Ein Ansatz zur einheitlichen Darstellung der Objekte einer beliebiger Instantiierungsstufe und des Übergangs zur nächstniedrigeren Instantiierungsstufe Teil II: Übergangsformalismus

Gerhard Kratz
Fachhochschule Frankfurt am Main
Fachbereich Mathematik, Naturwissenschaften und Datenverarbeitung (MND)
Nibelungenplatz 1
60318 Frankfurt am Main

Mai 1994

Zusammenfassung

Um aus einem Fachkonzept einen Prototypen für ein betriebliches Informationssystem, aus der Beschreibung einer Fachkonzeptmethode ein CASE-Werkzeug und aus methodologischer Information ein Werkzeug zum Generieren von CASE-Werkzeugen möglichst automatisiert herzustellen, ist es wünschenswert, die auf allen Stufen des Diskurses auftretenden Objekte sowie den Übergang von einer dieser Ebenen zur nächstniedrigeren allgemein zu beschreiben. Der bereits erschienene Teil I enthält eine allgemeine, auf Begriffen der Elementarmathematik fußende Beschreibung derjenigen Objekte, die in den Methoden des Software Engineering und in Fachkonzepten von betrieblichen Informationssystemen auftreten können. Im vorliegenden Teil II wird ein Formalismus zum Übergang von einer Stufe des Diskurses zur nächstniedrigeren angegeben, der aus der Instantiierung von Exemplaren, ihrer Uminterpretation in Klassen und der Vervollständigung ihrer Definitionen besteht.

Beschreibung der Objekte und Übergangsformalismus sollen die Grundlage für die Konstruktion eines oben skizzierten Werkzeugs mit dem Namen "Die Begriffsmaschine" sein.

3.1. Anforderungen

Instantiierung heißt, ein Exemplar einer Klasse zu erzeugen und ihrer Exemplarmenge hinzuzufügen. Andererseits muß man auch in der Lage sein, ein Exemplar aus der Exemplarmenge einer Klasse zu entfernen. Dies gilt nicht nur für Atomklassen wie PERSON, sondern auch für nicht-Funktionenklassen, deren Exemplare Tupel sind, sowie für Funktionenklassen, deren Exemplare Argument-Wert-Paare sind.

Üblicherweise legt man zusammen mit einem Exemplar einer Konzeptatomklasse auch Tupel und insbesondere Argument-Wert-Paare an, in denen dieses Exemplar vorkommt. Z.B. Legt man für einen neuen Mitarbeiter in einem Betrieb seine Personal-Nummer und seinen Namen an. Man speichert also zwei Argument-Wert-Paare, deren Argument jeweils eine eineindeutige und nicht veränderliche Kennung des Mitarbeiters ist, und deren Werte die Personal-Nummer bzw. der Name des Mitarbeiters sind. Darüberhinaus gibt man an, in welcher Abteilung er arbeitet bzw. arbeiten wird. Dies wird in Form eines Tupels, und zwar eines Paars gespeichert, dessen erste Komponente die Mitarbeiterkennung und dessen zweite Komponente die Abteilungskennung ist.

Entfernt man nun die Daten eines Mitarbeiters aus der Exemplarmenge der Klasse MITARBEITER, so sind auch diejenigen Argument-Werte-Paare zu löschen, in denen dieser Mitarbeiter als Argument auftritt. Ebenso sind dann alle Tupel zu löschen, in denen er als Komponente auftritt.

Es gibt darüberhinaus DV-Anwendungen, in denen ein bestimmtes Exemplar weder hinzugefügt noch entfernt wird, aber Argument-Wert-Paare bzw. Tupel angelegt werden, in denen es vorkommt. Eine solche Änderungsanwendung kann z.B. dazu dienen, den neuen Namen eines Mitarbeiters, z.B. nach einer Heirat, aufzunehmen und seinen alten Namen zu löschen oder den Mitarbeiter einer anderen Abteilung zuzuordnen.

In einer Bearbeitungsanwendung werden also Exemplare gelöscht und / oder Exemplare angelegt. Der Datenbearbeitungsteil eines betrieblichen Informationssystems enthält also eine Implementation der Instantiierung der Exemplare der jeweils vorgesehenen Klassen.

Die automatische Erzeugung der Bearbeitungsanwendungen, mit denen die Exemplarmengen eines Diskursmodells verändert werden, ist eine Teilaufgabe bei der Erarbeitung der "Begriffsmaschine"; sie wird hier nicht beschrieben. Vielmehr beschränke ich mich hier darauf, das Verhältnis der jeweils vorgesehenen Klassen zu ihren Exemplaren darzustellen.

3.2. Darstellung

Die Instantiierung wird beschrieben, indem man angibt, welchen Bedingungen die zu instantiierenden Exemplare genügen müssen:

- Ein Exemplar einer atomaren Klasse ist ein Atom (s. Abbn. 2-1 und 2-2).
  Die Exemplare einer Konzeptatomklasse sind Konzeptatome. Welche Konzeptatome anzulegen (und zu löschen) sind, läßt sich nicht berechnen, sondern ist rein empirisch bestimmt.
  Die Exemplare von Zahlenklassen sind Zahlen, die Exemplare von Textklassen Texte und die Exemplare der Klasse der Wahrheitswerte sind eben die Wahrheitswerte "wahr" und "falsch".
- Ein Exemplar einer n-stelligen Relation ist ein n-Tupel (s. Abb. 2-3).
  Im besonderen ist ein Exemplar einer Funktionenklasse ein Argument-Wert-Paar, wobei das Argument der Argumentenklasse und der Wert der Werteklasse der Funktionenklasse zum fraglichen Zeitpunkt angehört.
  Ein Exemplar einer n-stelligen nicht-Funktionenklasse ist ein n-Tupel, dessen k-te Komponente ein Exemplar der Werteklasse der k-ten Projektionenklasse der nicht-Funktionenklasse ist.
- Alle Integritätsbedingungen, die für die Exemplare einer Klasse gelten, sind nach der Instantiierung erfüllt.

4. UMINTERPRETATION VON EXEMPLAREN IN KLASSEN

4.1. Anforderungen

Nachdem man Exemplare einer Klasse instantiiert hat, möchte man zumindest einige dieser Exemplare als Klassen interpretieren, für die ebenfalls Exemplare anzulegen sind. Es erscheint intuitiv sinnvoll, ausschließlich bestimmte Konzeptatome in Klassen umzuinterpretieren. Denn jede Klasse, auch jede Relationenklasse und insbesondere jede Funktionenklasse wird als eigenständiges Objekt angesehen und deshalb nicht als Tupel oder im speziellen als Argument-Wert-Paar dargestellt.

Dann stellt sich jedoch die Frage, welcher Art eine so erzeugte Klasse sein soll (Konzeptatomklasse, nicht-Funktionenklasse, nicht-Projektionenklasse). Oder in entgegengesetzter Richtung gefragt: Was ist zu tun, um als Ergebnis der Uminterpretation eine Konzeptatomklasse, eine nicht-Funktionenklasse oder eine nicht-Projektionenklasse zu erhalten? Welches sind die an einer nicht-Funktionenklasse beteiligten Klassen? Welche ist die Argumentenklasse, welche ist die Werteklasse einer nicht-Projektionenklasse?

In der folgenden Abb. 4-1.1 ist zunächst je ein Exemplar der Konzeptatomklassen ENTITÄTENKLASSE, ATTRIBUT, TEXT sowie der nicht-Projektionenklassen VERWEIS_ATTRIBUT_ENTITÄTENKLASSE und VERWEIS_ATTRIBUT_TEXT dargestellt. Dabei sei die Bedeutung eines Attributs wie üblich die einer nicht-Projektionenklasse, deren Werteklasse eine Zeichenatomklasse und deren Argumentenklasse hier eine Entitätenklasse sei. Das Ziel sei diejenige Uminterpretation, in der NAME eine nicht-Projektionenklasse mit der Argumentenklasse PERSON und der Werteklasse NAMENSTEXT ist.

Möchte man dieses Ergebnis ableiten können, so müssen die Beschreibungen der Klassen der Instantiierungsstufe i+1 zusätzliche Information enthalten. In der Abb. 4-1.2 ist in den Kreisen eine Abkürzung für die Art derjenigen Klassen eingetragen, die durch Uminterpretation aus jedem Exemplar einer Konzeptatomklasse hervorgehen sollen (k: Konzeptatomklasse, np: nicht-Projektionenklasse, s: Zeichenatomklasse; s.u.). Darüberhinaus wird durch das Symbol ARG auf die Argumentenklasse und durch das Symbol WERT auf die Werteklasse einer durch Uminterpretation zu erzeugenden Funktionenklasse hingewiesen. Außerdem benötigen wird die eine Regel, die besagt, daß nur ein solches Exemplar f als Funktionenklasse und nur solche Exemplare a und w als Argumentenklasse bzw. Werteklasse von f uminterpretiert werden können, für die es die Argument-Wert-Paare (f, a) und (f, w) gibt.

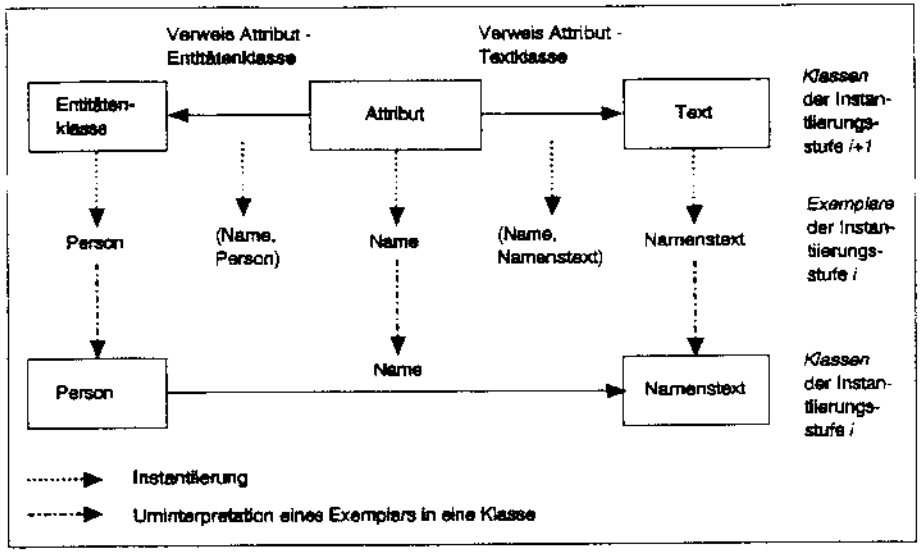

Abb. 4-1.1: Instantiierung von Exemplaren von Konzeptatomklassen (Rechtecke) und von nicht-Projektionenklassen (Pfeile) sowie deren Uminterpretation in eine Konzeptatomklasse, eine nicht-Projektionenklasse und eine Textatomklasse

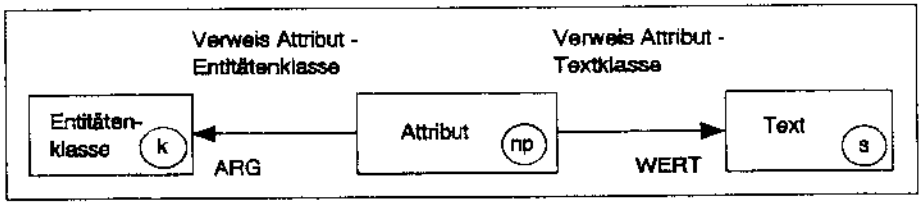

Abb. 4-1.2: Vervollständigung der Beschreibung von Konzeptatomklassen (Rechtecke) und von nicht-Projektionenklassen (Pfeile) zur eindeutigen Uminterpretation ihrer Exemplare in eine Konzeptatomklasse, eine nicht-Projektionenklasse und eine Textatomklasse

Wie läßt sich nun durch Uminterpretation eine nicht-Funktionenklasse mit ihren Projektionenklassen erzeugen? Das Muster der Klassen der Instantiierungsstufe i+1 in Abb. 4-2.1 ist dasselbe wie in Abb. 4-1.1. Wir fordern auch hier wieder die Gültigkeit einer entsprechenden Regel. Damit allerdings eindeutig aus einem Exemplar der Konzeptatomklasse BEZIEHUNGENKLASSE eine nicht-Funktionenklasse sowie aus einem Exemplar der Konzeptatomklasse ROLLE eine Projektionenklasse wird, müssen auch diese Konzeptatomklassen entsprechend gekennzeichnet werden, was in Abb. 4-2.2 geschehen ist (nf: nicht-Funktionenklasse, p: Projektionenklasse).

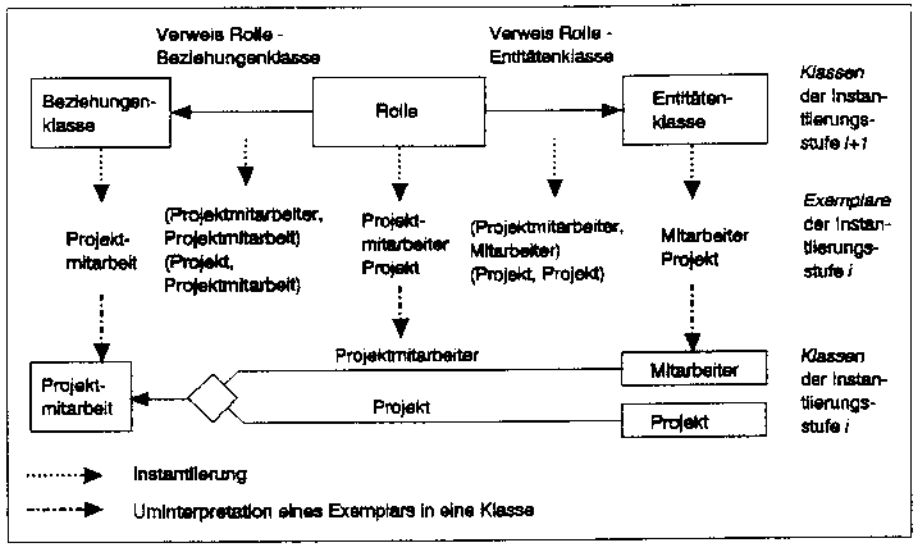

Abb. 4-2.1: Instantiierung von Exemplaren von Konzeptatomklassen (Rechtecke) und von nicht-Projektionenklassen (Pfeile) sowie deren Uminterpretation in Konzeptatomklassen, eine nicht-Funktionenklasse und in Projektionenklassen

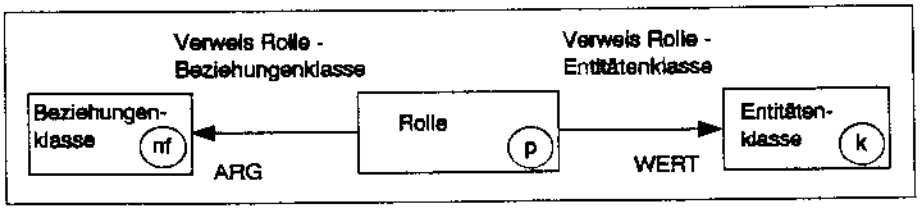

Abb. 4-2.2: Vervollständigung der Beschreibung von Konzeptatomklassen (Rechtecke) und von nicht-Projektionenklassen (Pfeile) zur eindeutigen Uminterpretation ihrer Exemplare in eine Konzeptatomklassen, eine nicht-Funktionenklasse und in Projektionenklassen

4.2. Darstellung

Ein Exemplar wird in höchstens eine Klasse uminterpretiert. Die Exemplare der Instantiierungsstufe 0 werden nicht uminterpretiert. Auch auf allen anderen Instantiierungsstufen kann es Exemplare geben, die nicht uminterpretiert werden. Alle Klassen sind Ergebnis der Uminterpretation genau eines Exemplars. Ein Exemplar x hat dieselbe Instantiierungsstufe wie die Klasse x, die aus ihm durch Uminterpretation entsteht:

$i(x) = j(x)$.

Die Symbole der Klassen von Konzeptatomen werden mit einer Kennzeichnung versehen, aus der hervorgeht, in welche Art von Klassen ihre Exemplare, wenn überhaupt, uminterpretiert werden. Diese Abbildung heißt das Uminterpretationsziel der Exemplare einer Klasse. In der nachstehenden Tabelle 4-1 wird für jede Art von Konzeptatomklassen angegeben, wie ihre Exemplare benannt werden und in welche Art von Klassen sie jeweils uminterpretiert werden.

Z.B. heißt eine Konzeptatomklasse, deren Exemplare in eine nicht-Funktionenklasse uminterpretiert werden sollen, nf-Atomklasse, ihre Exemplare heißen nf-Atome. Das Uminterpretationsziel der Exemplare einer Konzeptatomklasse kann durch einen Kreis mit der Abkürzung für ihre Bezeichnung angegeben werden.

Das Uminterpretationsziel der Exemplare einer Konzeptatomklasse K wird entlang der Spezialisierungen vererbt, deren Oberklasse K ist. Mehrfache Spezialisierungen von K müssen so in eine Hierarchie von Spezialisierungen überführbar sein, daß die Uminterpretationsziele aller Exemplare von K eindeutig bestimmt werden können.

Instantiierungsstufe i+1		Instantiierungsstufe i	
Art von Klassen uminterpretierbarer Konzeptatome Bezeichnung	Abk.	Art eines uminterpretierbaren Konzeptatoms	Uminterpretationsziel
k-Atomklasse	k	k-Atom	Konzeptatomklasse
s-Atomklasse	s	s-Atom	Zeichenatomklasse
np-Atomklasse	np	np-Atom	nicht-Projektionenklasse
p-Atomklasse	p	p-Atom	Projektionenklasse
nf-Atomklasse	nf	nf-Atom	nicht-Funktionenklasse

Tab. 4-1: Arten von Klassen uminterpretierbarer Konzeptatome, ihrer Exemplare und Ergebnisse der Uminterpretation dieser Exemplare

4.2.1. Erzeugen einer Konzeptatomklasse und einer Zeichenatomklasse

Regel a: *Jedes Exemplar einer Konzeptatomklasse der Instantiierungsstufe i+1 mit Uminterpretationsziel 'k' wird als Konzeptatomklasse der Instantiierungsstufe i interpretiert.*

Die Exemplare der Klasse ENTITÄTENKLASSE der Abbn. 4-1.1 und 4-1.2 sind Konzeptatome, wie z.B "Person", das als die Konzeptatomklasse PERSON der Instantiierungsstufe I interpretiert wird.

Regel b: *Jedes Exemplar einer Konzeptatomklasse der Instantiierungsstufe i+1 mit Uminterpretationsziel 's' wird als Zeichenatomklasse der Instantiierungsstufe i interpretiert.*

In ähnlicher Weise wird das Exemplar "Namenstext" der Konzeptatomklasse TEXT als Zeichenatomklasse interpretiert. Im Rahmen der Vervollständigung (Kap. 5) wird präzisiert, daß es sich um eine Textatomklasse handelt.

4.2.2. Erzeugen einer nicht-Projektionenklasse

Um ein Exemplar einer Konzeptatomklasse mit Uminterpretationsziel 'np' als eine nicht-Projektionenklasse zu interpretieren, müssen wir gleichzeitig angeben, welches Exemplar als Argumentenklasse und welches Exemplar als Werteklasse der so erzeugten nicht-Projektionenklasse interpretiert werden soll. In unserer hier anzuwendenden Regel c (s.u.) benutzen wir den Begriff des np-Musters, der zunächst erläutert wird.

Ein np-Muster (Abb. 4-3) besteht aus

(1) den folgenden Klassen der Instantiierungsstufe i+1:

- einer Konzeptatomklasse y_{np}, deren Exemplare als nicht-Projektionenklassen interpretiert werden
- einer Konzeptatomklasse y_d, deren Exemplare entweder als Konzeptatomklassen oder als nicht-Funktionenklassen interpretiert werden
- einer Konzeptatomklasse y_r, deren Exemplare entweder als Konzeptatomklassen, als nicht-Funktionenklassen oder als Zeichenatomklassen interpretiert werden
- einer nicht-Projektionenklasse f_d, deren Argumtenklasse y_{np} oder eine ihrer Oberklassen ist, und deren Werteklasse y_d ist
- einer nicht-Projektionenklasse f_r, deren Argumtenklasse y_{np} oder eine ihrer Oberklassen ist, und deren Werteklasse y_r ist

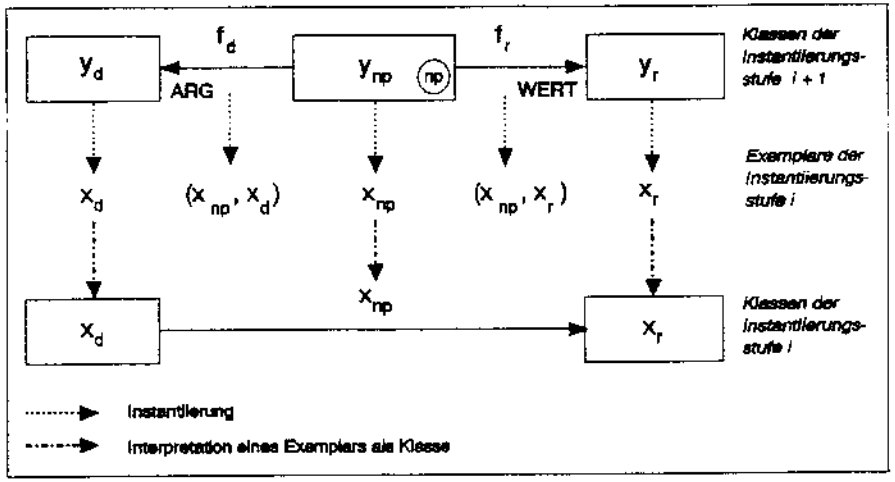

Abb. 4-3: np-Muster

(2) den folgenden Exemplaren der Instantiierungsstufe i

- ein Konzeptatom x_{np} , das Exemplar von y_{np} ist
- ein Konzeptatom x_d , das Exemplar von y_d ist
- ein Konzeptatom x_r , das Exemplar von y_r ist
- ein Argument-Wert-Paar (x_{np} , x_d), das Exemplar von f_d ist
- ein Argument-Wert-Paar (x_{np} , x_r), das Exemplar von f_r ist

(3) einer Funktion dr mit dr(f_d) ='ARG' und dr(f_r) = 'WERT'

Regel c: *Enthält ein Diskursmodell ein np-Muster mit den o.a. Bestandteilen, so wird x_{np} als nicht-Projektionenklasse mit Argumentenklasse x_d und Werteklasse x_r interpretiert.*

Die Funktion dr wird benötigt, um die Argumentenklasse bzw. die Werteklasse einer durch Uminterpretation erzeugten nicht-Projektionenklasse festzulegen. Eine Klassenzusammensetzung, die durch (1) und (3) beschrieben wird, kann in mehr als einem np-Muster auftreten.

Ein Beispiel für ein np-Muster wird in der Abb. 4-1.1 zusammen mit den Uminterpretationszielen und der Argument-Wert-Bestimmung dr der Abb. 4-1.2 gegeben.

4.2.3. Erzeugen einer nicht-Funktionenklasse

In analoger Weise benutzen wir zur Beschreibung dieses Vorgangs den Begriff des p-Musters.

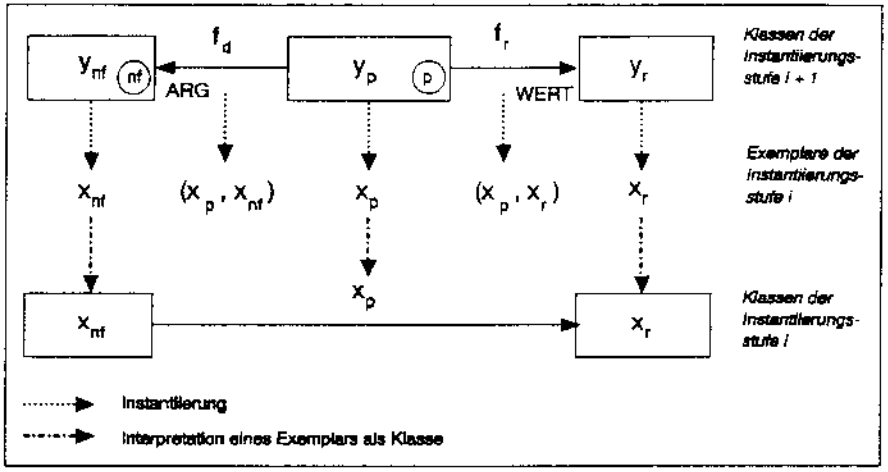

Abb. 4-4: p-Muster

Ein p-Muster (Abb. 4-4) besteht aus

(1) den folgenden Klassen der Instantiierungsstufe i+1:

- einer Konzeptatomklasse y_p , deren Exemplare als Projektionenklassen interpretiert werden
- einer Konzeptatomklasse y_d , deren Exemplare als nicht-Funktionenklassen interpretiert werden
- einer Konzeptatomklasse y_r , deren Exemplare entweder als Konzeptatomklassen, als nicht-Funktionenklassen oder als Zeichenatomklassen interpretiert werden
- einer nicht-Projektionenklasse f_d , deren Argumtenklasse y_p oder eine ihrer Oberklassen ist, und deren Werteklasse y_d ist
- einer nicht-Projektionenklasse f_r , deren Argumtenklasse y_p oder eine ihrer Oberklassen ist, und deren Werteklasse y_r ist

(2) den folgenden Exemplaren der Instantiierungsstufe i

- ein Konzeptatom x_p , das Exemplar von y_p ist
- ein Konzeptatom x_d , das Exemplar von y_d ist
- ein Konzeptatom x_r , das Exemplar von y_r ist
- ein Argument-Wert-Paar (x_p , x_d), das Exemplar von f_d ist
- ein Argument-Wert-Paar (x_p , x_r), das Exemplar von f_r ist

(3) einer Funktion dr mit dr(f_d) ='ARG' und dr(f_r) = 'WERT'

Regel d: *Gegeben seien $k \geq 2$ p-Muster (Indizes 1, 2, ... , k) mit*

- *identischen Klassen $y_{nf,1} = y_{nf,2} = \ldots = y_{nf,k}$ und*
- *identischen Konzeptatomen $x_{nf,1} = x_{nf,2} = \ldots = x_{nf,k}$.*

Dann interpretieren wir x_{nf} als eine nicht-Funktionenklasse und die $x_{p,1}$, $x_{p,2}$, ... , $x_{p,k}$ als Projektionenklassen mit der Argumentenklasse x_{nf} und den Werteklassen $x_{r,1}$, $x_{r,2}$, ... , $x_{r,k}$.

Ein Beispiel für das Erzeugen einer nicht-Funktionenklasse aus zwei p-Mustern wird in der Abb. 4-2.1 zusammen mit den Uminterpretationszielen und der Argument-Wert-Bestimmung dr der Abb. 4-2.2 gegeben.

5. VERVOLLSTÄNDIGUNG DER KLASSENBESCHREIBUNGEN

Nachdem die Exemplare nach den Regeln a - d zu Klassen uminterpretiert wurden, sind den Klassenbeschreibungen weitere Angaben hinzuzufügen, die sich nicht ableiten lassen. Dies sind:

- Anlegen der Diskursmodelle der Instatiierungsstufe i und Zuordnung jeder erzeugten Klasse zu einem Diskursmodell
- Einführen von Spezialisierungen/Generalisierungen mit ihren Oberklassen und Unterklassen
- Angabe der Berechnungsvorschriften
- Angabe von expliziten Integritätsbedingungen
- Kennzeichnung von Funktionenklassen als Klassen für Beginn- bzw. Endezeitpunkte der Existenzzeitintervalle von Exemplaren
- Bezeichnung der Umkehrauswertungen von Funktionenklassen, Angabe ihrer Kardinalitätenbeschränkungen und die Angabe, ob sie explizit vektorwertig geführt werden
- für nicht-Projektionenklassen wird angegeben, ob sie partiell oder total definiert sind, und im letzteren Fall, ob sie ein-eindeutig sind oder nicht
- für Konzeptatomklassen der Instantiierungsstufen i > 1werden die Uminterpretationsziele bestimmt
- der Wert der Funktion dr einer nicht-Projektionenklasse der Instantiierungsstufen i > 1 kann angegeben werden
- für jede Zeichenatomklasse wird angegeben, ob es sich bei ihren Exemplaren um ganze oder reelle Zahlen, um Texte oder um Wahrheitswerte handelt.

6. BEISPIEL

Gegeben sei das in Abb. 6-1 dargestellte Diskursmodell der Instantiierungsstufe i = 3 mit den Exemplaren der nächstniedrigeren Instantiierungsstufe i = 2.

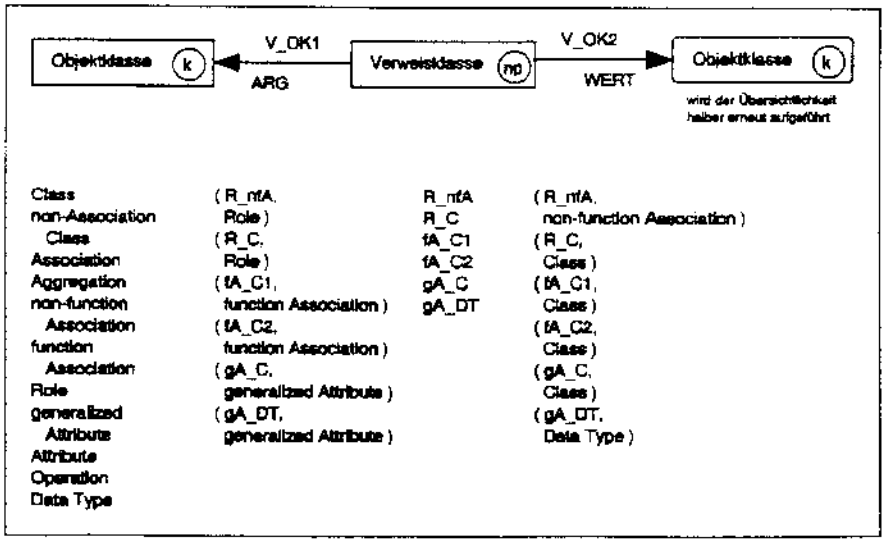

Abb. 6-1: Diskursmodell der Instantiierungsstufe 3 mit Exemplaren

Auf die in diesem Diskursmodell enthaltenen Konzeptatomklassen OBJEKTKLASSE und VERWEISKLASSE mit ihren jeweiligen Exemplaren kann zur Uminterpretation die Regel a angewandt werden. Dadurch entsteht auf der Instantiierungsstufe 2 z.B. die Objektklasse CLASS.

Durch Anwendung der Regel c auf die vorhandenen np-Muster kann z.B. die nicht-Projektionenklasse R_C mit der Argumentenklasse ROLE und der Werteklasse CLASS hervorgehen.

Das gesamte Diskursmodell, das durch Uminterpretation erzeugt wird, ist anschließend zu vervollständigen und zwar im wesentlichen um die Spezialisierungen, um die Uminterpretationsziele sowie um die Argument-Wert-Bestimmung der nicht-Projektionenklassen.

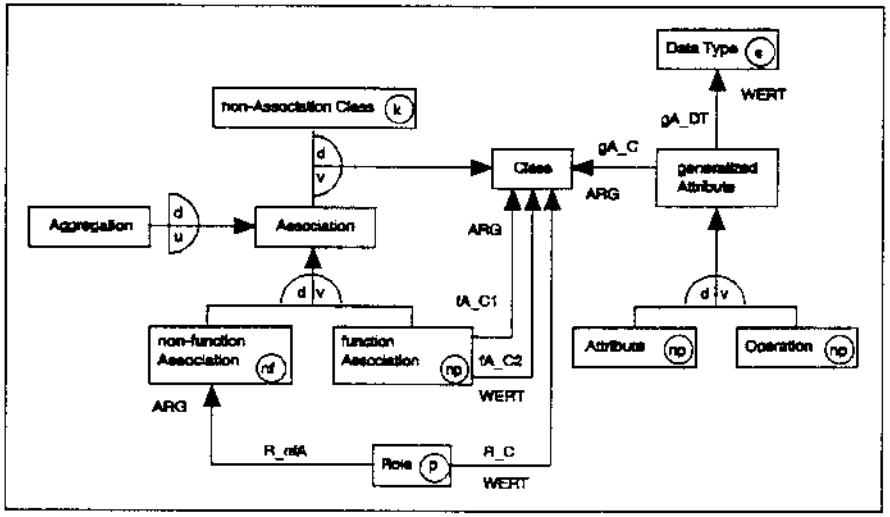

Abb. 6-2: Diskursmodell der Instantiierungsstufe 2
d: disjunkte Spezialisierung
v/u: vollständige / unvollständige Spezialisierung

Eine "Class" ist demgemäß entweder eine "Association" oder eine "non-Association Class". Eine "Association" ihrerseits ist entweder eine "function Association", also eine nicht-Projektionenklasse, oder eine "non-function Association", also eine nicht-Funktionenklasse. Argumenten- bzw. Werteklasse einer "function Association" sowie Rollenträger einer "non-function Association" kann jede Art von "Class" sein.

Zu den Exemplaren der Klasse ATTRIBUTE gibt es keinen Berechnungs- bzw. Ableitungsalgorithmus, zu den Exemplaren der Klasse OPERATION jedoch sehr wohl.

Das Diskursmodell der Abb. 6-2 lehnt sich stark an den entsprechenden Ausschnitt der Objektmodellierungstechnik von Rumbaugh u.a. (1991) an.

7. ZUSAMMENFASSUNG UND AUSBLICK

Vorausgesetzt, ein Diskursmodell mit Klassen einer bestimmten Instantiierungsstufe liegt vor. Es kann dann Klassen unterschiedlicher Arten (Atomklassen, Relationenklassen, Funktionenklassen), Kardinalitätenbeschränkungen, Spezialisierungen/Generalisierungen, Funktionenklassen für Beginn- und für Endezeitpunkte von Existenzzeitintervallen von Exemplaren sowie Berechnungen und Integritätsbedingungen enthalten. Im nächsten Schritt der Arbeit soll es möglich sein, Exemplare der vorhandenen Klassen anzulegen und zu bearbeiten, die den Integritätsbedingungen entsprechen (Instantiierungsschritt). Die dazu erforderlichen DV-Anwendungen, ihre Benutzeroberfläche und die von ihnen belieferten und ausgewerteten Datenspeicherstrukturen sollen soweit automatisiert wie möglich aus dem vorhandenen Diskursmodell abgeleitet werden.

Anschließend kann die in Kap. 4 vorgestellte Uminterpretation der eingegebenen Exemplare in Klassen auf algorithmische Weise stattfinden.

Die DV-Anwendungen, die dazu dienen, daran anschließend die Klassenbeschreibungen zu vervollständigen, basieren auf dem Beschreibungsmodell und werden deshalb nicht generiert, sondern können für alle Diskursmodelle vorgegeben werden. Zu ihnen gehören auch Editoren, die die syntaktische Richtigkeit von Berechnungen und Integritätsbedingungen überprüfen.

An der Fachhochschule Frankfurt am Main wird derzeit das Forschungsprojekt "Die Begriffsmaschine" mit dem Ziel durchgeführt, ein Software Engineering-Werkzeug zu entwickeln, das die oben skizzierten Möglichkeiten bietet. Es soll uns in die Lage versetzen,

- aus einem methodologischen Modell ein gewünschtes Diskursmodell für eine beliebige Software Engineering-Methode zu erzeugen. Diese Anwendung der Begriffsmaschine wäre das Meta-CASE-Werkzeug (s. Abbn. 6-1, 6-2)
- aus einem methodischen Modell ein fachliches Modell zu erzeugen. Diese Anwendung der Begriffsmaschine enthält die Funktionalität eines CASE-Werkzeugs
- für die Klassen eines fachlichen Modells Exemplare einzugeben, zu bearbeiten und auszuwerten. Diese Anwendung wäre ein Prototypengenerator.

Es sind also folgende Komponenten der Begriffsmaschine anzufertigen:

- ein Programm zum möglichst automatisierten Erzeugen von DV-Anwendungen zum Anlegen und Bearbeiten von Exemplaren gemäß den Beschreibungen ihrer Klassen. Dieses besteht aus:
  - dem Generator der Anwendungsprogramme,
  - dem Generator für einen Vorschlag der Struktur der Datenbasis und einem Editor, mittels dessen diese Struktur so verändert werden kann, daß sie nicht in Widerspruch zum Diskursmodell gerät
  - dem Generator für einen Vorschlag der Objekte der Benutzeroberfläche sowie auch hier einem Editor zum konsistenzerhaltenden Verändern dieser Struktur
- ein Programm zur Uminterpretation von Exemplaren in Klassen
- ein Programm zur Vervollständigung einer Klassenbeschreibung, das auch Editoren zur korrekten Eingabe und Bearbeitung von Integritätsbedingungen und Berechnungen enthält
- ein Programm, mit dem sich auf benutzerfreundliche Weise Auswertungen definieren und ausführen lassen.

Literatur

J. Rumbaugh, M. Blaha, W. Pemerlani, F. Eddy, W. Lorensen:Object-Oriented Modeling and Design. Prentice-Hall, Englewood Cliffs, N.J., 1991

Danksagung

Für wertvolle Diskussionsbeiträge bedanke ich mich sehr herzlich bei meinem Kollegen Gerd Mitschke und bei meiner Kollegin Wilhelmine Rottmann-Söde, beide Fachhochschule Frankfurt am Main.

Die Durchführung eines Forschungsprojekts an einer Fachhochschule einschließlich der Gewährung von Finanzmitteln und Arbeitszeit ist nicht selbstverständlich. Für die Unterstützung bei der Antragstellung und in organisatorischen Fragen bedanke ich mich bei Herrn Prof. Dr. B. Dumbacher, Herrn Prof. Dr. J. Schneider, Herrn P. Sulzbach und Herrn Prof. Dr. U. Timm, alle Fachhochschule Frankfurt am Main.