Bestimmung von Objektkandidaten mit Hilfe von Formaler Begriffsanalyse

Stephan Düwel und Wolfgang Hesse, FB Mathematik/Informatik, Univ. Marburg

1 Einleitung und Zielsetzung

Die Objektorientierung hat sich als Software-Entwicklungsprinzip von der Programmierung (OOP) über den Entwurf (OOD) und die Analyse (OOA) über den gesamten Software-Prozeß ausgeweitet. Heute wird sie als durchgängiges Paradigma anerkannt, mit dem es gelingen kann, methodische und terminologische Brüche an den Phasenübergängen zu vermeiden.

Erster Schritt der objektorientierten SW-Entwicklung ist die objektorientierte Analyse. Oberstes Ziel der OOA ist es, anhand eines objektorientierten Modells zu beschreiben, was das zu entwickelnde SW-System leisten soll. Mögliche Beschreibungsmittel in einem solchen Modell sind Klassendiagramme, Interaktionsdiagramme, Zustandsübergangsdiagramme, etc. Zentrale Elemente der Modellierung sind Objekte und Klassen. Am Anfang eine gute Wahl der Objekte zu treffen, ist besonders wichtig, weil sie (mit Einschränkungen) bis zum Ende der Entwicklung eine tragende Rolle spielen sollen. Wie findet man aber die richtigen Objekte?

Die Auffassung, daß das Denken und Leben sich in Objekten abspiele und daß die objektorientierte Software-Entwicklung damit ganz natürlich sei, wird immer wieder geäußert. Nach B. Meyer stellt das Auffinden der Objekte kein Problem dar, weil die Welt ja aus Objekten bestünde ([Mey 88] S. 51: "... in the physical or abstract reality being modeled, the objects are just there for the picking!"). Man muß dem aber entgegenhalten, daß diese Sicht sehr vereinfachend ist und spätestens dann versagt, wenn die Anwendung neuartig, sehr umfangreich oder komplex ist.

Andere Autoren schlagen zum Auffinden der Objekte in der OOA eine grammatikalische Analyse der Anforderungsdefinition vor (z.B. [WWW 90]). Kandidaten für Klassen seien die Substantive, für Attribute die Adjektive und für Operationen die Verben. Dies kann aber leicht in Irrwege führen, weil man z.B. in der deutschen Sprache (speziell im "Bürokratendeutsch") oft in gehäufter Anzahl substantivierte Verben antrifft. Außerdem ordnet man mögliche Kandidaten falsch oder ungerechtfertigt ein, weil sie etwa nur Füllfunktion innerhalb eines Satzes haben oder metaphorisch gebraucht werden.

Es ist unbestritten, daß mit jeder Wahl der Objekte, die zur Beschreibung eines Systems herangezogen werden sollen, bereits eine Entwurfsentscheidung getroffen wird. Der Erfolg dieser Entscheidung hängt von vielen Faktoren ab, dazu gehören die Erfahrung des Entwicklungsteams und das Systemumfeld. So geben viele Autoren von objektorientierten Entwicklungsmethoden (Check-) Listen von Möglichkeiten an, wo, wie und als was Objekte eines Systems gefunden werden könnten (z.B.: [Boo 94], [C-Y 90], [S-M 91]). Sie geben aber kein explizites Verfahren zur Objektbestimmung an. Das beabsichtigen wir auch nicht, aber wir wollen mit der Formalen Begriffsanalyse ein Werkzeug vorstellen, das besonders im Frühstadium der Analyse zur Objektauswahl eingesetzt werden kann.

Die Verfahren der grammatikalischen Analyse des Pflichtenhefts stoßen auch dann an ihre Grenzen, wenn es darum geht, den Kontext einer SW-Entwicklung zu betrachten. Heute ist man oft bestrebt, den Auftraggeber so früh wie möglich in die Analyse (und damit in die Entwicklung) einzubinden. Hier soll ein Diskurs zwischen Fachexperten und Entwickler stattfinden. U. Andelfinger weist in [And 96] in diesem Zusammenhang auf das Problem der „pragmatischen Lücke“ hin. Fachexperten und SW-Entwickler sprechen nicht die gleiche Sprache. Die Dokumente der OOA sollten in der Sprache der Fachexperten verfaßt sein, weil sie als Auftraggeber das entwickelte Modell beurteilen müssen.

Deshalb ist Entwicklungsarbeit ein Stück weit begriffliche Arbeit, gerade wenn es um das Auffinden der tragenden Elemente eines Modells geht. Bei Martin und Odell finden sich bereits ähnliche Ideen [M-O 92]. *Formale Begriffsanalyse* ist eine mathematische Theorie, die begriffliches Denken mit Hilfsmitteln der Mathematik (Verbandstheorie) zu unterstützen versucht. In Abschnitt 3 wird an einem kurzen einführenden Beispiel verdeutlicht, wie man Formale Begriffsanalyse zur Bestimmung von Objektkandidaten innerhalb der objektorientierten Analyse einsetzen kann. Vorher werden die wichtigsten Grundzüge der Formalen Begriffsanalyse erläutert.

2 Formale Begriffsanalyse (FBA)

Die *Formale Begriffsanalyse* (kurz: *FBA*, vgl. [W-G 96]) behandelt Begriffe als Paare aus Begriffsumfang und -inhalt. Ausgangspunkt sind eine Menge G von Gegenständen und eine Menge M von Merkmalen. Den Gegenständen können Elemente aus M als Merkmale zugesprochen werden. Das ergibt eine Inzidenzrelation I, die beschreibt, welcher Gegenstand welches Merkmal trägt. Das Tupel (G,M,I) bezeichnet man als *formalen Kontext*. Beschrieben wird er durch eine Tabelle (vgl. Abb. 1).

	Fußball	Klavierspielen	Radfahren	Schwimmen	Töpfern
Hans	X		X	X	
Eva			X	X	X
Karl	X			X	
Anna		X			X

Abb. 1: Beispiel eines formalen Kontexts

In diesem Beispiel sind Personen und ihre Hobbies aufgeführt. Die Personen werden als Gegenstände und ihre Hobbies als Merkmale behandelt. Die Inzidenzrelation ist durch die Kreuze in der Tabelle gegeben und gibt an, wer welches Hobby ausübt.

Ein *formaler Begriff* eines Kontexts (G,M,I) ist ein Paar (A,B) mit $A \subseteq G$, $B \subseteq M$ und:
$A = \{ g \in G \mid g\ I\ m\ \forall m \in B \}$, $B = \{ m \in M \mid g\ I\ m\ \forall g \in A \}$, d.h. jeder Gegenstand aus A trägt alle Merkmale aus B und jedes Merkmal aus B trifft auf alle Gegenstände aus A zu.
A ist der *Umfang* und B der *Inhalt* des formalen Begriffs (A,B).

Auf der Menge aller formalen Begriffe eines formalen Kontexts betrachtet man die Ober-/Unterbegriff-Ordnung. Ein formaler Begriff (A_1,B_1) ist *Unterbegriff* eines formalen Begriffs (A_2,B_2), wenn $A_1 \subseteq A_2$ gilt, d.h. alle Gegenstände, die zum Unterbegriff (A_1,B_1) gehören, auch unter den *Oberbegriff* (A_2,B_2) fallen (Äquivalent ist die Bedingung $B_1 \supseteq B_2$.). Diese Relation ist eine mathematische Ordnungsrelation und mit ihr ist die Menge der formalen Begriffe ein vollständiger Verband, der sog. *Begriffsverband*. Als geordnete Menge kann man diesen Begriffsverband in einem Liniendiagramm darstellen. Zu dem Kontext in Abb. 1 ergibt sich folgendes Liniendiagramm:

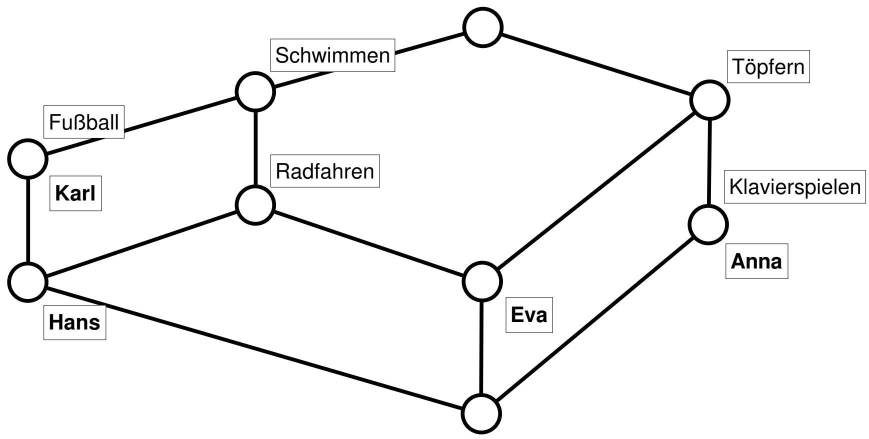

Abb. 2: Liniendiagramm zum Beispiel aus Abb. 1

Die Knoten des Liniendiagramms stellen die Begriffe des Begriffsverbands dar. Die Ober-/Unterbegriff-Ordnung ist dergestalt wiedergegeben, daß vom Knoten eines Unterbegriffs zu den Knoten seiner Oberbegriffe aufsteigende Linienzüge führen. Aus Gründen der Übersichtlichkeit sind nicht jeweils der Umfang und der Inhalt der Begriffe angegeben. Vielmehr reicht es, Gegenstände an dem untersten Knoten, unter dessen Begriff sie fallen, und Merkmale an dem obersten Knoten, dessen Begriff sie kennzeichnen, zu notieren. Dabei werden Gegenstände etwas unterhalb und Merkmale etwas oberhalb der Knoten angegeben. Des weiteren sind hier die Gegenstände fett gedruckt.

Das Liniendiagramm enthält die volle Information des formalen Kontexts. Sucht man zu einem Gegenstand die zutreffenden Merkmale, verfolgt man vom Knoten, an dem der Gegenstand notiert ist, alle aufsteigende Linienzüge. Die an ihnen auftauchenden Merkmale sind die auf den Gegenstand zutreffenden. Genau umgekehrt findet man durch die absteigenden Linienzüge zu einem Merkmal alle Gegenstände, die es tragen.

Aus dem obigen Diagramm kann man z.B. ablesen, daß Eva sich mit Schwimmen, Radfahren und Töpfern beschäftigt und Fußball von Karl und Hans gespielt wird. An der Struktur des Liniendiagramms erkennt man weitergehend eine mögliche Klassifizierung in „Sportler“ und „Künstler“. Diese Aufteilung wird durch die zwei Knoten direkt unterhalb des obersten Knotens suggeriert. Dem mit „Schwimmen“ beschriebenen Knoten folgen nach unten nur sportliche Hobbies als Merkmalsangaben an Knoten. Der entsprechende Begriff ist „Sportler“. Unter ihn fallen Karl, Hans und Eva. Dagegen folgt dem mit „Töpfern“ beschriebenen Knoten nach unten nur „Klavierspielen“ als weitere Merkmalsangabe. Als Bezeichnung für den Begriff dieses Knotens könnte man „Künstler“ wählen. Zu diesem Begriff gehören Eva und Anna. Dieses kleine Beispiel veranschaulicht schon, wie durch die FBA Sachverhalte strukturiert dargestellt und mögliche Begriffe entwickelt werden können.

3 Beispiel: Ein Recycling-Automat

Generell kann man zwei Strategien für das Suchen nach passenden Objekten unterscheiden:

- *Top down:* Man beginnt mit der Untersuchung des Systemumfelds, d.h. des Fachgebiets, zu dem die zur Debatte stehende Anwendung gehört (*domain analysis* oder Bereichsanalyse).

- *Bottom up:* Man beginnt mit der Untersuchung der speziellen Anwendung.

Aus Gründen der Wiederverwendbarkeit könnte man die *top down*-Strategie bevorzugen. Doch wo soll man anfangen? Wie stellt man sicher, sein Ziel nicht aus den Augen zu verlieren? Primäres Ziel ist es, daß das fertige System die Anforderungen des Auftraggebers erfüllt und nicht, daß die Entwickler in anderen Projekten viele ihrer Klassen wiederverwenden können.

Deswegen wählen wir eine bottom up-Strategie: Ausgangspunkt der Analyse ist also das gewünschte Verhalten des Systems. Ein Beschreibungsmittel für dieses Verhalten, das sich heute wachsender Popularität erfreut, sind die von I. Jacobson eingeführten *Use Cases.* Dabei handelt es sich um typische Anwendungsfälle, die von einem „Aktor“ außerhalb des Systems angestoßen werden und zu bestimmten beispielhaften Systemdurchläufen führen. Ein Beispiel dafür aus [Jac92] zeigt Abb. 3.

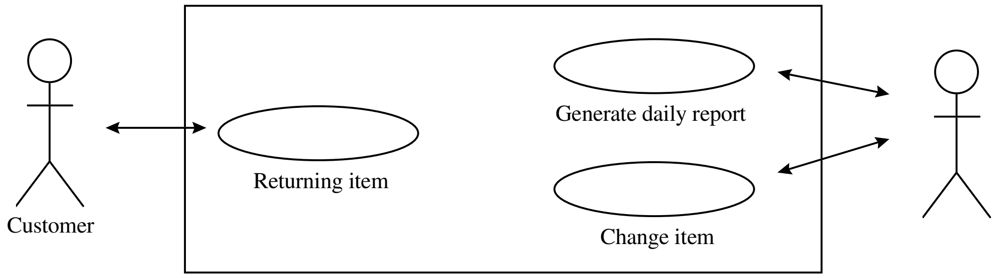

Abb. 3: Der Recycling Automat von I. Jacobson

Die Anwendungsfälle (*use cases*) werden in einem solchen Diagrammen aufgelistet. Dann müssen sie durch die Fachexperten beschrieben werden:

Returning item is started by *Customer* when she wants to return cans, bottles or crates. With each item that *Customer* places in the recycling machine, the system will increase the received number of items from *Customer* as well as the daily total of this particular type. When *Customer* has deposited all her items, she will press the receipt button to get a receipt on which the returned items have been printed as well as the total return sum.

Generate Daily Report is started by *Operator* when he wants to print out information about the deposit items returned during the day. The system will print out how many of each deposit item type have been received, as well as the overall total for the day. The total number will be reset to zero to start a new daily report.

Change Item is used by *Operator* to change information in the system. The return value as well as the size of each returnable item can be changed, and new types of item can be added.

Um aus den Beschreibungen der Anwendungsfälle mit Hilfe der FBA Kandidaten für Objekte und Klassen zu gewinnen, führt man mit Hilfe der Fachexperten eine Indizierung der Anwendungsfälle durch. Zu jedem Anwendungsfall werden alle Dinge aufgelistet, die bei ihm eine Rolle spielen. Das sind Kandidaten für Objekte, Klassen, Attribute, Operationen und Rollen im späteren Klassenmodell. Für die Verarbeitung mit FBA sind die Anwendungsfälle die Gegenstände und die bei der Indizierung gewählten Dinge die Merkmale (im Sinne der FBA-Terminologie). Die Inzidenzrelation I gibt an, ob ein als Merkmal aufgeführtes „Ding“ bei einem Anwendungsfall eine Rolle spielt. Bei der Bearbeitung des Beispiels ergab sich der folgende formale Kontext:

	customer	can	bottle	crate	item	received number of items	daily total	receipt button	receipt	total return sum	operator	day	daily overall total	daily report	return value	size
Returning item	×	×	×	×	×	×	×	×	×	×						
Generate daily report					×		×				×	×	×	×		
Change item					×						×				×	×

Abb. 4: Formaler Kontext zum Recycling-Automat

Dieser formale Kontext gibt die Indizierung der Anwendungsfälle wieder. Dabei unterliegt die Auswahl der für einen Anwendungsfall wichtigen Dinge dem Entwickler. Z.B. fehlt oben beim Anwendungsfall ‘Returning item’ das grundlegende Ding ‘recycling machine’. ‘recycling machine’ ist gerade das zu modellierende System. Entwurfsentscheidung war hier, das System selber nicht als Objekt zu modellieren.

Ein Problem bei der grammatikalischen Analyse sind Synonyme. Im obigen Beispiel sind die Synonyme ‘item’, ‘deposit item’, ‘deposit item type’ und ‘type of item’ durch ‘item’ ausgedrückt (Bei der Modellierung von Klassen wird hier nicht zwischen „items“ und ihrem Typ unterschieden.). Diese Entwurfsentscheidung wurde getroffen, nachdem bei einer vorausgegangenen anderen Indizierung der Anwendungsfälle und Betrachtung des entstandenen Liniendiagramms die Synonyme identifiziert wurden.

Abb. 5 zeigt das Liniendiagramm zum formalen Kontext aus Abb. 4:

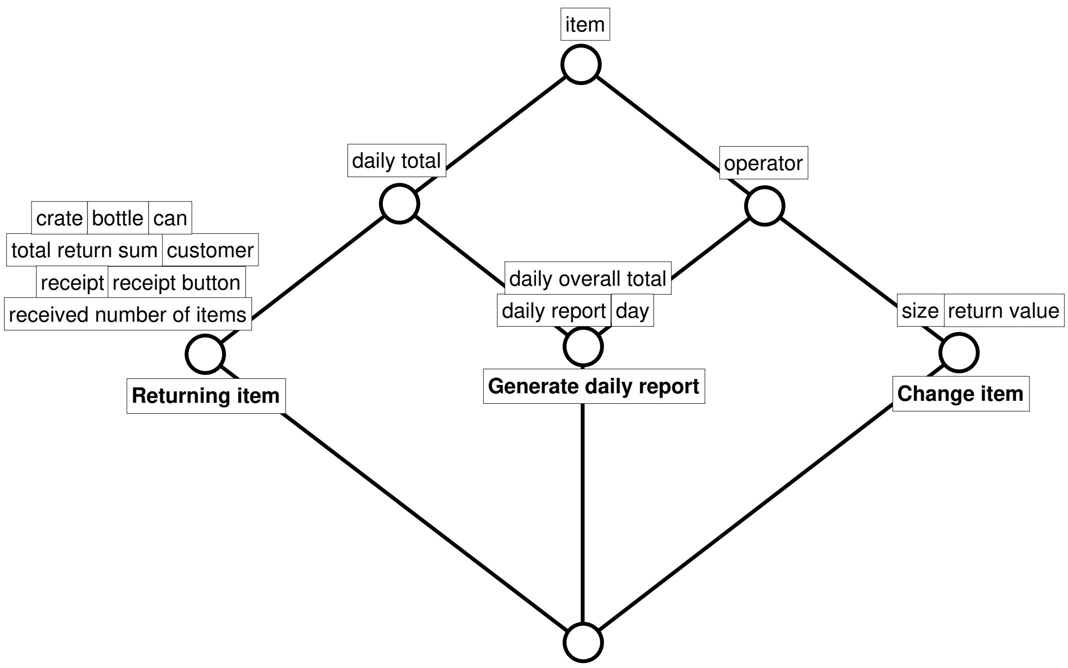

Abb. 5: Liniendiagramm zum Recycling-Automat

Im entstandenen Liniendiagramm stehen die Dinge oben, die für viele Anwendungsfälle interessant sind. Dies sind die "heißesten" Klassenkandidaten, die systemübergreifend für den ganzen Arbeitsbereich entscheidend sind. Nachdem man Klassenkandidaten für die eigentliche Anwendung gefunden hat, kann man dann im umfassenden Systemumfeld weiter nach ihnen und nach verwandten Dingen suchen. Die Attribute der (potentiellen) Klassen stehen, soweit sie schon zu Beginn als interessante „Dinge“ erfaßt wurden, in der Regel weiter unten im Diagramm. Bearbeiten aber verschiedene Anwendungsfälle verschiedene Attribute der gleichen Klasse, sind die Klasse und ihre Attribute mitunter beliebig weit über das Liniendiagramm verteilt.

In unserem Diagramm steht die für alle Anwendungsfälle interessante Klasse *Item* (die auch von Jacobson als zentrale Klasse zuerst betrachtet wird) ganz oben, ihre Attribute *Return value* und *Size* unten rechts und ihre Subklassen *Can, Bottle* und *Crate* links unten.

Nur für einzelne Anwendungsfälle interessante Klassen sind diesen im Liniendiagramm direkt zugeordnet oder benachbart. Es reicht also nicht aus, sich allein auf die Spitze des Diagramms konzentrieren.

4 Fazit und Ausblick

Bei unserem Vorgehen bleibt die Wahl der Objekte und Klassen eine Entwurfsentscheidung des Entwicklers. Wir schlagen also nicht vor, diese Entscheidung zu automatisieren, sondern die von einigen Autoren vorgeschlagene grammatikalische Untersuchung der Anforderungsbeschreibung durch einen Diskurs zwischen Fachexperten und Entwicklern zu ersetzen und durch ein Werkzeug zur Darstellung seiner Ergebnisse zu unterstützen. Dieses Verfahren ist dem anderen überlegen, da es nicht rein sprachanalytisch vorgeht, sondern Systemgesichtspunkte (bei der Wahl des formalen Kontexts) einbezieht.

Die hier dargestellte Vorgehensweise und das entstandene Liniendiagramm beleuchten nur einen Aspekt der Modellierung. Die Unterstützung anderer Sichten durch FBA ist vorstellbar. Z.B. kann FBA ebenfalls bei der Strukturierung des Klassenmodells sowie als Hilfsmittel für das Verständnis bereits bestehender objektorientierter Modelle eingesetzt werden.

5 Literatur

[And 96] U. Andelfinger: Diskursive Anforderungsanalyse, Europäische Hochschulschriften, Reihe Informatik, vol. 25, Peter Lang, Frankfurt am Main 1996

[Boo 94] G. Booch: Object-Oriented Analysis an Design with Applications, Benjamin/Cummings 1994

[C-Y 90] P. Coad, E. Yourdon: Object-Oriented Analysis, Prentice Hall 1990

[Jac 92] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard: Object-Oriented Software Engineering - A Use Case Driven Approach, ACM-Press, Addison-Wesley 1992

[M-O 92] J. Martin, J. Odell: Object-Oriented Analysis and Design, Prentice Hall 1992

[Mey 88] B. Meyer: Object oriented software construction, Prentice Hall 1988

[S-M 91] S. Shlaer, S.J. Mellor: Object Lifecycles, Modeling the world in states, Yourdon Press 1991

[WWW 90] R. Wirfs-Brock, B. Wilkerson, L. Wiener: Designing Object-Oriented Software, Prentice Hall 1990

[W-G 96] R. Wille, B. Ganter: Formale Begriffsanalyse, Mathematische Grundlagen, Springer 1996

Adresse der Autoren:

Stephan Düwel, Prof. Dr. Wolfgang Hesse
FB Mathematik/Informatik, Philipps-Universität
Hans Meerwein-Str.
D-35032 Marburg
Tel.: +49-6421-281515 (wh), -281522 (sd)
Fax: +49-6421-285419, email: <duewel,hesse>@informatik.uni-marburg.de