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

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

Dezember 1993

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. 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 später erscheinenden 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.

1. Einleitung

1.1. Anwendungsbezug: Generator von Prototypen für betriebliche Informationssysteme

In vielen, vor allem großen Betrieben (öffentliche Verwaltungen und Unternehmen der privaten Wirtschaft) werden gegenwärtig kommerziell-administrative Informationssysteme betriebseinheitlich nach einer bestimmten Vorgehensweise entwickelt. Die meisten dieser Software Engineering-Methoden sehen vor, daß zunächst der relevante Ausschnitt der betrieblichen Realität sowie die Anforderungen der späteren Benutzer in strukturierter Form modelliert werden (Fachkonzept), anschließend deren Umsetzung in Datenspeicherstrukturen und Programme entworfen wird (DV-Entwurf) und schließlich die Datenspeicherstrukturen und Programme implementiert und getestet werden.

Da der Übergang vom Fachkonzept bis zur Inbetriebnahme relativ lange Zeit in Anspruch nimmt, erweist sich in der Praxis dieses Vorgehen häufig als nicht einhaltbar, weil sich zwischenzeitlich die maßgebliche Ansicht sowohl der betrieblichen Realwelt als auch der Benutzeranforderungen ändern kann. Diese Änderungen werden zumeist entweder einfach ignoriert, so daß die fertiggestellte DV-Anwendung nur einen Teil ihres potentiellen Nutzens bringen kann, oder sie werden ausschließlich in der Implementation berücksichtigt, nicht aber im Fachkonzept nachgehalten, was die Dokumentation mehr und mehr von der real implementierten Software entfernt. Es kommt also darauf an, möglichst schnell zu einem stabilen, tragfähigen Fachkonzept zu kommen.

Einen Beitrag zur Lösung dieses Problems sollte ein Werkzeug leisten, mit dem man in kurzer Zeit aus den fachlichen Informationen Prototypen für eine herzustellende DV-Anwendung generieren kann (Prototypengenerator). Damit wäre den Benutzern ein Mittel an die Hand gegeben, das das, was sie von der von ihnen beschriebenen DV-Anwendung zu erwarten haben, in sehr anschaulicher Weise vor Augen führt und damit eine schnelle Rückmeldung über die Angemessenheit der Lösung und ihrer fachlichen Beschreibung erlaubt.

Darüberhinaus läßt sich (unter sehr weit gefaßten Einschränkungen[1]) ein DV-Entwurf, der mit einem Prototypengenerator erzeugt wurde, auch als Ausgangspunkt für die Implementation des Zielsystems verwenden, d.h. höchstens ein Teil des generierten Prototypen muß anschließend verworfen werden.

1.2. Anwendungsbezug: Meta-CASE-Werkzeug

Zur Erst- und Weiterentwicklung von Software nach einer bestimmten Vorgehensweise stehen den damit befaßten Personen in vielen Fällen Software Engineering-Werkzeuge zur Verfügung bzw. sind verbindlich zu benutzen. Aber auch die Software Engineering-Methoden, auf denen diese Werkzeuge basieren, werden weiterentwickelt.

Da die Herstellung von SE-Werkzeugen und ihre Vermarktung allerdings vergleichsweise teuer und ihr Absatz riskant ist, werden methodische Verbesserungen nur langsam in neue Werkzeuge oder in neue Versionen von Werkzeugen übernommen.[2] Es erscheint also im Sinne der Verantwortlichen für die in einem Betrieb angewandten Software Engineering-Methoden und im Sinne eines Herstellers von Software Engineering-Werkzeugen wünschenswert, aus methodologischen Vorgaben Werkzeuge oder Änderungen von Werkzeugen[3] generieren zu können. Ein solches Werkzeug zum Bau von CASE-Werkzeugen heißt Meta-CASE-Werkzeug.

[1]Diese Einschränkungen sind gegeben durch die Art der Datenbankverwaltungssysteme (hierarchisch, netzwerkartig, relational), die Art der verwendeten objekt-orientierten Datenbank), die Art der Benutzungsoberfläche (Terminal, grafisch-interaktiver Bildschirm) sowie den Darstellungsstil (Formular, Diagramm usw.).

[2]Dies führt zu der paradoxen Situation, daß Software-Entwickler ihre (veralteten) Hilfsmittel weiterhin "krummbiegen" mussen, um (so gut es geht) zu den erforderlichen Ergebnissen zu kommen, obwohl der Entwicklungsstand der Methoden ein problem-angemesseneres Arbeiten zuließe.

[3]Dabei ist natürlich das Problem der fortgesetzten Nutzung der mit einem Vorläufer-Werkzeug angelegten Daten zu lösen.

1.3. Problemstellung

Die Arbeitsweisen von Prototypengenerator und Meta-CASE-Werkzeug lassen sich in folgende Struktur einordnen:

- Gemäß einem methodologischen Modell und unter Verwendung eines Meta-CASE-Werkzeugs legen Benutzer (hier z.B. die Verantwortlichen für Software Engineering-Methoden in einem Betrieb) Objekte an, die in einem methodischen Modell vorkommen sollen, bearbeiten sie und werten sie aus. Solche Objekte können z.B. "Entitätenklasse", "Beziehungenklasse", "Attribut" sein.
- Gemäß einem methodischen Modell und unter Verwendung eines CASE-Werkzeugs legen Benutzer (hier die Systemanalytiker und die Auftraggeber eines betrieblichen Informationssystems) Objekte an, die in einem fachlichen Modell vorkommen sollen, bearbeiten sie und werten sie aus. Solche Objekte können z.B. "Person", "Kraftfahrzeug", "Kraftfahrzeugzulassung" sein.
- Gemäß einem fachlichen Modell und unter Verwendung eines betrieblichen Informationssystems bzw. eines Prototypen dafür legen Benutzer (die DV-Endanwender) Einzelobjekte an, die modellhafte Abbilder der Gegenstände ihrer Arbeitswelt sind, bearbeiten sie und werten sie aus. Solche Objekte können z.B. die Personen Ford Prefect und Zaphod Beeblebrox, die Kraftfahrzeuge mit den amtlichen Kennzeichen DO-DUdidelidu und Herz_aus_Gold sowie Kombinationen davon wie z.B. die Paare (Ford Prefect, DO-DU-didelidu) und (Zaphod Beeblebrox, Herz_aus_Gold) sein. (weitere Einzelheiten s. Adams, 1979)

In allen diesen Fällen handelt es sich um Übergänge von einer Zusammenstellung von Klassen zu deren Exemplaren (engl.: instances), der Instantiierung. (Die in diesem Abschnitt vorläufig verwendeten Begriffe "Klasse" und "Exemplar" werden in Kap. 2 beschrieben.) Sowohl für Klassen als auch für Exemplare ist eindeutig die Instantiierungsstufe gegeben, und zwar befindet sich eine Klasse immer auf derjenigen Instantiierungsstufe, die um 1 höher als diejenige ihrer Exemplare ist. Die niedrigste vorkommende Instantiierungsstufe ist 0. Eine Menge von Exemplaren und Klassen derselben Instantiierungsstufe wird hier als ein Diskursmodell (ebendieser Instantiierungsstufe) bezeichnet.

Es soll möglich sein (s. Abb. 1-1),

- die Exemplare eines methodologischen Diskursmodells in Klassen zu überführen und diese zum methodologischen Diskursmodell zu ergänzen
- aus einem methodologischen Diskursmodell ein Meta-CASE-Werkzeug zu erzeugen, mittels dessen man anschließend die Exemplare der Klassen dieses Modells, die Bestand teile eines methodischen Modells sind, instantiiert (sowie bearbeitet und auswertet),

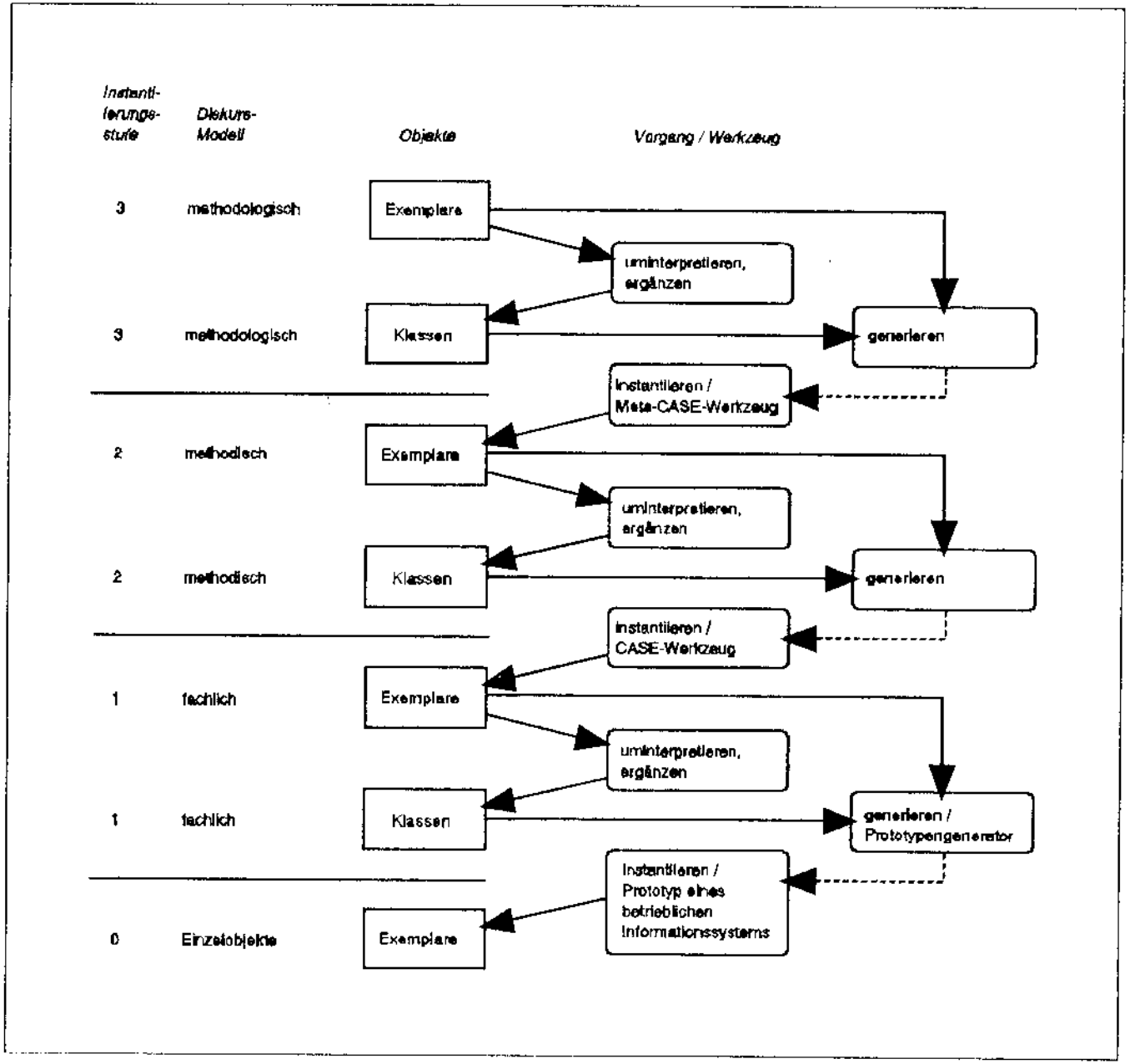

Abb. 1.1: Instantiierungsstufen von Modellen und Objekten sowie Übergänge von einer Instantiierungsstufe zur nächstniedrigeren.

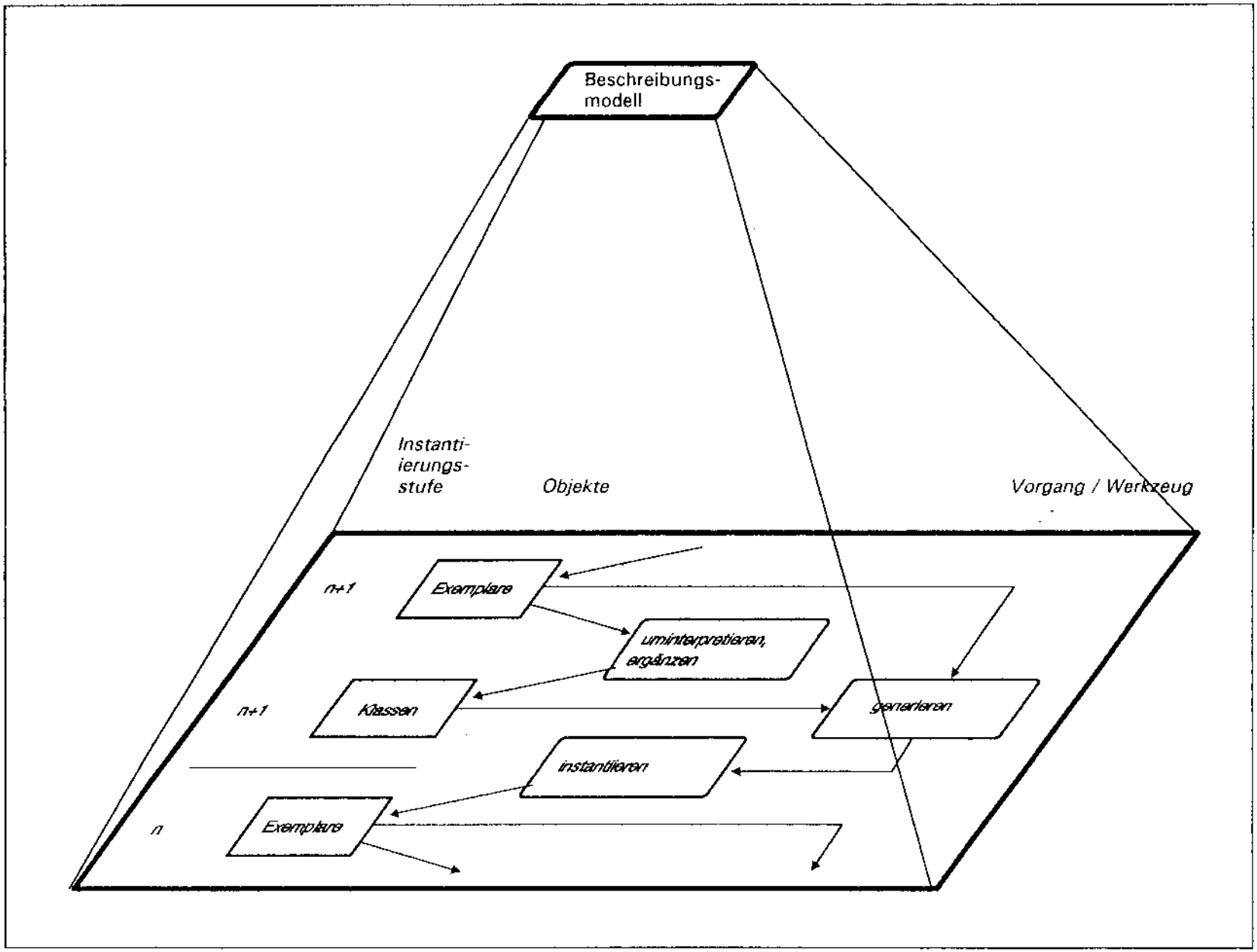

Abb. 1-2: Beschreibungsmodell und Übergangsformalismus

- die Exemplare eines methodischen Diskursmodells in Klassen zu überführen und diese zum methodischen Diskursmodells zu ergänzen
- aus einem methodischen Diskursmodell ein CASE-Werkzeug zu erzeugen, mittels dessen man anschließend die Exemplare der Klassen dieses Modells, die Bestandteile eines fachlichen Modells sind, instantiiert (sowie bearbeitet und auswertet),
- die Exemplare eines fachlichen Diskursmodells in Klassen zu überführen und diese zum fachlichen Diskursmodells zu ergänzen
- aus einem fachlichen Diskursmodell einen Prototypen für ein betriebliches Informationssystem zu erzeugen, mittels dessen man anschließend die Exemplare der fachlichen Klassen, d.s. die Einzelobjekte, instantiiert (sowie bearbeitet und auswertet).

Diese Aufstellung legt nahe, eine einheitliche, für beliebige Instantiierungsstufen geltende Beschreibung der Diskursmodelle und der in ihnen vorkommenden Objekte sowie des Übergangs von einem Diskursmodell der Instantiierungsstufe $n+1$ ($n \geq 0$) zu einem Diskursmodell der nächstniedrigeren Instantiierungsstufe n zu versuchen. Dem scheint zunächst entgegenzustehen, daß sich CASE-Werkzeuge meistens als Editoren von Graphen präsentieren, die Ein-/Ausgabestruktur von betrieblichen Informationssystemen dagegen häufig formularartig ist. Es handelt sich in beiden Fällen lediglich um unterschiedliche Formen der Darstellung zu demselben Zweck, nämlich der Bearbeitung von Exemplardaten.

Gegenstand der vorliegenden Arbeit ist eine einheitliche Beschreibung der Diskursmodelle beliebiger Instantiierungsstufen und des Übergangs von einem Diskursmodell einer beliebigen Instantiierungsstufe $n+1$ ($n \geq 0$) zu einem Diskursmodell der nächstniedrigeren Instantiierungsstufe n. (Abb. 1-2)

Folgende Anforderungen werden an diese Beschreibung gestellt:

Anforderung 1:

> Es gibt ein Modell zur begrifflichen Fassung derjenigen Objekte, die als Bestandteile eines Diskursmodells irgendeiner Instantiierungsstufe auftreten können. Dies ist das Modell zur Beschreibung der Diskursmodelle (Abk.: Beschreibungsmodell) (s. Kap. 2).

Anforderung 2:

> Es gibt einen allgemeinen Formalismus für den Übergang von einem Diskursmodell einer beliebigen Instantiierungsstufe zu einem Diskursmodell der nächstniedrigeren Instantiierungsstufe. Er besteht aus der Beschreibung der Instantiierung von Exemplaren einer Klasse (Kap. 3.), der Beschreibung ihrer Uminterpretation zu Klassen der nächstniederigeren Instantiierungsstufe (Kap. 4) und deren Vervollständigung (Kap. 5).

Anforderung 3:

> Der Übergangsformalismus ist in dem Sinne abgeschlossen, daß keine Information zu dem in Anforderung 2 skizzierten Übergang verwendet wird, die im Formalismus nicht beschrieben wird. Nur wenn diese Anforderung erfüllt ist, ist die Einheitlichkeit des Übergangsformalismus gesichert.

Gegenstand dieser Arbeit sind demnach weder die Generatorprogramme noch Editoren zur Beeinflussung der generierten Software noch die generierten Anwendungssysteme, sondern Diskursmodelle und ein Übergangsformalismus, die den genannten Anforderungen genügen müssen.

Nun ist der Vorgang der Instantiierung von Exemplaren bekannt; immerhin wurden einige Werkzeuge und viele betriebliche Informationssysteme entwickelt.

Jedoch wurde meines Wissens bis heute nicht beschrieben, in welchem Zusammenhang Exemplare und Klassen derselben Instantiierungsstufe stehen. Vielmehr scheint man allgemein davon auszugehen, daß jedes Exemplar einer Instantiierungsstufe $n \geq 1$ genau einer Klasse entspricht. In Kap. 4 wird ein Formalismus zur Uminterpretation von Exemplaren in Klassen angegeben, mittels dessen dieser Zusammenhang verdeutlicht werden soll. Des weiteren gibt es meines Wissens bisher keinen Formalismus für den vollständigen Übergang von den Klassen einer Instantiierungsstufe über ihre Exemplare bis zu den Klassen der nächstniedrigeren Instantiierungsstufe.

1.4. verwandte Arbeiten

In den aus der Literatur bekannten Ansätzen zur semantischen Datenmodellierung und zur Objektmodellierung (z.B. Ansätze zur semantischen Datenmodellierung (Borgida, 1990; Chen, 1976; Hull, King, 1987; Ortner, Söllner, 1989; Peckham, Maryanski, 1988; Sinz, 1990), Strukturierte Analyse (DeMarco, 1979; Raasch, 1991), Ansätze der Objektmodellierung (Rumbaugh u.a. 1991)) wird in keinem mir bekannten Fall ein *formales* Meta-Modell vorgestellt, das die Objekttypen eines fachlichen Modells als Klassen enthalten würde. Eine solcher Objekttyp wäre z.B. in der Objektmodellierungstechnik (OMT) von Rumbaugh u.a. (1991) die ASSOCIATION, deren Exemplare die in einem fachlichen Modell auftretenden zweistelligen Relationen wären. Möchte man ein Werkzeug zur Konzeption nach der OMT herstellen, so benötigt man aber genau eine solche Klasse.

Unter der Bezeichnung "Objekt-Verbindungs-Ansatz" (engl.: object-link approach) werden in der Literatur Modelle von Objekten derselben Instantiierungsstufe beschrieben, die in Form von Graphen dargestellt werden. Die Knoten eines solchen Graphen stehen für Klassen, denn sie können Exemplare haben, wobei ein solches Exemplar je nach Klassenzugehörigkeit entweder ein Knoten oder eine Kante ist.

Ein Modell von Objekten der Instantiierungsstufe 2 des Ansatzes von Habermann und Leymann (1993), der stellvertretend für andere Objekt-Verbindungsansätze untersucht werden soll, besteht aus Knoten der beiden Knotentypen "Rechteck" und "Raute" sowie aus ungerichteten Kanten genau eines Typs. Der Formalismus zum Übergang zu einem Modell der Instantiierungsstufe 1 sieht nun vor, daß Exemplare einer durch ein Rechteck dargestellten Klasse Knoten gleichen Typs sind, während die Exemplare aller durch eine Raute dargestellten Klasse ungerichtete Kanten gleichen Typs sind. Eine Kante steht nicht für eine Klasse, hat also keine Exemplare. Ein Diskursmodell der Instantiierungsstufe 1 kann also Knoten von mehr als zwei Knotentypen und Kanten von mehr als einem Kantentypen enthalten. Unter den so erzeugten Knoten können auch Rechtecke und Rauten sein.

Dieser Ansatz entspricht den hier gestellten Anforderungen aus den folgenden Gründen höchstens zum Teil:

- Es ist nicht angegeben, wie der Übergang von den Objekten der Instantiierungsstufe 1 zu den Einzeldaten (Instantiierungsstufe 0) zu vollziehen ist. Die Modelle der Instantiierungsstufe 1 können Knoten enthalten, die nicht Rechtecke oder Rauten sind. Sie können darüberhinaus Kanten enthalten, die nicht je ein Rechteck und eine Raute miteinander verbinden. Für Knoten bzw. Kanten solcher Typen von Objekten der Instantiierungsstufe 1, die nicht im Diskursmodell der Instantiierungsstufe 2 vorkommen, ist der Übergang zur Instantiierungsstufe 0 nicht erklärt. Anforderung 2 ist also nicht erfüllt.
- Es geht aus einem Diskursmodell dieses Ansatzes nicht hervor, welche durch ein Rechteck dargestellte Klasse Exemplare hat, die ihrerseits als Rechtecke bzw. als Rauten dargestellt werden. Es ist nicht ersichtlich, wie diese Information eingebracht und abgelegt bzw. dargestellt wird, weshalb Anforderung 3 nicht erfüllt ist.
- Benannte Kanten sind nicht darstellbar. Dies führt insbesondere zu Schwierigkeiten bei dem Versuch, Rollen von Komponenten einer Beziehungenklasse darzustellen, die z.B. bei Klassen von rekursiven Beziehungen erforderlich sind. Es ist nicht erklärt, auf welche Weise Objekte wie Spezialisierung / Generalisierung in ein Diskursmodell irgendeiner Instantiierungsstufe eingebracht werden kann. Auch aus diesen Gründen ist Anforderung 3 nicht erfüllt.

2. Modell zur Beschreibung der Diskursmodelle

Das Modell zur Beschreibung der Diskursmodelle wird im folgenden abgekürzt als "Beschreibungsmodell" bezeichnet. Die Abbn. 2-1 bis 2-3 geben einen Überblick.

2.1. Anforderungen

Anforderung 1. (s. Kap. 1.3) bedeutet für die Objekte des Beschreibungsmodells insbesondere:

A.1 Das Beschreibungsmodell soll von einem bestimmten Daten- bzw. Objektmodellierungsansatz unabhängig sein.
Durch das Beschreibungsmodell sollen also möglichst viele Diskursmodelle beschrieben werden. D.h. insbesondere, daß beliebige, aus der Literatur bekannte Modellierungsansätze (s. Abschnitt 1.4) und in Software Engineering-Projekten erarbeitete Konzepte durch das Beschreibungsmodell dargestellt werden können.

A.2 Das Beschreibungsmodell soll für beliebige Instantiierungsstufen gelten.

Alle der in Abschnitt 1.4 genannten Modellierungsansätze enthalten Meta-Objekte, bei denen abgelegt ist, welche Art von Information für die zu ihnen gehörenden Objekte geführt wird, welche Operationen mit diesen Objekten ausgeführt werden können und/oder welche Integritätsbedingungen für diese Objekte gelten sollen. Dies legt die Einführung des Begriffs der Klasse und ihrer Exemplare nahe und zwar in der unten (Kap. 2.2) angegebenen Weise.

Den angesprochenen Modellierungsansätzen ist weiterhin der Gebrauch von Relationen (z.B. als Beziehungenmengen) und Funktionen (z.B. als Attribute) gemeinsam, die im Beschreibungsmodell als Relationenklassen und Funktionenklassen auftreten. Den elementaren Datentypen für Zahlen und Zeichenketten, die ebenfalls in vielen Modellierungsansätzen vorgesehen sind, entsprechen im Beschreibungsmodell vordefinierte Mengen (Zahlen, Texte, Wahrheitswerte). In vielen Fällen sind besondere Teilmengen der vordefinierten Mengen von Zahlen und Texten von Belang; sie werden als Klassen, die auf Zahlen- oder Textmengen basieren, dargestellt.

2.2. Darstellung

Zur Beschreibung der Objekte des Beschreibungsmodells, eben den voranstehend genannten Arten von Klassen, werden elementarmathematische Begriffe wie Mengen und ihre Elemente, Teilmengen, Relationen und Funktionen verwendet. Die Wahl fiel auf diese Darstellung des Beschreibungsmodells, weil sie so allgemein ist, daß die Chance groß ist, damit die Anforderungen A.1 und A.2. zu erfüllen.

2.3. Klassen

Ein Objekt behält seine Identität über den gesamten Existenzzeitraum des Diskursmodells, in dem es vorkommt. (Zur Objektidentität s.z.B. Maier, Zdonik, 1990; Koshafian, Copeland, 1986)

Eine Klasse ist eine Zusammenfassung von Objekten, die zeitlich variabel sein kann. Die so zusammengefaßten Objekte heißen die Exemplare der Klasse. Eine Klasse ist ein Objekt.

Jeder Klasse y ist zu jedem Zeitpunkt die Menge ihrer Exemplare eindeutig zugeordnet; die Exemplarmenge von y zur Zeit t wird mit y_t bezeichnet; sie heißt auch Wertevorrat oder Domäne (engl.: domain) von y zur Zeit t. Es ist immer möglich, festzustellen ob ein Exemplar x zur Zeit t einer Exemplarmenge y_t als Element angehört (hat) oder nicht. Dieser Tatbestand wird ausgedrückt durch $x \in y_t$. Die Exemplarmenge der Klasse y zum aktuellen Zeitpunkt wird mit y, also ohne den Index "t" bezeichnet. Im Rahmen dieser Darstellung führt dies nicht zu Widersprüchen.

Es können also Objekte zu einer Klasse hinzukommen, es können Objekte aus einer Klasse verschwinden.

Die im Beschreibungsmodell auftretenden Zusammenfassungen werden sämtlich als Mengen und nicht als Klassen angesehen. Z.B. ist in der Abb. 2-1 das Rechteck mit der Beschriftung "K. Klasse" das Symbol für die Menge aller Klassen.

Ein Objekt kann Exemplar mehrerer Klassen sein.

Nur Klassen haben Exemplare.

Jedes Exemplar einer Klasse ist eindeutig identifiziert.

Es werden solche Objekte zu Klassen zusammengefaßt, für die dieselben Informationen geführt, dieselben Berechnungen definiert und dieselben Integritätsbedingungen gelten sollen.

Für jede Klasse y ist eine Funktion j mit $j(y) \geq 1$ definiert; sie hat die Bedeutung der Instantiierungsstufe einer Klasse. Für jedes Exemplar x ist eine Funktion i mit $i(x) \geq 0$ definiert, die die Bedeutung der Instantiierungsstufe eines Exemplars hat. Zwischen der Instantiierungsstufe einer Klasse und derjenigen eines ihrer Exemplare soll folgende Beziehung gelten:

$$x \in y_t \Rightarrow i(x) + 1 = j(y)$$

Die niedrigste Instantiierungsstufe (0) ist diejenige, auf der ausschließlich Exemplare, aber keine Klassen vorkommen. Ein Diskursmodell enthält ausschließlich solche Klassen und Exemplare, deren Instantiierungsstufen übereinstimmen; diese Instantiierungsstufe ist auch die Instantiierungsstufe des Diskursmodells.

Dieser Klassenbegriff ist im wesentlichen der der semantischen Datenmodellierung und des objekt-orientierten Ansatzes (Borgida, 1991). Es handelt sich nicht um den Klassenbegriff der formalen Logik.

Es werden Atomklassen und Relationenklassen voneinander unterschieden.

2.4. Atomklassen

Sind die Exemplare einer Klasse y atomar, d.h. nicht Tupel, so ist y eine Atomklasse (s. Abb. 2-1 und 2-2).

Zahlen, Texte und Wahrheitswerte werden als Zeichenatome bezeichnet. Atome, die keine Zeichenatome sind, heißen Konzeptatome (s. Abb. 2-1). Dementsprechend ist eine Atomklasse entweder eine Klasse von Zeichenatomen oder eine Klasse von Konzeptatomen. Eine Klasse von Zeichenatomen ist entweder eine Zahlenklasse, eine Textklasse oder die Klasse der Wahrheitswerte (s. Abb. 2-2). Eine Zahlenklasse kann z.B. auf den reellen oder den ganzen Zahlen basieren, je nach dem, ob ihre Exemplare reelle bzw. ganze Zahlen sind[4].

[4]Eine Klasse, die auf der Klasse der ganzen Zahlen basiert einschließlich der Menge der ganzen Zahlen, ist nicht Element der Menge derjenigen Klassen, die auf den reellen Zahlen basieren. Es ist für das Auftreten der Zahlenklassen als Elemente bestimmter Mengen (des Beschreibungsmodells) unerheblich, inwieweit die Exemplarmenge der einen Teilmenge der Exemplarmenge der anderen ist.

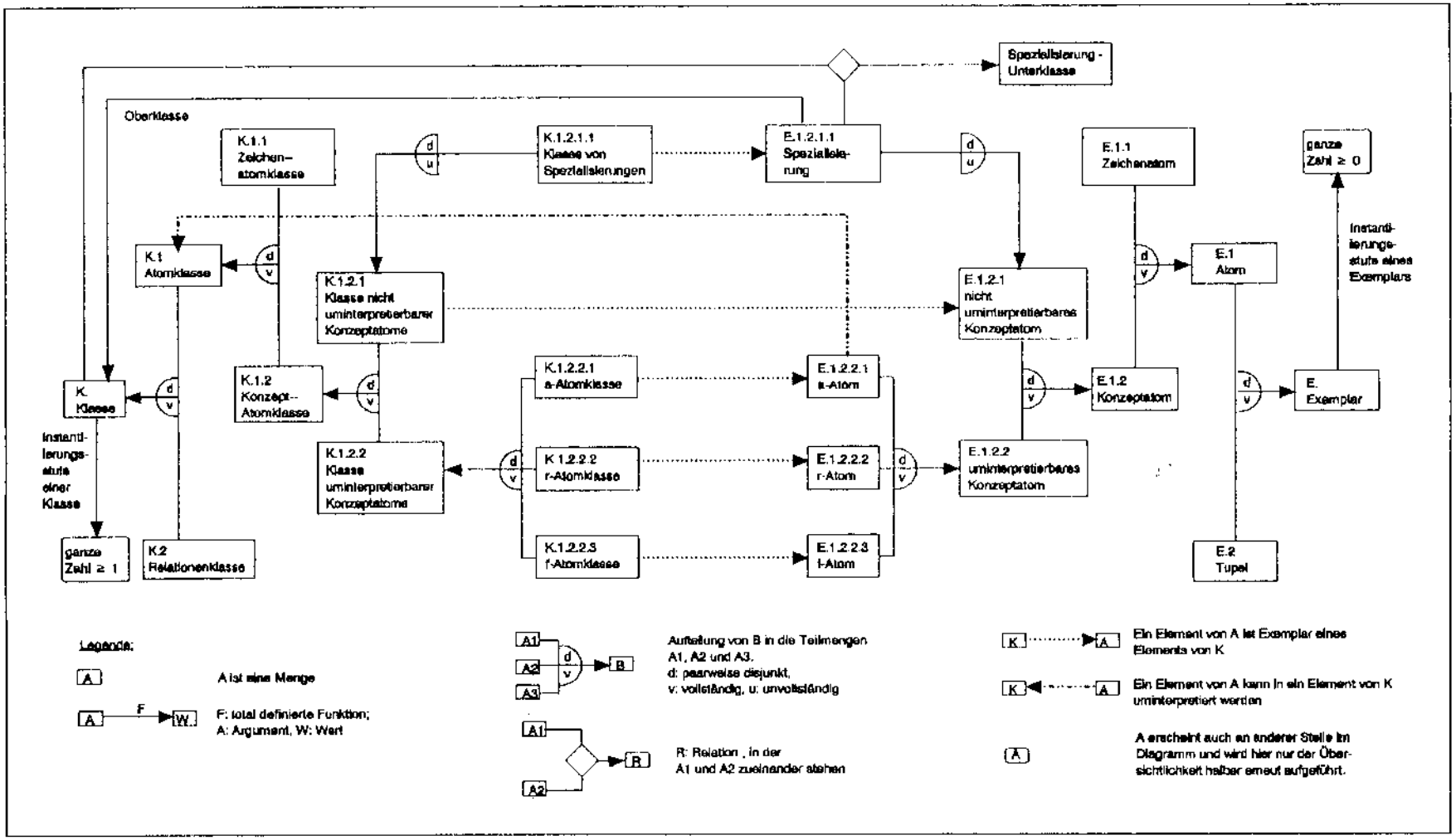

Abb. 2-1: Beschreibungsmodell: Konzeptatomklassen und ihre Exemplare

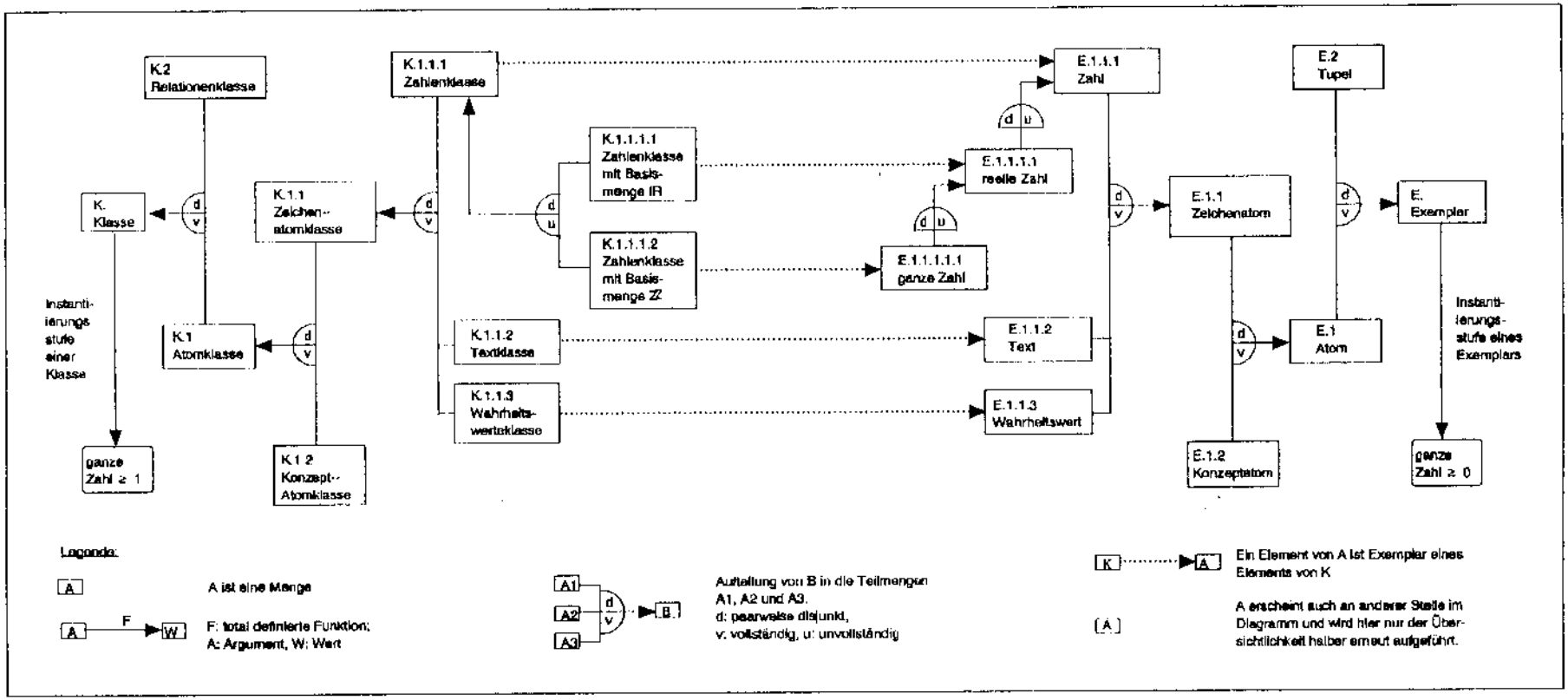

Abb. 2-2: Beschreibungsmodell: Zeichenatomklassen und ihre Exemplare

Unter den Zeichenatomklassen gibt es nun diejenigen Klassen, deren Exemplarmengen die Menge der reellen Zahlen, die Menge der ganzen Zahlen, die Menge aller Texte bzw. die Menge der Wahrheitswerte "wahr "und "falsch" sind. Die Exemplarzusammensetzungen dieser Klassen sind zeitlich nicht variabel. Jedes Diskursmodell enthält höchstens eine Klasse, deren Exemplarmenge die reellen Zahlen, eine Klasse deren Exemplarmenge die ganzen Zahlen, eine Klasse, deren Exemplarmenge die Texte, und eine Klasse, deren Exemplarmenge die Wahrheitswerte "wahr "und "falsch" sind. Man kann die Betrachtung auf diejenigen endlichen Teilmengen dieser z.T. unendlichen Mengen beschränken, die auf einem bestimmten Rechner als Werte der Variablen entsprechender Datentypen darstellbar sind.

Auf der fachlichen Ebene finden wir Atomklassen in Form der sogenannten Basisklassen wie PERSON und ARTIKEL, deren Exemplare für sich allein und ohne Beziehungen zu anderen Exemplaren existenzfähig sind.

Auf der methodischen Ebene finden sich Klassen in den Modellierungsansätzen unter verschiedenen Namen wie z.B. ENTITY-SET (Chen, 1976) oder CLASS (Rumbaugh u.a., 1991).

2.5. Relationenklassen

Eine Relationenklasse (s. Abb. 2-3) ist eine Klasse r, deren Exemplarmenge r_t zur Zeit t eine Teilmenge des kartesischen Produkts (d.h. eine Relation) von Exemplarmengen $y_{1,t}, \ldots, y_{n,t}$ der Klassen $y_1, \ldots, y_n$ ist:

$r_t \subseteq y_{1,t} \times \ldots \times y_{n,t}$

Die Klassen $y_1, \ldots, y_n$ heißen Komponenten von r, sie müssen nicht paarweise voneinander verschieden sein. Unter ihnen können auch Relationenklassen sein. Alle Komponenten y_i einer Relationenklasse r haben dieselbe Instantiierungsstufe wie r.

Die Exemplarmenge von r zum aktuellen Zeitpunkt ist:

$r \subseteq y_1 \times \ldots \times y_n$

Ein Exemplar der Relation r ist das n-Tupel $(x_1, \ldots, x_n)$,
wobei zur Zeit t
$x_1 \in y_{1,t}, \ldots, x_n \in y_{n,t}$
bzw. zum aktuellen Zeitpunkt
$x_1 \in y_1, \ldots, x_n \in y_n$
ist.

Eine zweiseitige Relationenklasse $r_t \subseteq y_{1,t} \times y_{1,t}$, deren Komponenten identisch sind, heißt eine rekursive Relationenklasse. Es kann sich dabei entweder um eine nicht-Projektionenklasse oder um eine nicht-Funktionenklasse handeln.

Beispiele für Relationenklassen auf der fachlichen Ebene sind z.B. ZUGEHÖRIGKEIT_EINES_MITARBEITERS_ZU_EINER_ABTEILUNG oder ZUSAMMENSETZUNG_EINER_KOMPONENTE_AUS_KOMPONENTEN.

2.5.1. Funktionenklassen und nicht-Funktionenklassen

Eine Funktionenklasse (s. Abb. 2-3) ist eine Relationenklasse f, deren Exemplarmenge zu jeder Zeit t eine Funktion f_t: $y_{1,t} \rightarrow y_{2,t}$ ist.

Eine Funktion f_t: $y_{1,t} \rightarrow y_{2,t}$ kann sein:

- eine total definierte Funktion f^1_t: $y_{1,t} \rightarrow y_{2,t}$
  zu jedem Zeitpunkt t gilt: zu jedem $x_1 \in y_{1,t}$ gibt es genau ein $x_2 \in y_{2,t}$
  mit $(x_1, x_2) \in f^1_t$
- eine partiell definierte Funktion f^c_t: $y_{1,t} \rightarrow y_{2,t}$
  zu jedem Zeitpunkt t gilt: zu jedem $x_1 \in y_{1,t}$ gibt es höchstens ein $x_2 \in y_{2,t}$
  mit $(x_1, x_2) \in f^c_t$

Fehlt der obere Index "1" bzw. "c" am Funktionensymbol, so ist über die Totalität der Abbildung nichts ausgesagt.

Für den aktuellen Zeitpunkt ergibt sich die bekannte Schreibweise::

$f: y_1 \rightarrow y_2$.

Die Funktionswerte können entweder explizit genannt oder durch eine Zuordnungsvorschrift berechnet werden.

y_1 heißt Argumentenklasse, y_2 heißt Werteklasse der Funktionenklasse f. Sowohl die Argumentenklasse wie auch die Werteklasse einer Funktionenklasse können Relationenklassen sein.

Soll eine Reihenfolge der Argumente der Exemplare einer Funktionenklasse f mit demselben Wert festgehalten werden, so werden diese Argumente als Komponenten eines Vektors geführt, die in den Beipieldiagrammen durch einen Vollkreis in der Nähe des Anfangs des Pfeilsymbols für f als solche kenntlich gemacht werden. (Eine explizit vektorwertige Auswertung ist immer dann der mengenwertigen vorzuziehen, wenn gleiche Objekte mehrmals im Auswertungsergebnis enthalten sein können und auch erhalten bleiben sollen.)

Eine nicht-Funktionenklasse ist eine Relation, die keine Funktionenklasse ist.

Das Symbol für eine nicht-Funktionenklasse in den Beispieldiagrammen besteht aus einer Raute, einem Rechteck mit dem Namen der nicht-Funktionenklasse und einem Pfeil, der von der Raute zum Rechteck gerichtet ist. (Diese Notation der nicht-Funktionenklassen geht zurück auf Ortner und Söllner, 1989.)

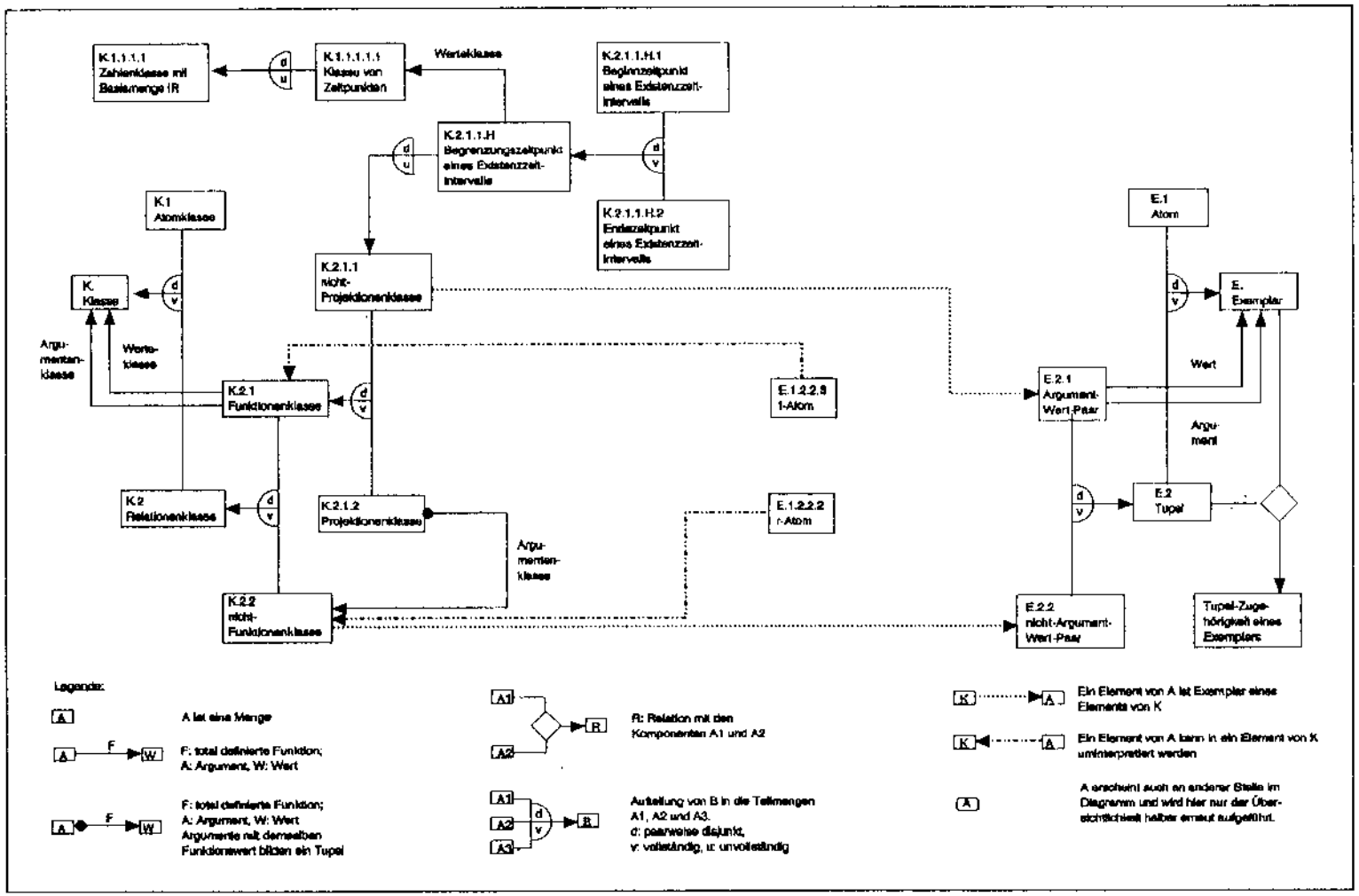

Abb. 2-3: Beschreibungsmodell: Relationenklassen und ihre Exemplare

2.5.2. Projektionenklassen und nicht-Projektionenklassen

Eine Projektionenklasse (s. Abb. 2-3) p_k^1 ist eine Funktionenklasse, deren Exemplarmenge zur Zeit t die Projektion einer Relation r_t auf ihre k-te Komponente $y_{k,t}$ ist:

$p_{k,t}^1$: $r_t \rightarrow y_{k,t}$

Die Argumentenklasse einer Projektionenklasse ist immer eine nicht-Funktionenklasse. Die Projektionenklassen einer nicht-Funktionenklasse bilden ein Tupel, d.h. ihre Reihenfolge ist von Bedeutung. Eine nicht-Funktionenklasse darf nicht als eine ihrer eigenen Komponenten auftreten, noch darf es einen sonstigen Verweis von ihr zu einer ihrer Komponenten geben. Alle Projektionenklassen sind total definiert.

Eine Projektionenklasse wird in den Beispieldiagrammen durch eine ungerichtetete Kante zwischen einer Raute und einem Rechteck dargestellt, die das Symbol der Argumentenklasse der Projektionenklasse (eine nicht-Funktionenklasse) mit dem Symbol ihrer Werteklasse verbindet.

Eine Funktionenklasse, die keine Projektionenklasse ist, ist eine nicht-Projektionenklasse. Die Exemplare einer nicht-Projektionenklassen sind Argument-Wert-Paare, d.s. Paare, deren erste Komponente (das Argument) ein Exemplar der Argumentenklasse und dessen zweite Komponente (der Wert) ein Exemplar der Werteklasse der nicht-Projektionenklasse ist. (Die Exemplare der Projektionenklassen werden nicht betrachtet.)

Eine nicht-Projektionenklasse wird in einem Beispieldiagramm als Pfeil dargestellt, der von dem Symbol für die Argumentenklasse der nicht-Projektionenklasse zum Symbol für ihre Werteklasse gerichtet ist, und der mit dem Namen der Funktion beschriftet wird. Die Totalität der Abbildung wird mit den Symbolen für die Kardinalitätenbeschränkungen angegeben. Eineindeutige Abbildungen werden durch Doppelpfeile dargestellt. Handelt es sich um eine Funktionenklasse, deren Werte berechnet werden, so wird dies durch das Zeichen "(calc)" zum Ausdruck gebracht.

2.6. Spezialisierung/Generalisierung

Eine Spezialisierung/Generalisierung (kurz: Spezialisierung) hat eine Oberklasse und eine oder mehrere Unterklassen. Die Exemplare der Unterklassen gehören auch der Oberklasse an; für die Unterklassen sind dieselben Funktionen und Integritätsbedingungen definiert wie für die Oberklasse (Vererbung).

Eine Klasse kann Unterklasse mehrerer Spezialisierungen sein (Mehrfachvererbung).

Existiert eine Spezialisierung s mit der Oberklasse y_2 und der Unterklasse y_1, so ist y_2 unmittelbare Oberklasse von y_1. Eine Klasse y_n ist Oberklasse von y_1, wenn y_n unmittelbare Oberklasse von y_1 ist oder wenn sie Oberklasse der unmittelbaren Oberklasse von y_1 ist. Die unmittelbare Unterklasse und die Unterklasse einer Klasse sind analog definiert.

Ist eine Klasse Oberklasse mehrerer Spezialisierungen $s_1, \ldots, s_n$, so lassen sich diese mehrfachen Spezialisierungen $s_1, \ldots, s_n$ in fortgesetzte Spezialisierungen umwandeln, indem man jede Unterklasse von s_1 zur Oberklasse von s_2 macht; jede Unterklasse von s_2 wird zur Oberklasse von s_3 usw. Überführt man so alle mehrfachen Spezialisierungen eines Diskursmodells in fortgesetzte Spezialisierungen, so ergeben sich einer oder mehrere azyklische Graphen, deren Kanten von der Oberklasse zu einer Spezialisierung und von dort zu ihren Unterklassen gerichtet sind. Dies bedeutet, daß eine Klasse y_1 nicht zur gleichen Zeit Oberklasse und Unterklasse einer Klasse y_2 ist.

Durch eine Spezialisierung kann die Exemplarmenge ihrer Oberklasse in paarweise disjunkte Exemplarmengen der Unterklassen aufgeteilt werden. In diesem Fall heißt die Spezialisierung disjunkt, sonst nicht-disjunkt oder überlappend. Außerdem kann durch eine Spezialisierung die Exemplarmenge einer Oberklasse vollständig in die Exemplarmengen der Unterklassen aufgeteilt werden. In diesem Fall gehört jedes Exemplar der Oberklasse mindestens einer Unterklasse an; die Spezialisierung heißt dann vollständig, sonst unvollständig.

Eine Spezialisierung s ist also ein Exemplar einer Klasse von Spezialisierungen. Oberklasse und Unterklassen von s haben dieselbe Instantiierungsstufe wie s, die mindestens 1 ist. Jede Spezialisierung innerhalb eines Diskursmodells gehört mindestens einer Klasse von Spezialisierungen an. Gibt es in einem Diskursmodell genau eine Klasse von Spezialisierungen, so muß diese nicht in den Diagrammen dargestellt werden.

Oberklasse und alle Unterklassen einer Spezialisierung sind Elemente derselben Menge von Klassen des Beschreibungsmodells. Sie sind also z.B. sämtlich Klassen, Atomklassen, nicht-Funktionenklassen oder Textklassen.

Hat eine nicht-Funktionenklasse r die Projektionenklassen

$p_{k,t}^1$: $r_t \rightarrow y_{k,t}$

so gibt es für jede Unterklasse r_1 von r die Projektionenklassen

$p_{1k,t}^1$: $r_{1,t} \rightarrow y_{1k,t}$

wobei y_{1k} eine Unterklasse von y_k ist.

In den Beispieldiagrammen wird eine Spezialisierung als Halbkreis dargestellt, in dessen linker Hälfte angegeben wird, ob sie disjunkt (d) oder überlappend (o) ist, und in dessen rechter Hälfte angegeben wird, ob sie vollständig (v) oder unvollständig (u) ist. Ein Pfeil ist vom Spezialisierungssymbol zum Symbol für die Oberklasse gerichtet; das Spezialisierungssymbol und die Symbole für die Unterklassen sind durch ungerichtete Kanten miteinander verbunden. Hat eine Spezialisierung genau eine Unterklasse, so ist die Spezialisierung unvollständig und die rechte Hälfte des Halbkreises, durch den die Spezialisierung dargestellt werden kann, kann leergelassen werden.

Man könnte daran denken, Spezialisierungen als nicht-uminterpretierbare Konzeptatome aufzufassen, die im Beschreibungsmodell nicht besonders aufgeführt werden und die mittels des Übergangsformalismus erzeugt werden. Da dieses Instantiierungsmuster (Spezialisierung mit Oberklasse und Unterklassen) allerdings auf allen Instantiierungsstufen (≥ 1) in derselben Weise auftritt, werden Spezialisierungen als Exemplare innerhalb des Beschreibungsmodells allgemein abgehandelt.

2.7. Historien

Sowohl eine Klasse als auch ein Exemplar kann historisch geführt werden (historische Klasse). Man bildet dadurch ab, daß ein Auftreten eines Objekts einen Anfang und ein Ende hat und evtl. mehrfach vorkommt. Jedes dieser Existenzzeitintervalle wird durch einen Zeitpunkt seines Beginns und einen Zeitpunkt seines Endes charakterisiert, wobei der Endezeitpunkt eines Zeitintervalls später liegt als sein Beginnzeitpunkt.

Ein Exemplar kann nur innerhalb eines Existenzzeitintervalls seiner Klasse vorkommen. Zu jedem Zeitpunkt seiner Existenzzeit muß es alle dann geltenden Integritätsbedingungen seiner Klasse erfüllen. Das bedeutet insbesondere, daß ein Exemplar einer Unterklasse einer Spezialisierung nur innerhalb desjenigen Zeitintervalls existieren kann, das als sein Existenzzeitintervall bei der Oberklasse abgelegt ist.

Eine Klasse existiert in denjenigen Existenzzeitintervallen, in denen das Exemplar existiert, aus dem sie durch Uminterpretation entsteht. Eine Klasse der höchsten betrachteten Instantiierungsstufe existiert innerhalb des Existenzzeitraums des abgebildeten Systems und wird nicht historisch geführt.

Historische Klassen, deren Exemplare einmalig auftreten, können in Diagrammen durch ein "(H, 1)", solche deren Exemplare mehr als einmal auftreten können, durch ein "(H, *)" gekennzeichnet werden.

Jede historische Klasse h ist Argumentenklasse von zwei Funktionenklassen b und e, deren Werteklassen Zeitangaben sind. (Wir nehmen an, daß Zeitpunkte als Julianische Tageszahlen angegeben werden und daß die Algorithmen zur Umwandlung in Zeitangaben gemäß einem bestimmten Kalender z.B. dem Gregorianischen bekannt seien.) b dient dem Nachhalten der Beginnzeitpunkte, e dient dem Nachhalten der Endezeitpunkte der Existenzzeitintervalle eines Exemplars von h. b bzw. e können durch einen Pfeil dargestellt werden, der von der Argumentenklasse zur Werteklasse (Klasse von Zeitangaben) gerichtet ist und durch ein "(+)" für Beginnzeitpunkte (b) bzw. ein "(-)" für Endezeitpunkte (e) gekennzeichnet ist. (s.a. Elmasri u.a., 1991)

2.8. Beispiel

In den Beispieldiagrammen werden die oben genannten Symbole verwendet, die Ähnlichkeiten mit den Symbolen des Beschreibungsmodells (Abb. 2-1 bis 2-3) aufweisen. Diese Symbolik ist allerdings frei wählbar.

Das hier vorgestellte Beispiel (s. Abb. 2-4) geht zum großen Teil auf das Beispiel aus einer der ersten Veröffentlichungen zur semantischen Datenmodellierung (Chen, 1976) zurück. Es handelt sich um Teile eines fachlichen (Spielzeug-)Modells (Instantiierungsstufe 1). Die Klassen PROJEKT, MITARBEITER, TEILEART, LIEFERANT, PROJEKT, KUNDE und GESCHÄFTSPARTNER sind Konzeptatomklassen.

PROJEKTMITARBEIT, ZUSAMMENSETZUNG und PROJEKT-LIEFERANT-TEILEART sind nicht-Funktionenklassen (s. Abb. 2-4). PROJEKT und MITARBEITER sind die Projektionenklassen von PROJEKTMITARBEIT. BAUGRUPPE und KOMPONENTE sind die Projektionenklassen von ZUSAMMENSETZUNG. Die Projektionenklassen von PROJEKT-LIEFERANT-TEILEART sind nicht namentlich erwähnt.

Die Klassen PERSONAL-NR, NAME, VORNAME, ADRESSE, STRASSE_HAUSNR, POSTLEITZAHL, ORT, GEBURTSDATUM, ALTER, BEGINN_DER_PROJEKTARBEIT, ENDE_DER_PROJEKTARBEIT und VERFÜGBARKEIT (s. Abb. 2-5) sind nicht-Projektionenklassen. Der Pfeil bezeichnet jeweils die Richtung von ihrer Argumenten- zu ihrer Werteklasse. NATÜRLICHE_ZAHL kann entweder als eigene Zahlenklasse oder als auf ZZ basierende Zahlenklasse eingeführt werden. FREMDSPRACHE ist eine Textklasse. ADRESSENKLASSE ist eine Konzeptatomklasse. Der Pfeil mit der Beschriftung "Fremdsprachenkenntnis" steht für eine nicht-Funktionenklasse, deren Projektionenklassen nicht genannt werden. Der Pfeil mit zwei Spitzen, der mit "Personal-Nr" beschriftet ist, besagt, daß es sich bei dieser nicht-Projektionenklasse um eine eineindeutige Abbildung handelt, durch deren Werte sich die Exemplare der Klasse Mitarbeiter eindeutig identifizieren lassen.

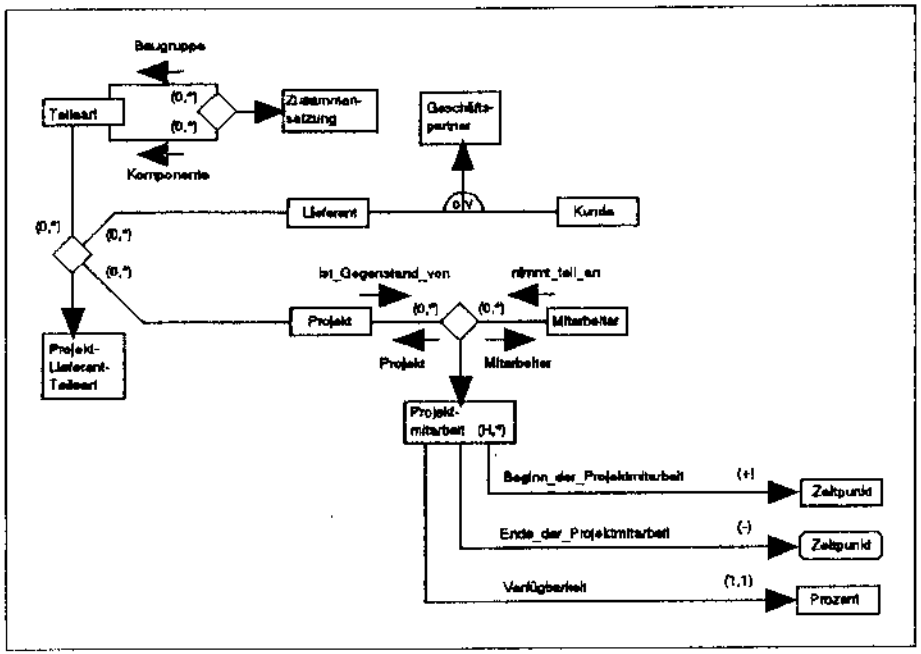

Abb. 2-4: fachliches Diskursmodell, Teil 1

In Abb. 2-4 wird ein Sinnbild für eine Spezialisierung / Generalisierung gezeigt, die nicht namentlich genannt wird. Ihre Oberklasse ist die Klasse GESCHÄFTSPARTNER, ihre Unterklassen sind KUNDE und LIEFERANT. Da ein Geschäftspartner sowohl als Kunde wie auch als Lieferant auftreten kann, sind die Exemplarmengen der Unterklassen nicht disjunkt (Symbol "o" für overlapping). Außer Kunden und Lieferanten soll es keine weiteren Geschäftspartner geben, weshalb die Spezialisierung als vollständig (Symbol "v") bezeichnet wird.

Es ist also festzustellen, daß Daten- bzw. Objektmodellierungsdiagramme wie das im Beispiel vorgestellte sowohl Klassen als auch Exemplare derselben Instantiierungsstufe enthalten können. Z.B. ist Mitarbeiter eine Klasse, während die durch das Halbkreissymbol dargestellte Spezialisierung ein Exemplar ist. Alle Objekte der in den Abbn. 2-4 und 2-5 dargestellten Diagramme haben die Instantiierungsstufe 1. Mittels eines solchen Daten- bzw. Objektmodellierungsdiagramms stellt man also ein Diskursmodell dar.

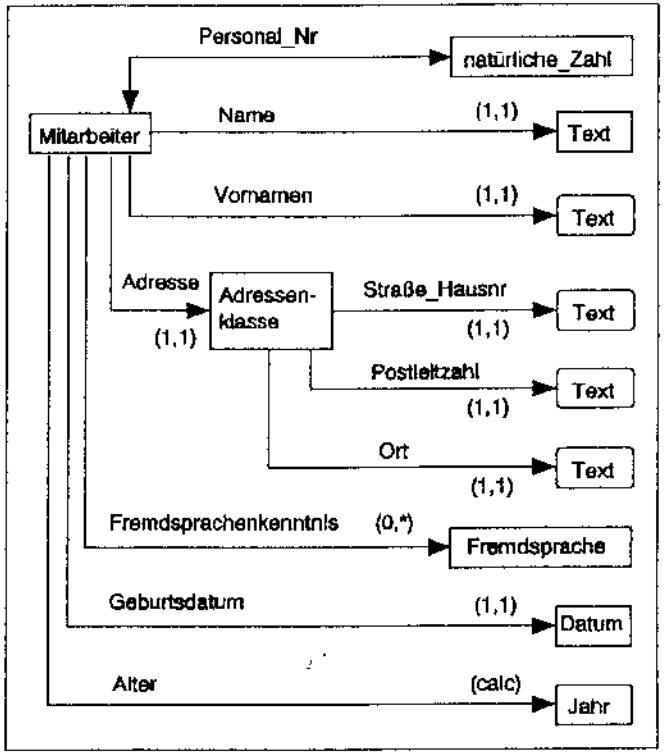

Abb. 2-5: fachliches Diskursmodell, Teil 2

Darüberhinaus ist ersichtlich, daß es durchaus Diagramme geben kann, in denen nicht alle vorhandene Information wiedergegeben wird; z.B. werden hier nur sporadisch Projektionenklassen aufgeführt. Jedes Diagramm bietet also eine mehr oder minder eingeschränkte Sicht auf das zugrunde liegende Diskursmodell.

Die in den Abbildungen vorkommenden Symbole "(a, b)" werden im Abschnitt 2.11. (Kardinalitätenbeschränkungen) erklärt.

Bisher wurden die im Beispiel vorkommenden Objekte durch das Beschreibungsmodell erklärt. Auf welche Weise sie als Exemplare der Klassen der nächsthöheren Instantiierungsstufe angesehen werden können, soll im Kap. 3. gezeigt werden.

2.9. Berechnungen

(Die Syntax für Berechnungen wird in Kratz (1993) angegeben.)

Wie gesagt, sind Funktionswerte entweder empirisch gegeben oder sie werden berechnet. Basiert die Werteklasse einer nicht-Projektionenklasse mit berechneten Werten auf der Menge der Wahrheitswerte, so ist die Berechnung ein Prädikat. Basiert sie auf einer Zahlenmenge, so wird die Berechnung als arithmetischer Ausdruck geschrieben. Analog gibt es Textausdrücke für die algorithmische Bearbeitung von Texten. Als Variablen in allen diesen Ausdrücken treten Exemplare und Verweise von Exemplaren auf Exemplare auf, die Wahrheitswerte, Zahlen oder Texte sein können.

Insbesondere kann man Berechnungen benutzen, um den Verweis von den Exemplaren einer Klasse über eine nicht-Funktionenklasse auf die Exemplare einer (evtl. weiteren) Klasse zu definieren. Sei z.B. vorausgesetzt, daß es eine nicht-Projektionenklasse ARBEITET_IN gibt, deren Argumentenklasse MITARBEITER und deren Werteklasse PROJEKT ist, könnte eine Beispielauswertung folgendermaßen geschrieben werden:

arbeitet_in(m ∈ Mitarbeiter : m.nimmt_teil_an.Projekt)

Dabei verweist NIMMT_TEIL_AN von den Exemplaren von MITARBEITER auf die Exemplare der nicht-Funktionenklasse PROJEKTMITARBEIT, und PROJEKT ist die Projektionenklasse, die von PROJEKTMITARBEIT auf die Konzeptatomklasse PROJEKT verweist. Es versteht sich, daß m eine Exemplarvariable für die Exemplare der Klasse MITARBEITER ist. Es gilt dann

m.arbeitet_in = m.nimmt_teil_an.Projekt.

Von besonderer Bedeutung für die beabsichtigten Diskursmodelle ist die Berechnung card(u), deren Ergebnis die Mächtigkeit der Menge oder des Vektors u ist und die auf allen Instantiierungsstufen für alle Mengen und Vektoren definiert ist.

2.10. Integritätsbedingungen

Zu jeder Klasse gibt es eine Menge von Integritätsbedingungen, die für alle ihre Exemplare gelten sollen. Eine Integritätsbedingung (IB), die für alle Exemplare einer Klasse y erfüllt sein soll, läßt sich darstellen als ein Tupel (P_n, P_{n-1}, ... , P_1, P_0) von Prädikaten über den Exemplaren von y mit folgender Bedeutung: Wenn für ein Exemplar x von y der Reihe nach die Prädikate P_n, P_{n-1}, ... , P_1 erfüllt waren, so gilt P_0 ab der nächsten Änderung, durch die P_1 nicht mehr erfüllt ist. Besteht das Tupel ausschließlich aus dem Prädikat P_0, so gilt dieses immer und unabhängig vom Vorzustand. Solche speziellen Integritätsbedingungen bezeichnet man als statisch; sie treten bei weitem häufiger auf als die Integritätsbedingungen mit Vorbedingungen, die Zustandsübergangsbedingungen heißen. (Zustandsübergangsbedingungen mit mehr als einem Vorzustand sind bei betrieblichen Informationssystemen außerordentlich selten.)

Darüberhinaus kann man unterscheiden zwischen expliziten IB.en, die sich auf Objekte eines bestimmten Diskursmodells beziehen - hierzu gehören z.B. die Wertebereichsbeschränkungen -, und den impliziten IB.en, die für Objekte einer bestimmten Instantiierungsstufe und aller höheren Instantiierungsstufen gelten. Diese können in generischer Weise angegeben werden und gehören damit zum Beschreibungsmodell. Zu den impliziten Integritätsbedingungen gehören

- die Kardinalitätenbeschränkungen,
- die Menge-Teilmenge-Beziehung der Exemplarmengen von Oberklasse und Unterklasse einer Spezialisierung,
- die Vererbung aller Auswertungen, Berechnungen, Prädikate und Integritätsbedingungen von der Oberklasse einer Spezialisierung auf ihre Unterklassen
- die Integritätsbedingungen für u.a. zyklische Auswertungen, wie sie z.B. durch rekursive Relationenklassen ermöglicht werden sowie
- die Integritätsbedingungen für Existenzzeitintervalle.

Unter Verwendung der Auswertung card(u) lassen sich auch Integritätsbedingungen für die Anzahl der Exemplare einer Klasse formulieren. Dabei ist u eine Menge.

Ein Beispiel für eine explizite Integritätsbedingung im Modell der Abb. 2-4 sei: "Ein Projektmitarbeiter darf zu keiner Zeit mit mehr als 120 % seiner wöchentlichen Arbeitszeit in Projekten mitarbeiten."

2.11. Kardinalitätenbeschränkungen

Es kann erforderlich sein, festzusetzen, wieviele Elemente eine Menge mindestens und höchstens enthalten darf. Besonders häufig und in Diskursmodellen aller Instantiierungsstufen ≥ 1 bezieht sich diese Integritätsbedingung auf das Ergebnis von mengenwertigen Auswertungen. Sie werden aber auch benutzt, um partiell definierte und total definierte Funktionenklassen als solche zu kennzeichnen.

Für jede Auswertung $a \in A_t$ (A_t: Menge der in den betrachteten Diskursmodellen zum Zeitpunkt t vorhandenen Auswertungen) ist eine total definierte Funktion

$card\_min^1$: $A \rightarrow \mathbb{N}_0$

sowie eine partiell definierte Funktion

$card\_max^c$: $A \rightarrow \mathbb{N}$

definiert mit folgender Bedeutung:

(card_min(a) = 1 $\wedge$ card_max(a) = 1) $\Rightarrow$ a ist eine total definierte Funktionenklasse

(card_min(a) = 0 $\wedge$ card_max(a) = 1) $\Rightarrow$ a ist eine partiell definierte Funktionenklasse

card_max(a) > 1 $\Rightarrow$
(a ist eine mengenwertige Auswertung
$\wedge$ ($\forall$(x $\in$ Argumentenklasse(a) :
(card(x.a) $\geq$ card_min(a) $\wedge$ card(x.a) $\leq$ card_max(a)))

Wenn card_max(a) existiert, so gilt: card_min(a) $\leq$ card_max(a). Hat card_max(a) keinen Wert, so ist die Mächtigkeit eines Auswertungsergebnisses x.a nach oben nicht beschränkt.

In den Beispieldiagrammen (s. Abbn. 2-4 und 2-5) werden Kardinalitätenbeschränkungen einer Auswertung a durch Paare (min, max) ausgedrückt, wobei min = card_min(a) ist. Wenn max = "*", so ist card_max(a) nicht gegeben (die Mächtigkeit der Auswertung ist nach oben nicht beschränkt), sonst ist max = card_max(a). (s. Schlageter, Stucky, 1983) Und zwar ersieht man aus diesen Diagrammen die Kardinalitätenbeschränkungen von nicht-Projektionenklassen und von Umkehrauswertungen von Projektionenklassen, zu denen auch FREMDSPRACHENKENNTNIS gehört. Projektionenklassen haben immer die Kardinalitätenbeschränkung (1, 1), die deshalb bei ihnen nicht aufgeführt wird.

2.12. Implizite Integritätsbedingungen für historische Klassen

(1) Ein Exemplar einer historischen Klasse kann nicht während seiner Existenzzeit ein zweites Mal auftreten.

(2.a) Gibt es zu jedem Zeitpunkt für jedes Exemplar x_t einer Klasse y_t genau ein Tupel x einer Relationenklasse y, so decken die Existenzzeitintervalle der x das Existenzzeitintervall von x_t vollständig und paarweise disjunkt ab (lückenlose Historie).

(2.b) Gibt es zu jedem Zeitpunkt für jedes Exemplar x_t einer Klasse y_t höchstens ein Tupel x einer Relationenklasse y, so decken die Existenzzeitintervalle der x das Existenzzeitintervall von x_t paarweise disjunkt, aber nicht notwendigerweise vollständig ab (Historie mit Lücken).

LITERATUR

D. Adams: The Hitch Hiker's Guide to the Galaxy. Pan Books, London, 1979

A. Borgida: Knowledge Representation, Semantic, Modelling: Similarities and Differences. *in:* Entity-Relationship Approach: The Core of Conceptual Modelling. Proc. 9th Int. Conf. on the Entitity-Relationship Approach, Lausanne, 8. - 10. 10. 1990, H. Kangassalo (Hrsg.), North Holland, Amsterdam, 1991, S. 1 - 24

P.P.-S. Chen: The Entity-Relationship Model - Toward a Unified View of Data. ACM Trans. Database Systems *1* (1976), 9 - 30

T. DeMarco: Structured Analysis and System Specification. Prentice-Hall, Englewood Cliffs, N.J., 1979

R. Elmasri, I. El-Assal, V. Kouramajian: Semantics of temporal data in an extended ER model. *in:* Entity-Relationship Appraoch - The Core of Conceptual Modelling. Proc. 9th Int. Conf. on the Entitity-Relationship Approach, Lausanne, 8. - 10. 10. 1990, H. Kangassalo (Hrsg.), North Holland, Amsterdam, 1991, S. 239 - 254

H.-J. Habermann, F. Leymann: Repository. Oldenbourg, Mnnchen, 1993

R. Hull, R. King: Semantic Database Modelling: Survey, Applications, and Research Issues. ACM Computing Surveys *19* (1987), 201 - 260

S. Koshafian, G.P. Copeland: Object Identity. ACM Proc. Conf. on Object-Oriented Programming Systems, Languages, and Applications. Portland, Oregon, September 1986. Nachdruck *in:* Readings in Object-Oriented Database Systems (S.B. Zdonik, D. Maier; Hrsg.), Morgan Kauffmann, San Mateo, Ca., 1990, S. 37 - 46

G. Kratz: Ein Ansatz zur einheitlichen Darstellung der Objekte beliebiger Instantiierungsstufen und des Zusammenhangs zwischen ihnen. 1. Zwischenbericht zum Forschungsprojekt "Die Begriffsmaschine", 1993 (internes Papier, Veröffentlichung in Vorbereitung)

D. Maier, S.B. Zdonik: Fundamentals of Object-Oriented Databases. *in:* Readings in Object-Oriented Systems (S.B. Zdonik, D. Maier; Hrsg.), Morgan Kauffmann, San Mateo, Ca., 1990, S. 1 - 32

E. Ortner, B. Söllner: Semantische Datenmodellierung nach der Objekttypenmethode. Informatik-Spektrum *12* (1989), 31 - 42

J. Peckham, F. Maryanski: Semantic Data Models. ACM Computing Surveys *20* (1988), 153 - 189

J. Raasch: Systementwicklung mit Strukturierten Methoden. Hanser, München, 1991

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

G. Schlageter, W. Stucky: Datenbanksysteme: Konzepte und Modelle. Teubner, Stuttgart, 1983

E. Sinz: Das Entity-Relationship-Modell (ERM) und seine Erweiterungen. Handb. d. modernen Datenverarbeitung, Heft 152 (1990), 17 - 29