierigkeiten in der programmiersprachlichen Umsetzung von statischen Modellen verringert werden?
5. Wie können die Bedeutungen einzelner Attribute und Methoden den Schülerinnen und Schülern in einem Softwareprojekt längerfristig präsent sein?

Die im Folgenden dargelegte Entwicklung eines Unterrichtskonzeptes und die parallele Weiterentwicklung eines einfachen CASE-Tools werden ergänzt durch Anmerkungen zu den Erfahrungen mit dem Konzept im Schuljahr 2001/2002.

2 Didaktische Überlegungen

Vor diesem Hintergrund werden im Folgenden einige Überlegungen zur Modellierung im Informatikunterricht dargelegt und verschiedene Optionen für eine unterrichtliche Umsetzung diskutiert.

2.1 Modellierung im Informatikunterricht

Die Bedeutung der Modellierung als Inhalt eines allgemein bildenden Informatikunterrichts ist unstrittig.

Die Entwicklung von Verfahren zur Modellierung von Informatiksystemen und von (zumeist grafischen) Modellierungstechniken ist ein zentraler Gegenstand der Softwaretechnik. Dabei haben leicht zugängliche grafische Methoden bereits längst auch in den Informatik-Unterricht Einzug gehalten: Programmablaufpläne, Flussdiagramme, Struktogramme etc. werden dazu eingesetzt, Abläufe zu beschreiben und nachvollziehbar zu machen. Damit wurden aber bislang nur Aspekte der dynamischen Modellierung erfasst. Darüber hinaus gelang es auch oft nicht, deren Nutzen für eine programmiertechnische Umsetzung transparent zu machen. Diese Schwierigkeit gilt nach meinen Erfahrungen auch im Bereich des objektorientierten Ansatzes im Hinblick auf die Klassenmodellierung im Rahmen der UML. Diese syntaktisch einfache, grafische Notation ist zwar relativ leicht zu erlernen, ihre Basisfunktion für die spätere Codierung wird jedoch von den Schülerinnen und Schülern oft nicht gesehen. Die grafische Darstellung erscheint als Zusatzaufgabe, die Codierung als darauf folgendes, neues Problem (vgl. zweite Ausgangsfrage).

Erst durch die Behandlung konkreter Modellierungstechniken sieht Hubwieser die Möglichkeit gegeben, Modellierung in angemessener Weise im Unterricht zu thematisieren und zu gewährleisten [Hu00, S.90]. Dabei ist jedoch zu beachten, dass die Techniken einzeln und behutsam eingeführt werden. Eine Konzentration auf einzelne Grundkonzepte und -techniken ist geboten.

Sollen von den Schülerinnen und Schülern auch im Anfangsunterricht eigenständige Modellierungen vorgenommen werden, so sind Beispiele auszuwählen, die einerseits nicht zu komplex und umfangreich sind, die aber andererseits auch noch etwas Modellierungsumfang besitzen. Reinsch/Schulte [RS01, S.4] schlagen dazu etwa ein Mini-Roulette vor.

2.2 Objektorientierte Methode in der Schule

Für den Informatikunterricht in der Schule sind die Konzepte, Notationen und Vorgehensweisen aus der Informatik zu modifizieren und aufzubereiten.

Zu den für die Schule relevanten *statischen* Konzepten zählen: Objekt, Klasse, Nachricht, Attribut, Methode, kennt-Beziehung (Assoziation), hat-Beziehung (Aggregation) und ist-Beziehung (Vererbung) (vgl. [LSW99b], [MSWWF99, S.29f]).

Damit ist auch in etwa eine Reihenfolge der Behandlung skizziert. Weitere statische Konzepte spielen in der Schule eine untergeordnete Rolle.

Von den Notationen, die sich auf statische Konzepte beziehen, ist das Klassendiagramm geeignet (vgl. z.B. [Sp00]). Diese können im Unterrichtsverlauf früh eingeführt werden und dann fortlaufend um die hinzugekommenen Aspekte erweitert werden.

Die Mühsal der Erstellung und vor allem der Modifikation einer grafischen Darstellung bleibt dabei aber auch ein Hindernis, wie dies für die bereits verbreiteten grafischen Techniken gilt. Erst ein interaktives grafisches Werkzeug ermöglicht eine einfache Erstellung und Modifikation der Darstellung, d. h. des Modells. Diese technische Hilfe unterstützt dann auch die *Modellentwicklung* in der Notation. Dies wird von Werkzeugen zum ‚computer aided software engineering‘ (CASE-Tools) geleistet.

Im Bereich der *dynamischen* Konzepte und Notationen herrscht weitgehend noch Uneinigkeit bzw. Unschlüssigkeit über die Relevanz und Eignung für den schulischen Unterricht. Die von mir mit Blick auf die Jahrgangsstufe 11 angedeutete Beschränkung auf Grundkonzepte legt es nahe, sich auf Klassendiagramme zu beschränken und diese von Beginn an schrittweise einzuführen.

2.3 Problembereiche des objektorientierten Anfangsunterrichts

Spolwig [Sp01] klassifiziert in seinem Artikel die generellen Schwierigkeiten des objektorientierten Ansatzes im Anfangsunterricht. Schwierigkeiten, die vor allem auf die vielfältigen Anforderungen an die Schülerinnen und Schüler zurückgeführt werden können.

Problembereich „Anwendungsproblem analysieren“

Spolwig sieht zwar selbst die Analyse des Anwendungsproblems nicht als das größere Problem bei der Bearbeitung mehr oder weniger kleiner Anwendungsprobleme, aber m. E. geht doch hiervon eine nicht zu unterschätzende Schwierigkeit aus. Dies gilt insbesondere, wenn es darum geht, funktionierende und sinnvolle Modellierungen zu finden und als Anfängerin bzw. Anfänger das Zusammenspiel der einzelnen Objekte genau zu durchschauen (vgl. unter anderem die erste Ausgangsfrage).

Da das Auffinden von ‚guten‘ Modellierungen auch stark von Erfahrungen abhängt, sollten den Schülerinnen und Schülern zunächst Modellierungen vorgegeben werden. Als ausreichende Anforderung verbleibt dann immer noch, dieses Modell nachzuvollziehen und auf seine Schlüssigkeit zu prüfen. Das Modell sollte allerdings in einer Gestalt vorgegeben werden, die eine weitere Bearbeitung ermöglicht. Für die Generierung und Erarbeitung von Modellen werden sowohl in Informatikpublikationen als auch in didaktischen Beiträgen (z.B. [RS01, S.4]) CRC-Karten vorgeschlagen (Class-Responsibilities-Collaborations). Diese bieten zunächst einen etwas geringeren Differenzierungsgrad als Klassendiagramme, können somit weiter ausdifferenziert werden. Im Zusammenhang mit der Einarbeitung in Modelle (oder der Überprüfung eigener Modellentwürfe) schlagen Reinsch/Schulte [RS01] aufbauend auf die CRC-Karten ein Objektspiel nach Bergin [Be00] vor, um den Schülerinnen und Schülern in diesem Rollenspiel die Beziehungen

und das Zusammenwirken der Objekte enaktiv nachvollziehbar zu machen. Damit wird eine Grundlage gelegt für die weitere Auseinandersetzung mit dem Modell.

Problembereich „Objektorientierte Designprinzipien kennen lernen“

Inwiefern Designprinzipien, die im objektorientierten Ansatz Verwendung finden, für die Schule Relevanz besitzen, möchte ich zunächst am MVC-Konzept (model-view-controller) diskutieren, das in besonderer Weise dazu geeignet erscheint, die von mir formulierte dritte Ausgangsfrage zu beantworten. Das MVC-Konzept spiegelt gerade die Trennung zwischen sachbezogener und darstellungsbezogener Modellierung wieder, Spolwig [Sp01] berichtet jedoch, dass Softwareprojekte mit solchen Ansprüchen mit Anfängerinnen und Anfängern nur mit einigen guten Schülerinnen und Schülern zu leisten sind. Die Umsetzung des MVC-Konzeptes stellt hohe (kognitive) Anforderungen und sollte deswegen erst zu einem späteren Zeitpunkt angestrebt werden.

Im Hinblick auf eine den Schülerinnen und Schülern deutlichere Trennung zwischen sach- und darstellungsbezogener Modellierung erscheint es mir sinnvoller, die Modellierung stärker im Sinne eines „idealtypischen Verlaufs“ zu gestalten (vgl. [B99]). Zuerst wird in einer Analysephase die Sachmodellierung in den Vordergrund gestellt und in einem OOA-Modell beschrieben. Die Analysephase schließt mit den deutlich abgetrennten Entwürfen für die Benutzungsoberfläche. Anschließend wird dieses Modell in der Designphase um technische und darstellungsbezogene Aspekte (weitere Klassen, weitere Attribute und Methoden) erweitert, die sich u. a. aus den Entwürfen zur Benutzungsoberfläche ableiten (Ergebnis: OOD-Modell). Nicht die Aufteilung von model und view, sondern eine *zeitliche* Aufteilung sollte den Unterschied schärfen.

Problembereich „Pascal lernen“

Auch für eine erfolgreiche Umsetzung eines (objektorientierten) Modells ist ein beachtenswertes Maß an Programmierkompetenz erforderlich, was Kenntnisse der Programmiersprache und deren grundlegender Konzepte umfasst. Spolwig vermerkt dazu [Sp00, S.54] „Es ist eindeutig zu beobachten, dass die Verlagerung des Unterrichtsschwerpunktes auf OOM mit einem dramatischen Verlust an Programmierkompetenz [...] einhergeht.“

Um z.B. die im Modell enthaltenen Methoden auch implementieren zu können, sind Kenntnisse der Konzepte Schleifen und Verzweigung und deren Realisierung in Objekt-Pascal erforderlich. Darüber hinaus werden dann zumeist auch Parameter und ggf. lokale Variablen benötigt.

Das Erarbeiten all dieser Kenntnisse allein im Rahmen umfangreicherer Softwareprojekte erscheint schwierig, insbesondere aufgrund der vielfältigen sonstigen Anforderungen. Anfängerinnen und Anfängern müssen entsprechende grundlegende Kenntnisse m.E. zunächst an kleinen Beispielen über einen Zeitraum von mehreren Wochen oder auch Monaten erarbeiten.

Dabei sollte jedoch keine Zuwendung zum imperativen Paradigmen erfolgen, die Objektsicht sollte streng eingehalten werden. Zur Erarbeitung der grundlegenden Kenntnisse erscheinen mir dazu Einstiegsbeispiele aus „Stifte und Mäuse“ [LSW99] geeignet, wobei die Beispiele gegebenenfalls im objektorientierten Sinne optimiert werden sollten: z.B. in der Einheit „Pfeilwurf“ (modifiziert zu [LSW01]), indem Pfeil und Zielscheibe als Klassen vorgegeben werden, die dann auch analysiert werden können und für die letzte Anwendungserweiterung auch selber erweitert werden müssen. So gelingt auch ein

Übergang zur eigenen Klassenerstellung im Rahmen der folgenden ersten größeren Modellierungstätigkeiten.

2.4 Weitere Aspekte

Implementation im Informatikunterricht

Auch wenn die Modellierung im Vordergrund steht, gehört selbstverständlich die Umsetzung in Programmcode immer dazu, denn erst eine Umsetzung mit einer Programmiersprache in ein reales Programm ermöglicht es auch, das erstellte Modell praktisch zu bewerten.

Dabei ist zu überlegen, inwiefern es ohne didaktische Verluste möglich ist, die Lernenden in der Code-Erstellung durch ein CASE-Tool zu unterstützen – ähnlich wie dies bei der professionellen Softwareerstellung der Fall ist. Der Code wird in Teilen automatisiert erstellt. Hubwieser [Hu00] sieht in Code-Generatoren eine interessante Perspektive. „Damit wird den Schülern unmittelbar die Bedeutung ihrer Modellierungsergebnisse vor Augen geführt.“ (S. 75). Dieser Vorteil sollte auch im Sinne der vierten und der zweiten Ausgangsfrage genutzt werden.

Ausreichende Anforderungen an eine Programmierkompetenz der Lernenden verbleiben bei der Ausgestaltung der Abläufe in den einzelnen Methoden, sofern sich die Code-Generierung auf den ‚statischen‘ Teil (Deklarationen, leere Methodenrumpfe) beschränkt. Die vom CASE-Tool übernommenen Anteile an der Code-Erstellung umfassen Kenntnisse, die die Lernenden zwar passiv in der Interpretation vorhandenen Codes beherrschen müssen, aber nur in eingeschränktem Umfang als aktives Wissen für spätere Anpassungen langsam aufbauen müssen (z.B. für das nachträgliche Einfügen eines Attributs). Gut strukturierter, vom Werkzeug erzeugter Code kann darüber hinaus Vorbildfunktion besitzen.

Dokumentation

In Bezug auf die fünfte Ausgangsfrage (langfristige Verfügbarkeit der Bedeutung von Attributen und Methoden) besteht eine Möglichkeit, bereits frühzeitig (in der ersten Modellierungsphase) Dokumentationen zu beginnen. Diese werden im weiteren Verlauf immer wieder ergänzt und für alle verfügbar gemacht, insbesondere für die Phase der Implementation. Zur Erleichterung der Arbeit und zur besseren Integration in die Modellierung sollte ein zu verwendendes CASE-Tool eine Dokumentationsfunktion bieten und diese auch in den erzeugten Code integrieren, um den Schülerinnen und Schülern die Bedeutungen auch in der Phase der Implementation präsent zu halten.

3 Auswahl und Bewertung von CASE-Tools für den Unterricht

Aus den bisherigen Überlegungen geht hervor, dass ein CASE-Tool im Informatikunterricht vor allem aus verschiedenen didaktischen Gründen seine Berechtigung besitzt (z.B. interaktives Erstellen und Bearbeiten von Modellen, Erledigung von wenig gewinnbringender Implementationsarbeit oder auch Integration der Dokumentation in Modellentwicklung und Implementation).

Werden nun CASE-Tools im Unterricht eingesetzt, so ist eine begründete Auswahl notwendig, um durch ein geeignetes Werkzeug den Modellierungsprozess sinnvoll unterstützen zu können und zugleich Probleme, die wiederum durch das Unterrichtsmedium verursacht werden, minimieren bzw. vermeiden zu können.

Grundlegende Überlegungen zur Bewertung von Software für Unterricht und Lernen von Leufen [Le96] und ein Kriterienraster zur analytischen Bewertung in LSW [LSW99a] liefern in Verbindung mit den spezifischen Anforderungen aus den obigen Überlegungen die folgende, stark zusammengefasste Kriterienliste für CASE-Tools:

- Ist die Darstellung sach- und adressatengemäß? D.h.: Werden zum Beispiel Begriffe und Bezeichnungen verwendet, die dem üblichen Unterrichtsgebrauch entsprechen?
- Sind die notwendigen Bedienungsschritte überschaubar? Bieten Menüs und Dialoge eine für die Schule und insbesondere den Anfangsunterricht gebotene eingeschränkte Auswahl an Möglichkeiten und Optionen? Entsprechen Möglichkeiten und Optionen dem zu vermittelnden Begriffsverständnis?
- Können alle Informationen (Klassen mit jeweiligen Attributen und Methoden sowie Beziehungen) in kompakter, übersichtlicher Form in einem Diagramm angezeigt werden?
- Gibt es eine durchgängige Dokumentationsmöglichkeit mit einer kompakten Anzeige- bzw. Speichermöglichkeit?
- Ist der automatisch erzeugte Programmcode übersichtlich, strukturiert und compilationsfähig – und damit für die Lernenden handhabbar und mit angemessenem Aufwand erweiterbar? Ist das zu Grunde liegende Modell darin gut erkennbar?

Die Erstellung von Sequenzdiagrammen und weiteren Diagrammformen wurden in der von mir durchgeführten Analyse nicht mit einbezogen.

Das Angebot möglicher Werkzeuge reduziert sich bereits stark durch die Prämisse der Unterstützung von Dephi/Objekt-Pascal. Für zwei der in Frage kommenden Werkzeuge ergibt sich dabei folgendes, zusammenfassende Bewertungsergebnis:

Die Eignung des Werkzeugs *ModelMaker 5* (Fa. Modelmaker, NL) erscheint für Anfängerinnen und Anfänger fragwürdig, da

- abweichende Begrifflichkeiten verwendet werden,
- eine Vielzahl unnötiger Optionen gleichzeitig angeboten werden,
- keine kompakte Ansicht der Methoden und Attribute aller Klassen möglich ist,
- die Dokumentation keine Gesamtansicht bietet und
- die Codegenerierung für die Zielgruppe unübersichtliche Teile enthält.

Das Werkzeug *Object Technologie Workbench (OTW) 2.4* (Fa. OTW Software, USA) erscheint nur eingeschränkt sinnvoll, da den positiven Punkten:

- einer übersichtlichen Darstellung der Klassendiagramme und
- eines guten vom Programm erzeugten Objekt-Pascal-Codes

die negativen Punkte entgegenstehen, dass

- Attribute und Beziehungen nicht in der für Anfänger gebotenen Klarheit getrennt werden und
- komplizierte Bedienungsschritte z.B. für die Erzeugung des Codes oder die Gesamtansicht der Dokumentation notwendig sind.

Neben den professionellen Werkzeugen stand eine Projektarbeit der Schülerakademie Mathematik – Informatik Münster (www.smims.de) aus dem Jahr 2001 zur Verfügung. Diese wurde vom Autor im Sinne des Anforderungsprofils in wesentlichen Teilen erweitert. Unter dem Arbeitstitel „UMLed“ entstand ein kleines Werkzeug, das durch folgende Eigenschaften gekennzeichnet ist:

- Erstellung von Klassendiagrammen,
- eine einfache, der Lerngruppe angemessene, sachgerechte Darstellung und Auswahlmöglichkeiten in Menüs, Funktionen und Dialogen,
- Dokumentationsmöglichkeiten für Attribute, Methoden und die jeweilige Klasse mit einer Gesamtansicht der Dokumentation und Exportmöglichkeit,
- Erzeugung eines übersichtlichen, strukturierten Codes (in Objekt-Pascal) als Ansicht bzw. Export für einzelne Klassen oder für alle Klassen in einem gesamten (Stifte-und-Mäuse-) Projekt und
- spezielle, nicht editierbare Bibliotheksklassen.

Die – zwar geringen – Möglichkeiten des Werkzeugs sind für die Modellierungsarbeit mit Anfängerinnen und Anfängern auf jeden Fall ausreichend.

4 Unterrichtskonzept

Der eigentlichen Modellierungsarbeit in zusammenhängenden Projekten geht (s.o.) eine mehrwöchige bzw. mehrmonatige Phase voran, in der grundlegende Konzepte der Objektorientierung in Anlehnung an [LSW99b] vermittelt werden. Dabei steht auch die Erarbeitung von Kontrollstrukturen im Vordergrund. In einer kleinen Überleitungseinheit werden weitere Klassen zunächst genutzt und später erweitert. Auch das CASE-Tool wird in dieser Phase bereits eingesetzt.

4.1 Ablauf eines Modellierungsprojekts

Die dann im Unterricht folgenden Software- bzw. Modellierungsprojekte sollen den folgenden Phasenablauf besitzen:

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

Für das erste Modellierungsprojekt habe ich ein Jahrmarktsspiel „Groschenroulette“¹ ausgewählt, das die Schülerinnen und Schüler zunächst an einem realen Modell spielen und in Form von CRC-Karten weiter erkunden und als Objektspiel nachspielen. Bei diesem ersten Projekt werden einzelne Schritte (s.u.) deutlicher erarbeitet, da sie z.T. neue Kenntnisse erarbeiten oder vertiefen. Bei späteren Projekten fallen gegebenenfalls verschiedene Schritte zusammen bzw. auch ganz weg.

Der Unterricht wurde von mir im Schuljahr 2001/2002 am Ratsgymnasium in Münster erprobt. Zu den einzelnen Schritten im Projektablauf, sind jeweils kurze Anmerkungen zu den Erfahrungen im Unterricht vorhanden.

OOA-Phase

1. Schritt: Den Anwendungskontext erkunden (das Spiel spielen)

2. Schritt: CRC-Modell aufstellen (bzw. erkunden) und simulieren (Objektspiel)

Bei der ersten Durchführung benennen die Schülerinnen und Schüler Ideen für Objekte, Klassen, Verantwortlichkeiten, aber führen keine eigenständige Modellentwicklung durch. Das Modell wird vorgegeben.

Objektspiel: CRC-Karte als Rollenkarte plus Wertekarte

Grundregeln des Objektspiels: – Objekte werden nicht von sich aus tätig, – Objekte haben „Scheuklappen“, kennen nur das was für sie festgelegt ist. – Objekte schreiben sich alles auf. – Objekte können auch nur über ihr Wissen Auskunft geben.

Erfahrungen: Die Schülerinnen und Schüler konnten einige Modellelemente auch selbst entwickeln. Schwierig in der Umsetzung des Spiels ist die Balance zwischen „Spielen lassen“ und Einhaltung der Regeln.

3. Schritt: Umsetzung des CRC-Modells in ein Klassendiagramm

Arbeiten mit dem CASE-Tool, Einbindung von Dokumentationen, Collaborations werden als kennt-Beziehungen umgesetzt.

Erfahrungen: Der Umgang mit dem Werkzeug fiel leicht, obwohl auch durch dieses Fragen aufgeworfen wurden (Was kommt bei Parameter hin?), die eigentlich späteren Schritten zugeordnet sind. Vereinzelt gab es Schwierigkeiten durch die nicht gewünschte Übernahme von Beziehungen als Attributen (nahegelegt durch die Wertekarten).

Es waren berechnete Schwierigkeiten festzustellen, was unter Dokumentation sinnvoll formuliert werden kann.

Schülerinnen und Schüler war die Bedeutung der Modellierungsarbeit für spätere Implementation noch nicht genügend bewusst. Eine frühe Ansicht des zum Modell generierbaren Codes wäre hier evtl. hilfreich.

4. Schritt: Entwurf einer Programmoberfläche und Festlegung der Programmaufgaben

In Kleingruppen werden Entwürfe erstellt und in der Klasse die Vorschläge diskutiert.

Erfahrungen: Es gab ähnliche Entwürfe in den einzelnen Gruppen, Kriterien für den Vergleich der Entwürfe waren für die Schüler schwer zu benennen.

OOD-Phase

5. Schritt: Erweiterung des Modells und Ergänzung der Klassen

Die Erweiterung des Modells wird zunächst auf CRC-Ebene vorgenommen, das erweiterte Objektspiel wird nur kurz angespielt, da es zu umfangreich ist. Die Ergänzungen

¹ Auf sechs verschiedene Felder kann von den Spielerinnen bzw. Spielern jeweils einmal ein Einsatz (ein Groschen) gesetzt werden. Ein Zufallsmechanismus lässt die Felder abwechselnd leuchten. Das Feld, auf dem das Licht stehen bleibt, gewinnt. Gewinn ist eine Gewinnmarke, die gegen kleinere oder größere Gewinne eingetauscht werden können.

werden im CASE-Tool umgesetzt. Da kein Rückgriff auf vorhandene Elemente (wie TLabel) stattfindet, muss die Klasse ‚Spielanweisung‘ selbst modelliert werden.

Erfahrungen: Die Modellierung von ‚Spielanweisung‘ gelang im Unterrichtsgespräch. Die Arbeit an der Dokumentation schärfte den Unterschied zwischen Sach- und Darstellungsmethoden (setzeText, zeigeText). Der Dokumentationsüberblick im Werkzeug wurde als (Selbst-) Kontrollmöglichkeit genutzt.

6. Schritt: Differenzierung/Charakterisierung der Beziehungen

Gestaltung der Beziehung zum Stift als Hat-Beziehung.

Erfahrungen: Schwierigkeiten entstanden dadurch, dass ‚hat Stift‘ nicht in typische hat-Schemata hineinpasst (wie „Das Ganze und seine Teile“).

7. Schritt: Kriterien für eine gute Dokumentation

Von Schülerinnen und Schüler erstellte Dokumentationen werden mit anonymisierten Bezeichnungen analysiert und verglichen.

Erfahrungen: Einige Kriterien wie Vollständigkeit, Sachlichkeit und Ausführlichkeit konnten gut herausgearbeitet werden, andere wie Korrektheit oder Prägnanz nur mit Hilfe.

8. Schritt: Zugriffsmethoden

Identifikation von Zugriffsmethoden im Modell und Untersuchung auf notwendige Zugriffsmethoden (mit Hilfe eines Arbeitsblattes).

Erfahrungen: Deutliche Schwierigkeiten bei der Formulierung solcher Methoden und auch bei deren (späterer) Implementation. Hinweise und schriftliche Informationen konnten von den Schülerinnen und Schüler nicht befriedigend genutzt werden.

9. Schritt: Konstruktoren und Destruktoren

Aufgabe dieser Methoden kennen lernen, Parameter und Realisation bedenken.

Rückgriff auf Startsituation bei Objektspiel liefert Vorstellung von Aufbau des Konstruktors (mit Hilfe eines Arbeitsblatts).

Erfahrungen: Grundlegende Fragen konnten gut herausgearbeitet werden, ansonsten analoge Schwierigkeiten wie bei Zugriffsmethoden. Aufgrund dieser Schwierigkeiten im 8. und 9. Schritt ist zu bedenken, inwiefern Zugriffsmethoden und Konstruktoren/ Destruktoren zumindest teilautomatisiert vom CASE-Tool zur Verfügung gestellt werden sollten. Ein didaktischer Verlust wäre darin m. E. nicht erkennbar.

Implementationsphase

10. Schritt: Implementation

In dieser Phase werden die einzelnen Klassen *arbeitsteilig* auf der Grundlage des generierten Codes mit Leben gefüllt (Methodendefinitionen). Eine Projektkommunikation mit einem geschlossenem Forum hält alle Beteiligten auf dem Laufendem, z.B. über Änderungen an der Dokumentation einer Klasse. Eine Arbeitsprozessdokumentation ist Grundlage für den späteren Austausch der Erfahrungen (wie z.B. neues Konzept der lokalen Variable oder von Zufallszahlen).

Erfahrungen: Auch Modelländerungen können von den Schülerinnen und Schülern selbstständig im Code vorgenommen werden (Codegenerierung mindert also nicht diese Fähigkeit). Allerdings macht sich fehlende Programmiersicherheit bemerkbar durch häufige Nachfragen von Seiten der Lernenden. Die Arbeitsprozessdokumentation und die anschließende Darstellung in der Gruppe gelingen nicht in allen Gruppen. Weitere Übungsaufgaben zu den neuen Konzepten aus den einzelnen Gruppenarbeiten müssen das Projekt ergänzen.

Insgesamt haben die Erfahrungen gezeigt, dass die Schülerinnen und Schüler auch in der Jahrgangsstufe 11 bereits frühzeitig mit Modellierungen und Modellierungstechniken umgehen können. Ein Teil der Schwierigkeiten war darauf zurückzuführen, dass die vorangehende Phase etwas knapp ausgefallen war. Auch Verbesserungen im CASE-Tool könnten bei einzelnen Schwierigkeiten Abhilfe schaffen.

Das CASE-Tool hat sich im Unterricht als unauffälliger Helfer erwiesen und wurde von den Schülerinnen und Schüler schnell angenommen und selbstverständlich eingesetzt.

Eine ausführlichere Darstellung der Planungen, des Ablaufs und der Erfahrungen aus der Erprobung findet sich in [Mo02].

Literatur

- [Ba99] Balzert, H.: Lehrbuch der Objektmodellierung. Analyse und Entwurf. Spektrum, Heidelberg, 1999.
- [Bu00] Bergin, J.: The Object Game. An Exercise for Studying Objects. 2000. URL: <http://www.wol.pace.edu/%7ebergin/patterns/objectgame.html>
- [Hu00] Hubwieser, P.: Didaktik der Informatik. Grundlagen, Konzepte, Beispiele. Springer, Berlin, 2000.
- [Le96] Leufen, S.: Ansätze zur Bewertung von Unterrichtssoftware. In: Bertelsmann Stiftung, Heinz Nixdorf Stiftung (Hrsg.): Neue Medien in den Schulen. Projekte – Konzepte – Kompetenzen. Bertelsmann Stiftung, Gütersloh, 1996; S. 55-72.
- [LSW99a] LSW (Hrsg.): Lernen mit neuen Medien: Grundlagen und Verfahren der Prüfung Neuer Medien, 4. erweiterte und überarbeitete Auflage. Kettler, Bönen, 1999.
- [LSW99b] LSW (Hrsg.): Von Stiften und Mäusen. Eine Einführung in die Grundlagen der objektorientierten Programmierung. Kettler, Bönen, 1999.
- [LSW01] LSW (Hrsg.): Projektbeispiel „Pfeilwurf“. LSW, Soest, 2001. URL: <http://www.learnline.de/angebote/oop/medio/kurs/einfuehrung/pfeilwurf/index.html>
- [Mo02] Moll, S.: Modellierung als zentrales Element des Informatikunterrichts. (Schriftliche Hausarbeit im Rahmen der zweiten Staatsprüfung). Studienseminar Sekundarstufe II Münster II, Münster, 2002.
- [MSWWF99] MSWWF (Hrsg.): Richtlinien und Lehrpläne für die Sekundarstufe II - Gymnasium/Gesamtschule in Nordrhein-Westfalen. Informatik. Ritterbach, Frechen, 1999.
- [RS01] Reinsch, T.; Schulte, C.: Objektorientierung im Anfangsunterricht. Arbeitspapier. Universität Paderborn, 2001.
- [Sp00] Spolwig, S.: Modellieren und Programmieren. In: LOG IN, 18 (2000) Heft 2, S. 53-59.
- [Sp01] Spolwig, S.: Methodische Probleme mit OOP im Anfangsunterricht. 2001, URL: http://www.oszhd.be.schule.de/gymnasium/faecher/informatik/aktuellekurse/spolwig/unterricht_oop/oop_probleme.htm