Objektorientierte Software-Entwicklung als Lehrinhalt
Anregungen und Erfahrungen

Horst Lichter
Institut für Informatik
Universitaet Stuttgart
Breitwiesenstr. 20-22
7000 Stuttgart 80

Heinz Züllighoven
Fachbereich Informatik
Universität Hamburg
Vogt-Koelln-Str 30
2000 Hamburg 54

1. Einleitung

Die objektorientierte Programmierung ist, da entsprechende Programmiersprachen (Simula-67, Smalltalk-80) seit mehreren Jahren zur Verfügung stehen, ein Lehrinhalt in der Informatikausbildung. Aber erst in den letzten Jahren werden Lehrveranstaltungen angeboten, die nicht nur den Aspekt der Codierung, sondern die gesamte objektorientierte Software-Entwicklung zum Inhalt haben.

Dies ist notwendig, weil die objektorientierte Programmierung nicht losgelöst von den anderen Tätigkeiten der Software-Entwicklung, wie Analyse oder Entwurf, behandelt werden kann. Daß diese Tätigkeiten aufeinander abgestimmt sein müssen, scheint unbestritten, da so erst die Vorteile des objektorientierten Ansatzes voll genutzt werden können. Vorrangig ist hier zu nennen, daß zwischen den Modellen für die Analyse des Anwendungsbereichs und den Modellen für die Softwarekonstruktion kein prinzipieller Bruch mehr besteht. Da jedoch noch wenige Methoden für die Konstruktion von objektorientierter Software existieren und publiziert sind, für spezielle Tätigkeiten, wie für das Dokumentieren oder Testen objektorientierter Programme, kaum gesichertes Wissen existiert, ist die Gestaltung der Inhalte einer Veranstaltung "Objektorientierte Software-Entwicklung" überwiegend offen und schwierig. Hinzu kommt, daß die objektorientierte Konstruktion bewährte Techniken der Software-Entwicklung wie Datenabstraktion oder das Geheimnisprinzip integriert und diese daher als zusätzliche oder vorausgesetzte Inhalte hinzukommen.

2. Anregungen und Erfahrungen

Wir möchten nachfolgend kurz versuchen, Themenschwerpunkte, Probleme und Anregungen für einen Lehrinhalt "Objektorientierte Software-Entwicklung" vorzustellen. Unser Aussagen basieren auf den Erfahrungen, die wir durch eigene Lehrveranstaltungen an der Hochschule, aber auch durch Industrie-Seminare gewonnen haben.

- Die Umstellung von der prozeduralen zur objektorientierten Denkweise braucht Zeit. Dieser sogenannte Paradigmenwechsel ist am Ende einer einzigen Lehrveranstaltung sicher noch nicht vollzogen. Es bedarf dazu praktischer Erfahrung und Anleitung; das Erlernen einer weiteren Programmiersprache oder Darstellungstechnik genügt nicht.

In: Ludewig, J., Schneider, K. (eds.) Software Engineering im Unterricht der Hochschulen – Studienführer Software Engineering, Berichte des German Chapter of the ACM 37, B.G. Teubner Stuttgart, pp 26 – 33

- Objektorientierte Software-Entwicklung sollte sowohl die Analyse und Modellierung eines Anwendungsbereichs als auch die konstruktive Umsetzung in einen Systementwurf und seine Realisierung umfassen. Dies geht im Regelfall über den Rahmen einer einsemestrigen Veranstaltung hinaus.
- Von zentraler Bedeutung der bei objektorientierten Analyse ist die Begriffsbildung in der betrachteten Anwendungswelt und die Strukturierung der Begriffe mithilfe der Spezialisierung und Generaliserung. Dies kann nur anhand von größeren praxisrelevanten Beispielen vermittelt werden. Unterbelibt dies, so kommen oft Entwürfe zustande, die schlecht strukturiert und unverständlich sind.
- Die sinnvolle Anwendung der Vererbung, als dem wesentlichen Modellierungsmechanismus, ist ein zentraler Aspekt im Bereich des objektorientierten Entwurfs. Die beiden Einsatzmöglichkeiten der Vererbung, erstens zur Spezialisierung und zweitens zur Wiederverwendung von Code, müssen aufgezeigt und diskutiert werden. Diese Diskussion ist besonders unter dem Aspekt der Mehrfachvererbung wichtig. Die Beispiele in der einschlägigen Literatur sind nicht immer hilfreich.
- Die genannten Probleme deuten darauf hin, daß eine objektorientierte Notation kein Garant für qualitativ hochwertige Software ist. Unsere Erfahrung zeigt, daß Leitmotive oder verständliche Metaphern für den Software-Entwurf dringend benötigt werden. Bewährt haben sich das Konzept des elektronischen Schreibtisches oder die Metaphern von Werkzeugen und Materialien. Im Bereich CSCW, d.h. für kooperative, arbeitsteilige Anwendungen stehen ähnlich griffige Metaphern noch aus.
- Damit die objektorientierte Konstruktion von Software vermittelt und umzusetzt werden kann, wird eine geeignete Notation benötigt. Eiffel ist eine geeignete Lehrsprache, da sie die objektorientierten Konzepte in sinnvoller Weise mit Aspekten des Software Engineering (z.B. Typisierung, Programming-by-Contracting) verbindet. Allerdings scheint Eiffel gegenwärtig für die professionelle Software-Entwicklung im industriellen Bereich noch nicht geeignet.
- Auch in der universitären Lehre sollte ein zyklisches und evolutionäres Prinzip des Software-Entwurfs vermittelt werden, daß analytische, konstruktive und expermientell bewertende Aktivitäten miteinander verbindet. In diesem Rahmen kommt der Arbeit mit einer ausgereiften Entwicklungsumgebung und dem Bau von Prototypen besondere Bedeutung zu. Hier läßt sich allenfalls das Smalltalk-System nennen. Für C++ oder Eiffel sind diese Forderungen nicht erfüllt.

3. Objektorientierte Software-Entwicklung als Inhalt von Lehrveranstaltungen

3.1 Beispiel 1: Universität Stuttgart

An der Universität Stuttgart werden verschiedene Veranstaltungen angeboten, die ganz oder zumindest teilweise die objektorientierte Software-Entwicklung zum Inhalt haben. Sie lassen sich in zwei Gruppen einteilen.

A. Vorlesungen

"Nichtprozedurale Programmierung"

Merkmale: angeboten für das Hauptstudium, 2 V, keine Übungen, mündliche Prüfung

Inhalte: Die Vorlesung erläutert die objektorientierte, die funktionale und die wissensbasierte Programmierung. Sie ist eine Grundlagen-Vorlesung. Im ersten Drittel werden die Konzepte der objektorientierten Programmierung vorgestellt. Es werden die zentralen Begriffe (Objekt, Klasse, Vererbung, Nachricht) definiert und das Berechnungsmodell erläutert. Zur Notation für die gezeigten Beispiele wird Smalltalk-80 verwendet.

Erfahrungen: Der Zweck dieser Vorlesung besteht darin, einen ersten Einblick in die verschiedenen Arten der nichtprozeduralen Programmierung zu geben. Da keine Übungen vorgesehen sind und nur wenige Vorlesungseinheiten für jedes Thema zur Verfügung stehen, können keine vertieften Kenntnisse vermittelt werden. Die Veranstaltung schafft jedoch eine ausreichende Grundlage für eine spätere Vertiefung in einem der behandelten Themengebiete.

"Konzeption und Aufbau objektorientierter Programme"

Merkmale: angeboten für das Hauptstudium, 2 V, 2 Ü, mündliche Prüfung

Inhalte: Die Veranstaltung vermittelt die speziellen Aspekte der objektorientierten Software-Entwicklung. Zu diesem Zweck wird sie in drei Aktivitäten aufgeteilt, die parallel ablaufen.

- Vorlesung: Hier werden die theoretischen Aspekte vermittelt: Begriffe, Konzepte, oo Analyse, oo Entwurf, Dokumentation oo Programme, Testen oo Programme.
- Übungen: Die Studenten bearbeiten parallel zur Vorlesung ein kleines Projekt in Gruppen. Sie durchlaufen dabei die Entwicklungsphasen Analyse, Entwurf und Implementierung. Am Ende jeder Phase werden die erzielten Ergebnisse überprüft. Das Projekt wird mit der Übergabe an die betreuenden Assistenten abgeschlossen. Zur Implementierung werden die Sprachen Objective-C, Eiffel und Smalltalk-80 verwendet.

In: Ludewig, J., Schneider, K. (eds.) Software Engineering im Unterricht der Hochschulen – Studienführer Software Engineering, Berichte des German Chapter of the ACM 37, B.G. Teubner Stuttgart, pp 26 – 33-

- Studentische Vorträge: Jede der Gruppen erarbeitet einen Kurzvortrag mit einer schriftlichen Zusammenfassung zu einem Thema aus dem Bereich der oo Software-Entwicklung. Dadurch werden allgemeine Themen (z.B. kurze Charkterisierungen von Sprachen wie Oberon-2, BETA usw.) vorgestellt, die in der Vorlesung nicht behandelt werden.

Erfahrungen: Die Veranstaltung wurde in SS 90 zum erstenmal durchgeführt und wird im kommenden SS wiederholt. Folgende Erfahrungen liegen vor:

- Das vorlesungsbegleitende Projekt ist geeigneter als unabhängige Übungen, weil die Zusammenhänge zwischen den einzelnen Tätigkeiten bei der Entwicklung besser klar gemacht werden können und die Probleme dabei deutlicher werden.
- Der Zeitraum für das Projekt ist etwas knapp, zum Teil werden die Lehrinhalte, die für die Projektarbeit notwendig sind, erst zu spät behandelt. Eine Aufteilung in Vorlesung mit anschießender Projektarbeit in den Semesterferien wäre sinnvoller.
- Der Aufwand für die Studenten ist sehr hoch, jedoch haben sie nach eigenen Aussagen auch einer höheren Nutzen als bei anderen Veranstaltungen.

B. Programmierkurse

"Kompaktkurs Eiffel" und **"Kompaktkurs Smalltalk-80"**

Merkmale: 2 V, mit begleitenden oder abschließenden Übungen

Inhalte: Die beiden Veranstaltungen haben zum Ziel, das programmiertechnische Rüstzeug, das die beiden Sprachen anbieten, zu vermitteln. Zu den Vorlesungen, die die Sprachkonstrukte und deren Einsatz aufzeigen, werden Übungen angeboten, die bearbeitet und abgegeben werden müssen.

Erfahrungen: Die Programmierkurse werden hauptsächlich von Studenten besucht, die für die Programmierung in ihrer Studien- oder Diplomarbeit eine dieser Sprachen verwenden müssen. Der Besuch des entsprechenden Programmierkurses ist in diesen Fällen sinnvoll und wird in vielen Fällen auch vorausgesetzt, damit eine Studien- oder Diplomarbeit begonnen werden kann.

3.2 Beispiel 2: Universität Hamburg

An der Universität Hamburg ist der Arbeitsbereich Softwaretechnik erst zum WS91/92 eingerichtet worden. Dadurch können keine großen Erfahrungen berichtet werden. Allerdings ist als Ausgangssituation bemerkenswert, daß die Studenten sehr wenig Erfahrung mit Software-Entwicklung allgemein haben.

"Objektorientierter Systementwurf"

Merkmale: angeboten für das Hauptstudium, 2 V, 2 Ü, keine Prüfung oder Rücksprache

Inhalte: Diese Veranstaltung behandelt objektorientierte Analyse und Entwurf ohne objektorientierte Programmierung. Sie ist als Grundlage für eine sog. Projektveranstaltung "Objektorientierte Programmierung in Eiffel" im Folgesemester. Themen der Lehrveranstaltung sind zum einen Grundbegriffe wie Objekt, Klasse, Vererbung und Typisierung aber auch das Entwurfsprinzip nach der Metapher von Werkzeug und Material und eine Skizze einer objektorientierten Entwicklungsstrategie. Zur Notation für die gezeigten Beispiele wird "Pseudo"-Eiffel verwendet.

Erfahrungen: Die Lehrveranstaltung will den genannten Paradigmenwechsel auslösen, d.h. hier soll im wesentlichen eine neue Sichtweise vermittelt werden. Auf fruchtbaren Boden fallen diese Gedanken vor allem bei solchen Studenten, die bereits umfangreichere Entwicklungserfahrung haben — am besten in einer kommerziellen Umgebung. Wo Grundkenntnisse wie abstrakte Datentypen fehlen, wird es mühsam.

3.3 Beispiel 3: TU Berlin

An der TU Berlin wurden im Bereich Softwaretechnik bis zum SS91 einige objektorientierte Lehrveranstaltungen im Hauptstudium angeboten, die sich ihrerseits auf softwaretechnische Grundveranstaltungen beziehen konnten. Beispielhaft seien genannt:

“Objektorientierte Systementwicklung mit Eiffel”

Merkmale: angeboten für das Hauptstudium, 2 V, 2 Ü, wahlweise Prüfung oder Rücksprache

Inhalte: Diese Vorlesung kombinierte objektorientierte Analyse und Entwurf mit objektorientierter Programmierung. Themen der Lehrveranstaltung waren ebenfalls Grundbegriffe und Entwurfsprinzipien. Die programmiersprachlichen Kenntnisse wurden in den Übungsgruppen vermittelt und erprobt. Als Beispiel wurden Teile der Fuhrparkverwaltung und der Kundenkartei eines fiktiven Versicherungsunternehmens modelliert und implementiert. Einzelne Systemkomponenten wie interaktive Bausteine oder elementare Klassen wurden vorgegeben.

Erfahrungen: Die Lehrveranstaltung erforderte einen hohen Arbeitsaufwand bei Studenten und Betreuern. Wo die Bereitschaft bestand, dies zu leisten, waren die Erfolge sehr gut. Auch hier waren Vorkenntnisse in der Systemkonstruktion, evtl. sogar mit einer objektorientierten Sprache extrem hilfreich.

“Objektorientierte Entwicklung interaktiver Werkzeuge mit Motif”

Merkmale: angeboten für das Hauptstudium, 2 V, 2 Ü, wahlweise Prüfung oder Rücksprache

Inhalte: Diese Vorlesung kombinierte den objektorientierte Entwurf interaktiver Software mit der Konstruktion in Motif. Themen der Lehrveranstaltung waren neben Grundbegriffen die Entwurfsprinzipien für interaktive Anwendungssysteme. Die Kenntnisse über X-Windows und OSF Motif wurden in den Übungsgruppen vermittelt und erprobt. Als Beispiel wurden Teile eines Bibliographiesystems entwickelt.

Erfahrungen: Die Lehrveranstaltung erforderte einen Arbeitsaufwand, der von den Studenten im Regelfall nicht bewältigt wurde. Wo Vorkenntnisse in der Konstruktion interaktiver Systeme und Programmierkenntnisse in C und X-Windows vorhanden waren, konnten zumindest einfache Prototypen erstellt werden. Motif in der Kombination mit C oder C++ erwies sich als ungeeignet — zumindest für eine einsemestrige Lehrveranstaltung.

4. Objektorientierte Entwicklung in der Lehre: Quo vadis?

Daß im Rahmen der Informatikausbildung die objektorientierte Software-Konstruktion vermittelt werden muß, ist unbestritten. Es muß jedoch diskutiert weden, wann und wie dies geschehen kann.

> *Soll der Übergang von der prozeduraler zur objektorientierten Denkweise bereits im Grundstudium stattfinden?*

Wir möchten dazu die folgenden Denkanstöße geben:

- Eine mögliche Vorgehensweise könnte die Vermittlung von CLU im Grundstudium sein, wobei der Schwerpunkt beim Entwurf und bei der Programmierung abstrakter Datentypen sein sollte. Wir haben die Befürchtung, daß der unmittelbare Einsatz einer objektorientierten Sprache zu einer andauernden Verwirrung bei den Konzepten “inherit” und “use” führt.
- Eine Alternative dazu könnte die initiale Beschäftigung mit Prolog sein - also weg von einer prozeduralen Sichtweise. Prolog bietet zudem die Vorteile einer interpretierenden Sprache mit generischer IO, was schnelle Entwicklungszyklen begünstigt.
- Im Hauptstudium sollten Konzepte und Praxis der objektorientierten Entwicklung zunächst mit einer Sprache wie Eiffel vermittelt werden. Darauf aufbauend könnten sich Veranstaltungen zu C++ und Motif, wegen ihrer industriellen Bedeutung, aber auch Smalltalk-80, wegen seiner Bedeutung für das Prototyping anschließen. Auf dieser Grundlage lassen sich dann auch Konzepte wie Hypertext und Systeme wie 4th Dimension einordnen, ohne das ein Absturz ins "Durchwursteln" zu befürchten ist.