Formale Modellierung mit Rollen

Habilitationsschrift
von

Friedrich Steimann

dem

Fachbereich für Elektrotechnik und Informationstechnik
der
Universität Hannover

vorgelegt
zur Erlangung
der

Venia legendi

für
das Fachgebiet

Angewandte Informatik

VORWORT

Die Idee, Rollen als Modellierungskonstrukt zu untersuchen, trage ich seit meiner Zeit als Softwareingenieur bei ALCATEL (1991–1992) mit mir herum. Damals beschäftigten wir uns mit objektorientierten Analyse- und Designmethoden, und Rollen lagen irgendwie in der Luft. Dabei war das Thema schon zu der Zeit alles andere als neu – vielmehr hatten bereits Codd [1970] und wenig später Falkenberg [1976] sowie Bachman und Daya [1977] jeweils verschiedene Rollenbegriffe in die Modellierung eingeführt.

Das erste Mal wieder aufgegriffen habe ich das Thema, als ich mich mit konzeptuellen Graphen befaßte und versuchte, formalisierte radiologische Befunde mittels einer Dependenzgrammatik in ihre kanonischen Bestandteile zu zerlegen [Steimann 1996]. Sowa hatte in seinen Buch [1984] Rollen erwähnt und formlos von natürlichen Typen abgegrenzt, sie aber nicht systematisch für die Modellierung eingesetzt. Am Beispiel der Analyse von Sätzen einer formalisierten Sprache zeigte sich jedoch, daß die mangelnde Unterscheidung von Rollen und natürlichen Typen zu erheblichen Problemen führt. Obwohl ich schon damals eine ziemlich konkrete Vorstellung davon hatte, wie Rollen definiert sein müßten, war mir die fundamentale Bedeutung des Konzepts für die Modellierung und deren Sprachen noch lange nicht klar.

Die Gelegenheit, die Grundlagen einer sinnvollen Rollendefinition für die Modellierung zu untersuchen, ergab sich für mich erst, als ich an meiner jetzigen Wirkungsstätte wieder mit der Softwaretechnik betraut und Modellierung generell ein Thema der Institutsarbeit wurde. Als ich dann beschloß, Rollen zum Gegenstand meiner Habilitationsbestrebungen zu machen, hatte ich keine Ahnung, was auf mich zukommen würde – zum Glück, denn hätte ich es gewußt, ich wäre wohl weniger zuversichtlich gewesen, selbst etwas beitragen zu können. Tatsächlich scheinen Rollen ein so zentrales Konzept für die Modellierung zu sein, daß sich der Vergleich mit dem If-Statement von Programmiersprachen anbietet [Fowler 1997b]. Wenigstens habe ich seither mit kaum jemand über Rollen gesprochen, der nicht selbst schon einmal zumindest darüber nachgedacht hätte. Entsprechend groß (oder klein, wenn man annimmt, daß Rollen wirklich als so selbstverständlich angesehen werden, daß das Schreiben darüber manchem gar nicht erst notwendig erscheint) ist die Anzahl der Veröffentlichungen dazu.

Allerdings, und das liegt in der Natur wissenschaftlicher Veröffentlichungen, habe ich kaum zwei gleiche Arbeiten zum Rollenbegriff gefunden; ja nicht einmal ein einheitlicher Ansatz scheint sich bis heute etabliert zu haben. Gleichwohl lassen sich verschiedene Linien ausmachen, die in der Literatur vorherrschen. Die Sichtweise, die ich von Anfang an favorisiert habe, gehört nicht dazu, und die wenigen Ausführungen, die in dieselbe Richtung zielen, bleiben schemenhaft. Insbesondere führt keine einzige mir bekannte Arbeit alle Facetten des so vielseitigen Konzepts in einer Definition so zusammen, wie ich es auf den folgenden Seiten tun werde.

Man ist nie allein, und so möchte ich an dieser Stelle den Menschen Dank sagen, die mich in meiner Arbeit unterstützt, mich auf den richtigen Weg gebracht und mir Bestätigung ausgesprochen haben. Das sind (in chronologischer Reihenfolge): meine ehemaligen Kollegen der Software-Engineering-Abteilung beim ehemaligen ALCATEL AUSTRIA ELIN RESEARCH CENTER in Wien, insbesondere Gerhard Gabriel und Rudolf Lewandowski, die mit mir endlose Diskussionen zum Thema Objektorientierung durchgestanden haben, Jürgen Holzhausen und Robert Pfeifer von der Bausparkasse Mainz, die mir Gelegenheit zum Einblick in die Tiefen eines großen Modellierungsprojekts gegeben haben, Professor Dietrich Peter Pretschner, der nie müde geworden ist, auf die Bedeutung der Modellierung für die Medizin hinzuweisen, Professor Wolfgang Nejdl, der mir die Möglichkeit gegeben hat, diese Arbeit durchzuführen, Franziska Pfeffer, die mir dabei den Rücken freigehalten und mich an alles erinnert hat, Christian Kruggel, der als einziger die unfertigen Manuskripte Korrektur gelesen hat, und Michael Mrva, der mir mit seinen Hinweisen die Augen für manches wichtige Verständnisproblem geöffnet hat. Eine ganz besondere Würdigung verdient aber die Universitätsbibliothek/Technische Informationsbibliothek Hannover, ohne die diese Arbeit unmöglich gewesen wäre, die in ihrer Vollständigkeit einen unschätzbaren Wert darstellt und deren Mitarbeiter immer schnell und zuverlässig gearbeitet haben und dabei auch noch stets freundlich und hilfsbereit waren.

Hannover, im Frühjahr 2000

F. S.

INHALTSVERZEICHNIS

Kapitel 1

Einleitung

Modellierung ist ein allgemeines Problem, das Menschen seit langem beschäftigt. Ziel der Modellierung ist es, ein Repräsentation der Realität mit einem der Aufgabenstellung angemessenen Grad der Genauigkeit zu erstellen. Eine solche Repräsentation, das Modell, kann im wesentlichen zwei Zwecken dienen:

1. einen bestehenden Sachverhalt wiederzugeben, um ihn zu kommunizieren und der Untersuchung mit abstrakten Mitteln zugänglich zu machen, und
2. einem Plan oder Entwurf eine Form zu geben, die als Vorlage für die Umsetzung in die Realität taugt.

Ein Modellierungsprojekt kann beide Zwecke zugleich verfolgen – der Entwurf einer integrierten Softwarelösung beispielsweise beinhaltet, zuerst das Problem, dann die Lösung als Modell zu formulieren und schließlich, das Modell in die (Software-)Realität umzusetzen.

Das verbreitetste Mittel der Modellierung ist die natürliche Sprache. Menschen drücken ihre Beobachtungen in Sprache aus, und sie teilen die Pläne, die sie entwerfen, anderen in ihrer Sprache mit. Doch obwohl (oder gerade weil) natürliche Sprache ungemein ausdrucksstark ist, bereitet sie Probleme; die zahlreichen Paradoxa, die in ihr ausdrückbar sind, mögen als Hinweis darauf gesehen werden, daß sie nicht zur strengen Analyse und Beweisführung taugt. Andere, formale Sprachen müssen da Abhilfe schaffen.

Die Menschheit hat im Laufe der Zeit verschiedene formale Sprachen hervorgebracht. Die älteste ist wohl die Mathematik, direkt gefolgt von der Logik, mit deren formalem Ableger sie heute unauflösbar verbunden ist. Die Linguistik hat eine allgemeine Theorie formaler Sprachen entwickelt, die den natürlichen recht ähnlich und deren prominenteste Vertreter die heutigen Programmiersprachen sind. Auch die Modellierung wartet mit eigenen Sprachen auf, die jedoch aufgrund der enormen Komplexität der Modellierungsprobleme zum Teil nur so beschaffen sind, daß sie nicht unbedingt als formal bezeichnet werden können.

In der Theorie der Sprachen, formaler wie natürlicher, taucht ein Begriff immer wieder auf: der der Rolle. Rollen komplettieren die für die Sprachtheorie so wichtigen Konzepte Prädikat (als Träger der Aussage eines Satzes) und Objekt (als Ergänzung des Prädikats) um die Beschreibung der Funktion, die das Objekt in der Aussage ausfüllt. Rollen sollten damit, genau wie Prädikate und Objekte, fundamentaler Bestandteil jeder Sprachtheorie sein. Doch während die Formalisierung von Prädikaten und Objekten heute eine Selbstverständlichkeit ist, tut man sich mit der Einordnung des Rollenbegriffs in formale (Modellierungs-)Sprachen vergleichsweise schwer.

1.1 Zielsetzung

Ziel dieser Arbeit ist es, einen allgemeinen, formalen Rollenbegriff vorzulegen, der in den verschiedenen Anwendungsgebieten der Modellierung durchgängig bis hin zur Implementierung angewendet werden kann. Dabei soll insbesondere das Verhältnis von Rollen zu Typen geklärt und mit den Mehrdeutigkeiten, die sich daraus ergeben, aufgeräumt werden. Idealerweise besetzt die Definition eines solchen Rollenbegriffs eine inhaltliche Leerstelle, die zuvor behelfsweise von anderen Modellierungskonzepten ausgefüllt wurde.

Die Definition eines Rollenbegriffs, eines formalen zumal, kann nicht isoliert erfolgen, sondern muß in den Rahmen einer Modellierungssprache eingebettet werden. Diese muß natürlich die für eine formale Definition notwendigen Voraussetzungen einer präzisen Syntax und Semantik mitbringen. Zugleich sollte sie aber keine Festlegungen treffen, die für die Einführung des Rollenbegriffs ohne Belang sind, um der geforderten Allgemeinheit nicht zu widersprechen. Auch ist es sinnvoll, daß sich die Modellierungssprache nicht mehr als notwendig von den in der Praxis üblichen unterscheidet, um einer direkten Umsetzung der Ergebnisse nicht im Wege zu stehen. Die Definition einer solchen Sprache ist ein zweites, untergeordnetes Ziel dieser Arbeit.

Ausdrücklich kein Ziel dieser Arbeit ist es hingegen, für die Einführung eines Rollenbegriffs in die Modellierung zu werben – viele andere haben das bereits ausführlich getan. Vielmehr sollen die verschiedenen Möglichkeiten der Darstellung von Rollen in der Modellierung untersucht und eine möglichst universelle Rollendefinition gefunden werden, die auch als Grundlage einer Standardisierung dienen kann. Ziel dieser Arbeit ist es also insbesondere nicht, irgendeine weitere mehr oder weniger sinnvolle Formalisierung von Rollen vorzulegen, sondern eine, die auf einer möglichst breiten Basis steht. Damit soll einer bis heute fehlenden einheitlichen und allgemein akzeptierten Fassung des für die Modellierung so wichtigen Begriffs der Weg bereitet werden.

1.2 Hintergrund

Menschen bedienen sich vor allem ihrer Sprache, um Modelle als Repräsentanten der wirklichen oder einer gedachten Welt zu schaffen. Die natürlichen Sprachen sind jedoch kaum dazu geeignet, Modelle mit für formale Zwecke ausreichender Qualität, besonders hinsichtlich Präzision und Kompaktheit, zu formulieren. An ihrer Stelle empfehlen sich formale Sprachen. Die Mathematik kann als eine solche Sprache angesehen werden, doch da viele Modellierungsaufgaben es mit diskreten und ungeordneten Grundmengen, Zuständen und Zustandsübergängen zu tun haben, ist sie nur begrenzt einsetzbar.

Andere klassische Disziplinen, die sich mit der Formalisierung von Sprachen im Kontext der Modellierung befassen, sind die Sprachphilosophie (inkl. Linguistik) und die Ontologie. Die Sprachphilosophie beschäftigt sich u. a. mit der Darstellungsfunktion von Sprache und berührt die formale Modellierung überall dort, wo Zeichen und den daraus gebildeten Ausdrücken eine Bedeutung zugeordnet werden soll. Die Ontologie befaßt sich u. a. (als Metaontologie) mit den Grundbausteinen der Modellierung, also mit der Adäquatheit von Modellelementen wie Klassen, Individuen, Eigenschaften und Beziehungen. Beide Disziplinen haben einen eigenen Rollenbegriff, und dieser sollte bei der Definition eines allgemeinen Rollenbegriffs für die Modellierung nicht außer acht gelassen werden.

1.2.1 Formale Modellierung

Es war wohl Aristoteles, der mit seiner auf Genus et differentiae basierenden Kategorienlehre den Grundstock auch für die Modellierung, wie wir sie heute sehen, schuf. Doch war sein Ausdrucksmittel trotz der Verwendung einer „perfekten Form“ [Scholz 1959] die natürliche Sprache, und die Anwendung seiner Lehrsätze auf größere Aufgabenstellungen blieb daher inhärent schwierig.

Leibniz (1646–1716) war sicher nicht der erste, dem die schlechte Eignung der natürlichen Sprache für den philosophischen Diskurs aufgefallen war [Salmon

1972; Eco 1994], aber auf ihn gehen wohl die ersten brauchbaren Ansätze zurück, für angemessene Abhilfe zu sorgen. Mit seiner Instrumentierung einer Lingua characteristica universalis, die auf der Primteilerschaft und damit auf Multiplikation und Division basierte, versuchte er gegen Ende des 17. Jahrhunderts, die Grundlage für einen Calculus ratiocinator, ein Kalkül des logischen Schließens, zu schaffen, bei Vollendung dessen zwei Philosophen, wenn sie sich über eine Sache uneins wären, sich an einen Tisch setzen und sagen würden: „Calculemus – laßt uns rechnen !“ Auch wenn Leibniz' Characteristica als der erste Entwurf einer Begriffs- oder Klassenalgebra angesehen werden kann [von Kutschera 1989] und im Prinzip bereits die Theorie formaler Sprachen sowie den Aufbau eines vollständigen und korrekten Logikkalküls vorwegnimmt [Scholz 1942], waren, wie sich herausstellte, seine praktischen Beiträge zum Rechnen ungleich viel größer als die zu formalen Sprachen und Modellierung – angesichts seiner Zielsetzung und Genialität ein Hinweis darauf, wie schwer es uns Menschen fällt, uns von unserer Sprache zu lösen.

Ungefähr 200 Jahre später griff Frege (1848–1925), als er sich mit der formalen Grundlegung der Mathematik beschäftigte, das Problem wieder auf. Dabei versuchte er zunächst, zwischen einer Sprache und dem auf ihren Aussagen angewendeten Schlußfolgerungsbegriff zu trennen:

> *„Ich wollte* [...] *einen Inhalt durch geschriebene Zeichen in genauerer und übersichtlichere Weise zum Ausdruck bringen, als es durch Worte möglich ist. Ich wollte in der Tat nicht einen bloßen calcuclus ratiocinator, sondern eine lingua characterica im leibnizischen Sinne schaffen, wobei ich jene schlußfolgernde Rechnung immerhin als einen notwendigen Bestandteil einer Begriffsschrift anerkenne.“* (nach [von Kutschera 1989, S. 22] zitiert, die Auslassung ist meine.)

Die natürliche Sprache oder, wie Frege sie nannte, die Sprache des Lebens sei dafür ungeeignet, denn, so Frege, sie

> *„erweist sich als mangelhaft, wenn es darum geht, das Denken vor Fehlern zu bewahren. Sie genügt schon der ersten Anforderung nicht, die man in dieser Hinsicht an sie stellen muß, der, eindeutig zu sein.“* [von Kutschera 1989, S. 22]

und weiter:

> „[Es] *muß immer wieder betont werden, daß es der Wissenschaft erlaubt sein muß, ihren eigenen Sprachgebrauch zu haben, daß sie sich der Sprache des Lebens nicht unterwerfen kann. Eben darin sehe ich*

> *die größte Schwierigkeit der Philosophie, daß sie für ihre Arbeiten ein wenig geeignetes Werkzeug vorfindet, nämlich die Sprache des Lebens, für deren Ausbildung ganz andere Bedürfnisse mitbestimmend gewesen sind, als die der Philosophie. So ist auch die Logik genötigt, aus dem, was sie vorfindet, sich erst ein brauchbares Werkzeug zurechtzufeilen.“* [von Kutschera 1989, S. 44][1]

Dabei ging es Frege nicht unbedingt um die Modellierung der Realität, sondern um die Darstellung von Inhalten ganz allgemein. Herausgekommen bei seinen Bemühungen ist eine formale Sprache, die heute als die erste vollständige Fassung einer Prädikatenlogik erster Stufe anerkannt ist [Scholz 1941; von Kutschera 1989].

Tatsächlich hat auch eine Modellierungssprache zunächst keinen Schlußfolgerungsbegriff, kommt aber ebenfalls nicht ohne gewisse Ersetzungs- oder Einsetzungsregeln aus, nach denen sich Modelle aufbauen. Aufgabe dieser Regeln ist, zu gewährleisten, daß sich möglichst nur sinnvolle, d. h. mit einer realen Entsprechung behaftete Ausdrücke erzeugen lassen. In seinem Vorwort zu Ogdens Übersetzung von Wittgensteins Logisch-philosophischer Abhandlung, die Modellierung als einen Abbildungsprozeß der Realität erscheinen läßt, schreibt dann auch Russell:

> *„In practice, language is always more or less vague, so that what we assert is never quite precise. Thus, logic has two problems to deal with in regard to Symbolism: (1) the conditions for sense rather than nonsense in combinations of symbols; (2) the conditions for uniqueness of meaning or reference in symbols or combinations of symbols. A logically perfect language has rules of syntax which prevent nonsense, and has single symbols which always have a definite and unique meaning.“* (Russell in: [Wittgenstein 1922, S. 8])

Zu (1): Formale Sprachen werden heute im allgemeinen als Termersetzungssysteme formuliert und gehen damit auf die Arbeiten Thues und Chomskys zurück [Becker & Walter 1977]. Aus einem endlichen Alphabet und einer endlichen Anzahl von Produktionsregeln läßt sich durch deren wiederholte Anwendung

[1] Später, unter dem Eindruck der eigenen Fehler, die er angesichts der Russellschen Antinomie einräumte und die er nicht zu beheben vermochte, schrieb er sogar: „Es ist gar schwer, vielleicht unmöglich, jeden Ausdruck, den uns die Sprache darbietet, auf seine logische Unverfänglichkeit zu prüfen. So besteht denn ein großer Teil der Arbeit des Philosophen – oder sollte wenigstens bestehen – in einem Kampfe mit der Sprache.“ [von Kutschera 1989, S. 138]

eine oft unendlich große Menge von Ausdrücken oder Sätzen, die Sprache, erzeugen. Idealerweise sind die Regeln tatsächlich so gestaltet, daß sich nur sinnvolle Ausdrücke damit bilden lassen. Doch allein die Fassung des Sinns der Sätze einer natürlichen Sprache bereitet heute noch die größten Schwierigkeiten, und wie Syntaxregeln aussehen müßten, damit sich alle und nur die grammatikalisch korrekten Ausdrücke einer Sprache ergeben, ist nach wie vor unklar.

Zu (2): Den Zusammenhang von Symbolen und deren Bedeutung aufzuklären, haben viele versucht, darunter auch Aristoteles und Frege. Zu einiger Beachtung gebracht hat es das sog. semiotische Dreieck, das den Zusammenhang von Symbolen, Gedanken und den Dingen der Realität darstellen soll [Ogden & Richards 1923]. Daß dennoch die Modellierung der Realität mittels einer Sprache immer mit Problemen verbunden ist, zeigt sich in folgender entmutigender Konklusion:

> *„It must be remembered, disconcerting though the fact may be, that so far from a grammar—the structure of a symbol system—being a reflection of the structure of the world, any supposed structure of the world is more probably a reflection of the grammar used.“* [Ogden & Richards 1923, Fußnote S. 96]

1.2.2 Rollen

Der Begriff der Rolle ist so etabliert im Sprachgebrauch, daß es schwer fällt, ohne ihn auszukommen. Dennoch hat auch der Rollenbegriff einen Ursprung, und der liegt im Theater.

Der Rollenbegriff im Theater

Im Theater bezeichnet Rolle zum einen die von einem Schauspieler darzustellenden Gestalt eines (Bühnen-)Stückes und zum anderen den den einzelnen Darstellern zugeordneten Text [Meyers 1977]. Im weiteren Sinne ist dieser Text (nebst den dazugehörigen Regieanweisungen) eine Art Protokoll, das vorschreibt, wie der Schauspieler sich in der Rolle zu verhalten hat, die jedoch vom Schauspieler selbst weitgehend unabhängig ist. Wie sich zeigen wird, ist dies auch eine recht gute Umschreibung des Rollenbegriffs in der Modellierung.

Rollen wurden früher in sog. Rollenfächer eingeteilt, und zwar anhand von Kriterien wie Alter, Geschlecht oder sozialem Stand. Einteilungen nach der Art

der Rolle wie tragisch oder komisch sowie in Haupt- und Nebenrollen sind auch heute noch üblich. Dabei kann die Klassifikation durchaus hierarchisch sein: Die Rolle des gehörnten Ehemanns in einer Komödie ist beispielsweise eine tragikomische Rolle. Die Einteilung in Rollenfächer suggeriert zudem, daß Rollenspieler im Prinzip austauschbar sind, solange sie nur dasselbe Fach ausfüllen können – auch das ist, wie sich zeigen wird, ein wesentliches Merkmal des Rollenbegriffs in der Modellierung.

Der Rollenbegriff der Soziologie

Rolle ist neben sozialem Handeln einer der Grundbegriffe der Soziologie. Die Encyclopædia Britannica definiert den soziologischen Rollenbegriff wie folgt:

> *„role, in sociology, the behaviour expected of an individual who occupies a given social position or status. A role is a comprehensive pattern of behaviour that is socially recognized, providing a means of identifying and placing an individual in a society. It also serves as a strategy for coping with recurrent situations and dealing with the roles of others (*e.g., *parent-child roles). The term, borrowed from theatrical usage, emphasizes the distinction between the actor and the part. A role remains relatively stable even though different people occupy the position: any individual assigned the role of physician, like any actor in the role of Hamlet, is expected to behave in a particular way. An individual may have a unique style, but this is exhibited within the boundaries of the expected behaviour.* [...]
>
> *Role expectations include both actions and qualities: a teacher may be expected not only to deliver lectures, assign homework, and prepare examinations but also to be dedicated, concerned, honest, and responsible. Individuals usually occupy several positions, which may or may not be compatible with one another: one person may be husband, father, artist, and patient, with each role entailing certain obligations, duties, privileges, and rights vis-à-vis other persons.“* [1999]

Wie sich zeigen wird, entspricht diese soziologische Definition in geradezu frappierender Weise der des Rollenbegriffs in der Modellierung. Insbesondere legt sie nahe, daß eine Rolle eine Art Protokollspezifikation ist, die Verhalten und Eigenschaften, nicht aber den Rollenspieler selbst festlegt.

Der Rollenbegriff in der Sprachwissenschaft

Die Ursprünge des Rollenbegriffs liegen, ähnlich wie die der Modellierung, in der Sprachphilosophie. Eco nennt Lodwick (1619–1694) als den ersten Vorläufer einer lexikalischen Semantik:

> *„Lodwick hat also die originelle Idee, nicht von den Substantiven auszugehen (den Nomina oder Namen von Individuen und Gattungen, wie es in der aristotelischen Tradition noch bis zu seiner Zeit üblich war), sondern von Handlungsmustern, und er bevölkert dann diese Handlungsmuster mit Akteuren bzw. mit jenen Figuren, die wir heute Aktanten nennen würden, d.h. mit abstrakten Rollen, die sich dann mit Namen von Personen oder Dingen oder Orten verknüpfen lassen als Handelnde, Gegen-Handelnde, Objekte der Handlung und so weiter."* [Eco 1994, S. 268]

Tatsächlich stellt Lodwick in seinem Werk „A common writing" [1647] die Verben (die ja für gewöhnlich das Prädikat eines Satzes stellen und damit dessen zentrale Relation benennen) an den Anfang und leitet in seiner Entwicklung einer Universalsprache die Wörter der anderen Wortformen systematisch davon ab.

> *„Next the verbes, follow in order the nounes substantives, of which there are two sorts.*
>
> *Appellative.*
> *proper.*
>
> *Appellative I thus distinguish. To be a name by which a thing is named and distinguished, but not continually, only for the present, in relation to some action done or suffered, as for instance, Speech being of a murther committed; he that committed the same, will, from the act, be called a murtherer, and the party on whom the act is committed, the murthered, these names thus given in reference to the action done, continues no longer with the party, then thought is had of the action done, but on the contrary the specificall proper name, remaineth continually with the denominated, as the specificall name of man, beast, so also the individuall denomination of any particular man, as* Peter, Thomas, *&c.*
>
> *A proper name is that, by which any thing is constantly denominated, specifically, as* Man, dog, horse.*"* [Lodwick 1647, S. 7–8]

Lodwick nimmt damit etwas vorweg, was erst viel später und anscheinend unabhängig davon von Guarino [1992] als objektive, formale Definition des Rollenbegriffs angeführt wurde. Lodwicks Unterscheidung von „Appellative nounes“ und „proper nounes“ basiert nämlich auf zwei Eigenschaften, deren eine von Husserl (1859-1938) [1901] eingeführt und Fundierung genannt wurde und deren andere Guarino als *semantische* oder *ontologische Rigidität* bezeichnet.

Nach Guarino ist ein Konzept fundiert[2], wenn keine seiner Instanzen für sich allein existieren kann, wenn also jede Instanz, um unter das Konzept zu fallen, in Beziehung zu mindestens einer anderen stehen muß. Dabei sind Teil-Ganzes-Beziehungen ausdrücklich ausgeschlossen. So ist z. B. Student fundiert, weil ein Student, der keinem Studium nachgeht, kein Student ist. Ein Rad dagegen ist nicht fundiert, obwohl es in der Regel Teil von etwas anderem ist und selbst aus Teilen besteht.[3]

Semantische Rigidität hingegen bedingt, daß ein Konzept die Identität seiner Instanzen ausmacht, eine Instanz also nicht der Extension dieses Konzeptes beitreten und diese wieder verlassen kann, ohne daß sich das auf die Identität der Instanz auswirkt. Das Konzept *Kleinkind* beispielsweise ist nicht semantisch rigide, da ein Säugling im Alter von einem Jahr zum Kleinkind (und damit zu einer Instanz des Konzepts *Kleinkind*) wird, dies aber genauso wie die spätere Aufgabe des Konzeptes nichts zur Identität der Person beiträgt. *Person* dagegen ist semantisch rigide, denn wer einmal eine Person ist, wird immer eine Person bleiben (oder muß seine Identität aufgeben) [Guarino 1992].

Rollen sind nach Guarino nun genau die Konzepte, die fundiert, aber nicht semantisch rigide sind, und natürliche Typen sind die Konzepte, die nicht fundiert, dafür aber semantisch rigide sind. Lodwick verwendete aber gerade diese beide Eigenschaften zur Unterscheidung von „Appellative nounes“ und „proper nounes“: „murtherer“ und „murthered“ sind (als Konzepte) fundiert („these names thus given in reference to the action done“ – ungefähr „diese Namen werden in Bezug auf die Handlung vergeben“) und nicht semantisch rigide („these names [...] continues no longer with the party, then thought is had of the action done“ – ungefähr „diese Namen verbleiben nicht länger bei den Beteiligten, als man an die Handlung denkt“); sie sind Rollen. „Man“, „dog“ und

[2] Husserls ursprüngliche Bestimmung des Begriffs ist länglich; Guarino gibt dafür eine modallogische Formalisierung an.

[3] Zur Problematik dieser Definition, die von der der Teil-Ganzes-Beziehung abhängt, siehe Abschnitt 3.3.1.

„horse“ dagegen sind semantisch rigide („A proper name is that, by which any thing is constantly denominated“) und offensichtlich nicht fundiert – sie sind natürliche Typen.

Dem sprachlich interessierten Leser wird aufgefallen sein, daß „murtherer“ und „murthered“ nicht nur konzeptuell, sondern auch grammatikalisch Rollen sind: sie sind Ergänzungen des Prädikats morden („murther committed“). Und während Rollen bei den meisten formalen Sprachen einschließlich der Logik keine eigenständige Rolle spielen, kommt ihnen in der Linguistik sogar eine zentrale Funktion zu. So konstatierte der Sprachtheoretiker Bühler [1879–1963]:

> *„Es bestehen in jeder Sprache Wahlverwandtschaften; das Adverb sucht sein Verbum und ähnlich die anderen. Das läßt sich auch so ausdrücken, daß die Wörter einer bestimmten Wortklasse eine oder mehrere Lehrstellen um sich eröffnen, die durch Wörter bestimmter anderer Wortklassen ausgefüllt werden müssen.“* [Bühler 1934, S. 173]

Etwa zur gleichen Zeit, doch erst viel später veröffentlicht, entstand Tesnières Dependenzgrammatik [1965], nach der ein Wort im Satz, das Regens, andere Wörter, seine Dependentien, fordert. Die Leerstellen Bühlers sind die Rollen eines Regens, die von dessen Dependentien gefüllt werden. In Fillmores Case grammar wird dem syntaktischen Kasus, der Kasusform, die sog. Kasusrolle oder auch semantische Rolle gegenübergestellt, die auf den Inhalt der logischen Ergänzung eines Prädikats abzielt. Anstatt es aber bei der Feststellung zu belassen, daß jedes Verb eine (seine eigene) Menge von semantischen Rollen hat, die durch andere Satzglieder (explizite oder implizite) zu füllen sind, besteht er darauf, daß eine grammatische Theorie ein Inventar von abstrakten Rollen bereitstellen muß, auf die sich alle konkreten reduzieren lassen: eben Handelnder, Gegen-Handelnder etc. [Fillmore 1971]. Aktuelle Bestrebungen, die Grammatik ganz in das Lexikon zu verlagern, gehen wieder davon ab; und auch für die Modellierung haben diese abstrakten Rollen wenig Wert.

1.3 Aufbau der Arbeit

In Kapitel 2 werden zunächst einige Grundbegriffe der Modellierung aus heutiger Sicht vorgestellt und dann eine einfache Modellierungssprache definiert. Gemäß der Zielsetzung dieser Arbeit ist die Sprache formal, an die Grundlagen gängiger Modellierungsformalismen angelehnt und frei von Festlegungen, die nicht unmittelbar der Einführung eines Rollenbegriffs dienen. Es soll also nicht verwundern, daß die Modellierungssprache im Vergleich zu anderen eher rudimentär ist.

Kapitel 3 ist ganz der Darstellung von Rollen gewidmet. Dazu werden zunächst die wichtigsten Anforderungen an den Rollenbegriff kurz motiviert und dann eine Rollendefinition präsentiert, die diese Anforderungen erfüllt. Anders als in [Steimann 1999b] erfolgt die Darstellung und Diskussion der aus der Literatur bekannten anderen Ansätze erst im Anschluß an die Präsentation der eigenen Arbeit, da die Ausführlichkeit der Literaturbesprechung den Zusammenhang von rollenloser und der Modellierung mit Rollen verlorengehen lassen würde.

In Kapitel 4 wird der in Kapitel 3 bestimmte Rollenbegriff auf die objektorientierte Softwareentwicklung, insbesondere auf die UNIFIED MODELING LANGUAGE (UML) und die objektorientierte Programmierung, übertragen. Es wird sich zeigen, daß dies im Fall von UML zu einer erheblichen Vereinfachung der Sprachdefinition führt und dabei nicht unbedingt die Syntax der Sprache, die geläufigen Notationen und Diagrammtypen, betreffen muß. Für die objektorientierte Programmierung ergibt sich eine direkte Umsetzung des Rollenkonzeptes und somit die in der Zielsetzung geforderte durchgängige Anwendbarkeit der Definition.

Kapitel 5 schließlich schließt die Arbeit mit einer Zusammenfassung, einer Bewertung und mit einem die offen gebliebenen Punkte nennenden Ausblick ab.

Wie jede Arbeit, so hätte auch diese ganz anders aufgebaut werden können. Ein alternativer Zugang ist aus Tabelle 1.1 ersichtlich. Zudem sind Teile der Arbeit an anderer Stelle veröffentlicht und werden deswegen hier nicht alle im vollen Umfang wiedergegeben. Dazu gehören

Tabelle 1.1: Alternative Struktur der Arbeit

THEMA	KAPITEL / ABSCHNITT
Motivation der Modellierungssprache und des darin enthaltenen Rollenbegriffs	Abschnitte 1.2.2, 2.2 und 3.1
formale Definitionen	Abschnitte 2.1, 2.3 und 3.2
bibliographische Diskussion	Rest des Kapitels 3
praktische Verwendung	Abschnitt 2.4 und das Kapitel 4
Zusammenfassung und Bewertung	Kapitel 5

- der Sinn und Zweck von Abstraktionshierarchien in [Steimann 1999a; 2000a],
- die Verwendung der rollenlosen Modellierungssprache zur Verhaltensbeschreibung dynamischer Systeme in [Steimann et al. 1999],
- die Grundlegung des Rollenbegriffs in [Steimann 2000b],
- der systematische Vergleich mit den Rollenbegriffen aus der Literatur in [Steimann 1999b],
- die Formalisierung des Rollenbegriffs in [Steimann 1999b; 2000b]
- die Bedeutung für die objektorientierte Modellierung in [Steimann 2000b; 2000d; 2001], und
- die Bedeutung für die objektorientierte Programmierung in [Steimann 2001].

Den besten Überblick über die Arbeit als ganzes bietet dabei [Steimann 1999b; 2000b], über ihre Einordnung [Steimann 1999b] und über ihre praktische Bedeutung [Steimann 2001].

Kapitel 2

Modellierung ohne Rollen

Nach einer einführenden Betrachtung des prinzipiellen Aufbaus und der Funktionsweise von Modellierungssprachen werden die gebräuchlichsten Modellierungskonzepte kurz vorgestellt. Es sind dies im wesentlichen die primitiven Sprachelemente Objekt, Typ, Relation, Nachricht und Subsumtionshierarchie. Auf dieser Grundlage wird eine einfache, aber in sich abgeschlossene und für viele Modellierungszwecke bereits ausreichende formale Modellierungssprache definiert. Sie ist vor allem als eine Vorbereitung auf das nächste Kapitel zu verstehen, in dem ich mit der Einführung von Rollen zum Kern meiner Arbeit komme. Insofern sei gerechtfertigt, daß hier manche Dinge unvollständig, andere nur knapp oder gar nicht angesprochen werden und daß insbesondere eine Berücksichtigung der umfangreichen Literatur zu dem Thema weitgehend fehlt; der Definition und Diskussion des Rollenbegriffs in Kapitel 3 wird dies keinen Abbruch tun.

2.1 Aufbau und Funktion einer Modellierungssprache

Eine Modellierungssprache ist eine formale Sprache. Als solche hat sie eine Syntax und eine Semantik.[4] Die Syntax gibt den Ausdrücken der Sprache ihre Struktur, die Semantik ihre Bedeutung.

Es gibt viele verschiedene formale Sprachen, die sich schon allein aufgrund ihres (historischen) Ursprungs unterscheiden. Zu den bekannteren zählen die Prädikatenlogik erster Stufe, Programmiersprachen wie PASCAL oder JAVA sowie die Modellierungssprache UML. Man kann die Sprachen anhand ihres Verwendungszwecks in verschiedene Typen einteilen und ihnen jeweils kollektiv bestimmte Eigenschaften beimessen. So sind Programmiersprachen eindeutig und haben eine operationale Semantik, Logiken beinhalten einen Schlußfolgerungsbegriff und dienen i. a. der Beweisführung etc. Manche Sprachen passen in mehrere Kategorien: Prädikatenlogik beispielsweise eignet sich auch als Programmiersprache und als Modellierungssprache.

Formale Sprachen sind in der theoretischen Informatik als Sprachen definiert, deren Sätze sich von einer (Chomsky-)Grammatik erzeugen lassen [Becker & Walter 1977]. Diese Definition trifft auch auf die meisten der hier angesprochen Sprachen zu[5]: Die Sprachen haben also eine Grammatik, aus der sich die Ausdrücke (oder Sätze) der Sprache ableiten lassen und die auch als Syntax bezeichnet wird (Tabelle 2.1). Neben dem (metasprachlichen; s. u.) Ableitungsbe-

Tabelle 2.1: Gegenüberstellung verschiedener Sprachtypen.

SPRACHTYP	GRAMMATIK	SPRACHE
Logik	Syntax	Menge aller Formeln
Programmiersprache	Syntax	Menge aller Programme
Modellierungssprache	Syntax (grafische Notation)	Menge aller Modelle

[4] Bei einer Modellierungssprache ohne formale Semantik spricht man besser von einer Modellierung*snotation*.

[5] Jedoch sind mit Sprachtypen nicht die Typen der Chomsky-Hierarchie gemeint, sondern die verschiedenen Arten, die sich im Laufe der Zeit herausgebildet haben.

griff der Theorie der formalen Sprachen haben manche Sprachen selbst wieder einen (objektsprachlichen) Ableitungsbegriff. Dieser Ableitungsbegriff dient i. d. R. dazu, mit einer vergleichsweise kompakten Spezifikation, die ein Ausdruck der Sprache ist, eine viel größere Menge von (anderen) Ausdrücken zu induzieren, die in ihrer Summe das Spezifizierte darstellen (Tabelle 2.2). Beispielsweise enthält die Prädikatenlogik erster Stufe einen Schlußfolgerungsbegriff, der aus einer Menge von Axiomen alle wahren Sätze abzuleiten erlaubt.

Entsprechend macht man es in anderen Sprachtypen auch. Beispielsweise ist es Ziel der formalen Linguistik, einen Grammatiktyp zu finden, der natürliche Sprachen so zu spezifizieren erlaubt, daß sie alle und nur die „richtigen“ (im Sinne des richtigen Wortgebrauchs, nicht des richtigen Inhalts) Sätze enthalten, so daß die Grammatiken also in gewisser Weise vollständig und korrekt sind. Ähnlich ist es bei den Modellierungssprachen: Da die zu modellierende Realität i. d. R. sehr komplex ist, wird eine Modellspezifikation sie i. a. nicht direkt eins zu eins wiederzugeben versuchen, sondern auf eine kompaktere Spezifikationsform zurückgreifen wollen, aus der sich dann mittels eines geeigneten Ableitungsbegriffs das gesamte spezifizierte Modell erzeugen läßt.

2.1.1 Objekt- und Metasprache

Über Sprache zu sprechen ist inhärent schwierig, und zwar insbesondere dann, wenn man dazu dieselbe Sprache bemüht. Man stelle sich nur vor, ein Wörterbuch schreiben zu müssen, dafür aber keine Wörter als bekannt voraussetzen zu können. Wenn Definiens und Definiendum zusammenfallen, beißt sich die Katze in den Schwanz.

Um das Sprechen über Sprache zu ordnen, unterscheidet man zwischen *Ob-*

Tabelle 2.2: Sprachen und ihr Ableitungsbegriff zur kompakten Spezifikation.

FORMALISMUS	SPEZIFIKATION	METHODE	SPEZIFIZIERTES PRODUKT
Logik	Mengen von Formeln (Axiome)	Deduktion	Menge der ableitbaren Formeln (Theorie)
Programmiersprache	Programm	Ausführung	Menge von Zuständen und Zustandsübergängen[6]
Modellierungssprache	Modell	Instanziierung	Menge von Szenarien und Verläufen

[6] Ein Programm ist ein endlicher Automat und damit streng genommen selbst wieder eine Sprache; entsprechendes gilt für ein Modell.

*jektsprache*als der Sprache, über die man spricht, und *Metasprache* als der Sprache, in der man das tut. Die Metasprache muß als bekannt vorausgesetzt werden und kann daher, wenn die Objektsprache das nicht ist, nicht mit dieser zusammenfallen.[7] Dazu ein Beispiel.

Die Definition einer Modellierungssprache kann selbst als ein Modellierungsprozeß aufgefaßt werden, was dann verlangt, daß die Metasprache wieder eine Modellierungssprache sein muß.[8] Die Modellierung der Modellierungssprache läßt sich wiederum modellieren, usw. Das grundlegende Problem der Modellierung, was denn deren primitive Sprachelemente sind, wird dadurch jedoch nie gelöst – dazu bedarf es immer einer ontologisch begründeten Festlegung [Steimann & Nejdl 1999].[9]

In dieser Arbeit werden als Metasprache der Modellierung keine spezielle Modellierungssprache, sondern die üblichen Schreibweisen der Mathematik und der formalen Logik verwendet. Gleichwohl teilen sich die Objektsprache und die Metasprache gewisse Konzepte: Typen entsprechen in etwa einfachen Mengen, Relationen Relationen und das Instanzsein dem Elementsein. Dennoch unterscheiden sich die beiden so, daß sich zum einen die Einführung einer neuen Sprache überhaupt lohnt (immerhin bringt sie einen Rollenbegriff, den es in der Mathematik und Logik nicht gibt) und zum anderen hinreichend klar sein sollte, welche Ausdrücke objektsprachlich und welche metasprachlich sind, ohne daß dies jedesmal dazu gesagt werden muß.

2.1.2 Syntax und Semantik

Ein *Modell* ist ein Ausdruck einer Modellierungssprache, der einen Ausschnitt der Realität mit einem den Möglichkeiten der Modellierungssprache entsprechenden und dem jeweiligen Modellierungsproblem angemessenen Grad der Genauigkeit wiedergibt. Ein Modell ist ein syntaktisches Konstrukt, das gemäß den Syntaxregeln der Modellierungssprache aus primitiven Modellierungselementen zusammengesetzt und induktiv über den Aufbau mit einer Semantik

[7] Es ist allerdings eine beliebte Übung, genau dies zu tun, z. B. die Darstellung der Backus-Naur-Form in sich selbst.

[8] Die abstrakte Syntax von UML beispielsweise ist in UML spezifiziert [OMG 1999].

[9] Da sich diese nicht erst für die x-te Metaebene der Modellierung aufdrängt, wird man sich bei der Metamodellierung im allgemeinen auf eine eher kleine Zahl von Metaebenen festlegen. Das Information Resource Dictionary System (IRDS) Framework der ISO z. B. schlägt dafür die Zahl drei vor [1990b].

versehen wird. Wie diese Semantik aussieht, wird mit der Modellierungssprache festgelegt. UML z. B. hat, insofern sich aus den Modellen Programmgerüste erzeugen lassen, eine operationale Semantik; die Modellierungssprachen, die im folgenden definiert werden, haben eine denotationale, die auf der der ordnungssortierten Prädikatenlogik [Oberschelp 1962; Bläsius et al. 1989] basiert.

Nicht unbedingt Ziel der Modellierung, doch i. a. als günstig erweist sich, wenn Modell und Realität einander isomorph entsprechen, d. h., wenn es zwischen den Objekten des Modells und denen der Realität eine Eins-zu-eins-Beziehung gibt und wenn Objekte, die im Modell miteinander in Beziehung stehen, auch in der Realität miteinander zu tun haben. Dies erleichtert die Modellierung als intellektuellen Prozeß, da sich der Modellierer bei seinen Entwurfsentscheidungen an der Realität orientieren kann und dem Betrachter eines Modells bei dessen Bemühungen, es zu verstehen, seine Kenntnis der Realität zur Verfügung steht. Die Isomorphie trägt jedoch nur zum intellektuellen Verständnis, nicht zur formalen Interpretation des Modells bei, denn die ist über die Semantik eindeutig festgelegt.

2.1.3 Intension und Extension

In vielen Fällen reicht die Unterscheidung von Syntax und Semantik aus, um zu einer sinnvollen Sprachdefinition zu kommen. In der Sprachphilosophie tauchen aber noch zwei weitere Begriffe auf, die auch und gerade in der Informatik eine Bedeutung erlangt haben: *Intension* und *Extension*.[10]

Carnap hat, ausgehend von Freges berühmt gewordenen Beispiel von Abendstern und Morgenstern (zwei Intensionen, die beide dieselbe Extension, nämlich den Planeten Venus, haben[11]), davon Abstand genommen, die Ausdrücke einer Sprache als Namen einer konkreten oder abstrakten Entität anzusehen, und hat statt dessen jeden Sprachausdruck mit einer Intension und einer Extension verbunden [1947]. Seine Zuordnung von Ausdrücken einer an die Prädikatenlogik angelehnten Sprache zu deren Intensionen und Extensionen ist in Tabelle 2.3 wiedergegeben. Wesentlich daran (und für die Formulierung von Modellie-

[10] zur terminologischen Verwendung von Intension und Extension s. [ISO 1990a]

[11] Man überzeuge sich, daß Abendstern und Morgenstern gemessen an Guarinos Definition aus Abschnitt 1.2.2 eigentlich Rollen, und zwar zwei Rollen eines Planeten sind.

Tabelle 2.3: Carnaps Zuordnung von Intension und Extension zu den wichtigsten Ausdruckstypen [Carnap 1954; S. 42] [12]

AUSDRUCK	INTENSION	EXTENSION
Satz	Proposition	Wahrheitswert
Individuenkonstante	Individuenbegriff	Individuum
einstelliges Prädikat	Eigenschaft	Menge [von Individuen]
n-stelliges Prädikat ($n > 1$)	*n*-stellige Relation	Klasse geordneter *n*-Tupel von Individuen
Funktor	Funktion	Werteverlauf

rungssprachen, wie ich sie vorhabe) ist, daß Intension und Extension sich weder gegenseitig bedingen noch Alternativen sind.

Die meisten formalen Sprachen sind entweder intensional oder extensional. Die Mengentheorie beispielsweise ist extensional (zwei Mengen sind gleich genau dann, wenn ihre Elemente gleich sind), während Modellierungssprachen typischerweise intensional sind: So sind *sprechendes Tier* und *federloser Zweibeiner* verschiedene Begriffe, auch wenn die Menge der Individuen, die darunter fallen (die Menge der Menschen), gleich sind. Anstatt, wie in der Praxis üblich, Intensionen mit Typen gleichzusetzen und ihnen direkt Extensionen zuzuordnen, werde ich den Symbolen der Modellierungssprache eine semantische Interpretation, eine Intension und eine Extension geben. Dabei gehören Intension und Extension mit zu dem, was ein Modellierer (direkt oder indirekt) festzulegen hat, wenn er ein Modell spezifiziert.

Wenn man davon ausgeht, daß sowohl die Interpretation eines Modells als auch die modellierte Realität extensional sind, dann sind Intensionen für die Modellierung vor allem deswegen interessant, weil sie eine kompakte Spezifikation der Extensionen erlauben. Anstatt, um beispielsweise auszudrücken, daß Männer und Frauen verheiratet sein können, alle möglichen Paare zwischen diesen als Elemente einer entsprechenden Relationen aufzulisten, ist es sinnvoll, anzugeben, daß Verheiratetsein unter anderem erfordert, daß von den Verheirateten einer ein Mann ist und die andere eine Frau. Wenn man aber weiß, wer Mann ist und wer Frau, lassen sich daraus alle möglichen Paare von Verheirateten erzeugen. Dies entspricht gerade dem eingangs angedeuteten Ableitungsbegriff der Modellierungssprachen.

[12] Die scheinbare Unstetigkeit, die sich aus der Unterscheidung von ein- und mehrstelligen Prädikaten ergibt, läßt sich so auflösen, daß eine Eigenschaft Attribut eines Objekts ist, während eine Beziehung Attribut von zwei oder mehr Objekten ist.

Allerdings gehört die geeignete Formalisierung der Intensionen wohl zu den hartnäckigsten Problemen, die die Modellierung zu bieten hat. Insbesondere die Abwägung zwischen Abstraktion und Genauigkeit bei der Fassung des dynamischen Modells mit seinen unübersehbar vielen Kombinationsmöglichkeiten ist in aller Regel nur schwer zu finden. Ich muß mich hier diesem Problem aber nur insoweit stellen, als es die Einführung eines Rollenbegriffs berührt, und dazu reicht es, zu wissen, daß es Intensionen gibt.

2.1.4 Statik und Dynamik

Statisch betrachtet läßt sich die Realität als aus Objekten und Beziehungen zwischen diesen bestehend auffassen. Dynamisch ändert sich daran nur, daß Objekte entstehen und wieder vergehen und daß sich die Beziehungen zwischen Objekten mit der Zeit ändern. Ein Modell ist eine Repräsentation dieser Realität, in der sich die Objekte und Beziehungen wiederfinden.

In der Modellierung wird üblicherweise zwischen statischen und dynamischen Aspekten unterschieden. Das liegt unter anderem daran, daß die zum Einsatz kommenden Formalismen jeweils für den einen oder den anderen Aspekt konzipiert sind: Ein Entity-Relationship-Diagramm beispielsweise drückt ein (statisches) Strukturschema aus, endliche Automaten und Petri-Netze dagegen (zeitliche) Abfolgen. Durch die strikte Unterscheidung wird die Modellierung nicht immer erleichtert; insbesondere die Konsistenthaltung der verschiedenen Teilspezifikationen eines Systems wird so zu einem ganz eigenen Problem.

In dieser Arbeit geht es um die Modellierung mit Rollen. Da Rollen, wie im nächsten Kapitel dargelegt werden wird, einen dynamischen Charakter haben, ist die Betrachtung der Dynamik eines Modells durchaus relevant. Allerdings ist sie das nur bis zu einem gewissen Grad; insbesondere ist zwar die Existenz, aber nicht die Form der Spezifikation von (Zeit-)Verläufen von Bedeutung. Ich werde mir daher ein paar grundlegende Vereinfachungen erlauben.

Ein *dynamisches Modell* sei im folgenden die (irgendwie geartete) Spezifikation einer Menge von möglichen Verläufen, die das Modell (als Repräsentant der Realität) nehmen kann. Ein *Verlauf* ist eine (zeitdiskrete) Folge von Szenarien. Ein *Szenario* ist die Menge der die zu einem gegebenen Zeitpunkt existierenden

Objekte der Realität repräsentierenden Symbole zusammen mit den Beziehungen, die zwischen diesen bestehen.[13]

Das *statische Modell* wird in dieser Arbeit als die Projektion des dynamischen Modells entlang der Zeitachse betrachtet. Seine Aufgabe ist es, unmögliche (weil in der Realität nicht und damit auch in keinem möglichen Verlauf vorkommende) Szenarien auszuschließen und nur die sinnvollen zuzulassen. Das statische Modell gibt also die Menge aller zu beliebigen Zeitpunkten existierenden Objekte und die Menge aller möglichen Beziehungen zwischen diesen vor. Das statische Modell hat in der Regel die Form einer (statischen) Strukturbeschreibung oder eines Schemas; man spricht daher alternativ auch von *Struktur* statt von Statik (und von *Verhalten* statt von Dynamik).

Anmerkung: Die Vorstellung von einem dynamischen Modell als einer Menge von möglichen Verläufen erinnert an die Menge von möglichen Welten der Interpretation einer modalen Logik. Das statische Modell wirft dann alle möglichen Welten zusammen. Es enthält aber nicht, wie man vielleicht erwarten würde, nur die Beziehungen, die in allen möglichen Welten zutreffen, sondern all die, die in irgendeiner möglichen Welt vorkommen. Insbesondere haben Mengenbeschränkungen, die sog. Kardinalitäten, im statischen Modell nicht die gewohnte Bedeutung; s. Abschnitt 2.4.1.

[13] Schnappschuß ist ein anderes Wort für Szenario. Nicht gemeint ist jedoch der Szenarienbegriff der objektorientierten Modellierung, wenn er Sequenzen oder Abläufe bezeichnet.

2.2 Allgemeine Modellierungskonzepte

Die hier getroffene Auswahl an Modellierungskonzepten stellt nur einen kleinen Teil der in der umfangreichen Literatur zum Thema Modellierung anzutreffenden dar. Sie orientiert sich zum einen an den Definitionen einer ordnugsortierten Logik und zum anderen daran, was für die Einordnung des Rollenkonzepts im nächsten Kapitel gebraucht wird. Man wird jedoch feststellen, daß sie dabei einen erstaunlich großen Teil dessen abdeckt, was man bei der Umsetzung von Modellen in Software wiederfindet.

2.2.1 Objekte und Typen

Obwohl es Modellierungsansätze gibt, die nicht prinzipiell zwischen Objekten und Typen unterscheiden (z. B. der Prototypenansatz [Lieberman 1986; Lakoff 1987], vgl. aber auch die Ansätze ohne strikte Ebenentrennung wie z. B. [Mylopoulos et al. 1990]), ist eine solche Betrachtungsweise eher dazu geeignet, Strukturen aufzulösen als zu schaffen. Nicht zuletzt entspricht die Unterscheidung von Objekt und Typ der von Element und Menge, die ganz wesentlich für die Neuordnung von Mathematik und Logik war [Gardies 1991]. Ich werde also in dieser Arbeit daran festhalten.

Objekte und Entitäten

Objekte werden durch ihre Eigenschaften (zu denen je nach Betrachtungsweise auch die Beziehungen zählen, in denen sie zu anderen Objekten stehen) charakterisiert. Ein Objekt bar seiner Eigenschaften ist eine *Entität*, eine Bezeichnung, die aus der Philosophie stammt und die für das Dasein von etwas im Gegensatz zu seinem Wesen steht. In der Tat haben Entitäten zunächst nichts als ihre Existenz – insbesondere haben Sie keinen inhärenten Zustand, denn dieser ist ausschließlich durch die (augenblicklichen oder zu einem bestimmten anderen Zeitpunkt bestehenden) Eigenschaften gegeben.

Objekte werden in Modellen durch Symbole repräsentiert. Dabei steht ein Symbol für die Identität eines Objektes. Es gehört daher zu jedem Symbol ge-

nau ein Objekt und zu jedem Objekt genau ein Symbol; das Symbol ist eine Art Object identifier, der ein Objekt einzig (aber nicht einzigartig) macht. Bei dynamischer Betrachtungsweise hat jedes Modellobjekt außerdem einen Lebenszyklus, der im allgemeinen mit dessen Eintritt in das dynamische Modell beginnt und mit dem Verlassen desselben endet. Die Identität eines Objektes hat jedoch über dessen Lebenszyklus hinaus Bestand, ein Umstand, der sich darin ausdrückt, daß das statische Modell stets alle Objekte (also aus der Sicht des dynamischen Modells auch die, die schon aufgehört haben zu existieren und auch die, die noch gar nicht existieren) umfaßt.

Typen, Sorten und Klassen

Der Begriff des Typs findet vielfach Verwendung. In die mathematische Logik wurde er Anfang des 20. Jahrhunderts von Russell eingeführt, um den Argumentbereich von Prädikaten höherer Ordnung einzuschränken und so die nach ihm benannte Antinomie und verwandte Paradoxa zu vermeiden [Whitehead & Russell 1910]. Die symbolische Logik verallgemeinerte den Typbegriff mit der Einführung zuerst der mehrsortigen [Herbrand 1930; Schmidt 1938] und dann der ordnungssortierten Prädikatenlogik [Oberschelp 1962; 1989], indem sie eine Klassifikation der Argumente auch für Prädikate erster Ordnung zuließ.[14] In die Informatik hat der Begriff des Typs mit der Typisierung von Variablen in ALGOL 60 und der Einführung von abstrakten Datentypen Anfang der 70er Jahre [Kutzler & Lichtenberger 1983] breiten Einzug gehalten.

Typen werden in der Modellierung auch häufig Klassen genannt. Dieser Terminus sollte jedoch vermieden werden, da er zum einen mathematisch belegt ist (als Begriff für alle Gesamtheiten, also neben Mengen auch für Unmengen [Glubrecht et al. 1983]) und zum anderen in der objektorientierten Programmierung in zunehmendem Maße der Implementierung eines Typs vorbehalten wird. Ich will daher durchgängig den Begriff Typ verwenden, auch wenn im einen oder anderen Kontext Sorte oder Klasse natürlicher klänge.

Durch die Typen findet eine Klassifikation der Objekte statt. Je nachdem, ob die Menge der Objekte durch die Typen partitioniert wird oder nicht, spricht man von einer Einfach- bzw. Mehrfachklassifikation der Objekte. Echte Mehrfachklassifikationen, das sind solche, die nicht allein auf Typsubsumtion (s. Abschnitt 2.2.4) beruhen, sind derzeit eher selten anzutreffen, für die natürliche Modellspezifikation aber durchaus von Interesse.

Allgemein wird ein Typ als eine intensionale Spezifikation verstanden: Ein Typ ist eine Beschreibung, die eine (potentiell unendliche) Menge von Objekten spezifiziert. Aber nicht mit jeder Menge korrespondiert ein Typ: Die Objekte eines Typs sind auf irgendeine Weise, meistens aufgrund ihrer Natur oder Abstammung, miteinander verwandt, und die Natur dieser Verwandtschaft wird durch den Typ ausgedrückt. Liegt eine natürliche Verwandtschaft nicht vor, kann die Zusammenfassung verschiedener Objekte zu einer Menge auch durch andere Kriterien wie z. B. die Fähigkeit, eine bestimmte Rolle zu füllen, gerechtfertigt werden; dies allerdings ist erst Gegenstand des nächsten Kapitels.

Typen und Metatypen

„Everything is an object." Zusammen mit der Festlegung, daß jedes Objekt Instanz eines Typs sein muß, der Tatsache, daß in endlichen Spezifikationen keine unendlichen monotonen Ketten von Instanz-Typ-Beziehungen auftreten können (so daß Zirkel unvermeidbar sind), und der Standardinterpretation von Typen als Mengen (und Instanzen als Elementen davon) führt dieser Slogan doch unmittelbar zur russellschen Antinomie. So ist z. B. in SMALLTALK *Metaclass* Instanz von sich selbst [Goldberg & Robson 1983] und hat damit keine mengentheoretische Interpretation. Statt dessen bedient man sich besser eines stufenweisen Aufbaus, wie er beispielsweise vom *Information Resource Dictionary System* (IRDS) Framework der ISO [1990b] vorgeschlagen wird (und der im wesentlichen Russells Auflösung seiner Antinomie entspricht).

2.2.2 Beziehungen und Relationen

Die zentrale Dichotomie der Modellierung, die Aufteilung der Modellelemente in Objekte und Beziehungen zwischen diesen, ist eine klassische. Auf ihr basieren nicht zuletzt die Prädikatenlogik und alle damit verwandten Modellierungsformalismen wie das Entity-Relationship-Modell [Chen 1976], die konzeptuellen Graphen [Sowa 1984] usw.

Genauso, wie Objekte zu Typen abstrahiert werden, werden in der Modellierung Beziehungen (zwischen Objekten) zu Relationen abstrahiert. Während die Unterscheidung von Objekten und Typen aber umgangssprachlich klar ist, ist sie das für Beziehungen und Relationen leider nicht – man verwendet die bei-

[14] Die Begriffe Typ und Sorte werden in der symbolischen Logik heute weitgehend synonym verwendet [Bläsius et al. 1989, S. 3].

den Begriffe gewöhnlich synonym.[15] Mengentheoretisch ist eine *n*-stellige Relation als eine Teilmenge des kartesischen Produktes von *n* Mengen und ein Element der Relation als ein *n*-Tupel von Elementen dieser Mengen definiert. Anders als in der Mathematik sind Relationen in der Modellierung jedoch intensionale Konstrukte. Sie werden daher im englischen Sprachgebrauch zur besseren Unterscheidung auch häufig relationships[16] oder relationship types genannt. Letzteres trägt dem Umstand Rechnung, daß die Extensionen der Relationen wie die der Typen Mengen sind. Dadurch aber, daß das einstellige kartesische Produkt einer Menge die Menge selbst ist, wird bei Betrachtung von Relationen als Typen die für die Modellierung so wesentliche Unterscheidung zwischen Typ und Relation wieder verwischt (vgl. Carnaps Tabelle in Abschnitt 2.1.3). Auf der anderen Seite wird so Relationen über Relationen und den Relationshierarchien (Abschnitt 2.2.3) der Weg bereitet.

Modellierung mit Relationen

Ein Modell deklariert eine Menge von Relationen, von denen jede eine Anzahl Stellen hat, die den Typ der Objekte angeben, die durch die Relation in Beziehung gesetzt werden können. Die Stellen der Relation sind durchnumeriert und werden zunächst, da Typen mehrfach auftreten können, nur über ihre Position eindeutig referenziert.[17] Im nächsten Kapitel werden die Typen dann durch Rollen ersetzt, die in jeder Stelle eindeutig sind.

Manche Autoren schränken die Menge der Relationen auf zweistellige ein und argumentieren, höherstellige ließen sich problemlos auf zweistellige zurückführen. Dies ist aber weder besonders natürlich noch sonderlich praktisch; die Relation etwa, die durch das Prädikat *geben* bestimmt wird[18], hat in der Regel drei Stellen, nämlich einen Geber, einen Empfänger und das Gegebene.

Die Existenz einer Beziehung ist von der Existenz der beteiligten Objekte abhängig. Das umgekehrte ist jedoch nicht der Fall – vielmehr drückt der Wechsel der Beziehungen, in denen ein Objekt zu anderen steht, seine Veränderung über die Zeit aus. Außerdem sind die Elemente einer Relation, die Beziehungen,

[15] In Abschnitt 2.3.3 dieses Kapitels wird daher der Terminus Assoziation für Beziehungen zwischen Objekten eingeführt, der aber in dieser Bedeutung nicht etabliert ist.
[16] was wiederum dem deutschen Beziehung entspricht
[17] Die Rollennamen, die den Stellen einer Relation zugeordnet werden, sind i. d. R. nur Kosmetik; man kann auch die Ordinalzahlen der Positionen als Rollennamen nehmen.
[18] Zwischen Relationen und Prädikaten besteht der übliche Zusammenhang: Mit jeder Relation korrespondiert ein Prädikat, und die Tupel der Relation sind genau die, die das Prädikat erfüllen.

i. d. R. bloße Tupel ohne eigene Identität – tauscht man ein Objekt an einer Stelle aus, dann erhält man eine andere Beziehung. Auch das unterscheidet Relationen von Typen (s. o.).

Modellierung mit Funktionen: Attribute

Funktionen werden in der formalen Modellierung gern dazu verwendet, Attribute darzustellen. Dabei entspricht der Funktor dem Namen des Attributs, das Argument dem attributierten Objekt und der Funktionswert dem Attributwert. Ist ein Attribut nicht ein-, sondern mehrwertig, kann man es als mengenwertig, also als Abbildung in die Potenzmenge eines Grundbereichs deklarieren. Dieser Ansatz steht in direkter Konkurrenz zur Repräsentation des Attributs als Relation, die im Attributwert ja nicht eindeutig sein muß und von daher immer „mengenwertig“ ist. Jedoch sind die beiden Ansätze nicht äquivalent: Je nachdem, ob man beispielsweise das Attribut *Geschwister* als mengenwertige Funktion oder als Relation deklariert, wird die Tatsache, daß *a* die Geschwister *b* und *c* hat, entweder als {*Geschwister*(*a*, {*b*, *c*})} (üblicherweise {*Geschwister*(*a*) = {*b*, *c*}} notiert) oder als {*Geschwister*(*a*, *b*), *Geschwister*(*a*, *c*)} dargestellt. Im ersten Fall ist *a* die Menge {*b*, *c*} als Einheit zugeordnet, während im zweiten zwei Geschwisterpaare spezifiziert werden.

Anmerkung: Die generative oder konstruktive Rolle, die Funktionen in algebraischen Spezifikationen [Ehrich et al. 1989] oder in der Logik (sog. Konstruktorterme [Aït-Kaci & Nasr 1986; Smolka & Aït-Kaci 1989]) spielen, findet in dieser Arbeit keine Anwendung.

Attribute (und je nach Betrachtungsweise auch Beziehungen) dienen in der Modellierung im wesentlichen zwei Zwecken:

1. der Klassifikation (dazu gehört sowohl die Eigenschaft, daß ein Objekt eine gewisse Eigenschaft wie z. B. eine Farbe hat, als auch, welche das ist) und
2. der Beschreibung des Zustandes (die Summe der veränderlichen Attributwerte macht den momentanen Zustand eines Objektes aus).

Auf den ersten Blick scheinen sich die beiden gegenseitig auszuschließen; es dienen nämlich typischerweise die konstanten Eigenschaften der Klassifikation und die variablen der Spezifikation des Zustandes. Tatsächlich aber kann auch der Zustand eines Objektes Grundlage für die Klassifikation sein, und zwar für die dynamische. Eine Diskussion dazu folgt im nächsten Kapitel in Abschnitt 3.4.3.

Ob man Attribute überhaupt benötigt, wenn man Relationen hat (wie z. B. von Kent [1978] diskutiert), ist im wesentlichen eine ontologische Frage. Ich werde

in der weiteren Darstellung auf die Verwendung von Attributen und Funktionen verzichten, allerdings nur in dem Bewußtsein, daß sie jederzeit und (z. B. auch als spezielle Relationen) ohne Schwierigkeiten in die Modellierungssprache aufgenommen werden können. Mehr dazu in Abschnitt 2.3.1.

Vorgegebene Relationen

Viele Modellierungssprachen sehen eine oder mehrere vordefinierte, d. h. fest benannte und mit einer speziellen Semantik ausgestattete Relationen auf der objektsprachlichen Ebene vor, die für jedes Modell zur Verfügung stehen. Ein typisches Beispiel hierfür ist die Teil-Ganzes-Beziehung, die ein Objekt in seine Bestandteile zerlegt. Manche Ansätze gehen sogar so weit, die Zahl der möglichen Relationen auf eine Anzahl Primitive zu beschränken und es zur Aufgabe des Modellierenden zu machen, alle anderen Relationen darauf zurückzuführen [Schank 1975].

Prinzipiell ist nichts gegen vordefinierte Relationen einzuwenden. In der Praxis ergeben sich jedoch zwei Probleme:

1. Da eine solche Relation mit der Modellierungssprache nicht durch Aufzählung der Extension vorgegeben werden kann, muß ihre Intension eindeutig und unmißverständlich festgeschrieben sein. Die Vielzahl der verschiedenen existierenden Auslegungen der Teil-Ganzes-Beziehung allein zeigt aber, daß dies nicht so leicht möglich ist.
2. Es ist zu prüfen, ob die Relation wirklich eine Beziehung zwischen Objekten herstellt (wie es z. B. die Teil-Ganzes-Relation tut), oder ob sie auf einer höheren Stufe angesiedelt ist. Da die meisten Modellierungssprachen aber die Definition eigener höherstufiger Relationen nicht erlauben (eine Ausnahme bildet z. B. TELOS [Mylopoulos et al. 1990]), wäre sie auch keine spezielle Relation eines Modells, sondern fester Bestandteil der Modellierungs- und damit der Metasprache des Modells.

Das Paradebeispiel für vordefinierte Relationen, die Is-a- oder Vererbungsrelation, steht im allgemeinen nicht auf einer Stufe mit anderen Relationen, da sie in der Regel keine Beziehung zwischen Objekten, sondern zwischen Typen wiedergibt. Mehr dazu in Abschnitt 2.2.3.

Eine sinnvolle Alternative zu vordefinierten Relationen, die die Probleme mit der eindeutigen Spezifikation vermeidet, ist die Einteilung von Relationen in

mehrere Arten, z. B. in Aggregationen[19] (von denen die Teil-Ganzes-Beziehung eine besondere wäre) und in Assoziationen [Rumbaugh et al. 1998]. Eine Zuordnung einer Relation zu einer solchen Art gibt dann nur einen Teil der Semantik der Relation vor, z. B. die Existenzabhängigkeit des Aggregats von seinen Teilen (oder umgekehrt).

Relationen als Typen und Relationen über Relationen

In Anerkennung der Tatsache, daß die Extension einer Relation, genau wie die eines Typs, eine Menge (wenn auch von Tupeln) ist, werden Relationen in der Modellierung häufig auch als Typen aufgefaßt und als Beziehungstypen (relationship types, in Unterscheidung von entity types) bezeichnet. Diese Verallgemeinerung hat allerdings die folgende weitreichende Konsequenz.

Wenn Relationen Typen sind, dann sollten Relationen auch auf Relationen deklariert werden können. Dies gehört z. B. zum Repertoire von NIAM [Nijssen & Halpin 1989; Halpin 1995], dessen Formalismus und Methodik beide stark linguistisch orientiert sind. Da Relationen Prädikaten entsprechen, entspricht eine Relation über eine Relation einer Aussage über eine Aussage, also einem Prädikat zweiter Stufe. Die Frage, die man sich stellen muß, ist allerdings, ob Aussagen zweiter Stufe in der Modellierung objektsprachlich überhaupt vorkommen. Die meisten Beispiele, die man in der Literatur zur Modellierung dazu findet, sind kein Beleg dafür – sie dienen, wie das Hantieren mit Assoziationsklassen in UML [OMG 1999, § 3.45.5], lediglich dazu, unvollständige (weil zu niedrigstellige) Relationen um die fehlende Information auf gleicher Stufe zu ergänzen. Wie es tatsächlich zu Aussagen über Aussagen in der Modellierung kommt und was dazu gehört, kann man z. B. bei Sowa [1984] nachlesen; vgl. aber auch Punkt 4 zu Metarelationen unten.

Relationen auf der Metaebene

So wie ein Typ von einem Objekt, so abstrahiert eine Relation von einer Beziehung zwischen Objekten. Wenn man auf der Ebene der Typen und Relationen nun selbst wieder Beziehungen zwischen diesen zuläßt, so lassen sich diese wiederum zu Relationen, den *Metarelationen*, zusammenfassen. Man kann verschiedene Arten von Metarelationen unterscheiden, je nachdem, was man in Beziehung setzt.

[19] Gelegentlich wird auch das kartesische Produkt als Aggregation bezeichnet [Abiteboul & Hull 1987]; es sind dann alle Relationen Aggregationen.

1. *Beziehungen zwischen Objekten und Typen*

   Die unvermeidliche Art der Beziehung zwischen Objekten und Typen bzw. Beziehungen und Relationen ist die des Instanzseins von[20]: Ein Objekt ist Instanz eines (oder mehrerer) Typen und eine Beziehung ist Instanz einer Relation. Eine weitere Beziehung zwischen Instanzen und Typen wird von machen Autoren im Zusammenhang mit Rollen eingeführt; mehr dazu im nächsten Kapitel.
2. *Beziehungen zwischen Typen*

   Das herausragende Beispiel für eine Relation, deren Elemente Paare von Typen sind, ist die Typsubsumtion (gewöhnlich auch als Is-a-Relation bezeichnet). Sie wird auf der Menge der Typen deklariert und ist damit eine Relation der Metaebene. Abschnitt 2.2.3 befaßt sich ausführlich damit.
3. *Beziehungen zwischen Typen und Relationen*

   Relationen werden auf Typen deklariert. Daher besteht auch zwischen einer Relation und ihren Typen eine Beziehung. Die Relation, die das einfängt, ist ebenfalls eine Metarelation.
4. *Beziehungen zwischen Relationen*

   Beziehungen zwischen Relationen sind ebenfalls Metarelationen. Eine besondere solche Metarelation ist die, die eine Relationshierarchie, z. B. eine Typsubsumtion auf Relationstypen, bildet. Mehr dazu in Abschnitt 2.2.4.

2.2.3 Signale und Botschaften

Die Zustandsänderungen in einem dynamischen Modell werden nur zum Teil von außen herbeigeführt; einmal angestoßen, beginnt im Modell i. d. R. eine Art Kettenreaktion, die zahlreiche weitere Veränderungen nach sich zieht. Diese Kettenreaktion braucht einen Ausbreitungsmechanismus, und der besteht aus dem Versenden von und dem Reagieren auf Nachrichten, das sind Signale und Botschaften.[21]

[20] Der Unterschied zwischen dem Instanzsein von und dem (wohl geläufigeren) Begriff der Instanziierung ist der, daß ersteres ein Zustand, letzteres ein Vorgang ist. Da ich mich nicht mit operationalen Aspekten befasse, spielt die Instanziierung in dieser Arbeit keine Rolle.

[21] Abweichend davon sind in der Datenmodellierung vielleicht Integritätsbedingungen und Trigger das Mittel der Wahl; Nachrichten sind aber konzeptuell klarer und werden daher hier vorgezogen.

Die Ausbreitungswege von Nachrichten in einem Modell sind durch die bestehenden Beziehungen zwischen den Objekten vorgegeben. Eine Nachricht kann also nur von einem Objekt zu einem anderen gesendet werden, wenn diese in Beziehung miteinander stehen, wenn der Sender den Empfänger „kennt". Botschaften unterscheiden sich von Signalen dadurch, daß sie neben der eigentlichen Nachricht auch noch Objekte enthalten, die mit der Nachricht in Verbindung stehen. Dies setzt wiederum ein Kennen dieses Objektes beim Sender voraus und impliziert es beim Empfänger.

Die genauen Mechanismen von Nachrichten und deren Verbreitung sind für die Spezifikation der dynamischen Aspekte einer Modellierungssprache von größter Bedeutung. Hier aber ist das Interesse daran von vorn herein auf die Existenz von möglichen Zeitverläufen, die in einer nicht näher beschriebenen Weise vom dynamischen Modell spezifiziert werden müssen, beschränkt; das Thema soll daher nicht weiter vertieft werden. Es liegt jedoch auf der Hand, daß sich Formalismen wie endliche Automaten und Petri-Netze gut zur Spezifikation dynamischer Modelle eignen; sie werden z. B. in UML verwendet.

2.2.4 Hierarchie und Vererbung

In der Modellierung wird häufig eine Halbordnung in Form einer transitiven, reflexiven und antisymmetrischen Relation, die sog. *Subtypenrelation*, auf der Menge der Typen definiert. Die in der Modellierung favorisierte Interpretation dieser Relation ist die der Subsumtion; andere Interpretationen kommen jedoch auch vor.

Die Elemente der Subtypenrelation sind Paaren von Typen. Sie steht also nicht auf einer Ebene mit anderen speziellen Relationen wie etwa der Teil-Ganzes-Beziehung (deren Elemente Paare von Objekten sind), sondern ist eine (mit fest vorgegebener Semantik versehene) Relation der Metasprache.[22] Diese Unterscheidung wird häufig vernachlässigt; dadurch kann es aber, wie gezeigt werden wird, zu schwer auflösbaren Widersprüchen kommen.

Die Subsumtionshierarchie

Die übliche semantische Interpretation der Subtypenrelation ist die als (mengentheoretische) Teilmengenrelation: Der Supertyp umfaßt oder subsumiert die

[22] S. hierzu auch Sowa [1992, S. 1504]: „a type-subtype link makes a second-order assertion about types".

Objekte seiner Subtypen; da durch die Subtypenrelation eine Hierarchie aufgebaut wird, spricht man von einer Subsumtionshierarchie.[23]

Vererbung

Durch die Interpretation der Subtypenrelation als Typsubsumtion ergibt sich automatisch das, was gemeinhin als *Vererbung* bezeichnet wird: Wenn eine Instanz eines Subtyps auch Instanz seiner Supertypen ist, so werden alle deren Objekte charakterisierenden Eigenschaften, die in der Intension zusammengefaßt sind, auch auf das Objekt des Subtyps zutreffen – es erbt also gewissermaßen deren Eigenschaften[24]. Die Intension eines Subtyps, das ist die Summe der Eigenschaften, die diesen Typ charakterisieren, schließt also die Intension des Supertyps mit ein; sie impliziert diese. Diese Beobachtung führt zu der etwas oberflächlichen Aussage, die Intension eines Subtyps sei eine Obermenge der Intension seines Supertyps [Wieringa et al. 1994; Hainaut et al. 1996], was erstens voraussetzt, daß die Intension als Menge dargestellt wird, und zweitens Subtypen, die durch Einschränkung eines Prädikats und nicht durch Hinzunahme weitere Prädikate entstehen (vgl. z. B. [Wegner 1987]), ausschließt.[25]

Prinzip der Substituierbarkeit

Viel wichtiger für die Modellierung als der Mechanismus der Vererbung, die gewissermaßen ein Abfallprodukt der Interpretation der Subtypenrelation als Teilmengenrelation ist, ist das Prinzip der Substituierbarkeit (principle of substitutability) [Wegner & Zdonik 1988], das sich ebenfalls aus der mengentheoretischen Interpretation der Subtypenrelation ergibt: Überall dort, wo ein (beliebiges) Objekt eines bestimmten Typs gefordert ist (beispielsweise an einer Stelle einer Beziehung), darf auch ein Objekt eines Subtyps auftreten, da die Menge der Objekte eines Subtyps eine Teilmenge der Menge der Objekte des Supertyps ist. Diese Eigenschaft ist für die objektorientierte Modellierung ins-

[23] Die Bezeichnung dieser Relation und der dadurch induzierten Hierarchie als Is-a ist zwar weit verbreitet, aber problematisch, da sie die Unterscheidung zwischen Untertyp- und Instanzsein verwischt.

[24] Diese Metapher ist freilich nicht ganz treffend, da es sich ja um dasselbe Objekt handelt, Vererbung aber verschiedene Objekte betreffen müßte. Tatsächlich wird die Intension eines Typs auf seine Untertypen vererbt.

[25] Den inversen Zusammenhang des Umfangs von Intension und Extension, das sog. Law of reverse correlation [ISO 1987], hatte übrigens bereits Aristoteles bemerkt, ohne die Begriffe jedoch selbst zu verwenden [Sowa 1984].

gesamt charakterisierend und wird oft (etwas frei; vgl. Abschnitt 3.2.6) als eine Art des Polymorphismus bezeichnet [Cardelli & Wegner 1985].

Allerdings, und das ist wichtig, ist das Prinzip der Substituierbarkeit nicht allein durch die Typsubsumtion gegeben: Wenn Supertypen selbst Instanzen haben, dann ist es durchaus möglich, daß diese Eigenschaften besitzen, die den Instanzen ihrer Subtypen aufgrund der Einschränkung der Intension abgesprochen werden. Wegner und Zdonik geben dazu das Beispiel des Typs *Person* und seines Subtyps *Retiree* an, wobei dem Rentner oder Pensionär kein Alter unter 65 Jahren zugewiesen werden kann, so daß er eben nicht überall dort auftreten kann, wo eine Person erwartet wird. Man beachte aber, daß nach dieser Argumentation umgekehrt eine Person sehr wohl da auftreten kann, wo ein Rentner erwartet wird – in der Tat ist, wie argumentiert werden wird, *Retiree* gar kein Subtyp von *Person*, sondern ein Zustand. Andererseits ist es auch nicht sinnvoll, einen Typ instanziierbar zu machen, der Subtypen hat [Hürsch 1994; Steimann 1999a; 2000a], und das Prinzip der Substituierbarkeit ist eigentlich keines der Substitution von Instanzen verschiedener Typen gegeneinander (die werden nämlich nie gleich sein, sonst wären ja ihre Typen nicht verschieden), sondern eins der Substitution von Abstraktionen (hier: Platzhaltern eines Supertyps) mit konkreten Instanzen, also im Prinzip eins der (erweiterten) Zuweisungskompatibilität [Steimann 2000c].

Instanzsein und Elementsein

Die Interpretation der Subtypen- als eine Teilmengenrelation impliziert eine subtile Asymmetrie: Während auf der semantischen Seite ein Element einer zu einem Typ gehörenden Menge immer auch Element der zu den Supertypen gehörenden Obermengen ist, kann (im vorherrschenden Fall der Einfachklassifikation) streng genommen ein (Modell-)Objekt als Instanz eines Typs nicht zugleich Instanz seiner Supertypen sein, da jedes Objekt genau einem Typen zugehört. Die Sichtweise in der Literatur ist diesbezüglich eher uneinheitlich (man unterscheidet einerseits zwischen direkten und indirekten Instanzen bzw. dem spezifischsten Typ einer Instanz – andererseits geht man aber auch wieder davon aus, daß es zu jeder Instanz eines Typs jeweils eine weitere Instanz in jedem seiner Supertypen, die sog. Images [Smith & Smith 1977; Wagner 1989], gibt); mehr dazu in Abschnitt 2.3.2.

Die Abstraktionshierarchie

Man kann den Übergang von einem Typ zu einem Supertyp als eine Art Abstraktion auffassen, wenn dabei Teile der Intension unterschlagen werden. Allerdings kann der Supertyp dann nicht instanziiert werden (also keine eigenen Instanzen haben), denn welches Individuum ist schon eine Abstraktion? Wenn man festlegt, daß alle Supertypen eines Modells Abstraktionen[26] sind, dann ist die Subsumtionshierarchie eine reine *Abstraktionshierarchie*, in der nur die Blätter selbst Objekte haben. Eine solche Hierarchie hat gewisse Vorteile [Steimann 1999a; 2000a]; sie ist auch Grundmerkmal der in den nächsten Abschnitten vorgelegten Modellierungssprache.

Spezialisierung und Generalisierung

Die oben beschriebene Subtypenrelation wird auch häufig als Spezialisierung und deren Umkehrung als Generalisierung bezeichnet. Die beiden Begriffe werden allerdings nicht immer symmetrisch gebraucht (so z. B. nicht in [Abiteboul & Hull 1987; ter Hofstede & van der Weide 1993; Halpin & Proper 1995]): Während die Generalisierung eine gängige Form der Abstraktion ist, die das Genus (Wortstamm!) verschiedener Objekte herauskehrt und daher im allgemeinen zu abstrakten Typen führt, versteht man unter Spezialisierung gelegentlich die Einschränkung bestehender oder das Hinzufügen weiterer Eigenschaften zu einem konkreten Typ von Objekten mit dem Zweck, diese für einen speziellen Fall (oder eine bestimmte Rolle; s. z. B. [Elmasri & Navathe 1994, S. 612-3]) verwendbar zu machen, eben zu spezialisieren. So wird zum Beispiel *Student* häufig als Spezialisierung des Typs *Person* aufgefaßt[27], *Person* aber nicht als Generalisierung von *Student*. Die Spezialisierung eines (konkreten) Typs kann also unter Umständen durch Generalisierung nicht rückgängig gemacht werden (z. B. weil sie zu einem abstrakten und damit nicht instanziierbaren Typ führt).

Für den Fall, daß eine Spezialisierung zwar die Intension eines Typs einschränkt (um zusätzliche Angaben ergänzt), seine Extension davon aber unberührt bleibt, will ich nicht von einer Spezialisierung, sondern von einer *Konkretisierung* sprechen. Konkretisierungen liegen z. B. dann vor, wenn ein abstrakter Typ (eine Generalisierung) nur einen Subtypen hat; sie haben aber auch im Zusammenhang mit Rollen (Kapitel 3) eine gewisse Bedeutung.

[26] in der objektorientierten Programmierung als abstrakte Klassen bekannt
[27] ein Fehler, wie sich zeigen wird

Typerweiterung

Eine alternative Basis der Typhierarchie ist die sogenannte Typerweiterung (type extension) [Wirth 1988], bei der von einem Basistyp (base type) durch Hinzufügen weiterer Komponenten (Felder) ein erweiterter Typ (extension) abgeleitet wird, der selbst wieder als Basis für weitere Ableitungen dienen kann. In einem klassischen Beispiel für die Typerweiterung werden Punkte im dreidimensionalen Raum als Erweiterung des Typs von Punkten im zweidimensionalen Raum um eine Koordinate implementiert. Während diese Sichtweise einen gewissen praktischen Vorteil mit sich bringt und zumindest eine Zuweisungskompatibilität (assignment compatibility) gewährleistet, ist sie für die Modellierung der Realität nicht angemessen: Semantisch ist die Menge der Punkte im dreidimensionalen Raum keine Teilmenge der Punkte im zweidimensionalen Raum (genauso wenig wie umgekehrt), selbst wenn sie die ersten beiden Koordinaten teilen. Es ist zu erwarten, daß eine solche, nicht den realen Verhältnissen entsprechende Modellierung über kurz oder lang zu Inkonsistenzen führt, weil die Substituierbarkeit semantisch nicht gegeben ist [Steimann 1999a].[28]

Anmerkung: Die Typerweiterung wurde zunächst nicht für die objektorientierte Modellierung, sondern zur Erweiterung PASCAL-artiger Typen (Records) in Programmiersprachen definiert, deren Instanzen Tupel im mathematischen Sinne sind und anders als Objekte keine Identität haben. Dazu schreibt Wirth:

> *„It is tempting to consider the instances of a type as a set characterized by the type. Then the instances of an extended type appear as a subset. However, the "merging" of different elements of the subset into the same element of the superset, as may happen in the case of projections* [z. B. im Zuge einer Wertzuweisung]*, is inconsistent with the set notion in which distinct elements retain their identity when being removed from that subset.“* [Wirth 1988, S. 207]

Inverse Typhierarchie

Immer wieder wird in der Literatur zur objektorientierten Modellierung und Programmierung die Möglichkeit der Umkehr der Vererbungsrichtung diskutiert (z. B. [Halbert & O'Brien 1987; Winkler 1992; Al-Ahmad & Steegmans 1999] oder auch, in einem anderen Kontext, [Halpin & Proper 1995]). Der

[28] Wenn überhaupt, dann sind beide Typen Einschränkungen (Spezialisierungen) eines abstrakten Typs von Punkten im *n*-dimensionalen Raum, die beispielsweise durch einen *n*-dimensionalen Koordinatenvektor dargestellt werden. Allerdings ist solches nicht durch eine Typerweiterung darstellbar. Es können aber Punkte im dreidimensionalen Raum genau dieselben Rollen wie Punkte im zweidimensionalen Raum spielen, indem sie einfach eine Dimension verbergen. Mehr dazu im nächsten Kapitel.

Grund hierfür ist zumeist, daß der bereits erwähnte inverse Zusammenhang zwischen Intension und Extension in bestimmten Fällen nicht zuzutreffen scheint. Als Standardbeispiel hierfür wird die Hierarchie geometrischer Formen herangezogen, in der ein Quadrat zwar konzeptuell ein Rechteck ist, das Rechteck aber aufgrund seiner (im Verhältnis zum Quadrat) relativen Unregelmäßigkeit eine „größere“ Intension (nämlich zwei Seitenlängen statt nur einer) habe und damit sinnvollerweise als Unterklasse von Quadrat implementiert werde. Tatsächlich aber haben sowohl Quadrate als auch Rechtecke vier Seiten und damit auch vier Seitenlängen, nur daß die Intension von Quadrat um die Bedingung reicher ist, daß alle vier Seiten gleich lang sein müssen (so daß man sich nur eine merken muß) [Steimann 2000a].

Relationshierarchien und Überladung

Wie bereits in Abschnitt 2.2.2 bemerkt, kann man Relationen selbst als Typen auffassen. Gelegentlich wird daher auch in Analogie zur Subtypenrelation eine Subrelationen-Relation auf Relationen gefordert (z. B. [Sowa 1984, Annahme 3.2.8, S. 82]). Diese müßten dann in Analogie zur Subtypenrelation als Teilmengenrelation interpretiert werden; jede Subrelation entspräche also einer Teilmenge der ihren Superrelationen entsprechenden Tupelmengen. Weit häufiger ist jedoch die implizite Bildung von „Subrelationen“ durch Überladen der Relation.

Die Idee dahinter ist so einfach wie fundamental. Wenn man den Typ einer Stelle einer bestehenden Relation auf einen Subtyp einschränkt, so folgt daraus, daß die Typen der anderen Stellen ebenfalls eingeschränkt, nicht aber erweitert werden dürfen (die sog. *Kovarianz* [Abadi & Cardelli 1996] oder *Monotonität* [Meseguer & Goguen 1993]): Wenn die Elemente zweier oder mehrerer Mengen miteinander in Beziehung stehen, dann stehen Teilmengen von diesen Mengen stets mit Teilmengen in Beziehung. Die Einschränkung eines Typs in einer Relationsdeklaration führt damit automatisch zur Einschränkung der Relation selbst. Daß beim Überladen für die eingeschränkte Relation anders als bei Typen kein neuer Name vergeben wird, mag dabei als Vorteil oder als Nachteil gegenüber einer Subrelationenbildung gesehen werden. Es unterstreicht auf jeden Fall die unterschiedlichen Rollen, die Typen und Relationen in der Modellierung beigemessen werden.

Überladung und Relationshierarchien werden in Abschnitt 2.3.4 dieses und in den Abschnitten 3.2.3 und 3.2.4 des nächsten Kapitels ausführlicher behandelt;

die Austauschbarkeit von beiden in gewissen Grenzen ist Gegenstand von Abschnitt 4.1.

Objekthierarchien

Die Aggregation als Teil-Ganzes-Beziehung oder als Ausdruck der Existenzabhängigkeit ist ein typisches Beispiel für eine eine Objekthierarchie aufspannende Relation. Anders als die Typhierarchien werden die Objekthierarchien nämlich durch objektsprachlich definierte Relationen vorgegeben. Dabei kann die Objekthierarchie homogener (durch eine, möglicherweise überladene Relation definiert) oder heterogener Struktur sein. Das klassische Beispiel für eine homogene Objekthierarchie ist im Composite pattern [Gamma et al. 1995] eingefangen, das in den baumartigen Vezeichnisstrukturen von Betriebssystemen oder in der rekursiven Teil-Ganzes-Beziehung von Stücklisten seine Ausprägungen hat. Heterogene Objekthierarchien werden durch eine einfache Klassifikation von Relationen, z. B. als Aggregationen oder Assoziation, aufgespannt. So bildet ein Objekt mit all seinen über Aggregationen verbundenen Objekten und rekursiv deren Aggregationen stets eine heterogene (da es sich um unterschiedliche Relationen handelt) Objekthierarchie. Assoziationen führen im allgemeinen nicht zu einer Hierarchie, da sie Zyklen enthalten können.

Objekthierarchien werden aber auch zur Ergänzung der als unbefriedigend empfundenen Eigenschaften von Typhierarchien [Sciore 1989] oder gar als deren Ersatz herangezogen. Die Realisierung der Vererbungshierarchie als Objekthierarchie jedoch, wie man sie gelegentlich vorfindet, führt zu einem Widerspruch, wie die folgenden Überlegungen aufzeigen.

Gegeben seien vier Typen *A*, *B*, *C* und *D*, die Relationsdeklaration *has-a*: *A B* sowie die überladene Relation *is-a* mit den Signaturen *is-a*: *C B* und *is-a*: *D B*, die ausdrücken soll, daß *C* und *D* Subtypen von *B* sind (Abbildung 2.1). Diese Darstellung von Vererbung ist nicht unüblich, hat aber zur Folge, daß jedem Objekt eines Subtyps genau ein Objekt seines Supertyps zugeordnet ist, die Vererbung also durch eine degenerierte Objekthierarchie (Liste) mit Objekten von der Wurzel der Typhierarchie bis zum (Typ des) erbenden Objektes repräsentiert wird.

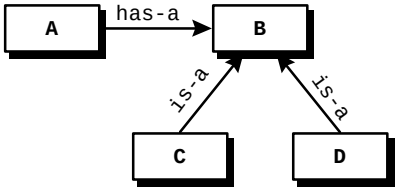

Abbildung 2.1: Vererbungs- als Objekthierarchie. *is-a* steht mit *has-a* auf einer Stufe, muß sich also selbst vererben. Dies führt zu einem Widerspruch (s. Text).

Nun gebietet die allgemein unterstellte Semantik der Vererbung und die Deklaration der Relation *has-a*: *A B*, daß Objekte vom Typ *A* auch Objekte der Typen *C* und *D* haben können, also *has-a*: *A C* und *has-a*: *A D*. Dies ist erwünscht. Da aber *is-a* auf derselben Ebene wie *has-a* definiert und lediglich mit einer speziellen Semantik ausgestattet ist, folgt unter anderem auch *is-a*: *C D* und *is-a*: *D C* (und damit, wenn *is-a* eine Halbordnung ist, *C* = *D*). Dies steht jedoch im Gegensatz zu dem, was mit den Deklarationen ausgedrückt werden sollte. Die Is-a-Relation vererbt sich eben nicht selbst, ein Konflikt, der sich nur auflösen läßt, indem man die Semantik der Vererbung mit einer entsprechenden Ausschlußklausel versieht[29] oder sie auf die Metaebene hebt.

[29] UML hat ein ähnliches Problem, das dadurch umgangen wird, daß für jede Art von Generalisierung festgelegt wird, was vererbt wird [OMG 1999].

2.3 Eine einfache Modellierungssprache

Wie bereits erwähnt, ist die Modellierungssprache, die im folgenden spezifiziert wird, ist in erster Linie als Vorbereitung der formalen Einführung des Rollenkonzepts im nächsten Kapitel zu verstehen. Die Erweiterung kann jedoch auch in eine andere Richtung gehen, wie das Anwendungsbeispiel aus Abschnitt 2.4.4 zeigen wird.

Da im weiteren Verlauf der Arbeit häufiger auf die hier definierte Modellierungssprache Bezug genommen werden wird, soll sie einen Namen haben. Nach langem Überlegen (und dem Verwerfen unzähliger unmöglicher Akronyme) habe ich mich für FREGE entschieden, weil auf Freges Begriffsschrift u. a. die Unterscheidung von Begriffen und Beziehungen zurückgeht, die für die Modellierung so wesentlich ist. Die Erweiterung der Sprache zu einer mit Rollen im nächsten Kapitel wird dann LODWICK heißen, was zwar nicht den geschichtlichen Abläufen entspricht, aber den zweiten historischen Ideenlieferant für diese Arbeit würdigt.

Die Modellierungssprache FREGE basiert im wesentlichen auf einem Ausschnitt der ordnungssortierten Prädikatenlogik [Oberschelp 1962]. Diese ist zwar nicht ausdrucksstärker als die einsortige Prädikatenlogik, ihre Ausdrücke sind jedoch kürzer, besser lesbar und nicht zuletzt den gängigen Daten-, konzeptuellen und objektorientierten Modellierungssprachen näher. Insbesondere kommt so die Sprache, wegen der Beschränkung auf sorten- oder typenrechte Ausdrücke, ohne einen expliziten Schlußfolgerungs- oder Ableitungsbegriff aus, was die Sache erheblich einfacher macht.

Einige der Schreib- und Sprechweisen, die im Umfeld der Ordnungssortiertheit üblich sind, sollen im folgenden kurz angeführt werden. Wenn M eine Menge ist, dann ist M^* die Menge aller endlichen Aneinanderreihungen (Ketten) von null oder mehr Elementen aus M. Ein $w \in M^*$ mit $w = m_1 \ldots m_n$ hat die Länge $|w| = n$. Eine Halbordnung $\leq$ auf M ist eine reflexive, transitive und antisymmetrische Relation $\leq \subseteq M \times M$. Für $a \leq b$ und $a \neq b$ schreibt man auch $a < b$, und $a \leq b$ und $a < b$ sind gleichbedeutend mit $b \geq a$ bzw. $b > a$. Die Ordnung $\leq$ setzt

sich auf Paare von gleich langen Ketten von Elementen aus *M* wie folgt fort: $a_1 \ldots a_n \leq b_1 \ldots b_n$ genau dann, wenn $a_1 \leq b_1 \wedge \ldots \wedge a_n \leq b_n$.

Die semantische Interpretation der Symbole der Modellierungssprache erfolgt über einen Grundbereich *M*. Wenn *a* ein Symbol des Modells ist, dann sei a^M die Interpretation dieses Symbols, int(*a*) seine Intension und ext(*a*) seine Extension. Für die gesamte Darstellung gelte die Annahme, daß alle Namen eindeutig sind, d. h., für $a \neq b$ sei $a^M \neq b^M$ und int(*a*) ≠ int(*b*). Für die Extensionen muß das jedoch nicht gelten.

Die Basis des dynamischen Modells ist eine (abzählbar unendlich große) total geordnete Menge *T* von aufeinanderfolgenden diskreten Zeitpunkten $t \in T$. Das dynamische Modell sei wie gesagt die Spezifikation einer (unendlich großen) Menge von möglichen Verläufen *V*. Jeder Verlauf $v \in V$ ist eine (zeitdiskrete) Folge von Szenarien (Schnappschüssen, Zuständen), wobei das *t*-te Element der Folge die Spezifikation all der Individuen und all der Beziehungen zwischen diesen sei, die zum Zeitpunkt *t* im gegebenen Verlauf bestehen.

2.3.1 Individuen und Spezies

Eine Modellspezifikation in FREGE umfasse zunächst eine Menge *S* von Typsymbolen, *Arten* oder *Spezies* genannt. Zu jedem $s \in S$ gehöre eine Menge von Bezeichnern I_s, deren Elemente $a \in I_s$ für die Instanzen der Spezies *s* stehen und die (und nur die) ich im folgenden *Individuen* nenne, weil sie elementar, also insbesondere weder Aggregationen noch Typen sind. Die Mengen I_s seien paarweise disjunkt – jedes Individuum gehört also genau einer Spezies an. Um auszudrücken, daß *a* Individuum der Spezies *s* ist, schreibe man (objektsprachlich) *a*:*s*.

$$\left(I_s\right)_{s \in S}$$

ist also eine *S*-indizierte Mengenfamilie von Individuenbezeichnern, und

$$I := \bigcup_{s \in S} I_s$$

die Menge aller Individuenbezeichner eines Modells. *I* kann abzählbar unendlich groß sein; da in den meisten Modellen aber die Ausdrücke, die Individuen enthalten, nur Beispielcharakter haben, reicht i. d. R. eine recht kleine Menge aus. Jedes $a \in I$ wird semantisch als ein Element a^M des Grundbereichs der Interpretation, *M*, interpretiert.

Zusätzlich zu den I_s sei mit jeder Spezies $s \in S$ assoziiert

- eine Intension int(s), die die Eigenschaften angibt, die allen Individuen der Spezies beigemessen werden (die die Spezies charakterisieren),
- eine Extension ext(s), die die Individuen der Spezies aufzählt, und
- eine Interpretation (oder Denotation) $s^M \subseteq M$, die die Semantik der Spezies ausmacht.

Man beachte den Unterschied zwischen semantischer Interpretation und Extension: Beides sind mit einem Typsymbol assoziierte Mengen, jedoch rangieren sie über verschiedene Grundbereiche – den Grundbereich der Interpretation M bzw. die Menge der Individuenbezeichner I.

Auch wenn ich auf die Formulierung der Intensionen von Spezies hier nicht weiter eingehen will, sei doch bemerkt, daß sie auf eine Verknüpfung einstelliger Prädikate im weitesten Sinne beschränkt sei. So werde beispielsweise der Umstand, daß alle Individuen einer Spezies rot sind, als ein einstelliges Prädikat (etwa *rot*(x)) aufgefaßt, auch wenn man es eher als ein Attribut der Form *Farbe*(x) = *rot* und damit als eine zweistellige Relation (oder Funktion) darstellen möchte. Ausdrücklich keine solchen Prädikate sind aber Relationen wie *ist Kind von*, wenn damit verschiedene Individuen in Beziehung gesetzt werden sollen. Damit soll die Beteiligung einer Spezies an Beziehungen als Bestandteil der Intension der Spezies ausdrücklich ausgeschlossen werden.

Anmerkung: Daß diese Regel keine absolute sein kann, zeigt die Charakterisierung der Spezies *Mensch* durch das Attribut *Kind von Adam*, das je nach Kontext durchaus als ein- oder zweistelliges Prädikat aufgefaßt werden kann. Der Modellierer hat die Freiheit, das eine oder das andere zu tun, nur soll die Regelung die sein, daß alles, was zur Intension einer Spezies gehört, keine Relation des Modells mehr sein kann. Man beachte also, daß es in FREGE nicht zur Intension einer Spezies gehört, welche Beziehungen dessen Individuen auf Modellebene eingehen können. Dies wird in FREGE allein und unabhängig davon durch die Deklaration der Relationen festgelegt. Mit der Einführung von Rollen soll das jedoch anders werden.

Während die Intension einer Spezies üblicherweise invariant ist (sie ist die Beschreibung, die auf alle Individuen einer Spezies jederzeit zutrifft), ist die Veränderlichkeit der Extension charakteristisch für jedes dynamische zu modellierende System. Man unterscheidet daher zwischen

- der statischen Extension ext(s), die alle jemals existierenden Individuen der Spezies s umfaßt, und
- den dynamischen Extensionen $\text{ext}_{v,t}(s)$, die jeweils alle in einem Verlauf v zum Zeitpunkt t existierenden Individuen der Spezies umfassen.

Für die statische Extension ext(s), die Teil des statischen Modells ist, gilt

$$\text{ext}(s) = I_s .$$

Für die dynamischen Extensionen $\text{ext}_{v,t}(s)$ gilt stets

$$\text{ext}_{v,t}(s) \subseteq \text{ext}(s) ;$$

sie sind Teil des dynamischen Modells. Erzeugung und Destruktion eines Individuums manifestieren sich im dynamischen Modell durch eine entsprechende Änderung der dynamischen Extension seiner Spezies.

Anmerkung: Auch wenn die Intension zeitlich invariant ist, kann man diese in zwei Teile gliedern – einen, der die zeitunabhängigen Eigenschaften der beschriebenen Elemente formuliert und der Teil des statischen Modells ist, und einen für die zeitabhängigen. Zur Formulierung des zweiten wird man so etwas wie temporale Logik oder Automatentheorie benötigen; deren Ausdrücke sind Teil des dynamischen Modells, jedoch selbst zeitlich invariant.

2.3.2 Genera und natürliche Typen

Neben den Spezies umfasse eine Modellspezifikation eine weitere, zu S disjunkte Menge G von Typsymbolen, *Gattungen* oder *Genera* genannt. Zu jedem Genus $g \in G$ gehöre wiederum eine Extension, eine Intension und eine Denotation. Auf den Genera sei mit $\leq_{GG}$ eine Halbordnung definiert, so daß für $g, g' \in G$ mit $g \leq_{GG} g'$

$$\text{ext}(g) \subseteq \text{ext}(g')$$

und

$$\text{int}(g) \Rightarrow \text{int}(g')$$

gelte, wobei der Implikationspfeil ausdrücken soll, daß die Intension von g die von g' bedingt (diese, falls die Intensionen als logische Ausdrücke formuliert sind, impliziert). Die semantische Interpretation eines Genus g, g^M, ist wie die einer Spezies eine Teilmenge des Grundbereichs. Es gilt auch für die Interpretation, daß für $g \leq_{GG} g'$

$$g^M \subseteq g'^M.$$

Anders als bei den Spezies werden den Genera direkt keine Mengen von Individuen zugeordnet. Statt dessen rekrutieren die Genera ihre Extensionen aus denen der Spezies. Zu diesem Zweck wird eine Relationen $<_{SG} \subseteq S \times G$ eingeführt, so daß für $s \in S$ und $g \in G$ mit $s <_{SG} g$

$$\text{ext}(s) \subseteq \text{ext}(g)$$

und

$$\text{int}(s) \Rightarrow \text{int}(g)$$

gelte. Die Spezies s wird also mit $s <_{SG} g$ vom Genus g subsumiert.

Da die Genera $g \in G$ keine eigenen Individuen haben[30], gilt für die Extensionen

$$\text{ext}(g) = \bigcup_{s <_{SG} g' \leq_{GG} g} \text{ext}(s)$$

und

$$\text{ext}_{v,t}(g) = \bigcup_{s <_{SG} g' \leq_{GG} g} \text{ext}_{v,t}(s) \quad \text{für alle } v \in V,\ t \in T$$

und damit auch wieder

$$\text{ext}_{v,t}(g) \subseteq \text{ext}(g) \quad \text{für alle } v \in V,\ t \in T.$$

Sowohl die statischen als auch die dynamischen Extensionen der Genera sind also ausschließlich von denen der Spezies bestimmt, die sie subsumieren.

Anmerkung: Eine Hierarchie von Genera basiert in der Regel auf einer bestimmten Perspektive, einer Facette der Klassifikation. Die entwicklungsgeschichtliche Abstammung von Lebewesen ist eine solche Facette; die Einteilung nach ihrem Nutzen eine andere. In der Modellierungspraxis werden die Spezies häufig durch mehrere solcher Facetten (auch Cluster genannt [Smith & Smith 1977]) generalisiert. Anstatt aber die verschiedenen Facetten zu verschmelzen, ist es sinnvoll, mehrere Genushierarchien parallel aufzubauen, die nur die Spezies miteinander teilen. Man vgl. dazu auch die statischen Partitionen in der Arbeit von Wieringa et al. [1994; 1995].

Die Spezies s in S und die Genera g in G werden per

$$N := S \cup G$$

zur Menge der *natürlichen Typen* (in Abgrenzung von den Rollentypen aus dem nächsten Kapitel) zusammengefaßt. Da $\leq_{GG}$ und $<_{SG}$ jeweils eine Mengeninklusion der Extensionen und eine Implikation der Intensionen bedingen und beide Relationen semantisch gleich interpretiert werden, lassen sie sich per

$$\leq_{NN} := \leq_{GG} \cup <_{SG} \cup \bigcup_{s \in S} \{(s,s)\}$$

zu einer Halbordnung auf der Menge der natürlichen Typen vereinen. $(N, \leq_{NN})$ ist Teil jeder Modellspezifikation und wird im folgenden auch *Generalisierungshierarchie* genannt, weil die Intensionen der Genera allgemeiner (oder

[30] Ich unterscheide umgangssprachlich zwischen „Instanzen/Individuen haben“ und „eigene Instanzen/Individuen haben“. Genera haben i. a. Individuen (nämlich die der Spezies, die sie subsumieren), aber eben keine eigenen. Vgl. dazu auch die Diskussion von Instanzsein und Elementsein in Abschnitt 2.2.4.

abstrakter) sind als die der von ihnen subsumierten Spezies. ($N, \leq_{NN}$) ist eine Subsumtionshierarchie, in der nur die Blätter eigene Instanzen haben. Genera, die keine Spezies subsumieren, bleiben abstrakt (in dem Sinne, daß sie keine Individuen haben); Spezies, die nicht von Genera subsumiert werden, werden nicht verallgemeinert. Beides kann in einem Modell vorkommen.

Anmerkung: Die Aufteilung von Typen in Genera und Spezies und die Bedingung, daß die Spezies die Blätter der Hierarchie sein müssen, ist der Biologie entlehnt[31] und mag als unnötige Komplizierung und Rückschritt gegenüber anderen Formalisierungen empfunden werden. Tatsächlich ist aber das Einhalten dieser Regel in der objektorientierten Softwaremodellierung längst guter Stil [Hürsch 1994; D'Souza & Wills 1998; Steimann 1999a; 2000a], in der Realität die Realität (wie gesagt: welches Individuum ist schon eine Abstraktion?) und nicht zuletzt für die Modellierung eine Vereinfachung, wie sich noch zeigen wird. Außerdem werden dadurch bestimmte Probleme bei der Fassung des Prinzips der Substituierbarkeit vermieden (vgl. Abschnitt 2.2.4).[32]

In der Modellierung dienen Genera im wesentlichen zwei Zwecken: Sie stellen benannte Disjunktionen von Extensionen zur Verfügung, und sie explizieren die gemeinsame Abstammung verschiedener Spezies, indem sie das Gemeinsame ihrer Intensionen hervorheben. Die Vereinigung von Extensionen ist häufig ad hoc: Sie dient z. B. der Angabe heterogener Domains bei der Deklaration von Relationen [Kent 1978] (und macht dabei die Überladung bis zu einem gewissen Grad überflüssig; s. u.). Die Explikation der Gemeinsamkeiten hingegen ist tiefgründiger: Sie gestattet die relative Definition der Intensionen verwandter Spezies, ein Mechanismus, der sowohl in der Natur als auch in der Modellierung als Vererbung bekannt ist. Die beiden Zwecke zu trennen ist eins der Ziele des nächsten Kapitels; in diesem (sowie in vielen anderen Modellierungssprachen) werden sie noch gemeinsam verfolgt. Die Wahl der Bezeichnungen Art/Spezies und Gattung/Genus legt jedoch nahe, daß ich diese der Betonung der gemeinsamen Abstammung vorbehalten möchte.

[31] Allerdings bezeichnen die Genera nur ein relativ schmales Band der biologischen Taxonomie; darüber stehen noch Familien, Ordnungen, Klassen, Stämme und Organismenreiche. In der Biologie ist man auch bemüht, die klassische Baumform (wegen der Einteilung in Organismenreiche eigentlich Waldform) der Taxonomie aufrechtzuerhalten. Dies wird von der heutigen Vorstellung von der Entwicklungsgeschichte oder Artenbildung durchaus unterstützt. Die Untersuchung genetischer Vererbung läßt jedoch auch eine Mehrfachhierarchie als sinnvoll erscheinen; es finden sich immer wieder Zusammenschlüsse vorher getrennter Arten und manchmal auch ganz neue Lebensformen, die einer aufwendigen Reorganisation der Baumstruktur erforderlich machen.

[32] zur terminologischen Verwendung der Begriffe Genus und Spezies s. a. [ISO 1990a]

2.3.3 Relationen

Eine Modellspezifikation in FREGE umfasse weiterhin eine Menge von zwei- und mehrstelligen Relationssymbolen *P*, die, nach Signaturen geordnet, üblicherweise als eine *N**-indizierte Mengenfamilie

$$\left(P_w\right)_{w \in N^*}$$

mit $P_w = \varnothing$ für $|w| < 2$ dargestellt werden.[33] Die Mengen P_w seien paarweise disjunkt zu den I_s, so daß mit

$$P := \bigcup_{w \in N^*} P_w$$

$P \cap I = \varnothing$. $w = n_1 ... n_m$ deklariert die natürlichen Typen, die die Stellen eines $p \in P_w$ besetzen, und $|w| = m$ ist die Stelligkeit der Relation. Für $p \in P_{n_1 ... n_m}$ schreibt man auch $p : n_1 ... n_m$.

So, wie jede Spezies ihre Menge von Individuen hat, so hat auch jede Relation eine Menge von Instanzen, im folgenden *Assoziationen* genannt. Eine *typenrechte Assoziation einer Relation* $p \in P$ ist ein *m*-Tupel

$$(a_1 : s_1, ..., a_m : s_m) \quad (\text{mit } a_1 \in I_{s_1}, ..., a_m \in I_{s_m}),$$

für das es ein $w = n_1 ... n_m \in N^*$ gibt mit

$$p \in P_w \quad \text{und} \quad s_1 \leq_{NN} n_1, ..., s_m \leq_{NN} n_m.$$

Alle Assoziationen einer Relation müssen typenrecht sein; die Deklaration einer Relation schränkt also Menge ihrer Assoziationen ein.

Anmerkung: Ein Szenario ist in FREGE eine Menge von Assoziationen einschließlich der in ihnen vorkommenden Individuen.

Zu jeder Relation $p \in P_{n_1 ... n_m}$ gehören wiederum

- eine Intension int(*p*), die die Eigenschaften der Assoziationen der Relation beschreibt (die Deklaration der natürlichen Typen $n_1 ... n_m$ an den Argumentstellen ist ein Teil davon),
- eine statische Extension ext(*p*), die alle Assoziationen umfaßt, die jemals unter die Relation fallen (wenn int(*p*) nur aus der Angabe einer Deklara-

[33] *P* soll an Prädikat erinnern; der Buchstabe *R* bleibt den Rollensymbolen im nächsten Kapitel vorbehalten.

tion besteht, dann ist – vorbehaltlich einer Überladung, s. u. – ext(p) = ext(n_1) × ... × ext(n_m)),

- dynamische Extensionen $\text{ext}_{v,t}(p)$, die alle Assoziationen umfassen, die zum Zeitpunkt t im Verlauf v unter die Relation fallen, und
- eine Interpretation im semantischen Sinne.

Die semantische Interpretation einer Assoziation (a_1:s_1, ..., a_m:s_m) ist das m-Tupel

$$(a_1^M, ..., a_m^M) .$$

Wie die Interpretation einer Relation beschaffen ist, kann, genau wie der Zusammenhang zwischen den In- und Extensionen von Relationen und natürlichen Typen, erst im Zusammenhang mit Überladung im nächsten Unterabschnitt geklärt werden. Es gelte aber auch für Relationen, daß

$$\text{ext}_{v,t}(p) \subseteq \text{ext}(p) \quad \text{für alle } v \in V \text{ und } t \in T$$

ist.

Anmerkung: Wesentlich für den Gebrauch von Relationen in der Modellierung ist die Einschränkung der Extensionen auf Teilmengen des kartesischen Produkts der Extensionen der beteiligten natürlichen Typen. Dazu nennt die Intension einer Relation über die Signatur hinaus i. a. weitere Bedingungen, denen die Instanzen der Relation genügen müssen. Das können allgemeine Eigenschaften wie z. B. Reflexivität, Symmetrie etc. (von homogenen zweistelligen Relationen) sein, aber auch spezielle konzeptuelle, wie z. B. die Bedingung, daß a als Vater von b auch älter sein muß als b. Wie man solche Einschränkungen formuliert, ist nicht Gegenstand dieser Arbeit.

2.3.4 Überladung

Bislang unberücksichtigt blieb der Fall, daß eine Relation $p \in P$ mehr als eine Deklaration hat. Bei der Einführung der Relationen wurde ja nur verlangt, daß sich die P_w keine Namen mit den I_s teilen. Wenn nun ein p mehrfach deklariert wird, d. h., wenn es mindestens $w_1 \neq w_2$ mit $p \in P_{w_1}$, $p \in P_{w_2}$ und $|w_1| = |w_2|$ gibt, dann nennt man p *überladen*. Ein p soll jedoch nicht mit unterschiedlichen Stelligkeiten überladen werden, und anders als z. B. in der Logik üblich, verzichte ich der einfacheren Formulierung wegen hier auch auf die Unterscheidung der Relationssymbole aufgrund ihrer Stelligkeit.

Überladungen werden in FREGE zunächst wie folgt gehandhabt. Sei

$$W_p := \{w \in N^* \mid p \in P_w\}$$

die Menge der Signaturen einer (überladenen) Relation p.[34] Jede Deklaration von p mit Signatur $w \in W_p$ trägt zur Intension von p bei, und zwar derart, daß wenn die Intensionen als logische Ausdrücke formuliert sind, diese durch eine Disjunktion verknüpft werden. Für die Extensionen gilt, da alle Assoziationen einer Relation typenrecht sein müssen und die Existenz einer Assoziation (a_1:s_1, ..., a_m:s_m) die Existenz der Individuen a_1 bis a_m voraussetzt,

$$\text{ext}(p) \subseteq \bigcup_{n_1 \ldots n_m \in W_p} \text{ext}(n_1) \times \ldots \times \text{ext}(n_m)$$

und für die dynamische Extension eines Verlaufs v zum Zeitpunkt t stets

$$\text{ext}_{v,t}(p) \subseteq \bigcup_{n_1 \ldots n_m \in W_p} \text{ext}_{v,t}(n_1) \times \ldots \times \text{ext}_{v,t}(n_m) \subseteq \bigcup_{n_1 \ldots n_m \in W_p} \text{ext}(n_1) \times \ldots \times \text{ext}(n_m).$$

D. h. insbesondere, daß eine Relation p keine Assoziationen (a_1:s_1, ..., a_m:s_m) in ihrer Extension haben kann, für die es kein $n_1 \ldots n_m \in W_p$ gibt mit $s_1 \leq_{NN} n_1$, ..., $s_m \leq_{NN} n_m$ – die Deklaration schließt also bestimmte Assoziationen definitiv aus. Weitere Angaben der Intension einer Relation schließen weitere Assoziationen aus; wenn die Intension jedoch nur aus der (überladenen) Deklaration besteht, dann ist die Teilmengenbeziehung in den obigen beiden Ungleichungen sogar eine unechte.

Die semantische Interpretation einer Relation $p \in P$, p^M, ist wiederum eine Teilmenge des kartesischen Produkts

$$\bigcup_{n_1 \ldots n_m \in W_p} n_1^M \times \ldots \times n_m^M .$$

Zu beachten ist, daß es sich trotz der Überladung einer Relationsdeklaration hier nur um ein Relationssymbol mit einer Intension, einer Extension und einer Interpretation handelt. Diese Auffassung unterscheidet sich von anderen (z. B. [Cardelli & Wegner 1985]), nach denen überladene Operatorsymbole für verschiedene Operatoren stehen. Goguen und Meseguer [1992] gehen etwas weiter und verlangen, daß in einer ordnungssortierten Algebra überladene Funktionen als auf den Schnittmengen ihrer Definitionsbereiche identisch interpretiert werden; es handelt sich aber auch bei ihnen um verschiedene Symbole. In FREGE hingegen ist eine überladene Relation genau eine Relation, nur daß diese gewissermaßen abschnittsweise definiert ist.

[34] Man spricht nur dann von einer überladenen Relationsdeklaration, wenn W_p mehr als eine Signatur enthält.

Welchen Sinn kann das Überladen von Relationen haben? Zunächst vor allem den, einen möglichst großen Teil der Intension einer Relation in deren Deklaration unterzubringen. Die zweistellige Relation *verheiratet* beispielsweise könnte man per

verheiratet: *Person*·*Person*

deklarieren. Um aber auszudrücken, daß nur Personen unterschiedlichen Geschlechts verheiratet sein können (was Teil der Intension von *verheiratet* sein soll), deklariert man statt dessen

verheiratet: *Mann*·*Frau* und *verheiratet*: *Frau*·*Mann*.

Allgemein ist es sinnvoll, eine Relation durch Überladen abschnittsweise zu spezifizieren, wenn

1. mindestens eine Stelle einer Relation mit verschiedenen Spezies besetzt werden sollen, die kein gemeinsames (und alle anderen Spezies ausschließendes) Genus haben, wenn
2. mehrere Stellen einer Relation zwar jeweils für sich sinnvoll mit Genera besetzt werden können, sich (wie im Fall von *verheiratet*) daraus aber Kombinationen ergeben, die nicht erwünscht sind, oder wenn
3. die Intension einer Relation gezielt abschnittsweise definiert werden soll, z. B. weil eine Operation, die durch die Relation repräsentiert wird, in Abhängigkeit von den Typen ihrer Parameter (den Stellen der Relation) unterschiedlich definiert ist.

Der erste Fall und zum Teil auch der zweite wird im nächsten Kapitel mit Rollen und ohne Überladung abgedeckt; für den dritten gibt es keine Alternative zur Überladung.

Anmerkung: Grundsätzlich kann der Beitrag, den die Deklaration einer Relation zu deren Intension liefert, die Einschränkung auf die allgemeinsten sinnvollen Typen sein. Die Intension zu konkretisieren (und damit die Extension weiter einzuschränken) ist dann Aufgabe der restlichen Zutaten der Intension. Das hieße aber, jede Stelle einer Relation grundsätzlich mit dem kleinsten gemeinsamen Genus der beteiligten Spezies zu deklarieren und gar nicht zu überladen. Das andere Extrem wäre, die Spezies und Genera so fein abzustufen und die Relationen sooft zu überladen, daß durch die Deklaration der statische Teil der Intension vollständig abgedeckt wird. Alles dazwischen ist ein Trade-off, den der Modellierende zu leisten hat.

Es stellt sich noch die Frage, die sich aus dem dritten Punkt ergibt, nämlich ob und wie die Assoziationen einer Relation im Fall der Überladung den einzelnen Abschnitten der Deklaration zugeordnet werden können. Dies hat z. B. im Zusammenhang mit dem dynamischen Binden von Prozeduraufrufen in objektori-

entierten Programmiersprachen eine Bedeutung. Üblich (und sinnvoll) ist, eine Assoziation immer der kleinsten expliziten Deklaration zuzuordnen, unter die sie fällt, wenn diese denn eindeutig bestimmt ist. Andernfalls kann sie dann nicht zugeordnet werden; die Signatur (oder hier: die Überladung) ist nicht regulär [Meseguer & Goguen 1993].[35]

Anmerkung: In der Programmiersprache JAVA erzeugt die Deklaration überladener Methoden selbst niemals einen Fehler zur Übersetzungszeit, also insbesondere auch dann nicht, wenn die Signatur nicht regulär ist [Gosling et al. 1996, § 8.4.7]. Gleichwohl kann die Referenzierung (der Aufruf) einer Methode einen Fehler zur Übersetzungszeit hervorrufen, nämlich genau dann, wenn statisch keine spezifischste Deklaration, die zu dem Aufruf paßt, bestimmt werden kann [§ 15.11.2]. Zur Laufzeit wird im Rahmen des dynamischen Bindens nur noch überprüft, ob die zur Übersetzungszeit bestimmte Methode überschrieben wurde, also ob eine Methode eines spezielleren Typs als dem zur Übersetzungszeit bestimmten mit ansonsten gleicher Signatur zur Anwendung kommen kann [§ 15.11.4]. Ggf. theoretisch zur Anwendung kommende Überladungen der Methode werden dabei ignoriert. Daraus ergibt sich mit den folgenden Klassendefinitionen

```
class A {
        void hello (A a) {
                System.out.println("AA");
        }
        void hello (B b) {
                System.out.println("AB");
        }
}

class B extends A {
        void hello (A a) {
                System.out.println("BA");
        }
        void hello (B b) {
                System.out.println("BB");
        }
}
```

bei Ausführung von

```
        A x = new B();
        x.hello(x);
```

das unter theoretischen Gesichtspunkten unerwartete Ergebnis „BA“.[36] Man beachte die Sonderbehandlung der impliziten ersten Stelle der Methoden, *this*.

[35] Die Bedingungen für die Regularität können hier insofern abgeschwächt werden, als die Stellen einer Assoziation ausschließlich mit Individuen der Spezies, also mit Instanzen minimaler Typen besetzt sind.

[36] Diese Festlegung von JAVA ist wohl ein Ausdruck des Bemühens, eine vollständige statische Typüberprüfung zu garantieren. Sie handelt aber nur eine Art von Fehlern, nämlich ein „method ambiguous“ bzw. „method not understood“ [Hürsch 1994] zur Laufzeit, gegen einen logischen Programmfehler, der automatisch überhaupt nicht, auch nicht zur Laufzeit, gefunden werden kann.

2.4 Diskussion

Die Modellierungssprache FREGE erlaubt die Spezifikation von Modellen prinzipiell auf zwei Arten: durch die Angabe von Extensionen, und durch die Angabe von Intensionen. Speziell wenn das Modellierungsproblem eine ausgeprägte dynamische Komponente hat, ist die Auflistung aller möglichen Verläufe, von denen ja jeder wieder eine potentiell unendliche Folge von Szenarien ist, jedoch praktisch nicht möglich. Man wird daher stets zumindest teilweise auf intensionale Spezifikationen ausweichen.

Eine intensionale Modellspezifikation benennt die Bedingungen, denen die gemeinte Extension, das spezifizierte Modell, genügen muß. Die Spezifikation ist modular aufgebaut, d. h., die Intension wird nicht für das Modell als ganzes angegeben, sondern augeteilt auf dessen Elemente, das sind hier zunächst die (natürlichen) Typen und die Relationen. Die Spezifikation eines Modells setzt sich dann aus diesen Einzelintensionen zusammen.

Aus der Angabe einer intensionalen Modellspezifikationen läßt sich also in FREGE das statische und dynamische Modell in Form von Mengen von Szenarien bzw. Mengen von möglichen Folgen von Szenarien ableiten. Eine solche Spezifikationsform wird auch als *Schema* bezeichnet; sie ist in gewisser Weise selbst wieder eine Grammatik, und das spezifizierte statische bzw. dynamische Modell die dazugehörige Sprache. Genau damit steht eine Modellierungssprache aber vor demselben Problem wie jeder andere Grammatiktyp: Die spezifizierten Sprachen sind meistens zu allgemein (sie schließen unsinnige Ausdrücke ein) oder/und zu speziell (sie schließen sinnvolle aus). Wenn es sich nicht gerade um ein sehr umgrenztes Problem handelt, ist, wegen der Vielzahl der Nebenbedingungen, die die Realität bereithält, die Spezifikation eines vollständigen und korrekten Modells praktisch nicht erreichbar. (Man beachte aber, daß dies von einer formalen Spezifikation eines Problems und insbesondere von dem Programm, das es lösen soll, durchaus erwartet wird.)

Während also die intensionale Modellspezifikation inhärent schwierig ist, kann man die extensionale als einen konstruktiven Akt auffassen. Sei *M* der Grund-

bereich der Interpretation. Ein Modell wählt daraus einen Teil $I^M \subseteq M$ aus, z. B. $\{a_1^M, a_2^M, a_3^M, a_4^M\}$ durch Festlegung von

$$I = \{a_1, a_2, a_3, a_4\}.$$

Darüber hinaus wählt ein Modell aus

$$2^{I^M},$$

der Menge der Teilmengen von I^M, eine Teilmenge N^M, z. B.

$$N^M = \{n_1^M, n_2^M, n_3^M\}$$

mit

$$n_1^M = \{a_1^M, a_2^M\},\ n_2^M = \{a_3^M, a_4^M\} \text{ und } n_3^M = \{a_1^M, a_2^M, a_3^M, a_4^M\}$$

aus und deklariert diese, im gegebenen Fall mit $N = \{n_1, n_2, n_3\}$. n_1, n_2 und n_3 sind dann die Typen des Modells und ext(n_1) = {a_1, a_2}, ext(n_2) = {a_3, a_4} und ext(n_3) = {a_1, a_2, a_3, a_4} die dazugehörigen Extensionen. Aus den Überlappungen der ext(n_i) bzw. der n_i^M ergibt sich dann die Typhierarchie, hier $n_1 \leq_{NN} n_3$ und $n_2 \leq_{NN} n_3$. n_1 und n_2 sind die Spezies, n_3 ist ihr gemeinsames Genus.

2.4.1 Vergleich mit objektorientierter Modellierung

Die Basis der (statischen) objektorientierten Modellierung ist die Angabe eines Schemas, das meistens in einer dem Entity-Relationship-Diagramm ähnlichen Notation spezifiziert wird [Wieringa 1998] und in dem, ganz wie in FREGE, Typen über Relationen in Beziehung gesetzt werden. In der objektorientierten Modellierung unterscheidet man bei den Typen in der Regel jedoch nicht zwischen Spezies und Genera, sondern allenfalls zwischen Typen und Klassen, wobei letztere zumeist als die Implementationen von Typen angesehen und somit für die Modellierung nicht interessant sind. Die Typhierarchie eines objektorientierten Modells entspricht also der Hierarchie der natürlichen Typen (der Generalisierungshierarchie) von FREGE, wobei die Bedingung, daß Genera keine eigenen Instanzen haben dürfen, in der objektorientierten Modellierung weitgehend ignoriert wird (aber: [Hürsch 1994; Steimann 2000a]). Die Relationen eines objektorientierten Modells entsprechen den Relationen von FREGE und die Instanzen der Relationen den Assoziationen.

Die Einschränkungen von Relationen, die in objektorientierten Modellen zum Teil in Form von Kardinalitäten vorliegen, gehen in die Intensionen der Rela-

Tabelle 2.4: Gegenüberstellung der statischen Aspekte objektorientierter Modellierung und der Modellierungssprache FREGE

objektorientierte Modellierung	Modellierungssprache FREGE
Objekte	$a \in I$
Typen	$n \in N = S \cup G$
Typhierarchie	$(N, \leq_{NN})$
Relationen	$p \in P_w$ mit $w \in N^*$ als Signatur
Instanzen einer solchen Relation	$(a_1, ..., a_n) \in \text{ext}(p)$
Kardinalitäten	$\text{int}(p)$
Attribute	Funktionen (nicht behandelt)
Vererbung (von Relationen und Attributen)	implizit durch Typsubsumtion

tionen ein, auf deren Form hier jedoch nicht weiter eingegangen wurde. Man beachte aber, daß wegen der Definition der statischen Extension die Angabe von Kardinalitäten, die ja eine Gleichzeitigkeitsbedingung sind, im statischen Modell eine andere Bedeutung hat als allgemein üblich; sie schränkt nämlich nicht alle möglichen Szenarien jeweils isoliert betrachtet ein, sondern vielmehr die Vereinigung aller, also z. B. mit wie vielen anderen ein Individuum eines bestimmten Typs im Laufe seines Lebens insgesamt in Beziehung stehen kann. Andere Ausdrucksmittel zur Beschreibung von Relationen finden in FREGE ebenfalls keine gesonderte Berücksichtigung. So muß z. B. der Umstand, daß eine Relation eine Aggregation ist, ebenfalls mit der Intension der Relation festgelegt werden. Dies ist insofern sinnvoll, als es *die* Aggregation gar nicht gibt, und vielmehr von Fall zu Fall festgelegt werden muß, was mit Aggregation genau gemeint ist.

Eine Gegenüberstellung der Ausdrucksmittel zur Beschreibung der (statischen) Struktur eines Modells in der objektorientierten Modellierung und in FREGE ist in Tabelle 2.4 dargestellt. Ein entsprechender Vergleich der Modellierung dynamischer Aspekte wie die Sequenzen des Austauschs von Nachrichten und die dadurch erwirkten Zustandsübergänge ist wegen der schwachen Ausstattung FREGEs auf dynamischer Seite kaum sinnvoll. Es sei jedoch noch einmal darauf hingewiesen, daß die Spezifikation des dynamischen Modells, und hier insbesondere des Interaktionsmodells, in der objektorientierten Modellierungspraxis wegen der enormen Komplexität häufig nur auf der Angabe von Beispielabläufen beruht.

2.4.2 Vergleich mit den konzeptuellen Graphen Sowas

Sowa führt konzeptuelle Graphen als einen allgemeinen, auf Erkenntnissen der Psychologie, Philosophie und der Linguistik basierenden Formalismus zur Wissensrepräsentation ein [1984]. Ein konzeptueller Graph drückt etwas aus, indem er Konzepte über konzeptuelle Relationen in Verbindung setzt. Anders als ein semantisches Netz steht ein konzeptueller Graph zunächst für genau eine Aussage.

Aus bestehenden konzeptuellen Graphen können mittels sog. Formierungsregeln [S. 91] (die in etwa den Ersetzungsregeln einer Grammatik entsprechen) neue Graphen abgeleitet werden. Die beiden wichtigsten Regeln sind die der Spezialisierung und die der Verschmelzung von Graphen. Die Spezialisierung entspricht der Vererbung, die in FREGE durch die Definition der Typhierarchie als eine Subsumtionshierarchie impliziert wird und die bewirkt, daß Relationen, die für Genera deklariert wurden, auch von deren Subtypen instanziiert werden dürfen. Die Verschmelzung von Graphen dagegen ist in FREGE genau wie die Graphen selbst implizit: Graphen entstehen dadurch, daß sich Assoziationen Individuen teilen, und entsprechend können mehrere Relationen, die sich einen Typen teilen, einen Graphen bilden.

Konzeptuelle Graphen können zunächst alles mögliche Sinnvolle und Unsinnige ausdrücken. Ein *kanonischer Graph* aber ist ein konzeptueller Graph, der einen sinnvollen Sachverhalt beschreibt. Die Formierungsregeln sind nun so beschaffen, daß sich mit ihnen aus kanonischen Graphen stets wieder nur kanonische Graphen bilden lassen. Analog zum Axiomensystem einer ordnungssortierten logischen Theorie definiert Sowa einen *Kanon* als bestehend aus

1. einer Typhierarchie *T*,
2. einer Mengen von Individuenreferenten *I*,
3. einer Kompatibilitätsrelation von Individuen zu Typen und
4. einer endlichen Menge von kanonischen Graphen mit Typen aus *T* und Individuen aus *I*, der sog. kanonischen Basis.

Aus diesem Kanon läßt sich durch Anwendung der (kanonischen) Formierungsregeln eine Menge von kanonischen Graphen erzeugen [Sowa 1984, S. 96].

Wie leicht klar wird, entspricht ein Kanon im Sinne Sowas in etwa einer Modellspezifikation in FREGE: *T* entspricht der Generalisierungshierarchie $(N, \leq_{NN})$ der natürlichen Typen, *I* entspricht *I*, die Kompatibilitätsrelation der Definition der Extensionen der natürlichen Typen und die kanonische Basis den Relati-

onsdeklarationen (einschließlich der Angabe der Intensionen der Relationen). Die kanonischen Graphen schließlich entsprechen den Szenarien, die sich aus der Modellspezifikation ableiten lassen, und damit dem spezifizierten Modell.

2.4.3 Metamodellierung

Wie bereits in Abschnitt 2.1.1 dargelegt, wird bei der Definition von FREGE der (klassischen) metasprachlichen Definition der Modellierungssprache der Vorzug gegeben. Daß dies einer Betrachtung von Metaebenen der Modellierung prinzipiell nicht im Wege steht, zeigen die folgenden Überlegungen.

Die Typen $n \in N$ dienen der Strukturierung einer ansonsten ungeordneten Menge von Individuen *I*. Nun ist *N* selbst wieder eine Menge, die a priori nicht strukturiert ist.[37] Man kann daher eine Menge $\overline{N}$ einführen, deren Elemente $\overline{n} \in \overline{N}$ die Typen in *N* strukturieren, und zwar (in Analogie zur Mengenfamilie der Individuenbezeichner) durch eine Mengenfamilie

$$\left(N_{\overline{n}}\right)_{\overline{n} \in \overline{N}}$$

von Typbezeichnern, die zu jedem $\overline{n} \in \overline{N}$ ein $N_{\overline{n}}$ mit Elementen aus *N* angibt. Die $\overline{n} \in \overline{N}$ sind dann Metatypen und die $n \in N_{\overline{n}}$ sind die Instanzen des Metatyps $\overline{n}$. Das ganze läßt sich rekursiv fortsetzen, so daß die

$$\overline{\overline{n}} \in \overline{\overline{N}},$$

die der Strukturierung von $\overline{N}$ dienen, Meta-Metatypen heißen usw. Allgemein gilt, wenn man die extensionale Konstruktion eines Modells wie eingangs dieses Abschnitts beschrieben zugrunde legt:

$$M \supseteq I^M,\ 2^{I^M} \supseteq N^M,\ 2^{N^M} \supseteq \overline{N}^M,\ 2^{\overline{N}^M} \subseteq \overline{\overline{N}}^M \text{ usw.}$$

Die Reihe wird jedoch in der Regel nicht bis ins Unendliche fortgesetzt, da die Zahl der ausgewählten Konzepte von Metaebene zu Metaebene immer kleiner wird. In der Praxis ist die Führung einer Meta-Metaebene in einem Modell bereits eher selten, und darüber hinausgehende Abstraktionen sind allgemein unüblich.

[37] Zwar ergibt sich, wie oben gezeigt, aus der Teilemengenbeziehung der Extensionen eine Typhierarchie, doch die ist hier nicht gemeint.

Tabelle 2.5: Beispiel für den rekursiven Zusammenhang von Individuen, Typen und Metatypen

Ebene	Semantik	Syntax
0	$M \supseteq \{Hans^M, Fido^M, FidosLeine^M\} =: I^M$	$I = \{Hans, Fido, FidosLeine\}$
1	$Lebewesen^M := \{Hans^M, Fido^M\} \in 2^{I^M}$ $Gegenstand^M := \{FidosLeine^M\} \in 2^{I^M}$ $2^{I^M} \supseteq \{Lebewesen^M, Gegenstand^M\} =: N^M$	$N(=S) = \{Lebewesen, Gegenstand\}$ $I_{Lebewesen} = \{Hans, Fido\}$ $I_{Gegenstand} = \{FidosLeine\}$
2	$Typ^M := \{Lebewesen^M, Gegenstand^M\} \in 2^{N^M}$ $2^{N^M} \supseteq \{Typ^M\} =: \overline{N}^M$	$\overline{N} = \{Typ\}$ $N_{Typ} = \{Lebewesen, Gegenstand\}$

Das Beispiel eines kleinen Modells mit drei Individuen, zwei Typen und einem Metatyp ist in Tabelle 2.5 gegeben. Man beachte, daß obwohl sich die Zahl der möglichen Typen durch die Bildung der Potenzmenge von Ebene zu Ebene potenziert, das Modell immer schmaler wird: Das ist ein Effekt der Abstraktion, deren Ziel es ja ist, die Zahl der Konzepte zu reduzieren. So ist bereits $\overline{N}$ eine Singularität, und da die Potenzmenge einer Singularität wenig ergiebig ist, ist die Bildung weiterer Metatypen auch technisch nicht sinnvoll.

Das Beispiel aus Tabelle 2.5 wird also durch folgende Deklarationen in FREGE vollständig spezifiziert:

Hans:*Lebewesen*
Fido:*Lebewesen*
FidosLeine:*Gegenstand*
Lebewesen:*Typ*
Gegenstand:*Typ*

2.4.4 Anwendung

Die Modellierungssprache FREGE eignet sich aufgrund ihrer bisherigen konkreten Ausstattung zunächst nur, statische Strukturbeschreibungen und Szenarien zu spezifizieren. Um auch dynamische Verläufe einzubeziehen, bedarf sie einer Erweiterung. Diese kann in verschiedene Richtungen erfolgen.

Wie in Abschnitt 2.1.4 angedeutet, bieten sich vor allem endliche Automaten und verwandte Formalismen zur Ablaufspezifikation an. Daß es aber auch anders geht, konnte in [Steimann et al. 1999] gezeigt werden. Dort wurden objektorientierte Struktur- und Funktionsbeschreibungen aus einer normierten

Syntax, dem sogenannten Management Information Model [ISO 1993a; 1993b], in Ausdrücke ordnungssortierter Logik übersetzt, die sich von denen aus FREGE nur dadurch unterscheiden, daß sie neben den Individuenkonstanten auch Variablen enthalten und logische Formeln sowie Axiome zulassen, mit denen die Verhaltensbeschreibungen spezifiziert werden können. Mit Hilfe der axiomatisierten Verhaltensbeschreibungen werden dann Signale und Botschaften entlang bestehender Beziehungen zwischen Objekten ausgetauscht und so das Systemverhalten modelliert. Im konkreten Fall wurde diese Möglichkeit dazu ausgenutzt, um in einem Basisstationssubsystem eines GSM-Netzes modellbasiert Diagnose zu betreiben [Steimann et al. 1999].

Die strikte Aufteilung der natürlichen Typen eines Modells in Spezies und Genera, wobei die Spezies die Blätter der Hierarchie sind und als einzige Typen eigene Instanzen haben, führt bei direkter Umsetzung in ein objektorientiertes Programm dazu, daß alle Klassen entweder abstrakt oder final sind. Gerade dies entspricht aber guter Programmierpraxis [Hürsch 1994; D'Souza & Wills 1998; Steimann 1999a; 2000a], so daß die Modellierung mit FREGE durchaus zu auch unter praktischen Gesichtspunkten sinnvollen Softwareentwürfen führt.

Kapitel 3

Modellierung mit Rollen

Der Rollenbegriff ist ein allgemeiner und entsprechend mit Bedeutung überladen. Geprägt vom Theater wird er metaphorisch überall dort verwendet, wo ein Akteur in einem Kontext oder Zusammenspiel eine bestimmte Funktion über-, eben eine Rolle einnimmt. Die Metapher geht aber noch weiter: Dieses oder jenes „spielt eine/keine Rolle“ ist eine gängige Redewendung, die andeutet, daß auch Dinge oder Sachverhalte Rollen einnehmen können (und die jeden, der sich per Freitextsuche auf die Suche nach Arbeiten macht, die sich mit Rollen befassen, zu manch falsch positivem Treffer führt).

Auch die Modellierung macht regen Gebrauch von der Rollenmetapher. Tatsächlich steigt die Zahl der Ansätze zur objektorientierten Software-, zur konzeptuellen oder zur Datenmodellierung, die in der einen oder anderen Form auf Rollen zurückgreifen, ständig an. Dies kann man als Indiz dafür werten, daß Rollen Bestandteil einer natürlichen Sichtweise jedes Modellierungsproblems sind und deswegen, als Ergänzung von Typen und Relationen, in einer konsensfähigen Form in die Riege der etablierten Modellierungskonzepte aufgenommen werden sollten. Genau dem soll hier der Weg bereitet werden.

Dieses Kapitel beginnt mit der Erweiterung der Modellierungssprache FREGE aus dem vorigen Kapitel um eine formale Definition des Rollenbegriffs, die einfach und zugleich möglichst natürlich sein soll. Manchem Leser wird das eine oder andere daran bekannt vorkommen; ich verzichte jedoch zunächst (in den Abschnitten 3.1 und 3.2) weitgehend auf Literaturverweise zugunsten einer ausführlichen Literaturbetrachtung und Diskussion in den beiden folgenden Abschnitten. Das Neue meiner Arbeit wird dann in Kapitel 5 noch einmal zusammengefaßt.

3.1 Das Konzept der Rolle

Die Notwendigkeit der Einführung eines geeigneten Rollenkonzepts für die Modellierung ist von vielen konstatiert worden. Die beiden folgenden Gründe werden am häufigsten genannt:

1. Die Stellen einer Relation müssen, wenn ein Typ darunter mehrfach vorkommt, eindeutig identifizierbar sein. Dies geschieht am sinnvollsten über Rollennamen.
2. Bei bestimmten, in vielen Modellen vorkommenden Typen handelt es sich nicht wirklich um (natürliche) Typen, sondern um Rollen. Typische Beispiele hierfür sind *Angestellter*, *Arbeitgeber*, *Student*, *Dozent*, *Kunde*, *Lieferant* etc. Charakteristisch für diese Rollen ist, daß sie von Individuen anders als ihre natürlichen Typen dynamisch angenommen (auch mehrere gleichzeitig) und wieder abgelegt werden können.

Der zweite Punkt stellt Rollen in die Nähe der Objektmigration und läuft im wesentlichen auf eine dynamische Mehrfachklassifikation hinaus. Anders als der erste Punkt läßt er jedoch die Abhängigkeit der Rollen von Beziehungen außer acht. Die nächsten beiden Punkte konkretisieren diese Abhängigkeit:

3. Einzelne Stellen einer Relation können gelegentlich von Individuen verschiedenen Typs ausgefüllt werden. Wenn sich diese Typen nicht durch eine gemeinsame Generalisierung auf natürliche Weise vereinen lassen, ist entweder ein entsprechendes Überladen der Relation oder eine Art Ad-hoc-Disjunktion der betroffenen Typen notwendig. Die Extension einer solchen Disjunktion kann man aber auch als die Extension einer mit der Stelle verbundenen Rolle betrachten.
4. Assoziationen stehen u. a. für das Zusammen- oder Wechselwirken der beteiligten Individuen. Daß ein bestimmter Typ in einem Modell an einer bestimmten Stelle einer Relationsdeklaration auftaucht, ist Teil der Intension dieses Typs, d. h., es bestimmt einen Teil der Eigenschaften der Individuen, die Instanzen dieses Typs sind. Wenn ein Typ an verschiedenen Stellen derselben oder verschiedener Relationen auftaucht, erhält er da-

durch mehrere solcher partieller Intensionen. Jede dieser partiellen Intensionen kann man auch als die Intension einer Rolle ansehen.

In der Literatur werden Rollen zahlreiche weitere Eigenschaften zugeschrieben (insgesamt mindestens fünfzehn; eine Aufstellung findet sich in [Steimann 1999b]). Weitgehend unbestritten ist, daß Rollen als Typen spezifiziert werden, wodurch sich natürlich die Frage nach der Abgrenzung und dem Verhältnis von Rollentypen zu natürlichen Typen aufdrängt.

3.1.1 Verhältnis zu natürlichen Typen

Ein (natürlicher) Typ spezifiziert eine Menge von Individuen, die durch ihre Natur miteinander verwandt sind [Sowa 1984, S. 80]. So gehört z. B. zum Typ *Person* die Menge der Menschen, die im Rahmen des zu modellierenden Problems eine Rolle[38] spielen. Durch seine Position in der Generalisierungshierarchie eines Modells gibt ein Typ zudem an, in welchem verwandtschaftlichen Verhältnis er zu den anderen Typen des Modells steht (d. h., welche Typen er subsumiert und welche er spezialisiert). Betrachtet man beispielsweise *Lieferant* und *Kunde* beide als Typen, so wird man diese vielleicht, da zum Kundesein mehr zu gehören scheint als zum Personsein, als Spezialisierung des gemeinsamen Supertypen *Person* und damit als miteinander verwandt modellieren wollen. Nun kann aber in einem Modell ein und dieselbe Person sowohl Kunde als auch Lieferant sein, ein gewöhnlicher Sachverhalt, der sich nur ausdrücken läßt, indem man Mehrfachklassifikation zuläßt oder, wie in Abbildung 3.1 einen gemeinsamen Subtypen *Kunde&Lieferant* bildet, der den Schnitt beider Typen abdeckt.

Diese Lösung ist jedoch aus verschiedenen Gründen unbefriedigend. Zum einen kann ein Individuum vom Typ *Kunde*, wenn das im Laufe seiner Existenz not-

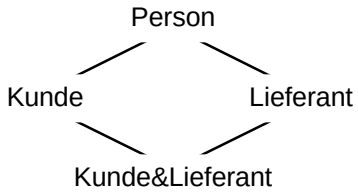

Abbildung 3.1: Modellierung von Personen, die zugleich Kunde und Lieferant sind, als gemeinsame Teilmenge beider.

[38] Schon wieder diese Metapher! Rolle ist hier aber nur umgangssprachlich zu verstehen.

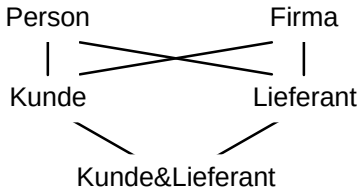

Abbildung 3.2: *Kunde* und *Lieferant* jeweils als gemeinsamer Subtyp von *Person* und *Firma*. Da *Person* und *Firma* sich aber gegenseitig ausschließen, müssen die Extensionen von *Kunde* und *Lieferant* stets leer sein.

wendig werden sollte, nur dann dynamisch zum Typ *Kunde&Lieferant* wechseln, wenn die sog. Objektmigration unterstützt wird (s. z. B. [Khoshafian & Copeland 1986; Su 1991; Mendelzon et al. 1994], oder auch die dynamische Klassen in [Wieringa et al. 1995]). Kunden, die auch Lieferanten werden können (und umgekehrt), müssen sonst von vornherein dem Typ *Kunde&Lieferant* angehören. Da dies aber prinzipiell für jeden Kunden und jeden Lieferanten der Fall sein kann, wären die beiden dazugehörigen Typen *Kunde* und *Lieferant* immer ohne eigene Individuen, also bloße Abstraktionen. Dies wäre zwar ganz im Sinne einer Abstraktionshierarchie (Abschnitt 2.2.4), entspräche aber wohl nicht der intendierten Semantik. Zum anderen ist die Unterscheidung von Personen in Kunden und Lieferanten kaum die einzig sinnvolle: In einer typischen Unternehmensmodellierung beispielsweise ist die nach Arbeitgeber, Angestellter und Freiberufler eine weitere, und da sich diese Subtypen fast beliebig miteinander kombinieren lassen, wächst die Zahl der notwendigen Schnittklassen kombinatorisch [Odell 1992; Wieringa et al. 1995].

Aber selbst wenn man die dynamische Mehrfachklassifikation als gegeben voraussetzt, ergibt sich eine Komplikation daraus, daß nicht nur Personen, sondern auch Firmen als Kunden und Lieferanten auftreten. Anders als Lieferanten und Kunden sind nämlich Firmen und Personen stets disjunkt (und, sieht man einmal von einer Verallgemeinerung im juristischen Sinne ab[39], nicht auf natürliche Weise miteinander verwandt); es gibt also kein Individuum, das sowohl Person als auch Firma ist.[40] *Kunde* bzw. *Lieferant* kann daher gar nicht gemeinsamer Subtyp von *Person* und *Firma* sein, so wie in Abbildung 3.2 angedeutet.[41]

[39] der Typ *Partei* (*Party*) [Fowler 1997a]

[40] Selbst wenn es so etwas wie Einpersonenfirmen geben sollte, wären das sicher nicht die einzigen, die Kunden und/oder Lieferanten sein können.

[41] Daß die vermeintliche Lösung, *Partei* als Verallgemeinerung von *Person* und *Firma* einzuführen und *Kunde* und *Lieferant* davon abzuleiten, auch keine ist, wird in Abschnitt 3.4.2 diskutiert.

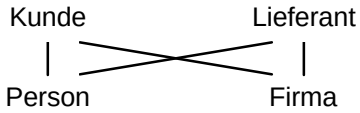

Abbildung 3.3: *Person* und *Firma* jeweils als gemeinsamer Subtyp von *Kunde* und *Lieferant*. Man beachte die zu Abbildung 3.2 inverse Anordnung.

Vielmehr gilt für die statischen Extensionen:

- Die Menge der (potentiellen) Kunden umfaßt alle Personen und Firmen, also

  $$\text{ext}(\textit{Kunde}) \supseteq \text{ext}(\textit{Person}) \cup \text{ext}(\textit{Firma}),$$

  und
- die Menge der (potentiellen) Lieferanten umfaßt alle Personen und Firmen, also

  $$\text{ext}(\textit{Lieferant}) \supseteq \text{ext}(\textit{Person}) \cup \text{ext}(\textit{Firma}).$$

Bei Berücksichtigung der Semantik der Subsumtionshierarchie aus Abschnitt 2.2.4 ergibt sich damit zunächst eine Typhierarchie, bei der *Kunde* und *Lieferant* Supertypen von *Person* und *Firma* sind, die jedoch der gängigen Vorstellung völlig zuwiderläuft (Abbildung 3.3).

Der scheinbare Widerspruch zur Intuition ergibt sich aus dem Unterschied von Statik und Dynamik eines Modells: Während prinzipiell alle Personen und alle Firmen als Kunden und Lieferanten in Frage kommen[42], werden doch in der Realität zu einem Zeitpunkt stets nur ein Teil der Personen bzw. Firmen als Kunden bzw. Lieferanten auftreten, so daß man letztere als Teilmengen der ersteren betrachten und als Subtypen darstellen möchte. Solange es jedoch kein eindeutiges, zeitinvariantes Merkmal (wie z. B. die Geschäftsfähigkeit) im Modell gibt, das Kunden und Lieferanten als Einschränkung (und damit als Subtyp) von Personen beschreibt, ist eine Darstellung beider als statische Restriktion nicht möglich. Zudem ist der Eindruck, daß *Kunde* und *Lieferant* spezieller sind als *Person* oder *Firma*, schlicht falsch: Ein Kunde beispielsweise hat kein Geburtsdatum, wenn er keine Person ist (und wird, da eine Migration von *Firma* zu *Person* nicht denkbar ist, auch nie eins bekommen). Vielmehr erben sowohl Personen als auch Firmen Attribute wie eine Lieferanschrift von *Kunde*, wenn sie als solche auftreten, und das sind ja wie gesagt zumindest möglicherweise alle. Die Typhierarchie aus Abbildung 3.3 ist also keine inverse Hierar-

[42] oder die Individuen der Typen sind nicht alle gleich und sollten daher auch nicht vom selben Typ sein [Steimann 1999b, 2001]

chie im Sinne von Abschnitt 2.2.4, sondern zumindest statisch eine echte Subsumtionshierarchie. Dennoch hat sie, da die Menge der Kunden und Lieferanten dynamisch eine Teilmenge der der Personen und Firmen ist, einen faden Beigeschmack.

Die Lösung kommt mit der Einführung von Rollen. Wie die umgangssprachliche Beschreibung des Problems bereits nahelegt, handelt es sich bei *Kunde* und *Lieferant* um eine andere Art von Typen als bei *Person* und *Firma*: Personen bzw. Firmen treten als Kunden und Lieferanten auf, beide Typen, *Kunde* und *Lieferant*, haben keine eigenen Individuen und eine Ableitung aufgrund ihrer (genetischen) Abstammung ist schon gar nicht gegeben.

Kunde und *Lieferant* sind vielmehr Rollen, die von den Individuen der Typen *Person* und *Firma* gespielt werden können. Solange nichts anderes spezifiziert ist, kommen alle Personen und Firmen als Rollenspieler in Frage – nur wenn z. B. *Person* weiter in die Typen *geschäftsfähige Person* und *nicht geschäftsfähige Person* unterteilt würde, wäre es möglich, die Rollen auf einen Teil der Personen einzuschränken, und zwar indem man bestimmt, daß sie nur von geschäftsfähigen Personen gespielt werden können. Man beachte jedoch, daß dann wieder das Problem der dynamischen Klassifikation (Wechsel zwischen *geschäftsfähige Person* und *nicht geschäftsfähige Person*) auftritt.[43]

3.1.2 Abhängigkeit von Relationen

Der dynamische Charakter von Rollen, den sie im Gegensatz zu natürlichen Typen fraglos haben, hängt mit der zweiten Eigenheit des Rollenbegriffs zusammen: der Abhängigkeit von Relationen. Eine Kunde ist kein Kunde, wenn er nicht (von einem anderen) etwas bezieht; ein Lieferant ist solange kein Lieferant, solange er nicht mit einem anderen in Beziehung steht, dem er etwas liefert. Zur Intension einer Rolle wie *Kunde* oder *Lieferant* scheint es also zu gehören, daß sie in bestimmten Relationen bestimmte Positionen besetzt, was man von natürlichen Typen wie *Person* nicht sagen kann. Das dynamische des Rollenbegriffs entspringt nun aber dem dynamischen des Relationsbegriffs: Ein Individuum spielt eine Rolle genau so lang, wie es in dieser Rolle mit einem ande-

[43] Außerdem sind *geschäftsfähige Person* und *nicht geschäftsfähige Person* wohl kaum als zwei verschiedene Spezies eines Genus *Person* aufzufassen. Eine solche Differenzierung wäre künstlich und bliebe sicher nicht die einzige. Statt dessen findet hier einer Unterteilung aufgrund von Attributwerten statt; mehr dazu wiederum in der Diskussion in Abschnitt 3.4.3.

ren Individuum in Beziehung steht. Die Extensionen von Rollen wachsen und schrumpfen im Gleichschritt mit denen von Relationen.

Der Einfluß, den Relationen auf die Definition von Rollen haben, geht aber noch weiter: Wie insbesondere die Darstellung des Rollenbegriffs in der Einleitung (Abschnitt 1.2.2) nahelegt, wird das Verhalten eines Individuums wesentlich von den Beziehungen, in denen es zu anderen steht, mitbestimmt. Die Spezifikation dieses Verhaltens ist aber, weil die Rollenspieler bis zu einem gewissen Grad austauschbar sind, primär an die Rolle, in der das Individuum die Beziehung eingeht, und erst sekundär an seinen natürlichen Typ gebunden. Diese Beobachtung spricht dafür, daß Rollen zumindest einen Teil der Intension von natürlichen Typen spezifizieren, nämlich den Teil, der die Interaktion mit anderen ausmacht.

3.2 Eine universelle Rollendefinition

Gemäß der Zielsetzung dieser Arbeit soll eine Rollendefinition gefunden werden, die so grundlegend ist, daß sie universell eingesetzt werden und damit als Basis für eine Standardisierung des Rollenbegriffs in der Modellierung dienen kann. Eine solche Definition soll nun geliefert und in die Modellierungssprache des vorigen Kapitel eingebettet werden. Zur klaren Abgrenzung der rollenlosen von der Modellierung mit Rollen werde ich die Erweiterung LODWICK nennen, weil Lodwick den Rollenbegriff schon 1647 so treffend herausgearbeitet hat.

3.2.1 Rollen

Neben der Menge der natürlichen Typen N (bestehend aus den Spezies S und den Genera G) und der Menge der Relationen P umfasse eine Modellspezifikation in LODWICK eine zu beiden disjunkte Menge von Rollensymbolen R. Die Rollen haben, genau wie natürliche Typen, Intension, Extension und Denotation. Die Extension einer Rolle sei, wie die eines natürlichen Typs, eine Menge von Individuen und die Denotation eine Teilmenge des Grundbereichs der Interpretation, M. Die Intension einer Rolle ist ähnlich der eines Genus eine partielle Spezifikation der Individuen, die darunter fallen; sie soll aber noch mehr beinhalten.

Auf den Rollen $r \in R$ sei mit $\leq_{RR}$ wiederum eine Halbordnung definiert, so daß $(R, \leq_{RR})$ eine *Rollenhierarchie* ist. Von $r \leq_{RR} r'$ heiße r *Unterrolle* von r' und r' *Oberrolle* von r. $(R, \leq_{RR})$ sei eine Subsumtionshierarchie, d. h., für $r \leq_{RR} r'$ gelte in Analogie zur Generalisierungshierarchie der natürlichen Typen

$$\text{int}(r) \Rightarrow \text{int}(r'),$$
$$\text{ext}(r) \subseteq \text{ext}(r')$$

und

$$r^M \subseteq r'^M.$$

Rollen haben keine eigenen Instanzen; sie sind wie Genera abstrakte Typen, aber eben keine natürlichen.[44]

Natürliche Typen und Rollen werden durch eine zweistellige Relation $<_{NR} \subseteq N \times R$ in Beziehung gesetzt. Diese Relation unterliegt keinerlei Restriktionen, d. h., die $n \in N$ und $r \in R$ können in beliebigen Kombinationen darin vorkommen. Die Elemente von $<_{NR}$ sind Teil der Modellspezifikation und geben an, aus welchen Typen Rollen ihre Individuen rekrutieren oder, im Rollenjargon, die Individuen welcher Spezies welche *Rollen spielen* können. $<_{NR}$ sei über $\leq_{NN}$ und $\leq_{RR}$ auf $N \times R$ transitiv fortgesetzt, so daß für alle $n <_{NN} n'$, $n' <_{NR} r'$ und $r' \leq_{RR} r$ auch $n <_{NR} r$ gelte. Ich werde für $n <_{NR} r$ auch sagen, daß n die Rolle r *füllt – spielen* ist also die umgangssprachliche Bezeichnung der (Meta-)Relation zwischen Individuum und Rolle und *füllen* die zwischen Typ und Rolle. $<_{NR}$ heißt daher auch im folgenden die *Rollenfüllerrelation*.

Genau wie die Genera $g \in G$ haben die Rollen $r \in R$ keine eigenen Instanzen, sondern rekrutieren diese aus den natürlichen Typen (und damit schließlich aus den Spezies), die die Rollen füllen. Anders als bei den Genera sind aber für die statischen Extensionen der Rollen $r \in R$ mit

$$\overline{\text{ext}}(r) = \bigcup_{n <_{NR} r} \text{ext}(n)$$

bzw. für die dynamischen

$$\overline{\text{ext}}_{v,t}(r) = \bigcup_{n <_{NR} r} \text{ext}_{v,t}(n) \quad \text{für alle } v \in V, t \in T$$

lediglich obere Schranken definiert. Die Extensionen der Rollen sollen nämlich nicht allein von denen der sie füllenden Spezies bestimmt werden, sondern auch von denen der Relationen, in denen sie vorkommen. Es gelte also insbesondere für $n <_{NR} r$ weder automatisch $\text{ext}(n) \subseteq \text{ext}(r)$ noch $\text{int}(n) \Rightarrow \text{int}(r)$.

Die Intension einer Rolle sei in zwei Teile geteilt, einen absoluten (d. h. von Relationen unabhängigen) Teil, int_{abs}, und einen relativen Teil int_{rel}, der ausdrückt, daß die Rolle auch über die Relationen, in denen sie vorkommt, definiert ist. Erst beide zusammen, also, wenn die Intensionen als logische Ausdrücke definiert sind, $\text{int}_{\text{abs}}(r) \wedge \text{int}_{\text{rel}}(r)$, machen die Intension einer Rolle r aus. Es gelte nun für $n <_{NR} r$ immerhin

[44] Wenn ich im folgenden abkürzend von Typen und Rollen rede, meine ich natürliche Typen bzw. Rollentypen, falls nichts anderes aus dem Kontext hervorgeht. Ausdrücklich nie bezeichnet hier Rolle eine Instanz, obwohl das in der Literatur des öfteren vorkommt.

$$\text{int}(n) \Rightarrow \text{int}_{\text{abs}}(r);$$

der absolute Teil der Intension einer Rolle bestimmt also einen Teil der Eigenschaften, die die Individuen, die darunter fallen (die die Rolle spielen können), haben müssen – er ist eine partielle Spezifikation, die auf die Typen, die die Rolle füllen, vererbt wird. Die Vererbung des relativen Teils stoppt hingegen am Übergang von Rollen zu Typen.

3.2.2 Relationen

Rollen sind aber nicht nur partielle Spezifikationen von Typen. Ihre zweite Aufgabe ist, die Typen, die sie füllen, in den Deklarationen von Relationen zu vertreten. Eine Modellspezifikation in LODWICK umfaßt also anders als eine in FREGE mit

$$\left(P_w\right)_{w \in 2^R}$$

eine 2^R-indizierte Mengenfamilie von Relationssymbolen, für die wiederum $P_w = \varnothing$ für $| w | < 2$ gelte sowie zunächst zusätzlich, daß die P_w paarweise disjunkt sind.[45] Relationen werden hier also nicht wie allgemein üblich auf Typen, sondern auf Rollen deklariert.[46] Außerdem ist die Familie nicht über Ketten, sondern über Mengen indiziert, weil in jeder Deklaration jede Rolle nur einmal vorkommen und die Reihenfolge der Stellen beliebig sein soll. Für die Deklaration einer Relation

$$p \in P_{\{r_1,\ldots,r_m\}}$$

schreibe man aber weiterhin

$$p\colon r_1 \ldots r_m.$$

Die Assoziationen einer Relation notiere man in LODWICK nach der Art sog. Feature-Strukturen ([Shieber 1986; Carpenter 1992]; s. a. Abschnitt 3.3.1), in denen die Reihenfolge der Stellen definitionsgemäß beliebig ist und die ihre Stellen über sog. Labels, die hier die Rollennamen sind, eindeutig benennen. Eine Assoziation wird demnach als

[45] Überladungen sind also zunächst ausgeschlossen.

[46] Die beiden einzigen mir bekannten Arbeiten, die das ebenso halten, sind das Entity-category-relationship-Modell von Elmasri et al. [1985] und das Entity-role-association-Schema von Chu und Zhang [1997]. Beide sind jedoch meines Wissen nicht weiter verfolgt worden und werden in Abschnitt 3.3.3 ausführlich diskutiert.

$$[r_1 \rightarrow a_1, \ldots, r_m \rightarrow a_m]$$

notiert, wobei die Reihenfolge der Rollenname/Individuum-Paare wie gesagt beliebig ist.[47]

Nur um die Semantik von Assoziationen und Relationen wie üblich mengentheoretisch definieren zu können, nehme ich zusätzlich eine (von $\leq_{RR}$ unabhängige) totale Ordnung < auf R an. Ein $[r_1 \rightarrow a_1, \ldots, r_m \rightarrow a_m]$ mit $r_1 < \ldots < r_m$ wird dann als

$$(a_1^M, \ldots, a_m^M)$$

interpretiert; entsprechend ist für ein $p: r_1 \ldots r_m$ mit $r_1 < \ldots < r_m$

$$p^M \subseteq r_1^M \times \ldots \times r_m^M .$$

Alle anderen Assoziationen und Relationen können entsprechend umgeordnet werden.

Die Extension einer Relation ist wiederum eine Menge von typenrechten Assoziationen. Dabei ist hier eine Assoziation $[r_1 \rightarrow a_1, \ldots, r_m \rightarrow a_m] \in \text{ext}(p)$ mit $p \in P_{\{r_1,\ldots,r_m\}}$ *typenrecht* dann und nur dann, wenn

$$a_1 \in \overline{\text{ext}}(r_1) \wedge \ldots \wedge a_m \in \overline{\text{ext}}(r_m) .$$

Für die statische Extension einer Relation $p \in P_{\{r_1,\ldots,r_m\}}$ gilt demnach

$$\text{ext}(p) \subseteq \left\{ [r_1 \rightarrow a_1, \ldots, r_m \rightarrow a_m] \,\middle|\, a_1 \in \overline{\text{ext}}(r_1) \wedge \ldots \wedge a_m \in \overline{\text{ext}}(r_m) \right\}$$

und für dessen dynamische Varianten

$$\text{ext}_{v,t}(p) \subseteq \left\{ [r_1 \rightarrow a_1, \ldots, r_m \rightarrow a_m] \,\middle|\, a_1 \in \overline{\text{ext}}_{v,t}(r_1) \wedge \ldots \wedge a_m \in \overline{\text{ext}}_{v,t}(r_m) \right\}$$

sowie selbstverständlich

$$\text{ext}_{v,t}(p) \subseteq \text{ext}(p) \quad \text{für alle } v \in V, t \in T.$$

Um auszudrücken, daß eine Assoziation zur Extension von p gehört, schreibe man in LODWICK

$$[r_1 \rightarrow a_1 : s_1, \ldots, r_m \rightarrow a_m : s_m] : p.$$

[47] Mengentheoretisch kann man eine Feature-Struktur als eine Menge von geordneten Paaren auffassen; die Schreibweise ist dann nicht unähnlich denen in [Chen 1976] und [Falkenberg 1976]. Mehr dazu in den Abschnitten 3.3.1 und 3.3.2. Anders als bei Feature-Strukturen üblich müssen aber in einer Assoziation alle deklarierten Stellen auch besetzt sein.

Anmerkung: Ein Rollenname ersetzt in der objektorientierten Modellierung häufig den Namen einer zweistelligen Relation, da er den Inhalt der Relation aus der Sicht des gegenüberliegenden Objekts wiedergibt [Rumbaugh et al. 1991, S. 34]. In der Tat ist es nicht immer einfach, für eine Relation einen Namen zu finden, der nicht zugleich der Name einer ihrer Rollen ist. Als Konvention bietet sich an, Relationen mit Prädikaten (Verb oder Kopula mit Prädikatsnomen) und Rollen mit Substantiven (oder substantivierten Verben) zu bezeichnen. Das führt beispielsweise zu *ist-Vater-von* als Name einer Relation mit den Rollen *Vater* und *Kind*. Die Redundanz, die in diesem Beispiel auftritt, ist rein metasprachlich, nicht immer so ausgeprägt und in Kauf zu nehmen.

Im allgemeinen wird die Extension einer Relation wieder eine echte Einschränkung des kartesischen Produkts der Extensionen der Rollen sein, die durch die Intension der Relation bestimmt ist.[48] Die Extensionen der Rollen sollen aber im Gegensatz zu denen der Genera nicht nur von denen der sie füllenden Spezies abhängen, sondern auch von denen der Relationen, in denen sie vorkommen. Die relative Intension einer Rolle und die Intensionen der Relationen, in denen sie vorkommt, bedingen sich also gegenseitig.

Wie der relative Teil der Intension einer Rolle genau auszusehen hat, hängt bis zu einem gewissen Grad davon ab, wie die Rolle definiert ist. In jedem Fall dazu gehört in der einen oder anderen Form die Bedingung, daß ein Individuum, um eine Rolle zu spielen (d. h., um zur Extension der Rolle zu gehören), mit der Rolle verbundene Beziehungen eingehen muß. Wenn es beispielsweise zum Dozenten gehört, daß er eine Lehrveranstaltung abhält, dann ist die Bedingung, daß eine entsprechende Assoziation bestehen muß, Teil der relativen Intension der Rolle *Dozent*. Dabei vererbt sich die relative Intension (genau wie die absolute) von Rolle zu Unterrolle, nicht aber von einer Rolle zu den sie füllenden Typen. So muß eine Person, um die Rolle *Professor* (als eine Unterrolle von *Dozent*) zu spielen, bei obiger Definition der Intension von *Dozent* ebenfalls eine Lehrveranstaltung abhalten; um Person zu sein, muß sie das jedoch nicht.

In LODWICK ist eine Rolle nicht an eine Relation gebunden, sondern kann in mehreren Relationen vorkommen.[49] Damit ist aber nicht automatisch klar, was genau der relative Teil der Intension einer Rolle zu beinhalten hat, also insbesondere an welchen der mit der Rolle deklarierten Relationen ein Individuum beteiligt sein muß, damit es zur Extension dieser Rolle zählt, eben diese Rolle spielt. Wie diese Beziehung im Einzelfall auszusehen hat, hängt vielmehr vom

[48] Teil der Intension einer Relation ist wiederum, weil alle ihre Assoziationen typenrecht sein müssen, ihre deklarierte Signatur.

[49] Dies steht im Gegensatz zu anderen Ansätzen, die Rollen mit Relationen in Verbindung bringen, z. B. denen von Bock & Odell [1998], Andersen [1997] und Riehle [2000]. Mehr dazu in den Abschnitten 3.3.4 und 3.4.

Begriffsinhalt der Rolle ab und ist Teil der Spezifikation der Rolle im Modell. Zwei Bedingungen müssen jedoch bei der Spezifikation des relativen Teils der Intension einer Rolle beachtet werden:

1. Sie muß verlangen, daß ein Individuum, um zur Extension der Rolle zu gehören, mindestens eine der damit verbundenen Beziehungen eingehen muß, und
2. die Extension einer Rolle muß in denen ihrer Oberrollen enthalten sein und ihre Intension muß die ihrer Oberrollen implizieren.

Die Freiheit in der Festlegung von $\text{int}_{\text{rel}}(r)$ besteht nun vor allem darin, welche mit der Rolle verbundenen Relationen (dazu gehören auch die von den Oberrollen geerbten oder die mit den Unterrollen zusätzlich dazukommenden) gleichzeitig eingegangen werden und ob manche Relationen (entsprechend der Angabe von Kardinalitäten) mehrfach ausgeprägt sein müssen, damit eine Rolle von einem Individuum als gespielt erachtet wird. Eine minimale Standarddefinition des relativen Teils von Rollenintensionen, die die obigen beiden Bedingungen erfüllt, ist mit

$$\text{int}_{\text{rel}}(r) \equiv \varphi(a) = \exists r' \leq_{RR} r \, \exists p \in P_{\{r', r_2, \ldots, r_m\}}, \{r', r_2, \ldots . r_m\} \in 2^R \, \exists a_2, \ldots, a_m \in I: \\ [r' \rightarrow a, r_2 \rightarrow a_2, \ldots, r_m \rightarrow a_m] \in \text{ext}(p)$$

gegeben. Demnach genügt es, daß ein Individuum nur an irgendeiner Relation der Rolle oder einer ihrer Unterrollen teilhat, damit das Individuum die Rolle spielt.[50]

Natürlich kann diese Bedingung verschärft werden. Eine andere, wesentlich stärkere ist mit

$$\text{int}_{\text{rel}}(r) \equiv \varphi(a) = \forall r' \geq_{RR} r \, \forall p \in P_{\{r', r_2, \ldots, r_m\}}, \{r', r_2, \ldots . r_m\} \in 2^R \, \exists a_2, \ldots, a_m \in I: \\ [r' \rightarrow a, r_2 \rightarrow a_2, \ldots, r_m \rightarrow a_m] \in \text{ext}(p)$$

gegeben, die besagt, daß ein Individuum eine Rolle genau dann spielt, wenn es alle direkt deklarierten und geerbten Relationen in jeweils mindestens einer Assoziation auch besetzt. Darüber hinaus kann man nur noch verlangen, daß ein Individuum dieselbe Rolle in derselben Relation mehrfach besetzt; dies entspricht dann der Angabe von Kardinalitäten mit einer Relation.

[50] Diese minimale Standarddefinition hat eine interessante Interpretation in der objektorientierten Programmierung. Demnach ist nämlich die dynamische Extension einer Rolle die Menge aller Instanzen, die den Variablen des die Rolle repräsentierenden Typs und seiner Subtypen aktuell zugewiesen sind. Mehr dazu in Abschnitt 4.2.

Damit wird die Extension einer Rolle von zwei Faktoren bestimmt: den Extensionen der Relationen, in denen sie vorkommt, und der Definition des relativen Teils der Intension der Rolle. Die statische Extension einer Rolle ist demnach die Menge aller Individuen, die diese Rolle gemäß den über die Deklarationen der Relationen hinausgehenden Einschränkungen, die damit verbunden sind (den Intensionen der Relationen), irgendwann einmal besetzen, und die gemäß den Vorgaben der relativen Intension diese auch spielen. Es ist

$$\text{ext}(r) \subset \overline{\text{ext}}(r)$$

dann und nur dann, wenn die Relationen, in denen *r* vorkommt, Individuen der Typen, die *r* füllen, vom Rollenspielen ausschließen.

Anmerkung: Es sind hier zwei Fälle zu unterscheiden: der *Regelfall*, in dem alle Individuen der Typen, die eine Rolle füllen, diese auch (in irgendeinem möglichen Verlauf zu irgendeinem Zeitpunkt) spielen, und der ungewöhnliche Fall, daß die (Intensionen der) Relationen, in denen die Rolle vorkommt, ein oder mehr Individuen der Typen, die die Rolle füllen, prinzipiell ausschließt. Im Regelfall ist

$$\text{ext}(r) = \overline{\text{ext}}(r)$$

und somit auch

$$\text{ext}(r) = \bigcup_{n <_{NR} r} \text{ext}(n) .$$

Für den ungewöhnlichen Fall, in dem das nicht so ist, ist zu prüfen, ob es nicht Subtypen gibt, die geeigneter wären, die Rolle zu füllen, denn sonst wären die Individuen der Typen zumindest nicht in der Hinsicht gleich, daß sie alle dieselben Rollen spielen können (vgl. Abschnitt 3.1). Die hier angeführten Definitionen decken jedoch beide Fälle gleichermaßen ab.

Interessanter als der statische ist der dynamische Fall: Während $\overline{\text{ext}}_{v,t}(r)$ alle im Verlauf *v* zum Zeitpunkt *t* existierenden Individuen vereint, die per Deklaration von $<_{NR}$ die Rolle spielen können, sind die dynamischen Extensionen $\text{ext}_{v,t}(r)$ auf die Individuen beschränkt, die zum Zeitpunkt *t* die gegebene Rolle auch tatsächlich spielen, also zu dem Zeitpunkt wie von der relativen Intension der Rolle gefordert in Assoziationen mit der Rolle in der Rolle auftreten. Für die Standarddefinition der relativen Intension einer Rolle gilt also

$$\text{ext}_{v,t}(r) = \left\{ a \in I \;\middle|\; \exists r' \leq_{RR} r \, \exists p \in P_{\{r',\ldots\}}, \{r',\ldots\} \in 2^R : [r' \rightarrow a, \ldots] \in \text{ext}_{v,t}(p) \right\} .$$

Insbesondere dynamisch ist also die Extension einer Rolle i. d. R. nur eine Teilmenge der der Typen, die sie füllen. Gerade darin unterscheidet sich die Semantik von $<_{NR}$ maßgeblich von $\leq_{NN}$ und $\leq_{RR}$, und darin kommt auch der prinzipielle Unterschied zwischen Rollen und Genera zum Ausdruck.

3.2.3 Überladung

Im Gegensatz zur Spezifikation von Relationen in FREGE müssen in LODWICK die Mengen der Relationssymbole P_w zunächst paarweise disjunkt sein – eine Relation darf also zunächst nicht überladen werden. Dies ist insofern gerechtfertigt, als mit der Einführung von Rollen der erste in Abschnitt 2.3.4 genannte Grund für das Überladen von Relationen, das Besetzen einer Stelle mit verschiedenen, ansonsten nicht verwandten (natürlichen) Typen, wegfällt – Rollen sind u. a. gerade dazu da, wenn eine Relation auf einer Stelle mit den Individuen unterschiedlicher Spezies besetzt werden kann, genau dies auszudrücken. Sie übernehmen diese Funktion von den Genera, die letzteren zuvor u. a. (und unangemessenerweise) zukam. Der zweite Grund für das Überladen, die wechselseitige Abhängigkeit der Typen an den Stellen einer Relation, behält aber seine Gültigkeit, solange es sich dabei nicht wie im Beispiel aus Abschnitt 2.3.4 lediglich um eine Permutation der Stellen handelt: *verheiratet*: *Ehemann*·*Ehefrau* mit *Mann* $<_{NR}$ *Ehemann* und *Frau* $<_{NR}$ *Ehefrau* deckt nämlich wegen der Unabhängigkeit von der Reihenfolge der Stellen beide Deklarationen ab. Für die anderen Fälle tut sich aber eine Lücke auf, die es zu füllen gilt.

Der eigentliche Grund, warum die Überladung von Relationen für die Modellierung mit Rollen im vorigen Abschnitt ausgeschlossen wurde, ist der, daß die Rollen einer Relation einen Teil der Bedeutung der Relation und damit ihrer Intension ausmachen, der über die bloße Nennung der rollenfüllenden natürlichen Typen an ihren Stellen[51] hinausgeht. Zwei Relationen mit unterschiedlichen Rollen sind also grundsätzlich genauso verschieden wie zwei Relationen mit unterschiedlicher Stellenzahl. Nun wird aber mit einer Modellspezifikation in LODWICK auf den Rollen eine Subsumtionshierarchie deklariert, so daß nicht alle Rollen „gleich verschieden“ sind. Es könnte z. B. durchaus sinnvoll sein, eine Relation, die zunächst mit allgemein gehaltenen Rollen deklariert wurde, zu überladen und dabei diese Rollen zu spezialisieren.

Nun ist es aber so, daß sich Relationsdeklarationen sowieso von Rollen auf deren Unterrollen vererben. Eine Relation mit Unterrollen zu überladen, scheint also gar nicht notwendig. Überladungen sind aber wiederum dann gerechtfertigt, wenn ausgedrückt werden soll, daß eine Relation abschnittsweise definiert ist, wobei jeder dieser Abschnitte durch eine andere Menge von Rollen aufgespannt wird. Um sicherzustellen, daß es sich semantisch auch wirklich um *eine*

[51] in der Linguistik auch als Selektionsbeschränkungen bekannt; s. Abschnitt 3.3.1

Relation handelt, ist es sinnvoll, zu verlangen, daß die Abschnitte alle Teile eines Bereichs sind, der durch eine (gedachte) allgemeinste Deklaration der Relation abgedeckt wird. Das heißt aber wiederum, daß die Rollen, die die Abschnitte aufspannen, jeweils Unterrollen der Rollen der allgemeinsten Deklaration sein müssen. Mit anderen Worten: Das Überladen einer Relation ist dann und nur dann sinnvoll, wenn es zu jeder Überladung bis auf eine eine andere gibt, die größer ist. Bezüglich der Zuordnung von Assoziationen zu den einzelnen Deklarationen gilt dann das in Abschnitt 2.3.4 für FREGE gesagte.

3.2.4 Relationshierarchie

Es gibt aber noch eine andere Möglichkeit, an die Sache heranzugehen. Relationen haben in LODWICK ja genau wie Typen und Rollen Intension, Extensionen (mit Assoziationen als Instanzen) und Interpretation. Man kann also auch auf Relationen eine Subsumtionshierarchie definieren. Sei mit $\leq_{PP}$ eine Halbordnung auf den $p \in P$ definiert, dann ist $(P, \leq_{PP})$ eine *Relationshierarchie* und für $p \leq_{PP} p'$ gelte wiederum

$$\text{int}(p) \Rightarrow \text{int}(p'),$$
$$\text{ext}(p) \subseteq \text{ext}(p')$$

und

$$p^M \subseteq p'^M.$$

Aus dem Zusammenhang der Intensionen der Relationen und der Tatsache, daß die deklarierten Rollen an den Stellen einer Relation Teil ihrer Intension sind, folgt, daß die kleinere, die Subrelation p nur Unterrollen der der größeren, der Superrelation p' deklarieren darf.[52] Es gilt also für $p \in P_{\{r_1,\ldots,r_m\}}$, $p' \in P_{\{r'_1,\ldots,r'_{m'}\}}$ mit $m \geq m'$

$$p \leq_{PP} p' \Rightarrow \{r_1,\ldots,r_m\} \leq_{RR} \{r'_1,\ldots,r'_m\}$$

wobei $\{r_1,\ldots,r_m\} \leq_{RR} \{r'_1,\ldots,r'_m\}$ genau dann gelte, wenn bis auf Umbenennung der $r_1,\ldots,r_{m'}$ jedem r'_i genau ein r_i mit $r_i \leq_{RR} r'_i$ eineindeutig zugeordnet werden kann. Wichtig ist also insbesondere, daß klar ist, welche Unterrolle welche

[52] LODWICK ist also insoweit intensional, als es hier tatsächlich um die (Rollen-) Typen, die in den Deklarationen der Relationen an deren Stellen stehen, geht; es reicht nicht aus, daß sich die Extensionen enthalten.

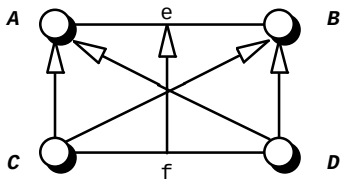

Abbildung 3.4: Relationshierarchie, die unzulässig ist, weil, da die Reihenfolge der Stellen beliebig ist, die Zuordnung der Rollen *C* und *D* zu den Rollen *A* und *B* nicht eindeutig ist. (Rollen sind hier als Kreise dargestellt, Relationen als Linien und $\leq_{RR}$ sowie $\leq_{PP}$ als Pfeile.)

Oberrolle spezialisiert. Eine Relationshierarchie wie in Abbildung 3.4 ist also nicht zulässig.

Prinzipiell ist denkbar, daß die Spezialisierung einer Relation mehr Stellen hat als die, die spezialisiert wird. Es ist dies dann so zu verstehen, daß die allgemeinere Relation eine Abstraktion ihrer Spezialisierungen ist, die u. a. das Weglassen von Stellen beinhaltet. Konzeptuell spricht das dafür, daß Relationen, die spezialisiert werden, selbst keine eigenen Assoziationen haben dürfen; die Relationshierarchie wäre also genau wie die Generalisierungshierarchie der natürlichen Typen eine Abstraktionshierarchie. Es bestehen jedoch gewisse Freiheitsgrade in der Festlegung, deren Diskussion hier zu weit führen würde.

Anmerkung: Die Deklaration einer Relationshierarchie ist oberflächlich vergleichbar mit der von Feature-Typen [Aït-Kaci & Nasr 1986; Smolka & Aït-Kaci 1989; Carpenter 1992]. Jedoch stehen Feature-Typen zunächst nicht für Beziehungen, sondern für (natürliche) Typen, und die Features eines Feature-Typs sind die Attribute, die die Eigenschaften der Individuen dieses Typs wiedergeben. Wenn trotzdem mit Feature-Typen Relationen modelliert werden, dann werden zwar die Rollen (Stellen) dieser Relationen durch die Features benannt, Rollentypen sind damit jedoch nicht definiert.

Der Unterschied zwischen Relationshierarchie und Überladung von Relationen besteht, so wie die beiden hier dargestellt werden, vor allem darin, daß es sich bei einer Relation und deren Subrelationen um verschiedene Relationen (mit verschiedenen Namen) handelt, während die Überladungen einer Relation alle zur Spezifikation derselben Relation beitragen (und denselben Namen tragen). Da aber jede Relation ihre Subrelationen subsumiert, kann man auch eine Superrelation, wenn sie selbst abstrakt ist, als abschnittsweise definiert auffassen. Das Verhältnis von Relation zu Superrelation entspricht dann dem von Spezies zu Genus: Superrelationen und Genera sind beides Abstraktionen, die eine Anzahl verschiedener konkreter Typen zusammenfassen. Trotzdem ist es fraglich, ob man Superrelationen überhaupt braucht; mehr dazu in Kapitel 4.

3.2.5 Dynamische Mehrfachklassifikation

In Abschnitt 3.2.2 wurde die Abhängigkeit von Intension und Extensionen einer Rolle von denen der Relationen, in denen die Rolle vorkommt, hervorgehoben. Aus dieser Abhängigkeit folgt aber umgekehrt ein mindestens ebenso interessanter Zusammenhang zwischen Individuen und den Rollen, die sie spielen. Im Fall der Standarddefinition der $\text{int}_{\text{rel}}(r)$ ist zu einem Individuum $a \in I$ mit

$$\left\{ r \in R \,\middle|\, \exists r' \leq_{RR} r \; \exists p \in P_{\{r',\ldots\}}, \{r',\ldots\} \in 2^R : [r' \rightarrow a, \ldots] \in \text{ext}_{v,t}(p) \right\}$$

die Menge der Rollen, die es zum Zeitpunkt *t* im Verlauf *v* spielt, gegeben. Da die Rollen $r \in R$ wie die natürlichen Typen $n \in N$ für Typen stehen, kann man sich vorstellen, daß Individuen durch die Rollen, die sie gerade spielen, dynamisch klassifiziert werden. Da jedes Individuum mehrere Rollen, auch nicht durch die $\leq_{RR}$ verwandte, gleichzeitig spielen kann, handelt es sich dabei sogar um eine dynamische Mehrfachklassifikation.[53]

Beispiel: Deklariert seien die Relation *liefern*: *Lieferant*·*Kunde*·*Ware*, die Individuen *a*:*Person*, *b*:*Person*, *c*:*Gegenstand* und *d*:*Gegenstand* sowie die Beziehungen *Person* $<_{NR}$ *Lieferant*, *Person* $<_{NR}$ *Kunde* und *Gegenstand* $<_{NR}$ *Ware* zwischen Typen und Rollen. Mit der Existenz der beiden Assoziationen [*Lieferant*→*a*, *Kunde*→*b*, *Ware*→*c*]:*liefern* und [*Lieferant*→*b*, *Kunde*→*a*, *Ware*→*d*]:*liefern*, die ein Tauschgeschäft ausdrücken sollen, sind *a* und *b* beide sowohl Lieferant als auch Kunde.

3.2.6 Polymorphie

Polymorphie ist der Wortbedeutung nach die Eigenschaft der Viel- oder Verschiedengestaltigkeit. Sie ist damit eine Eigenschaft von Objekten. Typische Beispiele für Polymorphie sind die Individuen der Ordnungen der Lepidopteren, die zuerst als Raupen und dann als Falter in Erscheinung treten, und Kohlenstoff mit seinen Vorkommen Kohle, Graphit, Diamant etc.

In der objektorientierten Programmierung wird Polymorphie häufig mit dem Binden von Prozeduraufrufen zur Laufzeit, dem sog. Late binding, gleichgesetzt. Genaugenommen handelt es sich dabei um den sog. Inclusion- oder Subtype-Polymorphismus, der nur einer von mehreren möglichen ist [Cardelli &

[53] Dabei wird hier nicht unterschieden, wie oft ein Individuum eine Rolle spielt, also ob es in der Rolle einfach oder mehrfach vorkommt. Dies mag für manche Definitionen des Rollenbegriffs von Interesse sein; für die hier angeführte ist es das jedoch nicht (vgl. dazu die Diskussion von Rollen als beigeordnete Instanzen, Abschnitt 3.4.4).

Wegner 1985; Wegner 1987; Goguen & Meseguer 1992]. Die Bezeichnung Polymorphismus ist hier insofern gerechtfertigt, als die Individuen der unter einem Typ zusammengefaßten Subtypen untereinander durchaus verschiedengestaltig sein können, ein Umstand, der sich u. a. eben darin äußert, daß diese Individuen auf den gleichen Aufruf jeweils verschieden reagieren, so daß in bestimmten Situationen ein Late binding notwendig ist.

Auf der anderen Seite ist die Praxis, die Instanz eines Typs als Instanz seiner Supertypen zu betrachten, also zum Beispiel ein bestimmtes Auto als ein Transportmittel, eine Art von Abstraktion, aber kein Ausdruck von Polymorphie im eigentlichen Sinn. So steht die Variable vom Typ *Transportmittel* zwar für Instanzen verschiedener Gestalt (nämlich Autos, Fahrräder etc.), aber keine dieser Instanzen (und auch nicht die Variable) ist selbst polymorph (vielgestaltig). Die wahre Polymorphie eines Individuums besteht vielmehr darin, daß es verschiedene Rollen spielen und somit in verschiedenen Formen auftreten kann. Ein Person, die Kunde, Lieferant und vielleicht andere Dinge mehr ist, ist polymorph; dasselbe gilt für eine Firma und überhaupt für alle Individuen, solange sie nur irgendeine Rolle spielen. [Steimann 1999b]

Man beachte, daß diese Auffassung von Polymorphie in keiner Weise dem Gleichsetzen mit dem Late binding oder der Definition als Inclusion-Polymorphismus entgegensteht: Im Regelfall umfaßt ein Rollentyp all die natürlichen Typen, die ihn füllen, und überall dort, wo eine Instanz eines Rollentyps gefordert ist, können Instanzen dieser Typen erscheinen (Substituierbarkeit). Anders als bei der herkömmlichen Form des Subtype-Polymorphismus, der auf der Generalisierungs- und damit auf einer Abstraktionshierarchie beruht, sind die Individuen, um die es hier geht, jedoch selbst wirklich polymorph, weil sie ja jeweils in der geforderten Rolle, also in der mit der Rolle verbundenen Form, auftreten. Dabei ist Polymorphie kein exklusives Begleitphänomen von Rollen: Raupe und Falter sind keine Rollen, sondern Zustände (s. Abschnitt 3.4.3); gleichwohl sind die dazugehörigen Individuen polymorph.

Wenn Polymorphie also die Eigenschaft ist, die besagt, daß ein Individuum verschiedene Formen annehmen kann, dann entspricht sie Guarinos Eigenschaft der mangelnden Rigidität (Abschnitt 1.2.2) und ist damit eine Eigenschaft von Rollen und Zuständen (bzw. zustandsdefinierten Subtypen, wenn es die denn gibt; vgl. Abschnitt 3.4.3). Genera hingegen bzw. Generalisierungshierarchien haben nichts mit dieser Art von Polymorphie zu tun, da ja die natürlichen Typen, für die sie stehen, rigide Konzepte sind. Schließlich kann man die rollenbezogene Polymorphie und den Subtype-Polymorphismus der objektorientierten

Programmierung als zwei Varianten eines allgemeinen Polymorphiebegriffs der Modellierung auffassen [Steimann 1999b].

3.2.7 Beitrag zur Modellierung

Was aber bringt nun die Einführung eines Rollenkonzepts wie in LODWICK gegenüber einer rollenlosen Modellierung wie in FREGE? Zuallererst, daß damit bestimmte Modellierungsprobleme wie z. B. die aus Abschnitt 3.1 überhaupt adäquat ausgedrückt werden können. Rollen sind eben keine Sub- oder Supertypen natürlicher Typen, und eine undifferenzierte Verquickung von Rollen und natürlichen Typen in einer Subsumtionshierarchie führt beinahe zwangsläufig zu einer gewissen Beliebigkeit des Entwurfs und zu Verwirrung beim Betrachter. Erst die klare syntaktische und semantische Abgrenzung von Typen und Rollen, wie sie in LODWICK festgeschrieben ist, kann dem abhelfen. Dabei ist hilfreich, daß die Rollendefinition LODWICKs dem Modellierer zumindest ansatzweise an die Hand gibt, wann eine Rolle zu verwenden ist und wann ein Typ, indem Rollen als obligatorische Bindeglieder zwischen natürlichen Typen und Relationen vorgesehen werden.

Rollen sind in LODWICK genau wie Genera partielle Spezifikationen von Typen, die ihre Instanzen aus Spezies rekrutieren müssen. Die Unterscheidung zwischen beiden basiert auf folgenden Punkten:

- Genera sind Abstraktionen und damit teilweise oder unvollständige Beschreibungen der unter ihnen zusammengefaßten Spezies. Ihr Wesen ist es, die Differenzierung ihrer Subtypen zu vernachlässigen und die gemeinsame Abstammung hervorzuheben. Dabei ist es einem Genus möglich, die gemeinsamen Eigenschaften seiner Individuen nicht nur zu deklarieren, sondern auch zu spezifizieren, in der Praxis, indem es Teile der Implementierung vorgibt. Die spezifizierten Eigenschaften eines Genus vererben sich zusammen mit deren Implementierung über seine Subtypen bis zu den Spezies, von denen das Genus abstrahiert.
- Rollen sind vollständige Beschreibungen eines Teils oder Aspekts der Typen, die die Rollen füllen. Sie vereinen die speziellen Eigenschaften, die in einem bestimmten Zusammenhang von den Individuen dieser Typen erwartet werden, so daß die Individuen im Kontext der Rolle gegeneinander austauschbar sind. Da diese Individuen jedoch verschiedener Abstammung sein können, beschränkt sich die Spezifikation darauf, festzuhalten, welche Eigenschaften das sind. Wie diese umgesetzt werden,

kann die Rolle nicht vorgeben; es wird somit zwar die Deklaration, aber keine Implementierung der Eigenschaften von Rollen auf die natürlichen Typen, die sie füllen, vererbt.

- Während die Individuen qua Existenz unter ihre natürlichen Typen (Spezies und Genera) fallen, müssen sie, um eine Rolle zu spielen, in dieser Rolle eine Beziehung eingehen. Genera sind selbständige Modellierungskonzepte, die die gemeinsame Abstammung von Individuen repräsentieren; Rollen haben nur im Kontext von Relationen einen Sinn und sind damit unselbständig.
- Damit einher gehen in LODWICK zwei unterschiedliche Arten der Klassifikation, denen in FREGE nur eine gegenübersteht: eine permanente, unabhängige, die auf der Typzugehörigkeit basiert, und eine temporäre, abhängige, die auf dem Rollenspielen basiert.[54] Beide Arten der Klassifikation sind über die Extensionen von Typen definiert, nur gehen in die Extensionen der Rollen stets auch die Extensionen der Relationen, an denen sie beteiligt sind, ein, und die sind nun mal dynamischer als die der natürlichen Typen, deren Individuen sie verbinden. Während also die Extension eines Genus immer die Extensionen aller Spezies, von denen es abstrahiert, enthält, ist nur die statische Extension einer Rolle (und das auch nur im Regelfall; s. die betreffende Anmerkung oben) eine Obermenge der Extensionen der Typen, die sie füllen. Dagegen ist die dynamische Extension einer Rolle die meiste Zeit eine echte Teilmenge derer der Typen, die sie füllen. Die Rollenfüllerrelation ist damit nicht etwa eine Subsumtionsrelation wie die Generalisierung.

Die Einführung von Rollen führt also zu einer echten Erweiterung der Möglichkeiten der Modellspezifikation. Sie erlaubt es, auf konzeptueller Ebene „genetische“ Verwandtschaft (nebst der damit einhergehenden Vererbung) und Substituierbarkeit aufgrund der Fähigkeit, dieselben Funktionen zu übernehmen, klar voneinander zu trennen. Damit wird der Modellspezifikation eine neue Dimension eröffnet, die die so oft kritisierte Überfrachtung der Typhierarchie mit Aufgaben [Armstrong & Mitchell 1994] zumindest teilweise abzustellen erlaubt.

Anmerkung: Während also das Rollenkonzept von LODWICK für die Modellspezifikation gegenüber FREGE einen Fortschritt bedeutet, der dem Modellierer einen differenzierteren Aufbau der Spezifikation erlaubt, unterscheidet sich das, was bei der Modellierung herauskommt, nämlich die spezifizierten Modelle, grundsätzlich nicht. In beiden Sprachen bein-

[54] Eine dritte Art der Klassifikation, die auf den Zuständen der Individuen basiert, wird in beiden Sprachen nicht berücksichtigt.

haltet ein Modell eine Menge von Szenarien und jedes Szenario wieder eine Menge von Assoziationen, nur daß in LODWICK bei den Assoziationen die Rollen, die die Individuen darin spielen, benannt sind. Diese Information ist aber zum einen redundant, da jede Assoziation eindeutig einer Relation zugeordnet ist und die Definition der Relation die Nennung der Rollen beinhaltet, zum andern ist die dynamische Mehrfachklassifikation, die durch die Bindung von Individuen an Rolle gegeben ist, für die Repräsentation der Realität dann ohne Belang, wenn Klassen in der Realität keine Entsprechung haben.

3.2.8 Übertragung auf die Metasprache

Die Metasprache, in der LODWICK spezifiziert ist, beruht im wesentlichen auf (den Schreibweisen) der Mengentheorie und enthält damit unter anderem, wie die Objektsprache auch, Relationen. Dies sind insbesondere die Relationen $\leq_{NN}$, $<_{NR}$ und $\leq_{RR}$ sowie die Elementbeziehung $\in$. All diese Relationen haben Rollen: $\leq_{NN}$ hat die Rollen *Subtyp* und *Supertyp*, $<_{NR}$ hat die Rollen *Füller* und *Gefüllter* und $\leq_{RR}$ hat die Rollen *Unterrolle* und *Oberrolle*. $\in$ hat die Rollen *Element* und *Gesamtheit*; wenn man will, kann man $\in$ auch als mehrfach überladen auffassen (mit entsprechenden Unterollen für Elemente von Typen, Rollen und Assoziationen etc.). Man beachte jedoch, daß, wenn man Rollen in der Metasprache einführt, daß man dann auch natürliche Typen und Relationstypen einzuführen versucht ist, so daß die Metasprache schnell zum Metamodell gerät (vgl. Abschnitt 2.1.1). Ein solches Metamodell mit Rollen ist am Ende von Abschnitt 4.1.4 dargestellt.

3.3 Andere Rollendefinitionen

Die im folgenden getroffene Aufteilung der zahlreichen Arbeiten zur Modellierung mit Rollen nach Disziplinen ist nicht immer eindeutig, und zwar schon allein deswegen nicht, weil die Disziplinen nicht klar voneinander abgrenzbar sind. So geht beispielsweise die Datenmodellierung fließend in die konzeptuelle Modellierung über und beide bilden die Grundlage für den strukturellen Teil der objektorientierten Softwaremodellierung. Wo immer die Zuordnung einer Arbeit also zweifelhaft ist, erscheint sie in der Kategorie, die dem Autor bzw. der Quelle, in der sie erschienen ist, am ehesten zugeordnet werden kann.

3.3.1 Rollen in der Wissensrepräsentation und Linguistik

Wie bereits in der Einleitung dargestellt, ist der Rollenbegriff der Sprachwissenschaft neben dem des Theaters einer der ältesten. Die Wissensrepräsentation versucht, von der Sprachwissenschaft zu abstrahieren, indem sie sich nur auf die Inhalte konzentriert; gleichwohl erbt sie gewissermaßen den Rollenbegriff der Linguistik.

Semantische oder thematische Rollen

Fillmore schreibt in seiner Arbeit über Typen lexikalischer Information [1971] den Einträgen eines Lexikons, die ihrer Bedeutung nach als Prädikate verstanden werden können, zwei besondere Eigenschaften zu, nämlich

1. die Anzahl der Argumente, die sie konzeptuell[55] erfordern, und
2. die Rollen, die die Argumente in der durch das Prädikat beschriebenen Situation spielen. [S. 370]

[55] Konzeptuell ist die von mir bevorzugte Übersetzung des englischen Adjektivs conceptual, das von englisch concept und damit der englischen Übersetzung des deutschen Wortes Begriff abgeleitet ist. Genaugenommen trifft weder konzeptuell noch konzeptionell die eigentliche Bedeutung, genauso wenig wie Konzept oder Konzeption Begriff entspricht.

Solche Einträge sind vor allem Verben. Die Verben rauben und stehlen beispielsweise haben drei konzeptuelle Argumentstellen, nämlich den Übeltäter, den Geschädigten und die Beute. Entsprechend hat kritisieren die Argumente Kritiker, den Kritisierten und das Kritisierte. Anstatt es aber nun bei der Abhängigkeit der Rollen vom konkreten Prädikat und der daraus resultierenden semantischen Differenziertheit der Rollen zu belassen, findet Fillmore, daß die Rollen der Prädikate einer natürlichen Sprache einem festen Inventar entstammen, das von einer dazugehörigen grammatikalischen Theorie bereitgehalten wird. So werden beispielsweise die Rollen Übeltäter und Kritiker von einer abstrakten Rolle Handelnder (oder Agent) subsumiert, die zugleich für alle Verben, die das erfordern, einen belebten Auslöser des mit dem Verb verbundenen Ereignisses zur Verfügung stellt. Andere abstrakte Rollen sind Objekt, Ergebnis, etc. Diese Rollen nennt Fillmore zunächst Kasus (cases) [S. 376]; später haben sich semantische, thematische oder auch θ-Rolle (in Abgrenzung von den syntaktischen Rollen oder den gewöhnlichen Kasus) dafür durchgesetzt.[56]

Fillmore verwendet die Kasusstrukturen der Wörter vor allem zur Klassifikation der Einträge eines Lexikons.[57] Sie können aber auch dazu verwendet werden, die Bedeutungen und möglichen Verwendungen eines lexikalischen Eintrags näher zu beschreiben. Es hat sich jedoch herausgestellt, daß diese abstrakten Rollen viel zu grob sind, um anhand ihrer die Bedeutung eines Satzes zu erschließen [Pustejovsky 1995, S. 6]. In den sogenannten Sinnaufzählungslexika (sense enumeration lexicons, SEL [Pustejovsky 1995]) werden dann auch die Argumentstellen eines Prädikats nicht mehr durch thematische Rollen, sondern durch sogenannte Selektionsbeschränkungen (selectional restrictions) in Form von Genera semantisch charakterisiert. Diese Genera sind natürliche Typen wie in FREGE (also insbesondere keine Rollen) und entstammen einer beliebig fein differenzierten Generalisierungshierarchie. Natürlich können die Selektionsbeschränkungen der Argumentstellen mit semantischen Rollen in Verbindung gebracht werden, nur füllen eben die aufgeführten Genera die Rolle lediglich im Kontext des genannten Prädikats – bei anderen Prädikaten können dieselben Rollen andere Selektionsbeschränkungen haben.

[56] Quasi nebenbei bemerkt Fillmore, daß ein Argument mehrere Rolle eines Prädikats füllen kann. Das Verb bewegen kann sowohl transitiv als auch intransitiv (reflexiv) verwendet werden; in letzterem Fall sind Handelnder und Objekt der Handlung identisch.

[57] Im Fall der Verben ist dies eine wesentlich stärkere Form der üblichen Unterscheidung zwischen transitiven und intransitiven.

Rollen in Schlagwortklassifikationen

Sog. Rollenindikatoren dienen in der schlagwortbasierten Klassifikation von Texten der Differenzierung von Begriffen anhand ihres Gebrauchs. In einem chemischen Aufsatz etwa kann ein bestimmtes Material selbst Gegenstand oder nur Mittel der Untersuchung sein, und je nachdem, in welcher Funktion man sich dafür interessiert, wird man es bei seiner Recherche mit der einen oder anderen Rolle qualifizieren wollen. Die Treffergenauigkeit kann auf diese Weise erheblich gesteigert werden. [Aitchison et al. 1997]

Feature-Strukturen

Feature-Strukturen (feature structures) oder auch Feature-Terme, die vor allem in der Computerlinguistik verwendet werden, sind im wesentlichen rekursive Mengen von Label/Value-Paaren (rekursiv deswegen, weil ein Wert selbst wieder eine Feature-Struktur sein kann). Sie sind damit verbundartige Strukturen wie der PASCAL-Datentyp Record, mit dem Unterschied, daß die Features einer Struktur anders als die Felder eines Records zunächst nicht festgelegt sind und die Reihenfolge der Felder stets beliebig ist. Der Wert nicht aufgeführter Features kann sowohl als unbekannt als auch als nicht anwendbar unterstellt werden. [Shieber 1986; Carpenter 1992]

Eine typische Feature-Struktur aus dem Bereich der Computerlinguistik ist die folgende

$$\begin{bmatrix} Infinitv : schlafen \\ Form : \begin{bmatrix} Person : 1. \\ Numerus : Singular \\ Modus : Indikativ \\ Tempus : Präsens \\ Genus\,verbi : Aktiv \end{bmatrix} \end{bmatrix}$$

für das Wort „schläft“. Aber auch ganze Sätze lassen sich durch eine Feature-Struktur darstellen:

$$\begin{bmatrix} Subjekt : \begin{bmatrix} Substantiv : Kind \\ Artikel : das \end{bmatrix} \\ Prädikat : \begin{bmatrix} Verb : schläft \end{bmatrix} \end{bmatrix}$$

steht für „Das Kind schläft“.

Man beachte die formale Ähnlichkeit zur Schreibweise von Assoziationen in LODWICK (Abschnitt 3.2.2): Das Feature label entspricht dem Rollennamen, der Feature-Wert dem Individuum, das die Rolle spielt. Allerdings ist festzuhalten, daß es sich bei Feature-Strukturen mit *Infinitiv*, *Form*, *Person* etc. nicht um Typen handelt, sondern um bloße Namen von Features (Attributen).

Etwas anderes ist es, wenn die der Satzanalyse und damit der Strukturbeschreibung eines Satzes zugrunde liegende Grammatik eine Dependenz- oder Valenzgrammatik ist: So faßt für den Satz „Peter malt ein Bild“ die Feature-Struktur

[*Akteur*:*Peter*, *Objekt*:*Bild*]

die Dependentien (oder Valenzen) des Verbs malen zusammen, die den semantischen Rollen Fillmores, Akteur und Objekt, entsprechen.

Feature-Strukturen werden zu Feature-Typen, indem per Deklaration sowohl jeder Struktur selbst als auch jedem Wert eines Label-value-Paares ein Typ zugeordnet wird [Aït-Kaci & Nasr 1986; Smolka & Aït-Kaci 1989; Carpenter 1992]. Mit jedem Feature-Typ sind dessen Features und die Typen ihrer Werte festgelegt; sie ähneln damit in gewisser Weise den Relationen LODWICKs (vgl. dazu die entsprechende Anmerkung in Abschnitt 3.2.2).

KL-ONE

Die Sprache des Wissensrepräsentationssystems KL-ONE unterscheidet zwischen Konzepten (concepts) und Rollen (roles) als „epistemologischen Primitiven“ (epistemological primitives). Diese Primitive lassen sich über strukturbildende Operationen zu größeren Komplexen zusammensetzen. Konzepte werden durch eine Subsumtionsrelation ins Verhältnis gesetzt; die resultierende Hierarchie wird auch Taxonomie genannt. Subsumtion und Differenzierung (mittels Rollen) eines Konzeptes bestimmen, entsprechend der klassischen oder aristotelischen Klassifikationstheorie, die Bedeutung eines Konzeptes. [Brachman & Schmolze 1985]

In KL-ONE übernehmen Rollen die Funktion zweistelliger Relationen zwischen Konzepten. Die Verwendung des Terms Rolle anstelle von Relation, Prädikat oder gar Attribut läßt sich insofern rechtfertigen, als KL-ONE eine konzeptzentrierte Sichtweise annimmt, aus der betrachtet ein mit einem Konzept in Verbindung stehendes anderes Konzept aus der Sicht des ersten eine Rolle einnimmt, die durch den Rollennamen bezeichnet wird. Mit jeder Rolle ist eine

Werteinschränkung (value restriction) durch Angabe eines Konzeptes nebst Kardinalitäten verbunden.

Die Rollen eines Oberkonzeptes können in dessen Unterkonzepten eingeschränkt und differenziert werden. Dabei entspricht die Einschränkung von Rollen (RoleSet restriction) in KL-ONE in der Konsequenz der Überladung von Relationen in LODWICK: Die Werteinschränkung der eingeschränkten Rolle entspricht der Konjunktion aller Einschränkungen durch Oberkonzept und Unterkonzept. Die Rollendifferenzierung (RoleSet differentiation) dagegen erlaubt es, von einer geerbten Rolle (als Relation im mathematischen Sinne, also als eine Menge von Paaren) eine oder mehrere Unterrollen abzuleiten, die einen anderen Namen als die geerbte Rolle tragen.

Das Konzept der Rollendifferenzierung wird dadurch motiviert, daß in KL-ONE Rollen Beschreibungen von Mengen, nämlich die der potentiellen Rollenspieler, sind [Brachman & Schmolze 1985, S. 185]. Die Differenzierung erlaubt die Spezifikation von Unterrollen, die eben nur von einem Teil der von der Oberrolle spezifizierten Spieler gespielt werden können. Rollendifferenzierung unterscheidet sich von der Rolleneinschränkung dadurch, daß bei der Differenzierung mehrere verschieden Rollen aus einer geerbten hervorgehen, während bei der Einschränkung die geerbte beibehalten wird. Die Rolle *Unternehmensleitung* beispielsweise des Konzeptes *Betrieb* kann im Unterkonzept *GmbH* in die Unterrollen *Geschäftsführer* und s*tellvertretender Geschäftsführer* differenziert werden, eine sinnvolle Unterscheidung, die mit einer bloßen Einschränkung nicht möglich wäre.

Die durch die Rollen von KL-ONE implizit gegebenen Mengenbeschreibungen werden durch den Rollenbegriff LODWICKs explizit gemacht: Die Rolle *Unternehmensleitung* wird als gemeinsame Oberrolle von *Geschäftsführer* und s*tellvertretender Geschäftsführer* und somit als Vereinigung der damit verbundenen Extensionen dargestellt. Eine Differenzierung der Oberrolle ergibt sich damit automatisch aus den deklarierten Unterrollen, deren Vereinigung sie darstellt. Eine Aufteilung in zwei verschiedene (d. h. verschieden benannte) Relationen ist nicht vorgesehen, es sei denn, man betrachtet die beiden Überladungen der Relation *leiten*: *Betrieb*·*Unternehmensleitung*, *leiten*: *GmbH*×*Geschäftsführer* und *leiten*: *GmbH*×*stellvertretender Geschäftsführer*, als zwei verschiedene Relationen (vgl. die Abschnitte 3.2.3 und 3.2.4). Man beachte, daß eine Unterteilung des natürlichen Typs *Person* in Subtypen nicht notwendig wird, da er alle genannten Rollen füllt. Eine solche Unterteilung wäre aber notwendig,

wollte man die Differenzierung als überladene Relation in FREGE (oder als Einschränkung in KL-ONE) darstellen.

In KL-ONE lassen sich auf verschiedene Weise gegenseitige Abhängigkeiten zwischen den Rollen eines oder mehrerer Konzepte definieren. So kann man beispielsweise die Rolle des Kunden und des Lieferanten miteinander in Beziehung setzen, um zu beschreiben, daß im Rahmen einer Lieferung etwas zwischen den beiden getauscht wird. Diese semantische Unterscheidung geht nämlich nicht aus der Differenzierung der Rolle *Teilnehmer* eines Oberkonzeptes *Transaktion* hervor, denn die Wahl der Namen der Rollen hat höchstens für den menschlichen Betrachter, nicht jedoch für das System eine definierende Bedeutung.

Obwohl die Ausarbeitung des Rollenkonzepts in KL-ONE fraglos zu den elaboriertesten zählt, beschreibt sie doch eher allgemeine Attribute denn das, was in der Literatur gemeinhin (und auch hier) unter Rollen verstanden wird.

Reimers May-be-a-Relation

Reimer ergänzt in seinem für die Wissensrepräsentation konzipierten Ansatz die Is-a-Relation durch eine rollenbildende May-be-a-Relation [Reimer 1985; 1991]. Diese Relation verbindet eine rollenspielende Klasse mit den Klassen, die eine Rolle dieser Klasse sein können. Der Name der Relation, *may-be-a*, soll ausdrücken, daß nicht jede Instanz der Klasse automatisch alle Rollen spielt, mit denen es über diese Beziehung in Verbindung steht, sondern diese bei Bedarf annimmt.

Die Semantik der May-be-a-Relation definiert Reimer durch das folgende modallogische Axiom:

$$\forall\, n, n' \in N, i \in \text{ext}(n) : n \text{ may-be-a } n' \Rightarrow \Diamond\, i \in \text{ext}(n') \,,$$

wobei $\Diamond$ die Möglichkeit ausdrückt, N die Menge der Klassen ist und ext(.) die Extension einer Klasse bezeichnet.[58] Über die Intensionen der beteiligten Klassen wird nichts ausgesagt. Das steht im Gegensatz zu der ebenfalls verwendeten Is-a-Relation, die, wie allgemein üblich, sowohl eine Teilmengenbeziehung der Extension als auch eine Implikation der Intensionen bedingt [Reimer 1991].

[58] Der modale Ausdruck des Rollenbegriffs ist übrigens genau das, was bei anderen Autoren als der dynamische Aspekt bezeichnet wird; in LODWICK wird die Modalität in der Unterscheidung zwischen statischer und dynamischer Extension aufgelöst.

Die Relationen *may-be-a* und *is-a* stehen nun laut Reimer wie folgt im Verhältnis:

1. Wenn alle Elemente einer Klasse eine Rolle spielen (und zwar nicht nur möglicherweise, sondern notwendig), dann ist dies durch die Is-a-Beziehung auszudrücken [1991, S. 283].
2. Wenn nicht alle Elemente einer Klasse eine Rolle spielen, dann ist *s* is-a *s'* und *s'* may-be-a *s* durchaus möglich, aber nicht notwendig der Fall; so ist *Politiker* is-a *Mensch* und *Mensch* may-be-a *Politiker* intuitiv sicher zu bejahen, *Taube* is-a *Vogel* und *Vogel* may-be-a *Taube* dagegen nicht [1991, S. 108].

Zu 1.: Hier ergibt sich eine Unstetigkeit. Der Unterschied zwischen alle notwendig und nicht alle notwendig kann sehr klein sein (z. B. nur einer möglicherweise nicht), bedingt aber die Entscheidung zwischen *is-a* und *may-be-a* und damit, ob das eine Konzept eine Rolle für die Instanzen des anderen ist oder nicht und ob die Intension vererbt wird oder nicht. Um Reimers eigenes Beispiel zu verwenden: Wenn alle Informatikprofessoren die Rolle eines E-Mail-Teilnehmers einnehmen, dann soll das durch die Is-a-Beziehung ausgedrückt werden. Hat aber nur einer keine E-Mail, dann ist die May-be-a-Beziehung zu verwenden.

Zu 2.: *Vogel* may-be-a *Taube* ist zwar nicht intuitiv, aber abgesehen davon kann einen nichts an Reimers Definitionen daran hindern, dies so festzulegen, denn wenn die Extension von *Taube* nicht leer ist, gibt es trivialerweise immer eine Instanz von *Vogel*, die *Taube* ist; das folgt aus der Semantik der Is-a-Beziehung.

Genau hier offenbart sich der Nachteil einer mangelnden Abgrenzung von Rollen und natürlichen Typen. *Politiker* ist eben kein natürlicher Typ wie *Mensch*, es ist vielmehr eine Rolle, und Rollen sind, wie in Abschnitt 3.1 bereits dargelegt wurde und in 3.4.2 nochmals diskutiert werden wird, keine Subtypen. Die Is-a-Beziehung ist nur zwischen Klassen, die May-be-a-Beziehung nur zwischen einer Klasse und einer Rolle anwendbar. Eine wechselseitige Is-a/May-be-a-Beziehung ist damit prinzipiell ausgeschlossen.

Bei der Instanziierung von Rollen geht Reimer zwei alternative Wege: [1991] erlaubt er einem Individuum, Instanz sowohl seiner Konzeptklasse als auch der mit dieser Klasse als deren Rollen verbundenen Konzeptklassen zu sein (sog. Mehrfachinstanziierung) und so dem Rollenspielen Ausdruck zu geben. [1985] führt er neben der May-be-a-Relation (als Relation zwischen Klassen, parallel zur Is-a-Relation) eine weitere Relation namens *role* ein, die zunächst eine In-

stanz mit ihren Rollen verbindet und dabei der Instanziierung entspricht. Um jedoch Konflikte zwischen der gewöhnlichen und der Rolleninstanziieung zu vermeiden, werden Rollen, die ja ebenfalls nur (Konzept-)Klassen sind, regulär instanziiert und die entstandenen Instanzen über die Relation *role* mit ihren konkreten Rollenspielern verbunden. Die Role-Relation ist damit eine vielfach überladene; s. dazu auch die Diskussion in Abschnitt 3.4.4.

Rollen in den konzeptuellen Graphen Sowas

Sowa unterscheidet auf konzeptueller Ebene zwischen natürlichen Typen (natural types) und Rollentypen (role types) [Sowa 1984; 1988][59]. Während natürliche Typen den Ursprung oder das Wesen (genus) einer Instanz benennen und allein und aus sich heraus existieren, sind Rollentypen über zufällige Eigenschaften und vor allem über den Zusammenhang mit anderen Typen definiert: „A natural type refers to the essence of some entity or substance, but a role type refers to accidental properties that might have been otherwise: a cat cannot stop being a cat as long as it lives, but it is a pet only when some human being adopts it; [...]“ [1984, S. 297]. In konzeptuellen Graphen werden Rollentypen über konzeptuelle Relationen (conceptual relations) definiert; sie gehen stets mit einem kanonischen Graphen einher, der die Beziehung zu anderen Typen festschreibt [Sowa 1984, S. 227, S. 297].

Als einfaches Entscheidungskriterium zwischen natürlichen Typen und Rollen gibt Sowa das folgende an [1988]:

- Etwas ist ein natürlicher Typ, wenn es isoliert als Typ identifiziert werden kann.
- Etwas ist ein Rollentyp, wenn es nur in gemeinsamer Betrachtung mit anderen Typen als Typ identifiziert werden kann.

So sind zum Beispiel *Mensch*, *Mann* und *Frau* natürliche Typen, während *Fußgänger*, *Ehepartner* und *Student* Rollentypen sind [Sowa 1984, S. 82]. Entsprechend ist z. B. *Zahl* der natürliche Typ der Zahl 7, während *Summe* oder *Divisor* verschiedene Rollentypen derselben sein können. Eine Rolle (role) bezeichnet das Auftreten einer Instanz in einer Beziehung; die Rolle wird durch diese Beziehung, eine konzeptuelle Relation, der Instanz zugewiesen. Anders als bei anderen Autoren ist der modale oder temporale Aspekt des Rollenbegriffs bei Sowa jedoch eher sekundär – insbesondere bei Zahlen fällt es schwer, von temporären Rollen zu sprechen.

[59] Die beiden Bezeichnungen habe ich für LODWICK übernommen.

Die Hierarchie natürlicher Typen, so Sowa [1988], hat in der Regel strikte Baumform. Im Widerspruch dazu steht jedoch, daß bei ihm natürliche und Rollentypen im selben Typenverband koexistieren und ggf. gemeinsame Subtypen haben [Sowa 1984, S. 82]. So ist z. B. *Ehemann* ein Subtyp von *Mann* und *Ehepartner*. Leider bleibt die Darstellung Sowas an dieser Stelle unformal, so daß auch nicht klar wird, ob eine Instanz, die stets zumindest einen natürlichen Typ hat, auch zugleich mehrere Rollentypen haben kann, die nicht über einen gemeinsamen Unter(rollen)typ zusammengeführt werden. Dies würde dann auf eine Mehrfachklassifikation hinauslaufen.

In seinem neuesten Buch zur Wissensrepräsentation [2000] geht Sowa erneut auf das Rollenkonzept ein. Und zwar definiert er Rolle (Role) als eine der sog. Top-level-Kategorien wie folgt:

$$Role < Actuality < \{Physical, Independent\} < Entity,$$

wobei *Entity* als Synonym für *Top* ganz oben steht. Rollen befinden sich auf einer Stufe mit Erscheinung (*Phenomenon*) und Zeichen (*Sign*). Dabei steht die Erscheinung für die inhärente Form oder Struktur einer Entität[60] unabhängig von allen anderen, die Rolle für die Rolle[61] einer Entität in einer Interaktion oder Beziehung mit einem anderen und das Zeichen für die Verwendung der Entität zur Bedeutung von etwas (zweitem) für einen dritten. In prädikatenlogischen Ausdrücken finden sich laut Sowa Rollen in Form zweistelliger Prädikate wieder: *Nahrung*(*x*, *y*) beispielsweise drückt aus, daß *x* die Rolle einer Nahrung für *y* spielt. Dieser Prädikate allgemeinster Typ ist das Prädikat *haben*: Mutter hat Kind, Kind hat Mutter etc.; so lassen sich nach Sowa schließlich alle Rollen definieren [S. 505].

Formal hängen Rollen nach Sowa also an einer zweistelligen Relation, und zwar derart, daß wenn Entität *x* unter einen Rollentyp fällt, dieses *x* in Beziehung zu einer anderen Entität *y* stehen muß [S. 80]. Anders als zuvor werden Rollen also nicht mehr als Subtypen dargestellt, und zwar schon allein deswegen nicht, weil jetzt auch Sowa anerkennt, daß Instanzen vollkommen verschiedener Erscheinungstypen in derselben Rolle auftreten können. Nur gelegentlich, so Sowa, legt ein Rollentyp einen Erscheinungstyp nahe, z. B. *Pet Animal*.

[60] genauer: Aktualität (im Sinne von tatsächlich existierend)

[61] Es ist tatsächlich schwierig, den Begriff der Rolle umgangssprachlich unter Umgehung desselben zu definieren; Funktion wäre vielleicht noch die beste Alternative.

Sowa verzichtet auch weiterhin auf eine syntaktische Trennung von Rolle und Nichtrolle, was aus seiner Warte verständlich ist, denn *Role* ist ja nur eine von vielen fundamentalen Kategorien seiner Ontologie, und schließlich umfaßt die Extension von *Entity* auch die aller Rollen. Es scheint aber, als ob sich das Rollesein dominant auf Kreuzungen zwischen Rollen und Nichtrollen vererben würde, denn weitere Rollen lassen sich laut Sowa aus den reinen Subtypen von *Role* und anderen Typen ableiten. Zum Beispiel sind

$$Argument < Data \cap Role$$

[S. 503] und

$$Part < Object \cap Component,$$

Rollen, wobei *Component* Subtyp von *Role* ist [S. 505]. Dies steht im Gegensatz zur Formalisierung LODWICKs aus Abschnitt 3.1, wonach ein Subtyp eines Genus immer ein Genus oder eine Spezies ist, aber nie eine Rolle. Die zu den neu gebildeten Rollen(typen) *Argument* und *Part* gehörenden zweistelligen Relationen(typen) heißen übrigens *Arg*(*Function*, *Data*) und *Part*(*Object*, *Object*); beide sind Subtypen des Relationstyps *Has*(*Entity*, *Entity*).

Die thematischen Rollen der Linguisten baut Sowa als spezielle Rollen ein, die sich allesamt von *Participant*, das selbst (über *Component* und *PrehendedEntity*) ein Subtyp von *Role* ist, ableiten. Die Menge der thematischen Rollen ist also eine Teilmenge der der (allgemeinen) Rollen. So ist *Fahrer* thematische Rolle des Verbs *fahren*; die (allgemeinere, auch was die Extension betrifft) Rolle *Führerscheininhaber* ist im Gegensatz dazu keine thematische [S. 511].

Leider bleiben bei Sowas Versuch einer ontologischen Klärung des Rollenbegriffs erneut viele Dinge schwammig oder unklar. So ist im obigen Versuch der Definition von *Argument Role* keineswegs Subtyp von *PrehendingEntity* oder *PrehendedEntity*, obwohl alle Rollen Teil einer Bezogenheit (prehension) sein müssen, und zwar entweder als Beziehende (*PrehendingEntity*) oder als Bezogene (*PrehendedEntity*) [S. 502], und es nur diese beiden Alternativen gibt. Außerdem ist die angedeutete Bivalenz von zweistelligem Relationstyp und Rolle eines allgemein *Has*-Relationstyps [S. 86] ein permanenter Quell der Konfusion, über den sich schon viele ausgiebig Gedanken gemacht haben (s. z. B. die Diskussion in [Guarino 1992]).

Eine letzte Bemerkung betrifft nicht nur Sowas Arbeit, sondern alle Ansätze, die eine Subsumtionshierarchie der allgemeinsten Konzepte (ein sog. Upper level model) zur Grundlage ihrer Ontologie machen: Wenn man *Phenomenon*

als oberstes Genus der Entitäten auffaßt[62], dann sind *Phenomenon* und *Role* gewissermaßen die Wurzeln der (natürlichen) Typ- bzw. der Rollenhierarchie und entsprechen damit den Wurzeln der Strukturen $(N, \leq_{NN})$ und $(R, \leq_{RR})$[63] eines Modells in LODWICK. Während aber bei allen Upper-level-Ontologien von der Art Sowas *Rolle* und *Phenomenon* selbst Konzepte sind, die sogar (gemeinsame) Verallgemeinerungen haben und die in der Entwicklung der Ontologie hin zu einem Modell (über Subtypenbildung) lediglich verfeinert (spezialisiert) werden[64], sind *N* und *R* Typen der Metasprache, die zur Bildung eines Modells instanziiert werden, und die Gemeinsamkeiten von *R* und *N*, z. B. daß beides Mengen von Typen sind, Sache der Metamodellierung. Die Konsequenz dieses Ansatzes ist, daß sich Rollen und natürliche Typen in der Objektsprache syntaktisch unterscheiden; es handelt sich um verschiedene Sprachkonstrukte.

Die Rollendefinitionen von Guarino

Angelehnt an Sowas ursprünglicher Definition von Rollen als Subtypen [1984; 1988], die in einem Beziehungsmuster (pattern of relationship) auftreten, stellt Guarino seinen Rollenbegriff auf ontologisch fundierte Füße. Für ihn ist ein Konzept eine Rolle genau dann, wenn

1. kein Individuum dieses Konzeptes unabhängig von, d. h. ohne Beziehung zu einem anderen Individuum (eines anderen oder desselben Konzeptes) existieren kann, wobei die Teil-Ganzes-Beziehung explizit ausgeschlossen ist, und wenn
2. die Eigenschaft eines Individuums, Instanz des Konzeptes zu sein, nicht zu seiner Identität beiträgt, d. h., wenn die Tatsache, daß ein Individuum unter das Konzept fällt, nicht dauerhaft Bestand haben muß. [Guarino 1992]

Nach dieser Definition ist Sohn eine Rolle, da das Konzept *Sohn* nicht unabhängig vom Konzept *Eltern* ist und keine Teil-Ganzes-Beziehung zwischen den beiden besteht (wohl aber zwischen *Familie* als einem übergeordneten Ganzen und den Teilen *Sohn* bzw. *Eltern*), die Eigenschaft, Sohn zu sein, aber zugleich nicht zur Identität des Individuums beiträgt. Das Konzept *Auto* hingegen ist

[62] tatsächlich bezeichnet Sowa *Phenomenon* als nahen Verwandten Aristoteles' Kategorie Substanz [2000, S. 87]
[63] wenn diese denn eindeutig sind
[64] vgl. dazu die einleitenden Bemerkungen zur Unterscheidung von Typen und Objekten in Abschnitt 2.2.1

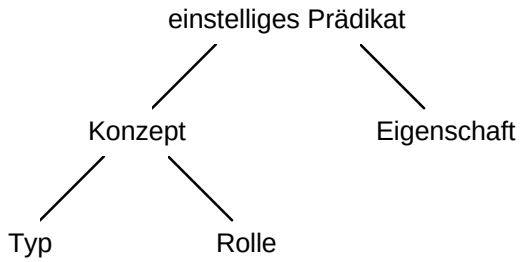

Abbildung 3.5: Vorschlag zur Aufteilung und Benennung einstelliger Prädikate für die Wissensrepräsentation nach Guarino et al. [1994]

keine Rolle, da es unabhängig von anderen Konzepten ist (Räder und Motor des Autos sind Teile und somit ausgeschlossen[65]). Eigenschaften wie Farbe, Gewicht und Geschwindigkeit hingegen, die ja auch nicht unabhängig von den Konzepten sind, die sie beschreiben, sind keine Rollen, da ihre Instanzen (wie *blau*, 180 *kg*, *schnell*) unabänderlich unter die Konzepte fallen: *Blau* kann nicht aufhören, eine Farbe zu sein.

Interessanterweise läßt sich durch Umkehrung der beiden Bedingungen auch gleich der Begriff des natürlichen Konzepts definieren: *Person* ist ein natürliches Konzept, weil seine Individuen unabhängig sind und zugleich nicht aufhören können, Person zu sein, ohne ihre Identität zu verlieren. Darüber hinaus läßt diese Klassifikationsordnung noch zwei weitere Arten von Konzepten zu: *Farbe* ist nicht nur keine Rolle, sondern auch kein natürliches Konzept, da es nicht unabhängig ist, und *Welpe* ist weder natürliches Konzept noch Rolle, da seine Individuen zwar unabhängig sind, aber nicht ewig Welpe sein werden. Gerade letzteres, nämlich die Eigenschaft von Individuen, ihr Konzept (oder ihre Klassifikation) zu wechseln, wird in der Literatur zur konzeptuellen und Datenmodellierung häufig als Rolle definiert; sie vernachlässigen aber den Beziehungsaspekt, der nach Sowa und Guarino für den Rollenbegriff maßgeblich ist (und für die Definition aus Abschnitt 3.1 schon sowieso).

In einem umfassenderen Versuch der ontologischen Grundlegung von Metakategorien der Wissensrepräsentation setzen Guarino et al. Rollen mit sogenannten nichtsubstantiellen Sortenprädikaten gleich [1994], wobei die Abgrenzung von substantiellen Sortenprädikaten anhand des Kriteriums der ontologi-

[65] Daß diese Begriffsdefinition nicht ganz unproblematisch ist, zeigen die folgenden Überlegungen. Linkes Vorderrad z. B. ist die Rolle eines Rades, obwohl es, wie Rad, Teil eines Fahrzeugs ist. Daß es dennoch eine Rolle ist, ergibt sich daraus, daß es ohne rechte und ohne Hinterräder auch kein linkes Vorderrad geben kann.

schen Rigidität, jedoch unabhängig von der Fundierung erfolgt. Die substantiellen Sortenprädikate entsprechen dann gewöhnlichen Typen; insgesamt ergibt sich die in Abbildung 3.5 dargestellte Begriffshierarchie.

Unnötigerweise folgen Guarino et al. der ursprünglichen Auffassung Sowas [1984] (und vieler anderer; vgl. die Diskussion in Abschnitt 3.4.2), nach der zu jeder Rolle ein übergeordneter Typ gehöre, dem die Rollenspieler entstammten. Daß dies den Rollenbegriff unnötig einschränkt, wurde in Abschnitt 3.1 gezeigt – es sei denn, der übergeordnete Typ ist so allgemein, daß er kaum diskriminatorischen Wert hat. Solche Typen werden aber von Guarino et al. a priori als sinnlos ausgegrenzt.

3.3.2 Rollen in der konzeptuellen Modellierung

Die konzeptuelle[66] Modellierung ist historisch aus der Datenmodellierung hervorgegangen und eine Folge der verstärkten Konzentration auf die logische Sicht einer Datenbank. Konzeptuelle Modelle basieren theoretisch zumeist in der einen oder anderen Form auf der Prädikatenlogik erster Stufe, wählen aber für die Darstellung von Datenschemata geeignetere Repräsentationsformen.

Rollen im Entity-Relationship-Modell und dessen Erweiterungen

Das Entity-Relationship-Modell [Chen 1976] basiert auf der üblichen Zweiteilung in Objekte und Relationen, Entitäten (entities) und Beziehungen (relationships) genannt. Die Rolle einer Entität in einer Beziehung bezeichnet die Funktion, die sie darin trägt [Chen 1976, S. 12]. Genau wie in anderen Ansätzen markieren Rollen im Entity-Relationship-Modell die Stellen einer Relation; ein Tupel läßt sich unter Verwendung der Rollen als Menge von Paaren $\{r_1/e_1,...,r_n/e_n\}$, also unabhängig von der Reihenfolge der Stellen darstellen. Im Gegensatz zu den Assoziationen LODWICKs (Abschnitt 3.2.2) sind die Rollen hier jedoch nur Namen; sie stehen insbesondere nicht für Typen.

Im erweiterten Entity-Relationship-Modell des DB-MAIN Projekts [Hainaut 1996; Hainaut et al. 1997] bezeichnen Rollen ebenfalls die Stellen einer Relation, sind jedoch nicht notwendig an einen Typ gebunden. Falls benötigt, können Entitätstypen ad hoc vereint werden (sog. multi-ET roles), wobei diese Vereinigungen nicht mit Generalisierungen gleichzusetzen sind; sie entsprechen vielmehr den sog. Domains von Kent [1978].

Rollen im Entity-category-relationship- und Extended-entity-relationship-Modell

Das Fehlen von Generalisierung und Spezialisierung der Entity sets im ursprünglichen Entity-Relationship-Modell hat Anlaß zu zahlreichen Erweiterungen desselben gegeben. Elmasri et al. nutzen die Gelegenheit, durch die Einführung eines zusätzlichen Modellierungskonzeptes in das Entity-Relationship-Modell einen Rollenbegriff zu integrieren, der u. a. Spezialisierung und Generalisierung abdeckt [Elmasri et al. 1985]. Ihrem Ansatz liegt die Erkenntnis zugrunde, daß die Entitäten, die in einer Relation eine Stelle (Rolle) füllen, nicht alle vom selben Typ sein müssen. Typen (entity types genannt) werden also zu sogenannten Kategorien (categories) zusammengefaßt, die durch die Rollen, die ihre Entitäten spielen können, charakterisiert werden. Da diese Kategorien eine zentrale Rolle im Modell von Elmasri et al. spielen, nennen sie es Entity-category-relationship-Modell.

Da Entity types (das Äquivalent der Entity sets im Entity-category-relationship-Modell) per Definition disjunkt sind, werden auch Generalisierung und Spezialisierung als Kategorien dargestellt. Dabei wird eine Generalisierung wie allgemein üblich als Vereinigung mehrerer Entity types und eine Spezialisierung als Einschränkung (Teilmenge) eines Entity types aufgefaßt. Allgemein gilt für eine Kategorie *C*

$$C \subseteq T_1 | P_1 \cup ... \cup T_n | P_n ,$$

wobei $T_i | P_i$ eine (optionale) Einschränkung des Typs T_i durch das Prädikat P_i ist. Eine Kategorie repräsentiert also, genau wie ein Entity type, eine Mengen von Entitäten

Anders als erwartet ist eine Generalisierung nicht als $C = T_1 \cup ... \cup T_n$, also als Sonderfall der obigen Ungleichung definiert: Eine Generalisierung muß nicht eine Vereinigung von ganzen Entity types, sondern kann auch eine Teilmenge davon sein [Elmasri et al. 1985, S. 83/84].[67] Alle Kategorien sind damit Generalisierungen und die Spezialisierung ist ein Spezialfall derselben (nämlich der, in dem nur über einen Typ generalisiert und/oder entsprechend eingeschränkt wird) – eine sicher nicht beabsichtigte Ungenauigkeit im Entity-category-relationship-Modell. Ausdrücklich als Spezialfall sieht die Entity-category-

[66] Zum Gebrauch von konzeptuell s. Fußnote 55 auf Seite 82.

[67] Entsprechendes findet man als Typkonstruktion bei Gogolla [1994, S. 7] in einer anderen Erweiterung des Entity-Relationship-Modells.

relationship-Notation vor, daß ein Entitätstyp und eine Kategorie zusammenfallen, also dieselbe Menge von Entitäten repräsentieren.

Der motivierenden Erkenntnis folgend wird eine Relation *R* im Entity-category-relationship-Modell nicht auf Entitätstypen, sondern auf Kategorien definiert:

$$R \subseteq C_1 \times ... \times C_n .$$

Dies führt zu dem gewünschten Effekt, nämlich daß Entitäten verschiedenen Typs ein und dieselbe Stelle einer Relation füllen können. Allerdings ist eine Kategorie nicht eindeutig an eine Rolle gebunden (vgl. deren Definition oben). Es ist daher durchaus möglich, daß eine Kategorie mehrere Rollen (auch innerhalb derselben Relation) besetzt. Damit aber sind, genau wie im Entity-Relationship-Modell, wiederum Rollen(namen) zur Unterscheidung der Stellen notwendig, ein Umstand, der angesichts der Tatsache, daß die Einführung der Kategorien gerade durch Rollen motiviert wurde, etwas enttäuscht.

Dem Entity-category-relationship-Modell von 1985 fehlt die (semantisch motivierte) Unterscheidung zwischen Rollenbildung und Generalisierung. Auch in späteren Arbeiten zum sog. Extended-entity-relationship-Modell, in denen Elmasri von seinem ursprünglichen Ansatz abrückt und zwischen konzeptuellen Entitäten (als Instanzen von Entitätstypen) und Rollenobjekten (als Instanzen von Rollentypen) unterscheidet, ist diese Unterscheidung vor allem technisch motiviert. So können dann auch Entitätstypen und Rollentypen gleichermaßen die Stellen einer Relation besetzen (allerdings nicht gemischt), und ob ein Objekt eine Rolle ist oder eine Entität, hängt nur davon ab, ob man es nach seiner Lebensspanne noch referenzieren können soll. Daß dieser Rollenbegriff recht weit von dem natürlichen entfernt ist, merkt man schon daran, daß bei Elmasri et al. [1991] *Person* und *Student* Entitätstypen sind und *lebende Person* und *eingeschriebener Student* dazugehörige Rollentypen.

Entity-role-association-Schema

Als eine weitere von insgesamt nur zwei Arbeiten betont die von Chu und Zhang [1997] mit dem Entity-role-association-Schema die doppelte Funktion von Rollen: einmal als Stellen einer Relation (Assoziation genannt) und einmal als Charakterisierung des Auftretens eines Objektes in dem durch die Relation gegebenen Kontext. Zu diesem Zweck führen die Autoren sog. Assoziationstypen (association classes) ein, die in ihrer Deklaration alle an einer Assoziation beteiligten Rollen aufführen.

Die Subtypenrelation wird im Entity-role-association-Schema in eine Is-a- und in eine Role-of-Variante aufgeteilt. Die Is-a-Relation bildet die klassische Typhierarchie, und die Role-of-Relation ordnet den sogenannten Rollentypen (role classes) Entitätstypen (entity classes) zu, deren Instanzen die Rolle spielen können. Die Rollentypen können selbst wieder in einer Is-a-Hierarchie angeordnet sein.

Das Überladen von Assoziationstypen, d. h. das Deklarieren von Assoziationen mit gleichem Namen und derselben Stelligkeit auf verschiedenen Unterrollentypen, impliziert im Entity-role-association-Schema eine Assoziationstyphierarchie. Assoziationen, deren Rollen mit Objekten der Typen, die die Rollen spielen, besetzt sind, werden aufgrund dieser Typen ihren Deklarationen zugeordnet, so daß Methoden und Eigenschaften einer Assoziation von den Typen ihrer Rollenspieler abhängen.

Obwohl dies gar nicht nötig gewesen wäre, verfallen Chu und Zhang in ihrem Ansatz der in den implementierungsnahen Arbeiten vorherrschenden Auffassung, wonach ein Objekt in einer Rolle durch ein separates Objekt repräsentiert wird, das mit seinem Rollenspieler über eine Instanz der Role-of-Relation verbunden wird. Ein Kritik dieser Auffassung folgt an gegebener Stelle (z. B. bei der Vorstellung des Rollenbegriffs von Wieringa et al. in Abschnitt 3.3.3 unten) sowie in der Diskussion in Abschnitt 3.4.4.

Falkenbergs Object-role model

Falkenberg konstruiert sein Object-role model aus nur zwei primitiven Arten von Elementen: den Objekten und den Rollen [1976]. Eine Assoziation ist bei Falkenberg genau wie in LODWICK eine Menge von Rolle/Objekt-Paaren, wobei ein Objekt selbst wieder eine Assoziation sein kann. Assoziationstypen (Relationen in LODWICK) sind dann als Mengen von Assoziationen definiert, die die gleichen Rollen beinhalten, und Objekttypen als Mengen von Objekten, die mindestens eine Rolle gemeinsam haben. Jede Rolle definiert damit implizit einen Objekttyp.

Falkenbergs Object-role model ist ein typisches Beispiel für eine extensionale Modellspezifikation: Die Typen sind ausschließlich über die Menge der Instanzen definiert, die darunter fallen. So kann es auch keine zwei verschiedenen Assoziationstypen geben, die dieselben Rollen haben, und keine zwei verschiedenen Objekttypen, deren Instanzen alle dieselben Rollen spielen.

Object-Role Modelling (ORM)

Im Object-Role Modelling (ORM[68]), einer Methode zur konzeptuellen Modellierung von relationalen Datenbankschemen, die aus der Verschmelzung vieler verwandter, von Falkenbergs [1976] und NIAM [Nijssen & Halpin 1989; Halpin 1995] abstammender Modellierungsansätze entstanden ist, wird der Gegenstandsbereich (universe of discourse) als eine Menge von Objekten aufgefaßt, die Rollen spielen [Halpin 1995]. Daß ein Objekt eine Rolle spielt, wird durch ein elementareres Fakt (elementary fact) ausgedrückt. Dabei entspricht ein solches Fakt einem Element einer Relation oder einer atomaren Formel in der Prädikatenlogik und die Rolle einer Stelle der Relation respektive des dazugehörigen Prädikats. Objekte werden zu Typen zusammengefaßt, deren Elemente alle dieselbe Rolle spielen. Anders als beispielsweise im Entity-Relationship-Modell wird hier nicht zwischen Relationen und Attributen unterschieden; statt dessen werden Eigenschaften (properties) durch einstellige Fakten ausgedrückt.

Die Auffassung von Rollen im ORM ist kaum verschieden von der im ursprünglichen Relationen- oder im Entity-Relationship-Modell (s. d.); lediglich die Betonung der Konzepte ist eine andere. Tatsächlich unterscheidet sich ORM insgesamt allenfalls methodisch von den zahlreichen erweiterten Entity-Relationship-Modellen, deren formale Äquivalenz von Halpin weitgehend negiert wird [1995, S. 404]. Hervorzuheben bleibt, daß sich die Methode von ORM an sprachlich formulierte Aussagen hält, deren grammatikalische Analyse in ein oder mehrere Prädikate mit den dazugehörigen Objekten als Rollenfüller die Struktur eines ORM mehr oder weniger automatisch ergibt. In diesem Punkt kommt der Ansatz der in Abschnitt 3.3.1 besprochenen Funktion der Rollen in der Linguistik sehr nahe.

Ausdrücklich nicht verwendet werden Rollen im ORM zur temporären Klassifikation von Objekten; statt dessen werden *Student* und *Angestellter* – das klassische Beispiel – als normale Subtypen eines Typs *Person* modelliert [Halpin & Proper 1995]. In Abgrenzung von der Spezialisierung sind die über Disjunktion gebildeten sog. polymorphen Typen (polymorphic types) nach Halpin und Proper die Vereinigung von Typen ohne gemeinsam zugrunde liegende Struktur. Als typisches Beispiel, für das die gewöhnliche Spezialisierung nicht in Frage käme, werden rekursive Datenstrukturen wie Listen (bestehend aus einem Kopf und dem Rest, der selbst wieder eine Liste ist) [Halpin & Proper

[68] Dieselbe Abkürzung ORM wird auch von Pernici [1990] verwendet; allerdings verbirgt sich dahinter ein anderer Ansatz (s. Abschnitt 3.3.3 unten).

1995, Abbildung 9] oder der induktive Aufbau von Formeln [ter Hofstede & van der Weide 1993] angeführt. Dagegen steht jedoch das über gewöhnliche Subtypenbildung realisierte Composite pattern [Gamma et al.1995], auf das allerdings tatsächlich der Einwand zutrifft, daß atomare und zusammengesetzte Komponenten nicht immer auch eine gemeinsame Generalisierung haben.[69]

Halpin und Proper versuchen, ihre Auffassung gegen die Argumentation, Polymorphismus und Spezialisierung seien lediglich zwei verschiedene Sichtweisen derselben Sache, zu verteidigen. Dazu führen sie, ter Hofstede & van der Weide [1993] folgend, an, daß mengentheoretisch die Spezialisierung der Einschränkung von Mengen (set comprehension) durch ein Teilmengen bildendes Prädikat entspräche, wo hingegen die Generalisierung im Sinne des Polymorphismus mit der Vereinigung von Mengen (set union) korrespondiere [Halpin & Proper 1995; ter Hofstede & van der Weide 1993]. Diese Position ist jedoch kaum haltbar, denn jeder durch Vereinigung gewonnene Typ läßt sich durch entsprechende (ggf. neu einzuführende) Prädikate in seine Konstituenten zerlegen, die dann nach der obigen Darstellung Subtypen sein müßten.

Rollen im OM Datenmodell

Das OM Datenmodell [Norrie 1993] unterscheidet grundsätzlich zwischen der Klassifikation und der Typisierung von Objekten. Dabei steht die Klassifikation für die inhaltliche, die Typisierung für die die Repräsentation betreffende Einordnung der Objekte. Bei der Klassifikation wird zusätzlich zwischen Arten (kinds) und Rollen (roles) unterschieden, wobei es sich bei ersteren um eine statische oder invariante, bei letzterem hingegen um die zeitlich veränderliche und damit dynamische Form der Klassifikation handelt. Rollen und Arten können gemischt in einer Spezialisierungshierarchie auftreten [Norrie et al. 1996].

Sinn und Zweck der Unterscheidung zwischen Arten und Rollen ist die Auferlegung von bestimmten Nebenbedingungen für die Evolution von Objekten in einem Modell. So ist es z. B. möglich, daß ein Objekt von einer Rolle in eine andere Rolle oder zu einer Art wechselt. Unmöglich ist jedoch, daß ein Objekt von einer Art in eine andere oder gar in eine Rolle schlüpft. Arten stellen damit gewissermaßen finale Rollen dar.

Allerdings wird, wenn eine Art die Spezialisierung einer Rolle ist, ein Objekt seine Art ablegen, wenn es auch die Rolle wechselt. (In OM existiert zu jedem Objekt in einer Art/Rolle jeweils ein Objekt in all denen Arten/Rollen, von de-

[69] Vgl. dazu auch die Darstellung des Composite patterns mit Rollen in Abschnitt 4.3.1.

nen es eine Spezialisierung ist. Dabei ist es grundsätzlich möglich, daß ein Objekt in mehrere Arten/Rollen spezialisiert wird. Wechselt ein Objekt eine seiner Rollen, dann muß es auch aus all deren Spezialisierungen entfernt werden, woraus sich die Aufgabe der Art automatisch ergibt. [Norrie et al. 1996])

Rollen bei Kristensen und Østerbye

Kristensen und Østerbye [1996] decken mit ihrer Darstellung von Rollen ein breites Spektrum von der konzeptuellen Modellierung bis hin zur Integration ihres Ansatzes in bestehende Programmiersprachen ab. Rollen werden von ihnen als Konzepte aufgefaßt, die, ähnlich wie Klassen, der Klassifikation dienen und sowohl aggregiert als auch spezialisiert werden können. Rollen existieren jedoch anders als Klassen nie isoliert, sondern sind stets an andere Rollen oder Klassen gebunden. Daraus ergibt sich eine Vielzahl von möglichen Konstellation, für die die Autoren eine intuitive, aber umfangreiche Notation vorlegen.

Kristensen [1995] interpretiert eine Rolle als eine Perspektive, einen Blickwinkel auf ein Objekt. Er knüpft daran die folgenden Eigenschaften:

- Von einem Objekt, das über eine seiner Rolle angesprochen wird, sind die Eigenschaften der Rolle und die des Objektes, nicht aber die anderer Rollen sichtbar. Spricht man das Objekt direkt an, ist keine seiner Rollen sichtbar.
- Eine Rolle ist immer (direkt oder indirekt über andere Rollen) an ein Objekt gebunden, und ihre Existenz hängt an der des Objektes. Die Rolle kann zur Erfüllung ihrer Aufgaben Eigenschaften ihres Objektes heranziehen, nicht jedoch umgekehrt.
- Das Objekt und seine Rollen bilden zusammen eine Einheit (Subjekt genannt) und haben demzufolge auch nur eine Identität. Tatsächlich sollen die Instanzen von Rollen keine eigene Identität haben; statt dessen teilt die Rolle die Identität des Objektes, der sie beigeordnet ist.
- Rollen können einem Objekt dynamisch beigeordnet und auch wieder entzogen werden. Ein Objekt kann dieselbe Rolle mehrfach spielen.
- Rollen klassifizieren ihre Objekte.

Die Rollenbildung wird von Kristensen und Østerbye [1996] als eine Art Spezialisierung betrachtet. Dem liegt zugrunde, daß die Extension der Rolle (nach Auffassung der Autoren) eine Teilmenge der Klasse ist, von der sie eine Rolle bildet, da nicht alle Objekte der Klasse die Rolle spielen, und daß die Eigenschaften des Objekts auch Eigenschaften der Rolle sind, also gewissermaßen

vererbt werden. Umgekehrt ist jedoch eine Klasse nicht die Generalisierung ihrer Rollen.

Die Eigenschaften eines Objektes, die durch eine Rolle hinzukommen, werden extrinsisch (extrinsic) genannt, was soviel bedeuten soll, wie daß diese Eigenschaften – im Gegensatz zu denen des Objektes, den sog. intrinsischen (intrinsic) – nur qua Existenz der Rolle vorhanden sind. Folgerichtig werden diese Eigenschaften auch mit der Rolle, nicht mit dem Objekt, spezifiziert, sind also, bis auf das Heranziehen von intrinsischen Eigenschaften zu deren Umsetzung, von der Rolle und nicht vom Objekt geprägt. Dies widerspricht der Auffassung, daß wenn Objekte verschiedener Klassen dieselbe Rolle spielen, daß dann jedes seine Rolle anders, nämlich so wie es kann, ausfüllt.

Problematisch an der Darstellung von Kristensen und Østerbye ist, daß die Autoren zwar eine Einheit von Objekt und Rolle, auch und gerade in bezug auf die Identität, proklamieren, diese jedoch selbst (durch die Instanziierung von Rollen, das Binden von Rollen an Rollen, den sog. Rollentransfer und „dangling roles“) immer wieder in Frage stellen. Leider beschränkt sich die formale Darstellung auf die Angabe einer Notation sowie auf die schemenhafte Umsetzung der Konzepte in zwei Programmiersprachen, BETA und SMALLTALK [Kristensen & Østerbye 1996]. In dieser Hinsicht ist dieser Ansatz unbedingt mit dem von Gottlob et al. [1996], in dem das Problem der Identität von Rollen ausdrücklich behandelt wird, zu vergleichen.

Rollen in M.E.R.O.D.E.

Snoeck und Dedene [1996] stellen den üblichen, per Spezialisierung abgeleiteten Typen (specialization types) eines konzeptuellen Modells sogenannte Rollentypen (role types) gegenüber, die Teile des Verhaltens von Typen (object types) modellieren. Das primäre Unterscheidungskriterium zwischen Spezialisierung und Rolle ist die Frage, ob es sich um eine permanente oder um eine temporäre Klassifikation der betroffenen Instanzen handelt. Weiterhin wird für Rollen gefordert, daß sie lediglich neue Ereignistypen und/oder -sequenzen spezifizieren, nicht aber neue Methoden oder gar die Wertebereiche von Attributen einschränken. Andernfalls, so Snoeck und Dedene, liegt ein Widerspruch vor, der aber durch Änderung des konzeptuellen Schemas aufgelöst werden kann, so z. B. durch Einführung einer Assoziation zwischen Objekt und Rolle.

Anders als gewöhnliche Typen haben echte Rollentypen keine eigenen Instanzen, sondern spezifizieren lediglich Sequenzen von Ereignissen, auf die ein Objekt reagieren kann. Die Tatsache, daß die Instanzen eines Typs *P* eine Rolle *R*

spielen können, wird durch $P\rho R$ ausgedrückt und bedeutet, daß das Verhalten des Objektes neben seiner eigenen auch der Sequenzspezifikation der Rolle genügen muß. Dabei darf die Rolle ausschließlich Ereignisse des Objektes und selbst davon keine Konstruktoren oder Destruktoren verwenden [Snoeck & Dedene 1996].

Die Arbeit von Snoeck und Dedene [1996] zielt allgemein darauf ab, das Prinzip der Substituierbarkeit auch unter dynamischen Gesichtspunkten zu garantieren, d. h., zu gewährleisten, daß jede Instanz einer Spezialisierung stets die gleichen Ereignisfolgen behandeln kann wie die Instanzen des Typs, die sie substituiert. Sie geben damit der auch in der konzeptuellen Modellierung üblichen Is-a-Hierarchie ein über das übliche Maß hinausgehendes formales Fundament. Welche Rolle die Rollen dabei spielen, wird jedoch nicht ganz klar. Vermutlich werden sie der Vollständigkeit halber (und in Anlehnung an JSD [Jackson & Cameron 1983]) integriert.

3.3.3 Rollen in der Datenmodellierung

Rollen im Relationenmodell

Codd unterscheidet in seiner initialen Arbeit über das Relationenmodell [1970] zwischen mathematischen Relationen (relations) als einer Menge von geordneten Tupeln und Beziehungen (relationships), in denen die Stellen beliebig permutiert sein dürfen und über den Namen des Wertebereichs angesprochen werden. Kommt ein Wertebereich innerhalb einer Beziehung jedoch mehrfach vor, so muß er über eindeutige Rollennamen, die seine jeweilige Funktion innerhalb der Beziehung wiedergeben, qualifiziert werden [Codd 1970].

Später dann werden im Relationenmodell grundsätzlich nicht mehr die Namen der Wertebereiche, sondern eindeutige Attributnamen (attributes) zur Qualifizierung der Stellen einer Relation verwendet [Codd 1979]. Dafür sieht Codd (in RM/T) vor, daß eine Entität (entity) mehrere verschiedene Typen haben und diese auch dynamisch annehmen oder ablegen kann, ohne jedoch dabei den Begriff Rolle explizit zu verwenden.

Rollen bei Kent

Kent [1978] verwendet den Begriff der Rolle, genau wie Codd, als Bezeichner der Stellen einer Relation. Er unterscheidet aber zwischen Kategorien (categories) als Arten (kinds) von Entitäten und dem Wertebereich (domain)

der Rollen (Stellen) von Relationen, der durchaus aus der Vereinigung mehrerer Kategorien bestehen kann [Kent 1978, S. 64]. Diese Festlegung ist heute weitgehend in die Typhierarchien objektorientierter Datenmodelle aufgegangen, in denen Supertypen als Vereinigung ihrer Subtypen interpretiert werden. Auf der anderen Seite bemerkt Kent, daß die einfache Verschachtelung von Typen durch Subtypenbildung nicht ausreicht, um die in der Realität häufig vorkommende Mehrfachklassifikation angemessen nachzubilden [S. 105].

Bachmans Role data model

Bachman, einer der Pioniere des Netzwerkmodells, betrachtete als einer der ersten Datenmodellierer die Unterscheidung zwischen Entität (entity) als Ausdruck der Existenz eines Objektes und Rolle (role) als Ausdruck des Verhaltens eines Objektes in einem gegebenen Kontext als natürlich. Mit seinem Rollenmodell (role data model) versuchte er, ein Datenmodell zu schaffen, das die bis dahin bekannten und verwendeten an semantischer Ausdrucksstärke übertraf und somit die Realität genauer abbilden konnte [Bachman & Daya 1977; Bachman 1980; 1989].

Bachmans Rollenmodell liegt die einfache Beobachtung zugrunde, daß die meisten aller in der Praxis vorkommenden Entitätstypen auf die Namen *Angestellter*, *Kunde*, *Lieferant*, *Patient*, *Student* etc. lauten, also Typen bezeichnen, die sich nicht durch ihre Identität oder Herkunft, sondern durch ihr Verhalten voneinander unterscheiden. Bachman differenziert daher zwischen dem statischen Aspekt eines Objektes, der Entität, und seinen dynamischen Aspekten, den Rollen, die das Objekt spielen kann [Bachman 1989, S. 30]. Die Menge der Rollen und die der Entitäten sind disjunkt; eine Rolle ist ein „definiertes Verhaltensmuster, das von Entitäten verschiedener Art angenommen werden kann“ [Bachman 1977, S. 465].

Ein Entitätstyp (entity type) wird über seine Fähigkeit, bestimmte Rollentypen (role types) zu spielen, definiert. Dabei kann ein und derselbe Rollentyp verschieden Entitätstypen charakterisieren (sharable roles). So ist z. B. in einem bestimmten Kontext *Arbeitgeber* ein möglicher Rollentyp der Entitätstypen *Person*, *Firma* und *Behörde*. *Arbeitnehmer* hingegen ist ausschließlich Rolle von *Person*. Der Entitätstyp *Person* selbst wird beispielsweise wiederum durch die möglichen Rollentypen *Arbeitnehmer*, *Arbeitgeber*, *Kunde*, *Aktionär* und *Lieferant* charakterisiert. Welche Rollen mit dem Auftreten einer Entität verbunden sein können oder müssen und welche Rollentypen von mehreren Entitätstypen geteilt werden, wird mit der Schemadefintion festgelegt [Bachman &

Daya 1977]. Zu beachten ist, daß eine Entität, von expliziten Einschränkungen durch die Schemadefintion abgesehen, jeden ihrer Rollentypen gleichzeitig, jedoch jeweils nur einfach besetzen kann [S. 466]. Für den Fall, daß jede Entität genau eine, seine eigene Identitätsrolle (identity role) spielt, reduziert sich das Rollenmodell auf das Netzwerkmodell [Bachman 1980].

Rückblickend motiviert Bachman sein Rollenmodell wie folgt [1980]: Wenn, was das Netzwerkmodell prinzipiell zuließ, die Member records eines Owner records verschiedenen Typs waren, dann mußte jedes Programm beim Durchlaufen der Member records typabhängig verzweigen, um die zu den Member records gehörenden Attribute ansprechen zu können. Da diese Records aber als eine Menge zusammengefaßt waren, sollten sie in den meisten Fällen trotz ihres unterschiedlichen Typs auch gleich behandelt werden, was zu erheblicher Redundanz im Programmcode führte. Indem man den unterschiedlichen Entitätstypen der Member records einen gemeinsame, d. h. von allen gespielten Rollentyp zuordnete und die angesprochenen Attribute über den Rollennamen adressierte, konnten Entitäten verschiedenen Typs durch denselben Code behandelt werden. Diese Eigenschaft ist heute unter dem Namen Polymorphie bekannt und wird als eine der großen Errungenschaften der objektorientierten Programmierung gepriesen.

Die pragmatische Motivation der Einführung des Rollenkonzepts wurde noch von einer theoretischen gestützt: Die akademische Gemeinde wollte das Netzwerkmodell dahingehend einschränken, daß in einer Owner-Member-Beziehung nur noch Member records gleichen Typs zugelassen würden. Damit wäre das Netzwerkmodell mit dem Relationenmodell, das aufgrund seiner mathematischen Definition keine variierenden Typen zuläßt, direkter zu vergleichen gewesen [Bachman 1980]. Indem man Ownership und Membership Rollen statt Entitäten zuordnet und ein Rollentyp von verschiedenen Entitätstypen gespielt werden konnte, wurde diese Anforderung erfüllt, ohne daß das Netzwerkmodell etwas von seiner Ausdrucksstärke verloren hätte.

Das Rollenmodell, das ursprünglich als Verallgemeinerung des Netzwerk- und des Relationenmodells gedacht war [Bachman & Daya 1977, S. 464, 466] wurde, bevor es Fuß fassen konnte, durch das Entity-Relationship-Modell ([Chen 1976]; s. Abschnitt 3.3.2 oben) verdrängt. Anstatt sich mit den dynamischen Aspekten der Rollen, die mit den auf die statische Modellierung ausgelegten Notationen wie dem Entity-Relationship-Diagramm kaum vereinbar sind, zu befassen, wurden Einschränkungen der Kardinalitäten von Relationen untersucht und populär gemacht [Bachman 1989, S. 30]. Die konzeptuelle Unter-

scheidung zwischen Entität und Rolle schien darüber in Vergessenheit geraten zu sein. Erst mit dem Aufkommen des objektorientierten Datenmodells, das genau wie das Netzwerkmodell und anders als das mengenorientierte Relationenmodell ein navigierendes ist, ist diese Unterscheidung wieder in das Bewustsein der Datenmodellierer zurückgekehrt.

Bachmans Rollenmodell geht über das seiner Nachfolger teilweise erheblich hinaus. Zwar wird der Wunsch nach Polymorphie im objektorientierten Datenmodell durch die Typhierarchie und die damit einhergehenden Substitutionsregeln zum Teil gedeckt, doch hat Bachman im Grunde bereits damals darauf hingewiesen, daß eine Rolle im wesentlichen ein Interface ist, über das Objekte verschiedenen Typs einheitlich angesprochen werden können. Auch wenn damals die Verwendung einer Typhierarchie nicht zur Disposition stand, so wird doch deutlich, daß die einzelnen Entitätstypen, die eine Rolle ausfüllen können, vollkommen verschieden sein können, also nicht über eine gemeinsame Generalisierung in Verbindung stehen müssen. Der Begriff der Rolle ist damit eine völlig eigenständige Dimension in der Datenmodellierung.

Rollen in Universal scheme interfaces

In einem Universal scheme interface zu einer relationalen Datenbank muß jedes Attribut oder Feld der Datenbank eindeutig benannt sein. Sind die Attribute das nicht, müssen sie umbenannt werden. Damit geht aber der implizite Zusammenhang, der sich aus der Verwendung gleicher Attributnamen in verschiedenen Relationen ergibt, verloren, und automatische Joins werden unmöglich.

Um den Zusammenhang wiederherstellen zu können, definieren Maier et al. eine Rollenbeziehung zwischen Attributen [1985]. Demnach ist Attribut *B* eine Rolle von Attribut *A*, wenn die Menge der Objekte von *A* die von *B* immer enthält. So ist z. B. *Employee* eine Rolle von *Person*. Diese Inklusion erlaubt nun die Übertragung von bestehenden Paarungen von Attributen auf andere Attribute, die Rollen davon sind. Aus der Paarung (*Person*, *DateOfBirth*) beispielsweise folgt auch (*Employee*, *DateOfBirth*). Es entspricht dies aber im wesentlichen dem Vererben von Relationen von Typen auf ihre Subtypen in einer Subsumtionshierarchie und ist wohl eher als eine zur objektorientierten alternative Façon de parler zu verstehen.

Objektspezialisierung

Der Grundgedanke der Objektspezialisierung (object specialization) [Sciore 1989] ist, daß Instanzen (Objekte) von anderen Instanzen erben, ohne daß de-

ren Klassen über eine Vererbungshierarchie miteinander verbunden sind. Anders als in prototypbasierten Ansätzen, in denen es gar keine Klassen mehr gibt, werden die Eigenschaften einer Instanz nach wie vor über eine Klasse spezifiziert; der Vererbungsmechanismus ist jedoch auf Instanzebene definiert.

Bei der Objektspezialisierung wird ein Objekt der Realität durch eine Instanzhierarchie repräsentiert. Jede dieser Instanzen stellt eine Rolle des Objektes dar. Dabei erbt eine Instanz weiter unten in der Hierarchie alle Eigenschaften der darüberliegenden Instanzen, die sie aber überschreiben kann.

Der Zugriff auf ein durch eine solche Hierarchie repräsentiertes Objekt erfolgt stets über einen Verweis auf eine ihrer Instanzen. Aus der Sicht des Zugreifenden stellt sich das Objekt als die Summe der durch die auf dem Pfad vom Einstieg bis zur Wurzel liegenden Instanzen dargestellten Rollen dar. Der Rest der Hierarchie bleibt für den Zugreifenden unsichtbar; er hat somit nur eine Sicht (view) oder Perspektive (perspective) auf das Objekt.

Neben dem strukturellen Vorteil der Partitionierung eines Objektes in Sichten und der daraus resultierenden Möglichkeit der perspektivweisen Überladung von Eigenschaften führt Sciore eine Erhöhung der Zugriffsgeschwindigkeit in Datenbanksystemen durch ein entsprechendes Clustering entlang der Perspektiven an. Daß durch die Repräsentation eines Objektes durch mehrere Instanzen die Identität des Objektes (jede Instanz hat ihre eigene Identität!) zunächst verlorengeht, wird nicht erwähnt. Es ist jedoch zu vermuten, daß dem bei der Implementierung der Objektspezialisierung im VISION-System [Sciore 1989] Rechnung getragen wurde, zumal sich das Problem lösen läßt (vgl. u.). Es bleiben jedoch die prinzipiellen Probleme der Vererbung auf Instanzebene, darunter auch der in Abschnitt 2.2.4 dargestellte Widerspruch.

Rollen bei Papazoglou

Auch Papazoglou [1991; 1995] versucht, die verschiedenen Facetten, die ein Objekt in verschiedenen Kontexten oder aus verschiedenen Perspektiven betrachtet haben kann, durch einen geeigneten Rollenbegriff abzudecken. Dazu unterscheidet er zwischen statischen und dynamischen Rollen eines Objektes, wobei die statischen bei den meisten anderen Autoren nicht als Rollen bezeichnet werden, sondern einfach nur Klassen heißen. Die Menge der statischen Rollen eines Objektes ist bei Papazoglou nämlich schlicht durch seine Position in der Klassenhierarchie bestimmt; sie besteht aus dem Pfad (oder den Pfaden im Falle einer Mehrfachhierarchie) von der Klasse bis zur Wurzel der Hierarchie.

Im Gegensatz dazu werden die dynamischen oder transienten Rollen einer Instanz nicht durch die Klassenhierarchie vorgegeben, sondern von den Instanzen dynamisch erworben. Dazu muß man wissen, daß Papazoglou eine Klasse als Realisierung eines Typs betrachtet, die neben den Methoden zur Instanzerzeugung auch noch einen sog. Extent hat, in dem all ihre Instanzen aufgezählt werden. Ist eine Instanz im Extent einer Klasse, bedeutet dies, daß die Instanz die mit der Klasse verbundene Rolle spielt. Da die Instanzen einer Klasse automatisch in deren Extent und im Extent all ihrer Oberklassen vertreten sind, wird die statische Rollenzugehörigkeit automatisch geregelt. Der dynamische Erwerb einer Rolle wird hingegen durch einen expliziten Eintrag in den Extent der betreffenden, die Rolle definierende Klasse (role-defining class) vollzogen. Dabei kann eine Instanz mehrere dynamische Rollen gleichzeitig spielen, wenn diese sich nicht gegenseitig ausschließen [Papazoglou 1991].

Object-with-roles-Modell (ORM)

Im Rahmen des Espritprojektes ITHACA (Integrated Toolkit for Highly Advanced Computer Applications), an dem neben der federführenden Siemens-Nixdorf AG etliche europäische Firmen sowie die Universität Genf beteiligt waren und aus dem die objektorientierte Programmiersprache COOL hervorgegangen ist [Computerwoche 1991], schlägt Pernici [1990] vor, die Spezifikation eines Objektes in mehrere Abschnitte, Rollen genannt, aufzugliedern und so die Komplexität der Spezifikation zu reduzieren. Dabei hat jede Klasse eine Basisrolle (mit Namen base-role) sowie eine Reihe von klassenspezifischen weiteren Rollen, die individuell benannt und die getrennt instanziiert werden. Bei der Instanziierung einer Klasse wird immer die Basisrolle instanziiert; weitere Rollen können bei Bedarf – auch mehrfach – instanziiert werden.

Jede Rolle verfügt über eine eigene Menge von Eigenschaften, Zuständen, Botschaften und Regeln. Eingehende Botschaften werden als Ereignisse interpretiert und lösen über die Regeln definierte Zustandsübergänge aus. Daneben spezifizieren die Regeln auch die Bedingungen für die gleichzeitige Existenz mehrerer Rollen sowie für den Wechsel zwischen den Rollen.

Eine Klasse wird bei Pernici ausschließlich über ihren Namen und ihre Rollen, die sämtlich lokal zu ihrer Klasse sind, spezifiziert. Auf diese Weise wird die Spezifikation der Klasse, insbesondere die des Verhaltens ihrer Instanzen, partitioniert. Durch die Aufteilung in logisch kohärente Abschnitte, die den Message categories von SMALLTALK [Goldberg & Robson 1989] ähneln, wird der

Spezifikationsprozeß insgesamt erleichtert – eine Bedeutung der Rolle außerhalb der Klasse ist jedoch nicht erkennbar.

Auch wenn Pernici keine Hinweise zur Implementierung ihres Rollenkonzeptes gibt, so läßt es sich doch leicht auf lokale Klassen (Klassen, die lokal zu einer übergeordneten Klasse definiert werden; eine lokale Klasse pro Rolle) und öffentlich zugängliche (public deklarierte) Aggregationen (Instanzen der lokalen Klassen als Bestandteil des Objektes) abbilden. Ähnliches schlagen allerdings auch Martin und Odell [1992, S. 416] als Alternative zum sog. Object slicing vor (s. Abschnitt 3.3.5). Wesentliche, darüber hinausgehende Aspekte eines allgemeineren Rollenkonzeptes läßt es vermissen. So kann, auch wenn die Rollen einer Oberklasse auf ihre Unterklassen vererbt werden, eine Rolle grundsätzlich nicht von verschiedenen Objekten ausgefüllt werden. Die Funktion eines Platzhalters in Relationen oder Kollaborationen zwischen Klassen kommt ihr damit nicht zu, und es bleibt unklar, welche weiteren Vorteile der Ansatz für die Modellierung bringen könnte.

Rollen in ASPECTS

Richardson und Schwarz [1991] beschreiten mit ihrem ASPECTS-Ansatz einen ähnlichen Weg. Ein Aspekt (aspect) eines Objektes ergänzt dieses um spezifische Zustände und Verhalten, die durch Variablen und Methoden repräsentiert werden. Dabei kommt es den Autoren darauf an, zu zeigen, daß dies trotz Repräsentation eines Objektes durch mehrere Instanzen unter Beibehaltung der Identität des Objektes selbst in streng typisierten Umgebungen möglich ist.

ASPECTS unterscheidet grundsätzlich zwischen abstrakten Typen und deren Implementierungen. Typkompatibilität in ASPECTS ist über Konformität (conformity) definiert: Ein Typ ist mit einem andern Typ konform, wenn er all dessen Signaturen (oder Einschränkungen davon) deklariert; eine Implementierung ist mit einem Typ konform, wenn sie mindestens dessen deklarierten Signaturen implementiert. Eine explizite Deklaration der Kompatibilität wie etwa die *extends*-Klausel bei der Type extension [Wirth 1988] ist nicht vorgesehen; dies soll dem Entwickler die Möglichkeit geben, neue Supertypen einzuführen, ohne die Definition existierender Typen abändern zu müssen.

Ein Aspekt eines Typs wird durch eine separate Implementierung (nebst des dazugehörigen Typs) spezifiziert. Diese gibt den Typ, von dem sie einen Aspekt darstellt (den sogenannten Basistyp), durch eine *extends*-Klausel (nicht zu verwechseln mit der Deklaration einer Typerweiterung, s. o.) bekannt. Der Aspekttyp wird unter Bezugnahme auf eine Instanz seines Basistyps instanzi-

iert; die erzeugte Instanz stellt dann einen Aspekt der referenzierten dar. Die Aspektinstanz ist mit einer Variable ihres Basistyps gemäß dem Prinzip der Konformität nur dann zuweisungskompatibel, wenn der Aspekt alle Signaturen des Basistyps implementiert. Um dies zu vereinfachen, kann ein Aspekt per Deklaration die Signaturen seines Basistyps durchscheinen lassen.

Die Identität von Objekten und deren Aspekten wird in ASPECTS durch zwei verschiedene Gleicheitsoperatoren entschieden: den üblichen Referenzidentitätsoperator und einen indirekten Operator, der die Identität der Basisinstanzen zweier Aspekte bzw. eines Aspekts und einer Basisinstanz überprüft. Diese zweite Defintion von Objektidentität läßt sich prinzipiell auch auf andere Ansätze zur Rollenbildung übertragen. Um sie zu implementieren, müssen lediglich die mit den Rollen verbundenen Basisinstanzen auf Identität überprüft werden.

Rollen in FIBONACCI

Genau wie in den vorgenannten Ansätzen kann auch in der objektorientierten Datenbankprogrammiersprache FIBONACCI ein Objekt verschiedene Rollen einnehmen [Albano et al. 1993]. Zustände und Verhalten des Objektes sind an seine Rollen gebunden; das Objekt selbst verfügt nur über seine Identität, die Liste der Rollen, die es gerade spielt, sowie Verfahren zur Überprüfung der Gleichheit mit einem anderen Objekt und die Annahme neuer Rollen. Alle Eigenschaften werden jedoch nicht direkt, sondern ausschließlich über eine seiner Rollen, die es gerade spielt, adressiert.

Die Deklaration eines Objekttyps, die in FIBONACCI dynamisch erfolgen kann, umfaßt demnach nicht mehr als die Nennung seines Namens. Jeder Objekttyp ist die Wurzel einer Menge von Rollentypen, der Rollenfamilie (role family) des Objektes. Die Rollentypen einer Rollenfamilie können per Deklaration in einer Subtypenbeziehung zueinander stehen; sie bilden eine Mehrfach-Rollenhierarchie. Die Menge der Botschaften, auf die eine Instanz des Objekttyps reagieren kann, wird durch seine Rollenfamilie bestimmt.

Anders als eine Klassendefinition in objektorientierten Programmiersprachen spezifiziert ein Rollentyp in FIBONACCI lediglich ein Interface in Form einer Menge von Signaturen. Bei der Instanziierung eines Rollentyps muß daher stets eine Implementierung des Interfaces angegeben werden. Diese kann grundsätzlich von Instanz zu Instanz variieren, was eigentlich für einen Prototypenansatz spricht [Lieberman 1986]; es ist aber auch möglich, einen Konstruktor zu definieren, der die Implementierung für alle Instanzen, die damit erzeugt werden,

einheitlich vorgibt. Jede Instanziierung liefert einen Verweis, über den die Rolle und das implizit dazugehörige Objekt adressiert werden können. Existieren keine Verweise mehr auf ein Objekt, so wird es mittels Garbage collection entfernt.

Ein Objekt kann (über eine seiner Rollen angesprochen) dynamisch weitere Rollen aus seiner Rollenfamilie annehmen, wobei jede Rolle nur einmal erworben werden darf. Mit jeder dieser dynamischen Erweiterungen des Objektes müssen die Implementierungen aller dazukommenden Signaturen aus der Deklaration der Rollentypen aufgeführt sein. Auch hier ist die Definition spezieller Konstruktoren vorgesehen.

Eine offensichtliche Besonderheit des FIBONACCI-Systems ist die Tatsache, daß (Rollen)-Typen lediglich Interfaces spezifizieren. Das Positive dieses etwas ungewöhnlichen Ansatzes wird jedoch dadurch wieder abgeschwächt, daß jede einzelne Instanz das Interface ihres Typs individuell realisieren kann, obwohl die Instanzen, die eine Rolle spielen, alle Instanzen desselben Objekttyps (der Wurzel der Rollenfamilie) und damit quasi genetisch miteinander verwandt sind, so daß man, wie bei anderen typbasierten Ansätzen auch, eigentlich erwarten dürfte, daß sie ihre Rollen auf dieselbe Art realisieren. Die individuelle Implementierung ist also hier gewissermaßen widersinnig (sie spricht eigentlich für einen prototypenbasierten Ansatz); sie wäre aber angebracht, wenn ein und dieselbe Rolle von Instanzen verschiedenen Objekttyps gespielt werden könnte. Befremdlich ist weiterhin, daß in einem Beispiel *Person*, der prototypische natürliche Typ, als Rolle angeführt wird.

Rollen in TROLL

In der objektorientierten Spezifikationssprache TROLL [Jungclaus et al. 1991; Jungclaus 1993; Wieringa et al. 1993], die an der TU Braunschweig entworfen wurde, wird die Möglichkeit eingeräumt, verschiedene Aspekte eines Objekttyps zu spezifizieren. Drei Arten von Aspekten stehen zur Verfügung: Spezialisierungen (specializations), Rollen (roles) und Generalisierungen (generalizations) [Jungclaus et al. 1991, S. 70]. Dabei werden interessanterweise Spezialisierungen als Spezialfälle der Rollen angesehen: Während Rollen während der Lebensdauer eines Objektes, auch wiederholt, angenommen und wieder abgelegt werden können, währt eine Spezialisierung stets von der Generierung bis zur Entfernung des Objektes [Jungclaus et al. 1991, S. 74; Ehrich et al. 1993; Wieringa et al. 1993]. Rollen sind also temporärer und damit dynamischer Natur, Spezialisierungen permanenter und damit statischer.

Eine Spezialisierung in TROLL ist eine detailliertere Sicht auf ein Objekt [Jungclaus 1993, S. 106]. Das spezialisierte Objekt wird dabei durch mindestens zwei Instanzen repräsentiert, die sich einen Identifikator teilen. Dadurch, daß die Menge der Identifikatoren der Spezialisierung eine Teilmenge der des Basistyps sind, wird der Basistyp partitioniert [Jungclaus 1993, S. 147]; Spezialisierungen werden also auch als Teilmengen interpretiert.

Da ein Objekt gleichzeitig mehrere Rollen und jede Rolle mehrfach spielen kann[70], werden Rollen in TROLL als Aggregationen realisiert. Sie bestehen aus einem Rollenteil (role part) und einem Basisteil (base part) [Jungclaus 1993, S. 154]. Aus der Darstellung eines Objektes in einer Rolle als Aggregation mehrerer Instanzen ergibt sich, daß die verschiedenen Auftreten in einer Rolle prinzipiell unterscheidbar sind, wodurch das sog. Zählproblem (s. u.) auf einfache Weise gelöst wird. Allerdings, und das räumen auch Ehrich et al. [1993] ein, ist die Darstellung von Rollen als Einschränkung eines Typs in Situationen, in denen Rollen von Objekten verschiedenen Typs gespielt werden, nicht haltbar; die Autoren denken deswegen über eine zweite Art von Rollen, die der speziellen Generalisierungen, nach.

Rollen bei Wieringa et al.

Wieringa et al. [1994; 1995] unterscheiden zwischen Objektklassen und Rollenklassen, deren Instanzen sie Objekte bzw. Rollen nennen. Die Extensionen beider Klassentypen sind Mengen; Unterklassen werden über eine Partitionierung dieser Mengen gebildet. Aus allen Paaren zueinander orthogonaler Partitionierungen werden automatisch alle Schnittklassen gebildet, die jeweils Unterklassen ihrer sie bildenden Partitionen sind. Die so entstehenden kleinsten Unterklassen werden wie in FREGE und LODWICK Spezies genannt; sie sind die einzigen, von denen Instanzen erzeugt werden dürfen.

Jede Instanz einer Unterklasse ist mit genau einer Instanz ihrer Oberklasse über die Is-a-Relation verbunden. Außerdem gehört zu jeder Rollenklasse genau eine Rollenspielerklasse, die durch die Deklaration eines Played-by-Attributs mit spezieller Semantik benannt wird und die selbst eine Objekt- oder eine Rollenklasse sein kann. Damit wird jeder Rolle genau ein Objekt (ggf. durch die transitive Verfolgung der Played-by-Relation) zugeordnet, das diese Rolle ausfüllt; das Objekt kann jedoch beliebig viele Rollen gleichzeitig oder nacheinander spielen.

[70] Letzteres zumindest trifft für Spezialisierungen nicht zu.

Wieringa et al. unterscheiden zusätzlich zwischen statischer und dynamischer Partitionierung. Wenn die Partitionierung statisch (also unveränderlich über die Zeit) ist, dann sprechen sie von statischen Unterklassen. Ist die Partitionierung hingegen veränderlich, so heißen die Unterklassen dynamische Unterklassen. Deren Instanzen können im Gegensatz zu denen der statischen Unterklassen die Unterklassenzugehörigkeit von Zeit zu Zeit wechseln, gehören aber zu einem Zeitpunkt immer nur genau einer Klasse an. Die Migration eines Objektes, die einer dynamischen Klassifikation entspricht, wird damit klar vom Spielen einer Rolle abgegrenzt.

Der Unterschied zwischen einer Rollenklasse, die über das Played-by-Attribut mit ihrer Rollenspielerklasse verbunden ist, und einer Unterklasse, die über die Is-a-Relation mit ihrer Oberklasse verbunden ist, ist der, daß eine Instanz der Unterklasse identisch mit einer Instanz der Oberklasse ist, während eine Instanz einer Rollenklasse immer verschieden von der ihrer Rollenspielerinstanz sein muß. Damit soll das sog. Zählproblem (counting problem) gelöst werden: Wenn man die Anzahl der Instanzen einer Rollenklasse (z. B. *Angestellter*) zählt, dann kann diese Zahl höher ausfallen, als wenn man die zu den Instanzen gehörenden Rollenspieler (Personen) zählt (wenn nämlich manche Personen mehrfach angestellt sind).

Der Ansatz von Wieringa et al. leidet nach theoretischen Gesichtspunkten darunter, daß die Is-a-Relation, die eigentlich eine Relation zwischen Klassen ist, mit der Played-by-Relation als Beziehung zwischen Instanzen auf dieselbe Stufe gestellt wird. („A person is therefore related by the is_a arrow to at most one employee“ [Wieringa et al. 1994]; vgl. dazu die Diskussion der Realisierung von Vererbung auf Instanzebene in Abschnitt 2.2.4.) Außerdem ist die Vorstellung, daß mit jeder Instanz einer Unterklasse eine Instanz der Oberklasse mit gleichem Identifikator (identifier) korrespondiere, wenn auch nicht unüblich (z. B. [Wagner 1989; Elmasri et al. 1991; Ryant 1997]), so doch zumindest merkwürdig.

Der wohl gravierendste Nachteil des Ansatzes ist aber, daß jede Rolle von Instanzen nur eines Typs (und seiner Subtypen) gespielt werden kann. Dadurch geht die mögliche Funktion einer Rolle als Platzhalter in einer Relation für Objekte verschiedenen Typs, die an sich charakteristisch für den Rollenbegriff ist (Abschnitt 3.1), verloren.

Rollen bei Kappel et al.

Auch Kappel und Schrefl verwenden eine vordefinierte Role-of-Relation, um bestimmte Instanzen als Rollen von anderen zu kennzeichnen [1991; 1996]. Dabei darf jede Instanz Rolle höchstens einer anderen sein, so daß durch die Role-of-Relation eine Objekthierarchie aufgespannt wird, die an die Objektspezialisierung von Sciore [1989] (s. d.) erinnert. Dabei kann auch hier eine (Instanz als) Rolle selbst wieder eine Rolle spielen, also z. B. ein Konferenzteilnehmer (als Rolle von Person) die Rolle eines Autors [Schrefl 1991].

Bemerkenswert an diesem Ansatz ist, daß die Role-of-Relation einerseits wie eine Relation zweiter Ordnung, also wie auch die Subtype-of-Relation als eine Beziehung zwischen Typen dargestellt wird, andererseits aber diese Beziehung zwischen Instanzen ausgeprägt wird [Kappel & Schrefl 1996]. Auch wenn hier im Gegensatz zu vielen anderen Arbeiten die Diskrepanz zwischen Deklaration und Instanziierung der Role-of-Relation deutlich gemacht wird, bleibt doch festzuhalten, daß es nicht möglich ist, diese in einer einfachen Modellierungssprache wie LODWICK aufzulösen. (Man vgl. dazu die Probleme mit der Vererbung auf Instanzebene sowie die Diskussion in Abschnitt 3.4.4).

In späteren Arbeiten rücken dann Kappel et al. neben dem evolutionären Aspekt des Rollenkonzepts, der mit dem Phänomen der Objektmigration [Mendelzon et al. 1994] verwandt ist, das kontextabhängige Verhalten von Objekten in den Vordergrund [1998], das ja ebenfalls durch die Rollenmetapher abgedeckt wird (Abschnitt 1.2.2). In [Hitz & Kappel 1999] schließlich wird die Äquivalenz von Rollen und der UML-Typdefinition diskutiert; man vgl. dazu jedoch die davon abweichende Gleichsetzung von Rollen und Interfaces in Abschnitt 4.2.

Rollen bei Gottlob et al.

Auch Gottlob et al. [1996] schlagen vor, ein Objekt der Realität durch mehrere Instanzen verschiedener Klassen zu repräsentieren, und zwar durch eine Instanz, die die Art des Objektes wiedergibt, und jeweils eine Instanz pro Rolle, die das Objekt gerade ausfüllt. Durch Erzeugen und Verwerfen von Rolleninstanzen kann das Objekt dynamisch beliebig viele Rollen annehmen und wieder aufgeben. Dabei tragen die Instanzen dafür Sorge, daß das Objekt insgesamt nur eine Identität (die derjenigen Instanz, die die Art festlegt) hat, einzelne Rollen aber unabhängig voneinander erzeugt und entfernt werden können. Als Nachteil des Ansatzes erweist sich wiederum, daß die Rollendefinition fest an die Klassenhierarchie gebunden ist, also nur Instanzen, die über eine gemeinsa-

me Oberklasse miteinander verwandt sind, dieselben Rollen spielen können. Dies führt nun wieder dazu, daß gemeinsame Oberklasse künstlich eingeführt werden müssen, wenn Instanzen verschiedener Klassen dieselbe Rolle ausfüllen können sollen, selbst wenn diese Verwandtschaft an sich gar nicht gegeben ist. Außerdem läßt sich der Ansatz, der für SMALLTALK und verwandte objektorientierte Programmiersprachen entwickelt wurde, kaum auf Programmiersprachen mit strenger Typprüfung übertragen, da Instanzen einer Rollenklasse nicht mit Variablen einer normalen Klasse typverträglich (zuweisungskompatibel) sind, solange sie nicht einen gemeinsamen Typ haben.

Rollen in DOOR

Ein weiterer Ansatz aus der Reihe, die die Rollen eines Objektes über eine Art beigeordnete Instanz (s. die Diskussion in Abschnitt 3.4.4) modellieren, ist der der DYNAMIC OBJECT-ORIENTED DATABASE PROGRAMMING LANGUAGE WITH ROLES (DOOR) von Wong et al. [1997]. Genau wie bei Wieringa et al. [1995] bemühen die Autoren eine Played-by-Relation, um eine (Instanz einer) Rolle ihrem Rollenspieler zuzuordnen. Anders als die meisten anderen Arbeiten dieser Kategorie (mit Ausnahme von [Kristensen 1995; Kristensen & Østerbye 1996]) wird aber nicht nur das Verhältnis von Rolle zu Rollenspieler, sondern auch das von Rolle zu Rollenbesitzer (role owner) berücksichtigt. So hat z. B. die Rolle eines Angestellten neben der Person, die sie spielt, auch eine Firma, die die Attribute dieser Rolle definiert, und diese Definition hat über einen Wechsel der rollenspielenden Person hinweg Bestand.[71]

Eine Rolle in DOOR wird als eine Brücke (besser: als ein Brückenpfeiler) der Relation zwischen Rollenspieler und Rollenbesitzer verstanden, deren Bestimmung es ist, die Relation von der Besetzung der mit dem Relationsende verbundenen Rolle unabhängig zu machen. So kann eine Firma für einen Angestellten ein Gehalt angeben, ohne sich dabei auf einen konkreten Rollenspieler beziehen zu müssen. Die Rollen DOORs ähneln damit in gewisser Weise den Rollen des Relationship-Dienstes von CORBA [OMG 1994; Orfali et al. 1998] (s. d.). Allerdings, und das wird in DOOR vernachlässigt, spielt der Rollenbesitzer in der Relation zum Rollenspieler auch eine Rolle, im obigen Beispiel die des Arbeitgebers, die aus der Sicht der Person austauschbar ist. Eine Relation

[71] Der Wechsel der Rollenspieler in einer Kollaboration ist auch in Andersens Formalisierung der Rollenmodelle von OORAM ein Thema [1997]; in LODWICK hingegen wird der Wechsel der Teilnehmer einer Assoziation, da diese keine von den Teilnehmern unabhängige Identität haben, durch eine neue Assoziation modelliert.

müßte also schon an allen Enden jeweils eine Rolle als Brücke haben (wie das ja auch in CORBA der Fall ist).

Rollen in ADOME

Li und Lochovskys Advanced Object Modeling Environment (ADOME) [1998] verwendet Rollen als Bindeglied zwischen den statisch typisierten Objekten einer objektorientierten Datenbank und den Regeln einer Wissensbasis. Als vorteilhaft erweist sich dabei die Möglichkeit der dynamischen Bindung von Objekten an Regeln, die über den Umweg der Rollen ermöglicht wird. Zusammen mit jeder Rolle sind zudem Zustände und Zustandsübergänge definiert. Außerdem können Rollen in einer Rollenhierarchie angeordnet werden, in der Eigenschaften selektiv, d. h. per Import/Export deklarierbar, vererbt werden.

Die dynamische Verknüpfung von Rollen und Klassen (als Typen der Datenbank) erfolgt in ADOME im Rahmen der Rollendefinition durch Auflistung der Klassen, deren Instanzen die Rollen füllen können. Die Alternative, nämlich die Rollen, die die Instanzen einer Klasse spielen können, als Eigenschaft der Klasse zu definieren, wurde aus praktischen Gründen verworfen; sie hätte einen Eingriff in die Definition bestehender Datenbankschemata bedeutet und wäre damit einer einfachen Integration bestehender Datenbanken im Wege gestanden.

ADOME ist ein vorwiegend praktisch orientierter Ansatz zur Integration von bestehenden Datenbasen mit deduktiven Komponenten. Die formale Definition von Rollen deckt sich weitgehend mit der von Wieringa et al. [1995]; es wurden aber auch Aspekte der Objektspezialisierung [Sciore 1989] (s. o.) übernommen.

3.3.4 Rollen in der objektorientierten Softwaremodellierung

Während in den Anfängen der objektorientierten Modellierung Rollen noch eine untergeordnete Rolle spielten, sind sie in einigen gegenwärtigen Ansätzen zu einem zentralen Modellierungskonzept erhoben worden [Reenskaug et al. 1992; 1996; D'Souza & Wills 1998; Coad & Mayfield 1999].

Rollen in OMT

Wie bereits bei Codd [1970] ist in OMT eine Rolle schlichtweg als ein Ende einer Assoziation (d. i. eine Relation) definiert [Rumbaugh et al. 1991, S. 34].

Der Rollenname identifiziert das Ende eindeutig. Dies ist insbesondere dann von praktischer Bedeutung, wenn die Assoziation Objekte gleichen Typs miteinander verbindet, also die Stelle nicht aufgrund der Klasse der Instanz identifiziert werden kann.

Während der Name einer Assoziation diese als ein eigenständiges Konstrukt Modellelement identifiziert, gibt ein gut gewählter Rollenname die Funktion des durch ihn bezeichneten Objektes aus der Sicht des anderen wieder. Er dient damit zum einen der Navigation entlang der Assoziationspfade, zum anderen trägt er zum inhaltlichen Verständnis der Assoziation durch den Betrachter bei.

Rollen bei Booch

Auch Booch versteht eine Rolle als die Stelle einer Relation, an der ein Objekt einem bestimmten, durch die Relation gegebenen Zweck dient [Booch 1994, S. 192, S. 518]. Zugleich ist eine Rolle bei Booch jedoch eine Eigenschaft eines Objektes: Sie bezeichnet einen Ausschnitt seines Protokolls und des damit verbundenen Verhaltens und ist somit eine Art partielle Spezifikation des Objekts. Die Rolle definiert damit einen Vertrag (contract) zwischen einem Objekt und seinen Klienten, die Aufgaben (responsibilities) eines Objektes werden seinen Rollen zugeordnet [Booch 1994, S. 90].

Für Booch hat die Rolle jedoch noch eine andere, grundsätzlichere Bedeutung: Für ihn beginnt die Analyse eines Problems häufig mit der Untersuchung der verschiedenen Rollen, die ein Objekt spielen kann. Während des Designs werden den Rollen dann einzelne Operationen zugeordnet, die wiederum die Aufgaben des Objektes erfüllen müssen [S. 240].

Rollen in OORAM

Nach der OORAM (Object oriented role analysis and modeling)-Methode wird zunächst die für andere Ansätze grundlegende Unterscheidung zwischen Klasse und Objekt aus der Modellierung auf die Ebene der Implementierung verbannt. Dies geschieht der Einsicht folgend, daß nach den gängigen Ansätzen Klassen (über Klassendiagramme) mit statischen, Objekte (über Interaktions- wie Sequenz- oder Kollaborationsdiagramme) hingegen mit dynamischen Aspekten der Modellierung verbunden sind, die Trennung des Modellierungsprozesses in Statik und Dynamik durch diese Zweiteilung also manifestiert wird, obwohl der objektorientierte Ansatz eigentlich auf der Modellierung interessierender Phänomene als eine Struktur interagierender Objekte basiert, also Statik und Dynamik vereint. [Reenskaug et al. 1992; 1996]

Ein Rollenmodell ist nach OORAM die Beschreibung eines Teils des betrachteten Systems, der aus einem gegebenen Blickwinkel eine Einheit darstellt. Es besteht aus einer Anzahl Rollen, die jeweils durch ihre Attribute und Operationen in Interaktion (Kollaboration) mit den anderen Rollen des Modells beschrieben werden. Ein Rollenmodell legt zunächst nicht fest, welche Objekte die Rollen spielen: Eine Rolle kann insbesondere von Objekten verschiedener Klassen gespielt werden, und ein Objekt kann mehrere oder sogar alle Rollen eines Rollenmodells spielen. Jede Rolle spezifiziert einen Objekttyp (object type) und Klassen implementieren einen oder mehrere solche Objekttypen [Reenskaug et al. 1996, S. 14]. Damit ist in OORAM eine Rolle die (idealisierte) Darstellung eines Objektes im Kontext eines Rollenmodells, und die Eigenschaften der Rolle stellen den Teil der Eigenschaften des Objektes dar, der im Rahmen des Rollenmodells von Interesse ist. Ein Objekt wird dann durch die Summe seiner Rollen vollständig beschrieben [Reenskaug et al. 1996, S. 59].

Die verschiedenen Rollenmodelle eines Systems werden in OORAM durch Synthese zusammengeführt. Dabei fallen gleiche Rollen verschiedener Rollenmodelle zusammen. Eine Verbindung zwischen verschiedenen Rollen ergibt sich zudem, wenn diese Rollen von demselben Objekt gespielt werden (so daß es auch eine Klasse geben muß, die alle dazugehörigen Objekttypen implementiert). Der Übergang von einer Rolle zur anderen erfolgt, wenn ein Objekt in einer Rolle eine Botschaft empfängt und daraufhin Botschaften in einer anderen Rolle, die es ebenfalls spielt, versendet.

Andersen formalisiert in seiner Dissertation die Rollenmodelle von OORAM, vor allem, um sein Gerüst formaler Verhaltensbeschreibungen mittels endlicher Automaten und deren Komposition im Zuge der Synthese von Modellen untersuchen zu können [1997]. Dazu trifft er eine Reihe von Festlegungen. So gibt es z. B. in einem Rollenmodell lediglich Pfade, keine benannten Relationen, entlang derer die Kommunikation der kollaborierenden Objekte verläuft. Außerdem ist jede Rolle eindeutig, d. h., sie darf in einem Rollenmodell nur einmal und nicht in mehreren Rollenmodellen zugleich vorkommen. Diese Festlegungen finden sich in der Arbeit von Riehle et al. wieder (s. u.); ein Vergleich mit den Definitionen von LODWICK erfolgt dort.

Rollen in UML

In der gegenwärtig vorliegenden Version 1.3 [OMG 1999] kennt die Spezifikation von UML neben der üblichen metasprachlichen vier weitere Verwendungen des Rollenbegriffs:

1. Rollen als Enden von Assoziationen,
2. Rollen als Teilnehmer an einer Kollaboration,
3. Rollen von Akteuren (in einem Use-case-Diagramm) und
4. Rollen zum Zweck der dynamischen Klassifikation.

Die erste Verwendung ist die traditionelle, die sich als das Erbe des relationalen Modells über das Entity-Relationship-Modell, das Basis der statischen Modellierung beinahe jeder objektorientierten Analyse- und Designmethode ist [Wieringa 1998], auch in UML eingeschlichen hat: Der Rollenname bezeichnet die Stelle einer Relation oder, im UML-Jargon, das Ende einer Assoziation, innerhalb derselben eindeutig. Ein Rollenname ist insbesondere dann angebracht, wenn dieselbe Klasse mehrfach an einer Relation beteiligt ist, so daß die verschiedenen Vorkommen der Klasse durch eindeutige Benennung unterschieden werden müssen.

Im UML-Metamodell ist der Rollenname nichts anderes als das Attribut *name* der Klasse *AssociationEnd* [OMG 1999, § 2.5.2, S. 2-22]. In der OBJECT CONSTRAINT LANGUAGE (OCL) von UML wird der Rollenname des gegenüberliegenden Endes einer (zweistelligen) Assoziation zum Navigieren verwendet; da er obendrein eindeutig in Bezug auf die Quelle sein muß, spricht man in UML auch von einem Pseudoattribut.[72] Noch deutlicher wird die Verwandtschaft von Rollenname und Attribut in UML bei Kompositionen, die nicht nur wie üblich als spezielle Assoziationen, sondern auch als geschachtelte Klassenboxen dargestellt werden können, wobei dann der Rollenname dem Klassenname eines Teils durch einen Doppelpunkt getrennt vorangestellt wird [OMG 1999, § 3.47.2, S. 3-75].

Das UML-Metamodell sieht neben der Angabe eines Rollennamens für ein *AssociationEnd* vor allem die Angabe eines Classifiers vor (über ein Pseudoattribut mit Namen *type*), die sich in einem konkreten Modell in der Angabe des mit dem Assoziationsende verbundenen konkreten Classifiers wiederfindet. Darüber hinaus ist im Metamodell jedes *AssociationEnd* über eine Relation mit Namen *specification* mit einer (möglicherweise leeren) Menge von Classifiern verbunden, die die Menge der Operationen spezifizieren, die jede an dem Assoziationsende auftretende Instanz zur Verfügung stellen muß, wenn sie über das Assoziationsende angesprochen wird [OMG 1999, § 2.5.2, S. 2-22], und die

[72] Tatsächlich verwendet die UML-Spezifikation gelegentlich den Begriff Rolle, wo eigentlich Assoziation oder Attribut stehen müßte, so z. B. bei der Beschreibung der Abbildung der Statechart-Diagramme auf endliche Automaten [OMG 1999a, § 9]. Es steht dies vermutlich in der Tradition von KL-ONE, wo Attribute allgemein Rollen heißen (s. Abschnitt 3.3.1).

zugleich den Zugriff auf das Objekt entsprechend beschränkt [OMG 1999, § 3.42.2, S. 3-66]. In einem konkreten UML-Modell wird diese Spezifikation durch einen (selten anzutreffenden) sog. Interface specifier vorgenommen.[73]

Die zweite Verwendung von Rollen ist die im Kontext von Kollaborationen. Die Spezifikation einer Kollaboration in UML (genauer: Kollaboration auf Spezifikationsebene) umfaßt auch Projektionen oder Sichten genannte Einschränkungen (der Intensionen, nicht der Extension) der Classifier, Assoziationen und Assoziationsenden, die an der Kollaboration beteiligt sind. So spezifiziert eine Kollaboration z. B. nur die Eigenschaften der Classifier, die deren Instanzen haben müssen, damit sie an der Kollaboration partizipieren können [OMG 1999, § 2.10.1, S. 2-103]. Dabei unterscheidet sich die Motivation für die Einführung von Rollen zunächst nicht von der der ersten Verwendung: „The same Classifier or Association can appear in several Collaborations, and several times in one Collaboration, each time in a different role." Kollaborationsrollen unterscheiden sich aber sehr wohl von Rollennamen: „In each appearance it is specified which of the properties of the Classifier or the Association are needed in that particular usage. These properties are a subset of all the properties of that Classifier or Association."[74] Die Rollen legen also die benötigten Eigenschaften fest; sie sind damit (wie in OORAM) partielle Spezifikationen der beteiligten Akteure.

Aufschlußreich ist wiederum ein Blick ins UML-Metamodell [OMG 1999, Abb. 2.17, S. 2-104]. Die Metaklassen *ClassifierRole*, *AssociationRole*, und *AssociationEndRole* sind jeweils Spezialisierungen von *Classifier*, *Association* und *AssociationEnd*. Gleichzeitig gibt das Metamodell vor, daß jede konkrete Classifier-Rolle, jede Assoziations- und jede Assoziationsenderolle eines Modells jeweils eine oder mehrere konkrete Classifier, Assoziationen bzw. Assoziationsenden zur Basis (spezifiziert über die Relation *base*) haben kann, nämlich genau die, von denen sie jeweils eine Projektion ist. Die Relation *base* ist also eigentlich eine Art Spezialisierungsrelation mit der Rolle als Generalisie-

[73] Eine explizite Überprüfung auf Überlappung von *type* und *specification* bzw. von Classifier und Interface specifier eines Assoziationsendes ist übrigens nicht beschrieben, obwohl sich hieraus Inkonsistenzen ergeben können. Statt dessen wird auf eine gewisse Parallelität zwischen Rollenname und Interface specifier einerseits und den Classifier-Rollen in Kollaborationen andererseits hingewiesen; mehr dazu folgt unten.

[74] Bemerkenswert an dieser Beobachtung ist auch, daß sie sich nicht nur auf Classifier, sondern auch auf Assoziationen erstreckt. Dazu später mehr. Wichtig ist schon hier der Hinweis darauf, das es um eine Teilmenge der Intension, nicht der Extension geht.

rung und der Basis als Spezialisierung, allerdings ohne daß dies in der UML-Spezifikation so dargestellt würde.[75, 76] Statt dessen liest man:

> „*Different instances may play the same role but in different instances of the collaboration. Since all these instances play the same role, they must all conform to the classifier role specifying the role. Thus, they are often instances of the base classifier of the classifier role, or one of its descendants. However, since the only requirement on conforming instances is that they must offer operations according to the classifier role, as well as support attribute links corresponding to the attributes specified by the classifier role, and links corresponding to the association roles connected to the classifier role, they may be instances of any classifier meeting this requirement.*“ [OMG 1999, § 2.10.4, S. 2-113]

Die Angabe einer Basis hat also lediglich eine indirekte Auswirkung auf den Typ der Instanzen, die die Rolle spielen, nämlich die, daß die Classifier-Rolle eine Projektion ihrer Basen ist, die Intension der Rolle denen ihrer Basen also nicht widersprechen darf.

Zwischen der Verwendung von Rollen als Enden einer Assoziation (die erste Verwendung) und als Akteure einer Kollaboration (die zweite) besteht eine gewisse Parallelität, die immer dann ins Auge fällt, wenn Assoziationsenden neben Rollennamen auch mit Interface specifiers versehen werden: „The use of a rolename and interface specifier are equivalent to creating a small collaboration that includes just an association and two roles, whose structure is defined by the rolename and attached classifier on the original association.“ [OMG 1999, § 3.42.2, S. 3-66] Allerdings wird diese Parallelität aus dem UML-Metamodell trotz seiner symmetrischen Struktur nicht ersichtlich. Vielmehr ist das, was (in einem Klassendiagramm) für das durch den Rollennamen bezeichnete Assoziationsende der Interface specifier ist, (in einem Kollaborationsdiagramm) für die Classifier-Rolle am Ende der zur Assoziation gehörigen Assoziationsrolle die Liste der Base classifer.

---

[75] Man beachte, daß *base* hier genau gegensinnig zur üblichen Verwendung der Begriffe Base class/Derived class eingesetzt wird: Die Rolle ist die Base class und die Basen sind die Derived classes.

[76] Im Gegensatz zur Generalisierung ist die Rolle zunächst auf den Kontext der Kollaboration beschränkt; sie vereint in der Regel Typen, die gemeinhin nicht als einer eigenen Generalisierung würdig erschienen. In der Praxis drückt sich das so aus, daß die unter die Rolle fallenden Klassen ihre Methoden in der Regel nicht gleich implementieren; sie sind eben nicht auf natürliche Weise miteinander verwandt.

Nun noch der oben angekündigte Kommentar zur Motivation der Assoziationsrollen:

> *„Each association role represents the usage of an association in the collaboration, and it is defined between the classifier roles that represent the associated classifiers. The represented association is called the base association of the association role. As the association roles specify a particular usage of an association in a specific collaboration, all constraints expressed by the association ends are not necessarily required to be fulfilled in the specified usage.“* [OMG 1999, § 2.10.4, S. 2-112]

Als Beispiel für die Abänderung der Constraints wird das Herabsetzen der Kardinalitäten an den Assoziationsenden genannt: Im Kontext einer Kollaboration stehen u. U. weniger Instanzen miteinander in Beziehung, als dies für die Assoziation frei von jedem Kontext gesagt werden kann. Dies ist intuitiv wohl richtig, aber gegen die Feststellung, daß die Classifier-Rollen einer Kollaboration allgemeiner sind als ihre Basen, also prinzipiell mehr Instanzen umfassen als jede Classifier-Basis für sich. Tatsächlich scheint hier eine Verquickung von Kardinalitäten als Mengenbegrenzung für eine Assoziation und als Mengenbegrenzung für Kollaborationspartner zu bestehen: Eine Instanz muß ja nicht mit allen Instanzen, mit denen sie überhaupt in einer bestimmten Beziehung steht, innerhalb einer Kollaboration auch zusammenarbeiten.

Die dritte Verwendung des Rollenbegriffs in UML trägt dem Umstand Rechnung, daß ein und derselbe Akteur in verschiedenen Use-case-Diagrammen auftreten kann und dort jeweils verschiedene Rollen spielt: „An actor may be considered to play a separate role with regard to each use case with which it communicates.“ [OMG 1999, § 3.55.1, S. 3-92] Dies entspricht einer Übertragung der ersten und zweiten Verwendung auf die Modellierung mit Use cases, wenn man Akteure als über Assoziationen mit ihren Use cases verbunden und jedes Auftreten eines Akteurs in einer solchen Assoziation als eine Rolle des Akteurs betrachtet, und ist insofern gerechtfertigt, als Akteure (und Use cases) spezielle Classifier sind [Rumbaugh et al. 1998, S. 43].

Die vierte und letzte Verwendung wird lediglich in einem Satz angedeutet und ist wohl als ein Zugeständnis an eine in der Literatur weit verbreitete Auffassung von Rollen zu werten: „A Type may be used [...] to characterize a changeable role that an object may adopt and later abandon.“ [OMG 1999, § 3.27.1, S. 3-46].

Das Rollenmodell von Riehle et al.

Das Rollenmodell von Riehle et al. [Riehle & Gross 1998; Riehle 2000] ist stark an das von OORAM [Reenskaug et al. 1996] angelehnt. Anders als in OORAM bewegen sich die Autoren aber nicht erst auf der Instanzebene, sondern von vornherein auf der Ebene der Klassen und Rollentypen. Dabei nennt jede Klasse eine Anzahl Rollentypen, wobei ein Rollentyp für eine Sicht auf die Klasse aus der Perspektive einer anderen steht. Die Rollentypen einer Klasse spezifizieren jeweils einen Teil des Verhaltens der Objekte dieser Klasse; sie sind also partielle Spezifikationen der Klasse.

Die Rollen, die eine Instanz einer Klasse gleichzeitig spielen kann, können bei Riehle et al. in bestimmte Abhängigkeiten (sog. Role constraints) zueinander gesetzt werden. Zu diesen Abhängigkeiten zählt jedoch nicht, ob ein Objekt dieselbe Rolle mehrfach spielen kann (und für den Fall daß, wie oft). Außerdem werden Rollen, obwohl sie Typen sind, nicht in einer Subsumtionshierarchie angeordnet. Jedoch haben die Rollenabhängigkeiten unter anderem einen Einfluß darauf, wie Rollentypen auf die Klassenhierarchie aufgeteilt werden können.

Ganz wie in OORAM werden Kollaborationen zunächst als reine Rollenmodelle, also ohne Klassen, dargestellt. Die vorherrschende Beziehung zwischen den Rollen ist die der Verwendung (Uses-Abhängigkeit in UML). Rollenmodelle können kombiniert werden; da aber alle Rollen global eindeutig sind (also keine zwei Rollenmodelle dieselbe Rolle enthalten), ist die einzige Verbindung zwischen den Bestandteilen des kombinierten Modells die der Rollenabhängigkeiten (die durch die Kollaborationen vorgegebenen Beziehungen zählen nicht dazu).

Ein Klassenmodell ist bei Riehle et al. ein Modell, in dem Klassen neben der Klassenhierarchie über ihre Rollen und deren Verwendungsbeziehung verbunden werden. Ein Klassenmodell verbindet somit (über gemeinsame Klassen) mehrere Rollenmodelle. Rollen, die unbesetzt bleiben, sind, ähnlich wie bei einem Puzzle, die Stellen, an denen andere Klassen ergänzt werden müssen. Klassen- und Rollenmodelle eignen sich daher ganz besonders zur Spezifikation von objektorientierten Frameworks (s. Abschnitt 3.3.5).

Der Modellierungsansatz von Riehle et al. unterscheidet sich von dem LODWICKs vor allem in den folgenden Punkten:

1. Rollen sind nicht an Relationen gebunden, sondern an Kollaborationen.

2. Rollen werden immer lokal zu einem Rollenmodell (stehend für eine Kollaboration) definiert und sind global eindeutig. Zwei verschiedene Kollaborationen (und damit auch die Relationen aus diesen) können also nicht dieselbe Rolle verwenden.
3. Rollen sind zwar auch hier Typen, bilden aber keine Typhierarchien, sondern werden statt dessen durch Rollenabhängigkeiten ins Verhältnis gesetzt [Riehle 2000]. Diese müssen bei der Synthese eines Klassenmodells berücksichtigt werden, erlauben dabei aber gewisse Freiheiten in der Umsetzung.

Rollen als Qua types

Eine interessante Variante des Rollenbegriffs findet man bei Bock und Odell [1998]. Sie definieren als Rollentyp (role type oder current type) in Abgrenzung vom deklarierten Typ (allowed type) einer Stelle einer Relation die Menge aller Objekte, die augenblicklich in der Relation auftreten. Da die Extension eines Rollentyps somit nicht größer werden kann als die seines deklarierten Typs, ist für Bock und Odell der Rollentyp ein Subtyp. Da zudem alle Elemente eines Rollentyps mit mindestens einem Element eines anderen Rollentyps in Verbindung stehen müssen (denn das ist ja die Definitionen des Rollentyps), wird eine Null in der unteren Schranke einer Kardinalität bei Rollentypen stets durch eine Eins ersetzt.

Die Rollentypen von Bock und Odell beziehen ihren praktischen Nutzen aus der Tatsache, daß ihnen jeweils das rollenspezifische Verhalten zugeordnet werden kann, das sonst zusammen im deklarierten Typ als gemeinsamen Supertyp aller durch Relationen mit diesem Typ definierten Rollentypen angesammelt würde. Auch führen Rollentypen in Aggregationen zu dem interessanten Effekt, daß die Teile aus einer Teil-Ganzes-Beziehung automatisch danach klassifiziert werden, von Objekten welchen Typs sie ein Teil sind. Auf der konzeptuellen Seite dient die Einteilung in deklarierte und Rollentypen vor allem dazu, zwischen dem, was möglich (Role type) ist und dem, was tatsächlich ist (Allowed type), zu vermitteln. So würde z. B. die Deklaration der Anstellungsbeziehung nur auf Arbeitnehmer und Arbeitgeber nicht nahelegen, daß Personen, die nicht angestellt sind, durchaus noch angestellt werden können.

Das Problem des Ansatzes von Bock und Odell ist, daß Statik und Dynamik eines Modells durcheinander geworfen werden. Die Rollentypen sind über die dynamische Extension einer Relation definiert, während die deklarierten Typen den üblichen statischen Typen entsprechen. Bezüglich der Auffassung von

Rollentypen als Subtypen ihrer deklarierten Typen gilt das bereits in Abschnitt 3.1 gesagte; man vgl. dazu auch die Diskussion in den Abschnitten 3.4.2 und 3.4.3.

Rollen in der OPEN modelling language (OML)

In OPEN ist mit CIRT (für Class, Instance, Role und Type) eine Meta-Superklasse definiert, deren Instanzen in einem Modell von der Unterscheidung auch zwischen Klassen, deren Instanzen und den Rollen, die sie spielen, abstrahieren. Etwas mißverständlich ist die Darstellung von Rollen als partielle Objekte (partial objects) [Firesmith & Henderson-Sellers 1998; Henderson-Sellers et al. 1998], die sowohl auf die Auffassung von Rollen als partielle Spezifikationen als auch auf die Aufteilung eines Objekts der Realität in mehrere (dann in der Tat partielle) Objekte im Modell zutreffen kann. Letzteres ist ein Relikt aus MOSES [Renouf & Henderson-Sellers 1995] und hat den Charakter eines Entwurfsmusters (s. Abschnitt 3.3.5). Ersteres hingegen wird in OPEN vor allem dadurch abgedeckt, daß mit einer Instanz von CIRT schließlich offengelassen wird, ob es sich dabei um eine Rolle handelt oder ein anderes Modellelement handelt. Da OPEN sich selbst eher als eine Sammlung von Techniken versteht, wird hier kein eigenständiges Rollenkonzept propagiert, sondern verschiedene aus der Literatur bekannte Ansätze zur Auflösung von Rollen angeboten [Henderson-Sellers et al. 1998, Anhang E].

Eine ausgesprochen interessante Hypothese stellt Henderson-Sellers im Zusammenhang mit der Reifizierung von Assoziationen auf [1998]: Demnach steht eine Klasse, wenn sie eine Assoziation repräsentiert, immer für eine Rolle und nicht für einen (natürlichen) Typ. Ein Indiz für die Richtigkeit dieser These findet man in Abschnitt 4.3.2.

Rollen in CATALYSIS

Die CATALYSIS-Methode von D'Souza und Wills [1998] propagiert Rollen als Metapher für zwei verschiedene Phänomene: für komplexe Objekte, die verschiedene Rollen ausfüllen können (Role delegation), und zur Entkopplung zwischen kollaborierenden Objekten (Role decoupling). Ersteres entspricht im wesentlichen der Darstellung eines Objektes durch mehrere Instanzen, wie es auch von einigen anderen Autoren (z. B. Coad und Mayfield [1999]) gesehen wird. Letzterem liegt die Beobachtung zugrunde, daß Klienten in der Regel nur einen Teil der Eigenschaften ihrer Kollaborateure benötigen, so daß die Bindung an eine bestimmte Klasse unnötig ist, zumal damit auch die Eigenschaften

der Kollaborateure festgeschrieben werden, die für die Kollaborateure gar nicht von Bedeutung sind. Die Entkopplung erfolgt dadurch, daß Variablen und formale Parameter als abstrakten Typs (abstrakte Klassen oder Interfaces) deklariert werden und somit mit Instanzen beliebigen Typs besetzt werden können, solange diese nur der Spezifikation des abstrakten Typs genügen.

Rollen bei Coad

Coad und Mayfield lassen ihre Designer zunächst die gewünschten Eigenschaften des zu entwickelnden Systems Punkt für Punkt unter anderem nach Rollen und Rollenspielern absuchen und so die Klassen eines initialen Modells identifizieren [Coad & Mayfield 1999]. Rollen sollen nicht als Unterklassen der Klassen, die sie spielen, modelliert werden [S. 54], sondern als separate Klassen, deren Objekte mit dem Rollenspieler mittels Komposition in Verbindung stehen. Gleichwohl zählen bei Coad Rollen neben Transaktionen und Dingen (things) zu den Klassen, die selbst typischerweise über Unterklassen spezialisiert werden.

Mit der Aufteilung von Rollenspieler und Rolle auf zwei Klassen geht eine Aufgabenteilung in Form von Delegation einher. Insbesondere Anfragen an ein Rolle, die durch den Rollenspieler abgehandelt werden können, werden an diesen weiterdelegiert. Um ein Objekt und seine Rollen austauschbar zu machen, d. h. insbesondere, um eine Interaktion davon unabhängig zu machen, ob es ein Objekt mit einem anderen Objekt oder mit einem seiner Rollen zu tun hat, sieht Coad vor, daß die Klasse und die dazugehörigen Rollen dasselbe Interface implementieren. Rollen werden also durch Interfaces nicht repräsentiert, sondern transparent gemacht.

Rollen in MON

Die MONASH OBJECT NOTATION (MON) [Maughan & Durnota 1995] ist eine graphische Notation für die formale Spezifikation objektorientierter Modelle. In MON sind Rollen spezielle Relationen zwischen Klassen und den Rollenklassen, die von (den Instanzen einer) Klasse gespielt werden können. Dabei können über eine Rollenrelation mehrere Rollenklassen mit einer Klasse verbunden werden; diese Rollenklassen sind dann einander gegenseitig ausschließend.

Durch Rollen werden in MON Klassen klassifiziert. Sie ähneln damit der Klassifikation, einem weiteren Relationstyp von MON, der im wesentlichen auf die sog. State-defined subtypes von SDM [Hammer & McLeod 1981] hinausläuft. Da Rollen von ihren Rollenspielerklassen erben, wird das Übernehmen einer

Rolle von Maughan und Durnota auch als eine Art dynamischer funktionaler Vererbung (dynamic functional inheritance) bezeichnet [1995]. Die genauen Mechanismen und insbesondere, wie die Rollen formalisiert werden, wird jedoch nicht deutlich.

Rollen bei Liao

Liao [1996] schlägt eine Technik vor, mit der sich aus während der Analysephase gewonnen heterogenen Klassen (Klassen mit Objekten verschiedenen Typs) durch Anwendung sogenannter Perspektiven (perspective application) homogene Unterklassen und Rollen ableiten lassen. Das Unterscheidungsmerkmal zwischen beiden ist, ob die Zuordnung von Objekten zu den abgeleiteten Klassen evolutionär (evolutionary), d. h. mit der Zeit veränderlich, oder fix (non-evolutionary) ist. In letzterem Fall handelt es sich um Klassen, die mit ihrer (heterogenen) Ursprungsklasse über die Unterklassenbeziehung in Verbindung stehen; in ersterem handelt es sich nicht um eigenständige Klassen, sondern um Abschnitte einer Klassendefinition, die die jeweilige Rolle repräsentieren. An dieser Stelle greift Liao den Ansatz von Pernici [1990] auf.

Liao unterstreicht mit seiner Auffassung, wie andere Autoren auch, den temporalen/temporären Charakter einer Rolle. Zur Ausprägung seines Rollenkonzeptes gilt allerdings das an anderer Stelle über ITHACA [Pernici 1990] Gesagte.

3.3.5 Rollen in objektorientiertem Design und Implementierung

Rollen sind zunächst kein eigenständiges Konstrukt gängiger objektorientierter Programmiersprachen.[77] Statt dessen werden sie in der Regel durch Entwurfsmuster simuliert. Gelegentlich werden jedoch Variablen als Rollen bezeichnet (z. B. [Hogg et al. 1992]), eine Sprechweise, die durchaus einen Sinn ergibt – schließlich haben Rollen als Typen nach Abschnitt 3.1 keine eigenen Instanzen, so daß nur Variablen, die ja vornehmlich im Kontext von Beziehungen auftreten, Rollenspieler repräsentieren können. Tatsächlich kann man, wenn eine Variable vom Typ eines Interfaces deklariert wurde, diese Variable als eine Rolle ihres Wertes (der Instanz, die den Wert bildet) interpretieren; mehr dazu im nächsten Kapitel.

[77] Man vergleiche dazu aber die Arbeiten zur Datenmodellierung aus Abschnitt 3.3.3, die zum Teil dahingehende Erweiterungen von Datenbankprogrammiersprachen vorschlagen.

Auch wenn eine Programmiersprache kein Rollenkonstrukt hat, kann man doch, wenn man den Definitionen LODWICKs folgt, Rollen als partielle Spezifikationen von Objekten oder Typen[78] auffassen. Partielle Spezifikationen aber sind abstrakte Klassen und die Interfaces von JAVA – nur werden diese eben nicht Rollen genannt (und für gewöhnlich auch nicht als Rollen verstanden). Auch dazu mehr im nächsten Kapitel.

Rollen in Entwurfsmustern und objektorientierten Frameworks

Entwurfsmuster (design patterns) stehen für das Bestreben, programmiersprachenunabhängig immer wiederkehrende Softwarestrukturen zu standardisieren und in einem Katalog verfügbar zu machen. Dabei fällt auf, daß die Strukturierungselemente der objektorientierten Programmierung, die Klassen, häufig nicht das richtige Konzept sind, um Entwurfsmuster darzustellen. Vielmehr treten in Entwurfsmustern typischerweise Rollen auf [Buschmann 1998; Riehle & Gross 1998]. Daß diese Rollen in den zu den Mustern gehörenden Strukturbeschreibungen häufig als abstrakte Klassen repräsentiert werden, liegt zum einen daran, daß sich der Rollenbegriff in der objektorientierten Softwaremodellierung noch nicht etablieren konnte (und schließlich immer noch eine einheitliche Rollennotation fehlt), und zum anderen daran, daß das Interface-Konzept von UML zu restriktiv ist, um Rollen in all ihren Facetten adäquat wiederzugeben (s. Abschnitt 4.2).

Den Entwurf und die Integration von objektorientierten Frameworks auf der Basis von Rollen schlagen Riehle und Gross [1998] vor. Ähnlich wie in OORAM [Reenskaug et al. 1996] werden zunächst Kollaborationen mittels eines Rollendiagramms modelliert. Diese Rollendiagramme können nach bestimmten Regeln kombiniert und über das Zuordnen von Rollen zu Klassen zu einem Klassendiagramm synthetisiert werden (s. Abschnitt 3.3.4).

Ein Framework ist nun ein durch ein Klassendiagramm mit offenen Rollen beschriebenes Stück Software. Um ein solches Framework zu benutzen, muß ein Anwendungsprogramm diese offenen Rollen implementieren, d. h., Klassen deklarieren, die diese Rollen füllen können. Die Kollaborationen beschreibenden Rollenmodelle, die das Framework beinhaltet, bilden also gewissermaßen die Brücken zwischen dem Framework und der Applikation, die das Framework verwendet [Riehle & Gross 1998]; die Rollen sind dabei die Anknüpfungspunkte (Plug points).

[78] nicht als partielle Objekte, wie beispielsweise in [Firesmith & Henderson-Sellers 1998;

Die Rolle als Entwurfsmuster

Rollen sind aber nicht nur Strukturelemente von Entwurfsmustern, sie sind, eben weil gängige Programmiersprachen kein Rollenkonstrukt aufweisen, selbst häufig der Gegenstand von Entwurfsmustern. Renouf und Henderson-Sellers [1995] haben für MOSES ein Entwurfsmuster vorgeschlagen, dessen Beteiligte (oder besser Rollen) sie *Role* und *Role Model* nennen. Dieses Muster trifft man in kleineren Abwandlungen häufiger an (z. B. in [Gamma 1996; Schoenfeld 1996; Bäumer et al. 1997; Fowler 1997b; Coad & Mayfield 1999]), hat aber immer dieselbe grundlegende Struktur: Eine Klasse, als Role Model[79] oder auch als Core [Bäumer et al. 1997] bezeichnet, liefert die zentrale Instanz, die Träger der Identität ist und zusätzlich die allen Rollen gemeinsamen Eigenschaften implementiert, während die andere, als *Role* bezeichnet, rollenspezifische Eigenschaften hinzufügt und die allgemeinen von ihrem Role model per Delegation erbt. Ein Objekt mit seinen Rollen besteht dann, genau wie beim Object slicing (s. u.), immer aus mehreren Instanzen und wird in Anlehnung an [Kristensen 1995] auch Subject [Bäumer et al. 1997] genannt.

Die Muster variieren nun vor allem darin, ob *Role* und *Role Model* (oder *Role* und *Core*) ein gemeinsames Interface implementieren [Bäumer et al. 1997] oder nicht [Renouf & Henderson-Sellers 1995; Fowler 1997b]. Daß sie das tun, ist sinnvoll, denn nur so ist es möglich, daß eine Rolle sein Role model bei strenger Typprüfung ersetzt, daß also beispielsweise ein Angestellter da auftritt, wo eine Person erwartet wird. Man kann sogar soweit gehen, die Rollen einer Klasse als Unterklassen der Klasse darzustellen, also z. B. die Klasse *Angestellter* als Unterklasse der (abstrakten) Klasse *Person*, deren personentypische Eigenschaften in einer konkreten Core-Unterklasse implementiert werden [Riehle & Gross 1998]. Dies steht jedoch im Gegensatz zu der Erkenntnis, daß Rollen keine Subtypen, sondern Supertypen sind.

Abgesehen davon, daß die Beteiligten eines Entwurfsmusters (hier *Role* und *Role Model*) in der Regel selbst Rollen sind [Buschmann 1998], so daß das Role-model-Muster eigentlich rekursiv auf sich selbst anwendbar sein müßte [Steimann 1999b], bereitet es ein grundlegendes konzeptuelles Problem: Ein Objekt der Realität wird, wenn es nur eine Rolle spielt, nicht mehr durch genau ein Objekt im Programm repräsentiert, sondern durch mehrere. Genau diesen

Henderson-Sellers 1998] formuliert

[79] Role model ist englisch für Vorbild und hier wohl in etwa so zu verstehen. In anderen Arbeiten steht Role model allerdings für ein Modell mit Rollen im Sinne von OORAM [Reenskaug et al. 1996], so z. B. in [Riehle & Gross 1998; Riehle 2000].

Mißstand haben aber Bachman und Daya schon [1977] zum Anlaß genommen, für die Datenmodellierung ein eigenständiges Rollenkonzept zu fordern.

Tatsächlich entspricht das Rollenentwurfsmuster strukturell der Nachbildung von Vererbung in einer relationalen Datenbank: Hat man Relationen wie *Angestellter*, *Kunde* etc. in seinem Datenbankschema, die jeweils alle Felder für die typischen Personenattribute enthalten (so daß man in einer objektorientierten Datenbank versucht wäre, diese Relationen als Subtypen von *Person* zu modellieren), so wird man aus Gründen derer Schemaänderungseffizienz eine Tabelle *Person* einführen und die anderen nur noch die zusätzlichen Attribute hinzufügen lassen.

Object slicing

Um dynamische Mehrfachklassifikation (und damit zumindest einen Teil des Rollenbegriffs) zu implementieren, schlagen Martin und Odell [1992; Odell 1992] eine Technik vor, die sie Object slicing nennen. Ein Objekt wird demnach durch ein oder mehrere Surrogate in der jeweiligen Klasse, die Slices, vertreten. Die verschiedenen Surrogate eines Objektes, die allesamt Instanzen von (geeigneten Unterklassen von) *Implementation Object* sind, werden durch einen gemeinsamen Repräsentanten, der Instanz der Klasse *Conceptual Object* ist, zusammengehalten [Martin & Odell 1992]. Heute würde man für die Definition von Object slices vermutlich ein Entwurfsmuster bemühen.

Martin und Odell schlagen selbst Alternativen zum Object slicing vor. So lassen sich dynamische Mehrfachklassifikationen von Instanzen einer Klasse zumindest im Ansatz durch Zustandsvariablen oder auch durch privat deklarierte Unterklassen der Klasse implementieren, die nur für diese sichtbar sind und deren Instanzen dann, ähnlich wie beim Object slicing, die dynamische Klassifikation repräsentieren [Martin & Odell 1992]. Gemeinsam ist jedoch allen aufgeführten Ansätzen, daß nicht klar wird, wie Polymorphie im Sinne einer Substituierbarkeit bei gleichzeitiger strenger Typprüfung gewährleistet werden soll.

Rollen bei Kilian

Kilian empfiehlt mit seinen Vorschlägen zur Entkopplung von Typen in objektorientierten Entwürfen und Komponentenbibliotheken u. a. die Verwendung von Rollentypen als reine Verhaltensspezifikationen [1991]. Ausgehend von der Beobachtung, daß eine Stelle einer Relation mit Objekten unterschiedlichen Typs besetzt werden kann, diskutiert er die Verwendung von Typdisjunktionen (type unions) und Generalisierung zur Bildung von Rollentypen, verwirft beide

aber mit der Begründung, daß damit die Menge der Typen, die für einen Rollentypen eintreten können, fixiert (und damit eine Kopplung gegeben) ist. Statt dessen fordert er eine neue Art von Rollentyp, dessen Kompatibilität mit einem (normalen) Typ erst bei Aufruf einer mit dem spezifizierten Verhalten verbundenen Operationen geprüft wird. Ein Konzept, wie dazu Programmiersprachen mit strenger Typprüfung geändert werden müßte, bleibt er bei seinem Vorschlag jedoch schuldig.

Rollen bei VanHilst & Notkin

VanHilst und Notkin schlagen ein auf Class templates basierendes Verfahren zur Implementierung von Rollen vor [1996]. Ihr Rollenbegriff ist im wesentlichen der von OORAM [Reenskaug et al. 1996] und verwandten Ansätzen, also der von einem Teilnehmer an einer Kollaboration. Insbesondere betrachten sie, wie andere auch, eine Rolle als eine Art partielle Spezifikation der rollenspielenden Klasse, die per Vererbung auf diese übertragen wird. Ihr Ansatz weicht aber in zwei Punkten wesentlich von denen anderer Autoren ab:

1. Rollen werden nicht als Interfaces (abstrakte Klassen) deklariert, sondern als Templates. Die Parameter dieser Templates stehen für die Klassen der Kollaborateure, die mit der Rollendefinition verbunden sind, sowie für eine anzugebende Superklasse.
2. Mehrere Rollen werden von einer Klasse nicht parallel geerbt, sondern seriell. Dazu gibt die Klasse bei der Instanziierung eines Templates eine Superklasse an, die selbst wieder die Instanziierung einer (anderen oder derselben) Rolle ist.

Diese Verwendung von Templates ermöglicht es insbesondere, die Rollen für eine Klasse individuell zusammenzustellen, ohne dabei auf Mehrfachvererbung zurückgreifen zu müssen. Allerdings müssen dafür immer auch eine Reihe von Zwischenklassen deklariert werden, die u. U. keine andere Verwendung finden. Dies kann man nun wieder auch über Interfaces haben. Schließlich ist eine Parametrisierung der Kollaborateure, die bei der Instanziierung einer Rolle aufgelöst werden muß, auch kein wesentlicher Vorteil; die Rollen (und damit partiellen Spezifikationen der Kollaborateure) einer Kollaboration stehen in der Regel mit der Spezifikation der Kollaboration fest und hängen nur gelegentlich von der jeweiligen Besetzung einer Rolle ab.

Rollen bei Mrva

In seiner Arbeit zum rollenzentrierten Entwurf (role-centered design) unterscheidet Mrva [1999] ausdrücklich zwischen Interfaces im JAVA-Stil und Rollen als einem darüber hinausgehenden Konstrukt. Er vertritt die Ansicht, daß Rollen den Objekten dynamisch (d. h. zur Laufzeit) und unabhängig von der Klassendefinition zugewiesen werden können müssen, wohingegen die Implementierung von Interfaces eine statisch festgelegte Verpflichtung der Klassen darstellt. Rollen bieten damit die Möglichkeit, Klassen in neue, bei ihrem Entwurf unvorhergesehene Kontexte einzubetten und erhöhen damit die Entkopplung von Entwürfen.

Rollenwechsel eines Objektes, so Mrva, können sowohl durch einen Wechsel der Perspektive durch den Betrachter, also den Objekten aus der Umgebung des Objektes, als auch durch eine Veränderung des Zustandes des internen Objektes erfolgen. Solche von Zustandswechseln abhängigen Rollen nennt Mrva Instanzrollen, im Gegensatz zu den abstrakteren Klassenrollen, die von Attributwerten unabhängig sind. Ziel eines Entwurfs, der die Weiterentwicklung und Wiederverwendung im Auge hat, sollte es sein, Instanzrollen zu vermeiden, da bei ihnen der Grad der Entkopplung niedriger ist.

Rollen in CORBA

Der CORBA-Relationship-Dienst [OMG 1994; Orfali et al. 1998] stellt eine Alternative zur Bildung von navigierbaren Objektverbünden über verzeigerte Objektstrukturen zur Verfügung. Anders als die Verzeigerung erlaubt er es, Objekte in Beziehung zu setzen, ohne die Objekte selbst zu manipulieren. Dadurch wird es möglich, daß Objekte, die vor der Definition einer Beziehung generiert wurden, noch nachträglich über diese Beziehung verbunden werden können.

Damit dies möglich ist, wird jedes Objekt in einer Beziehung des Relationship-Dienstes durch eine Rolle (role object) vertreten, wobei die Verbindung zwischen Rolle und Objekt aus einem Zeiger von der Rolle auf das Objekt besteht. Diese Rollen werden dann durch Relationen verbunden, und mehrere solche Relationen bilden einen navigierbaren Graphen, der wiederum als Objekt selbst Gegenstand von CORBA-Diensten sein kann.

Sowohl Rollen als auf Relationen sind Instanzen vollwertiger CORBA-Typen, die über CORBA-Interfacedeklarationen in der Interface Description Language (IDL) spezifiziert werden. Benutzer des Relationship-Dienstes können nach Bedarf Subtypen davon abgeleiteten und somit für Rollen und Relationen eige-

ne Namen, Attribute und Methoden deklarieren. Die Factory-Methoden zur Erzeugung von Rollen- und Relationsinstanzen überwachen die Einhaltung von mit den Typen definierten Typrestriktionen, so daß eine Relation nur die deklarierten Rollen an ihren Stellen hat und eine Rolle nur Objekte der deklarierten Typen repräsentiert. Rollenobjekte können ggf. Aufgaben an die von ihnen in der Rolle repräsentierten Objekte delegieren; eine partielle Spezifikation eines Objekttyps ist ein Rollentyp aber nicht.

3.4 Diskussion

Die Unterscheidung von Rollen und Typen ist in erster Linie semantisch motiviert. Guarino [1992] liefert dazu eine überaus hilfreiche ontologische Grundlage, die sowohl den dynamischen Aspekt von Rollen als auch deren Abhängigkeit von Relationen abdeckt (Abschnitte 1.2.2 und 3.3.1). Das Verdienst der Formalisierung des Rollenbegriffs in LODWICK (Abschnitt 3.2) besteht vor allem darin, daß sie diese ontologische Fassung umsetzt [Steimann 2000b], ohne daß die konventionelle, rollenlose Modellierung übermäßig umstrukturiert werden müßte – sie fügt den Rollenbegriff einfach als natürliche Brücke zwischen Relationen und Typen ein.

Auf der einen Seite steht der dynamisch-temporäre Charakter des so definierten Rollenkonzepts: Anders als ihre natürlichen Typen können Individuen ihre Rollen annehmen und wieder ablegen. Zwar hat auch das rollenlose FREGE dynamische Aspekte (im dynamischen Modell verändern sich die Extensionen der Typen und vor allem der Relationen ständig), doch jedes Individuum gehört, solange es existiert, immer demselben natürlichen Typ an. Die Klassifikation der Individuen durch ihre Typen ist also statisch. Nun wird in LODWICK zwar auch mit der Modellspezifikation vorgegeben, welche Rollen ein Individuum prinzipiell spielen kann[80], jedoch erfolgt die Zuordnung von Individuen zu ihren Rollen nicht mit ihrer Erzeugung (ihrem Eintritt in das dynamische Modell), sondern erst mit dem Eingehen entsprechender Beziehungen. Während das Eingehen von Beziehungen nichts mit dem natürlichen Typ eines Individuums zu tun hat, bestimmt es aber sehr wohl, welche Rollen es spielt, und weil die Beziehungen der Individuen während ihrer Existenz auch wechseln, ist auch die damit verbundene Klassifikation eine dynamische.

Damit ist bereits die andere Seite des Rollenbegriffs in LODWICK genannt: seine Abhängigkeit von Beziehungen. Ein Individuum, das in keiner Beziehung zu einem anderen steht, spielt auch keine Rolle. Doch die Abhängigkeit geht noch

[80] ein Umstand, der durchaus sinnvoll ist und einer dynamischen Klassifikation nicht im Wege steht; es darf z. B. ausgeschlossen werden, daß eine Person die Rolle eines Einrichtungsgegenstandes spielt

weiter: Im Gegensatz zu natürlichen Supertypen (Genera) gehören die in einer Rolle vereinigten Typen nicht auf natürliche Weise (aufgrund der gemeinsamen Abstammung) zusammen, sondern aufgrund der Tatsache, daß ihre Individuen alle dieselben Funktionen innerhalb einer Beziehung übernehmen können, und zwar genau die Funktionen, die die Rolle ausmachen. Die Beziehungen, in denen Rollen vorkommen, prägen diese also. Allerdings kann die Prägung wegen der heterogenen Zusammensetzung einer Rolle (als Vereinigung nicht verwandter Typen) nur das Was, nicht das Wie betreffen. Dieser Unterschied wird sich in der Praxis vor allem darin ausdrücken, daß die Umsetzung der mit den Rollen verbundenen Funktionen von Typ zu Typ verschieden ist, so daß diese (anders als bei den Genera) nicht vererbt werden kann. Rollen ähneln damit in frappierender Weise abstrakten Schnittstellenbeschreibungen (Interfaces); mehr dazu im nächsten Kapitel.

Obwohl diese Definition des Rollenbegriffs recht einleuchtend erscheint, gibt es, wie die Literaturbetrachtung des letzten Abschnitts gezeigt hat, eine Vielzahl anderer Definitionen, die zum Teil nur wenig bis überhaupt keinen Bezug auf die hier genannten Aspekte (oder aufeinander) nehmen. Trotzdem läßt sich die Mehrzahl der Arbeiten jeweils einer von vier grundsätzlichen Auffassungen zuzuordnen. Es sind dies die Auffassungen von:

1. Rollen als Bezeichnern von Stellen in Relationen,
2. Rollen als Generalisierungen und/oder Spezialisierungen,
3. Rollen als dynamischen Klassen und
4. Rollen als beigeordneten Instanzen.

Diese Auffassungen sollen nun der Reihe nach diskutiert werden.

3.4.1 Rollen als Stellenbezeichner

Die Benennung einer Stelle einer Relation [Codd 1970; Chen 1976] oder eines Prädikats [Fillmore 1971] durch einen Rollennamen ist gewissermaßen die minimale Verwendung des semantisch so reichen Rollenbegriffs. Auch wenn diese Verwendung außer acht läßt, daß die Rolle nicht nur für die Relation, sondern auch für die Individuen, die diese Rolle spielen können, eine prägende Bedeutung hat, so betont sie doch immerhin die Notwendigkeit einer Relation für das Bestehen einer Rolle, eine Eigenschaft, die die meisten anderen Definitionen vermissen lassen. Kein Student ohne Studium, kein Angestellter ohne Anstellung – es gibt eben keine Rolle ohne Relation.

Anmerkung: Die Formalisierung des Rollenbegriffs in LODWICK zählt im Prinzip zu den Ansätzen, die Rollen mit den Stellen der Relationen verbinden, geht jedoch über die bloße Benennung der Stellen hinaus, indem sie zugleich die (in der Linguistik als Selektionsbeschränkung bekannte) Funktion der Deklaration der Typen, deren Individuen die Stellen besetzen können, übernimmt. Ein Rolle (und nur eine Rolle) gibt an, Individuen welcher Typen an den mit der Rolle verbundenen Stellen der Relationen auftreten können – es gibt eben auch keine Relation ohne Rollen.

3.4.2 Rollen als Spezialisierung und/oder Generalisierung

Sobald Rollen nicht mehr nur als Namen, sondern selbst als Typen angesehen werden, muß ihr Verhältnis zu den natürlichen Typen geklärt werden. Anstatt wie in LODWICK eine neue Relation auf der Metaebene einzuführen, die das tut, setzen viele Autoren auf eine bereits bestehende: die Subtypenrelation. Damit aber müssen Rollen und Typen in einer Spezialisierungsbeziehung, in einer Generalisierungsbeziehung oder in einer Kombination aus beiden zueinander stehen.

Rollen als Spezialisierung

Die häufig anzutreffende Darstellung einer Rolle als Spezialisierung (z. B. in [Sowa 1984; 1988; Maier et al. 1985; Abiteboul & Hull 1987; Essink & Erhart 1991; Jungclaus et al. 1991; Jungclaus 1993; Ehrich et al. 1993; Elmasri & Navathe 1994]) und die Interpretation einer Spezialisierung als Teilmenge (Abschnitt 2.2.4) legen nahe, daß nur ein Teil der Individuen des übergeordneten Typs diese Rolle spielen können. Das widerspricht aber zumindest konzeptuell dem Verständnis von Spezies als den Typen, die die Individuen eines Modells stellen: Entweder können alle Individuen einer Spezies prinzipiell dieselbe Rolle spielen, oder es handelt sich nicht um Individuen derselben Spezies [Steimann 2001]. Sollen aber alle Objekte die Rolle spielen dürfen, müssen die Extensionen beider Typen gleich sein, so daß die Intension zwar zusätzliche Eigenschaften definieren kann, diese die Extension aber nicht weiter einschränken (sog. Konkretisierung, vgl. Abschnitt 2.2.4). Darüber hinaus schließt die Auffassung von Rollen als Spezialisierungen definitiv aus, daß ein und dieselbe Rolle von Individuen verschiedener (disjunkter) Typen gespielt wird [Steimann 1999b]. Gerade dies ist jedoch nicht nur der hier vertretenen Auffassung nach ein prägendes Element des Begriffs Rolle, und so ist wohl auch die im Kontext von Rollen als Spezialisierung vorgetragene Forderung nach einer zweiten Art von Rollen, nämlich nach Rollen als Generalisierung [Ehrich et al. 1993], zu verstehen.

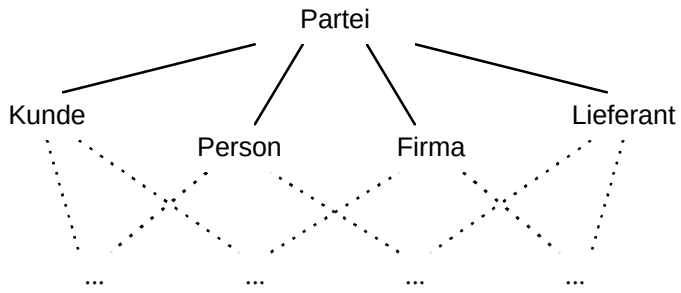

Abbildung 3.6: *Partei* generalisiert *Person* und *Firma*; *Kunde* und *Lieferant* sind Spezialisierungen davon, die als Rollen fungieren. Welche Typen allerdings Spezialisierungen und welche Generalisierungen sein sollen, ist hinterher nicht mehr zu erkennen. Außerdem muß eine Person als Kunde entweder zwei Typen haben, oder es muß einen Schnittyp *Person&Kunde* geben.

Tatsächlich liegt der Auffassung, Rollen seien Spezialisierungen, für gewöhnlich eine Beobachtung zugrunde, die dynamischer Natur ist. So ist es z. B. üblich, daß nur ein Teil aller Personen als Kunde auftreten kann, nämlich nur der, der geschäftsfähig ist. Anstatt nun die Rolle *Kunde* als eine Spezialisierung von Personen, die die Geschäftsfähigkeit implizit beinhaltet, darzustellen, könnte man genauso gut g*eschäftsfähige Person* als Subtyp von *Person* angeben und die Rolle *Kunde* darauf definieren. Geschäftsfähigkeit ist jedoch keine ontologisch rigide Eigenschaft [Guarino 1992; Guarino et al. 1994], d. h., sie kann von einem Objekt erworben und wieder abgelegt werden. *Geschäftsfähige Person* ist damit auch kein natürlicher Typ, sondern ein Zustand von *Person* (vgl. Abschnitt 3.4.3).

Rollen als spezialisierte Generalisierungen

Universeller als die Modellierung der Rollen eines Typs durch dessen Spezialisierung ist die Definitionen von Kategorien als Spezialisierung einer Generalisierung von Typen, wie sie z. B. von Elmasri et al. [1985] betrieben wurde. Durch die Generalisierung kann dem Umstand Rechnung getragen werden, daß Individuen grundverschiedener Typen ein und dieselbe Rolle spielen können; für die anschließende Spezialisierung gilt das oben Gesagte. Doch selbst wenn es sich bei den Spezialisierungen nur um Konkretisierungen handelt, entsteht durch die Kombination von Generalisierung und Spezialisierung ein weiteres Problem: Man muß dann entweder eine Mehrfachklassifikation der rollenspielenden Individuen erlauben (nämlich einmal durch ihren natürlichen Typ und einmal durch ihre Rolle) oder eine entsprechend große Menge von Schnittypen deklarieren, wie in Abbildung 3.6 angedeutet. Ersteres ist in der Regel aus

technischen Gründen nicht möglich, und letzteres wird von vielen Autoren aus Gründen der kombinatorischen Komplexität verworfen (z. B. [Sciore 1989; Richardson & Schwarz 1991; Odell 1992; Wieringa et al. 1995]).

Rollen als Generalisierung

Wie in Abschnitt 3.1 dargelegt, kommt nicht etwa die Spezialisierung, sondern die Generalisierung dem Konzept der Rolle am nächsten. Tatsächlich verbinden manche Autoren Vereinigungen von Typen mit den Stellen einer Relation, so z. B. Kent mit Domains [1978] oder Hainaut et al. mit ihren Multi-ET roles [1996]. Der Unterschied zur Generalisierung, die ja auch eine Vereinigung von Typen ist, ist in diesen Fällen nur schwer festzumachen.

Die Rollendefinition von LODWICK erweitert die Auffassung von Rollen als Generalisierungen um die (dynamische) Einschränkung der Extensionen dieser Generalisierungen auf die Individuen, die (aktuell) in mit den Rollen verbundenen Beziehungen zu anderen stehen. Dadurch wird der scheinbare Widerspruch zwischen der Definition von Rollen als Spezialisierung und als Generalisierung auf intuitiv einsichtige Weise aufgelöst.

3.4.3 Rollen als Mittel der dynamischen Klassifikation

Daß der Rollenbegriff etwas mit Zeit zu tun hat, wird von kaum jemandem bestritten.[81] Selbst Guarinos Unterscheidung von Rollen und Typen verwendet mit der Rigidität ein Kriterium, das auf die zeitliche Veränderlichkeit der Rollenzugehörigkeit abzielt. Es ist also nur folgerichtig, daß es eine ganze Reihe von Definitionen des Rollenbegriffs gibt, die sich an seiner dynamischen Natur festmachen.

Rolle als Zustand

Gelegentlich wird die Rolle eines Objekts mit seinem Zustand gleichgesetzt. So schreibt z. B. Booch zu Rollen unter anderem:

[81] Manche Autoren betonen anstelle des Temporalen das Modale des Rollenbegriffs, so z. B. Reimer [1991], Guarino [1992], Wieringa et al. [1995] und Bock und Odell [1998]. Daß es möglich ist, daß ein Individuum eine Rolle spielt, wird in LODWICK dadurch ausgedrückt, daß es einen möglichen Verlauf gibt, in dem das der Fall ist, was wiederum im statischen Modell (als Projektion des dynamischen) zeitunabhängig eingefangen ist.

> *„A bank account may be in good or in bad standing, and which role it is in affects the semantics of a withdrawal transaction.“* [1994, S. 90]

Ähnliche Aussagen machen andere auch, z. B. Su [1991] und Maughan & Durnota [1995]. Dabei wird geflissentlich ignoriert, daß das Verhalten eines Objektes immer von seinem Zustand abhängt, und zwar unabhängig davon, ob dieser Zustand als Rolle oder wie üblich als die Summe der momentanen Eigenschaften (Werte von Instanzvariablen) aufgefaßt wird. Tatsächlich wird durch den Zustand eher ein dynamischer Subtyp (in SDM ein sog. State-defined subtype [Hammer & McLeod 1981]; vgl. aber auch die dynamischen Klassen in [Wieringa et al. 1995]) denn eine Rolle definiert; der hat aber, wie bereits mehrfach bemerkt, nichts mit Rollen zu tun.

Gemessen an der Rollendefinition LODWICKs fehlt der Darstellung von Rollen als Zustand sowie allen vorigen, die Rollen nur als Generalisierungen und/oder Spezialisierungen hinstellen, der für den Rollenbegriff so wesentliche Kontext einer Relation – es handelt sich also eigentlich gar nicht um Rollen. Statt dessen bietet sich Guarinos in Abschnitt 3.3.1 dargestellte Basis der Definition natürlicher Typen und Rollen zur Bildung einer dritten Kategorie an: die all der Konzepte, die wie Rollen nicht semantisch rigide sind (d. h., die nicht zur Identität ihrer Instanzen beitragen) und die wie natürliche Konzepte unabhängig, also nicht durch eine Beziehung zu anderen Konzepten definiert sind. Die fehlende semantische Rigidität impliziert, daß sich diese Konzepte, wie Rollen, aus anderen, semantisch rigiden rekrutieren müssen. Das Konzept trägt also nicht zur Identität seiner Instanzen bei, vielmehr bestimmt es deren Attribute und Verhalten für die Dauer, die sie unter das Konzept fallen. Das Welpe sieht anders aus und hat ein anderes Verhalten als der ausgewachsene Hund, ohne daß dies etwas zu seiner Identität beitragen würde oder von anderen Instanzen abhinge. Entsprechendes gilt für das Kind, obwohl Kind zumindest im Deutschen zugleich Name einer Rolle, nämlich der im Eltern-Kind-Verhältnis, ist.[82]

Diese neben natürlichen Typen und Rollen dritte Kategorie von Konzepten könnte man in Anlehnung an die Literatur dynamische Klassen [Wieringa et al. 1995], Phasen [Ehrich et al. 1993], Modi [Taivalsaari 1993] oder, wie hier, Zu-

[82] Im Gegensatz zu dieser Doppelfunktion des Wortes Kind unterscheidet man im Deutschen (genau wie im Englischen) bei der geschlechtlichen Bezeichnung sehr wohl zwischen Sohn/Tochter (als Rolle) und Junge/Mädchen (als Altersgruppe), eine Unterscheidung, die im Ungarischen gar nicht und im Chinesischen bereits auf der Ebene von Kind gemacht wird [Mrva, persönliche Kommunikation].

stände nennen. Keinesfalls handelt es sich dabei jedoch um Rollen im eigentlichen Sinn; dazu fehlt ihnen einfach der Bezug.

Qua-definierte Rollen

Etwas anderes ist die Definition eines Rollentyps als der Teil der Individuen eines Typs, der aktuell in einer bestimmten Beziehungen mit anderen Individuen steht (in Anlehnung an den sog. Qua link [Bock & Odell 1998] aus KL-ONE [Brachman & Schmolze 1985]).[83] Auch hierbei wird eine Rolle als eine bestimmte Form der Spezialisierung (analog zu SDMs Existence subclass [Hammer & McLeod 1981; Snoeck & Dedene 1996]) verstanden, jedoch setzt die Definition immerhin die für den Rollenbegriff so wesentliche Forderung nach der Existenz einer Beziehung, die in den anderen Definitionen fehlt, voraus.

Ähnlich wie in LODWICK bestimmt bei qua-definierten Rollen die dynamische Extension einer Relation die dynamische Extension eines Rollentyps. Anders als in LODWICK wird dieser Typ jedoch als ein (dynamischer) Subtyp angesehen, und Individuen, die diese Rolle annehmen, migrieren von ihrem Typ zu diesem Subtyp. Dies führt jedoch strenggenommen bei der Typprüfung zu einem Problem: Dort, wo die Instanzen eines Rollentyps hingehören (also z. B. an eine Stelle einer Relation), darf nämlich erst gar keine Instanz des rollenspielenden Typs erscheinen, weil die Besetzung nicht typenrecht wäre. Die Migration muß also eigentlich vorher (und damit als expliziter Akt wie z. B. bei Papazoglou [1991; 1995]) erfolgen.

3.4.4 Rollen als beigeordnete Instanzen

Während also die Subtypenrelation der Spezialisierung bzw. Generalisierung vorbehalten bleiben sollte und sich, wie in Abschnitt 3.4.2 erörtert, nicht für die Modellierung von Rollen eignet, bietet sich zur Darstellung des Verhältnisses zwischen Rollen und Typen die Einführung einer neuen, einer Rollenspielerrelation an. Obwohl man eine solche Relation wie in LODWICK in Analogie zur Subtypenrelation auf der Menge der Typen definieren könnte, haben viele Autoren offenbar das einzelne Individuum und seine (individuellen) Rollen vor Augen und definieren diese Relation, anstatt als eine Metarelation, als eine vor-

[83] Eine ausführliche Definition des Qua link und damit verwandter Konzepte, die jedoch sehr KL-ONE spezifisch ist, findet man in [Freeman 1981].

definierte auf der Objektebene, wodurch das Verhältnis von Rolle zu Rollenspieler eine Sache zwischen Individuen wird.

So findet man in der Literatur die Deklaration von Funktionen oder Relationen wie *played-by* [Wieringa et al. 1995; Wong et al. 1997], *role-of* [Kappel & Schrefl 1996; Chu & Zhang 1997] oder *role* [Reimer 1985], die einer Instanz, dem Individuum in der Rolle, eine zweite, das rollenspielende Individuum, zuordnet. Ein Individuum, das eine Rolle spielen soll, wird in dieser Rolle also durch ein zweites, den Repräsentanten in der Rolle, vertreten, und die beiden sind durch eine der deklarierten Beziehung entsprechende Assoziation miteinander verbunden.

Diese einfache Festlegung hat mehrere attraktive Konsequenzen. Zunächst ist es so auch ohne Mehrfachklassifikation möglich, daß ein Individuum mehrere Rollen (einschließlich der gleichen mehrfach) spielt. Entsprechendes gilt für das Annehmen und Ablegen von Rollen, das nun auch ohne dynamische Klassifikation oder Instanzmigration vonstatten gehen kann. Dann hat jedes Individuum pro Rolle, die es spielt, automatisch einen eigenen Zustand, nämlich den des Repräsentanten in der Rolle. Und nicht zuletzt wird die Erscheinung des Individuums aus der Sicht des Kontexts der Rolle auf die Eigenschaften seines Repräsentanten in der Rolle eingeschränkt – inhärente Eigenschaften des Individuums kann der Repräsentant (über die sog. Delegation) gezielt sichtbar machen.

Der Preis für diese Annehmlichkeiten ist jedoch hoch: Anstatt mit einer Instanz hat man es jetzt mit mehreren verschiedenen zu tun, die alle ihre eigene Identität und ihren eigenen Typ haben und die allenfalls logisch eine Einheit bilden. In der Realität dagegen ist ein Individuum in einer Rolle immer noch dasselbe Individuum, nur daß es sich auf eine bestimmte, durch die Rolle vorgegebene Art gibt. Der eindeutige Zusammenhang von Realität und Modell ist mit der Aufteilung von Rollenspieler und Rolle in verschiedene Instanzen dahin, und es bedarf einer gehörigen Neuordnung insbesondere der Semantik der üblichen Modellierungsformalismen, um ihn wiederherzustellen. Meines Wissens hat sich bislang aber noch niemand diese Mühe gemacht, und so ist es dann auch nicht weiter verwunderlich, warum es zumindest auf Modellierungsebene bislang bei Ausarbeitungen mit Vorschlagscharakter geblieben ist.[84] Anders sieht es auf der Ebene objektorientierten Designs aus; mehr dazu gleich.

[84] Die einzige mir bekannte Ausnahme bildet hier die Arbeit von Wieringa et al. [1994; 1995], die aber bezeichnenderweise dem Individuum in einer Rolle ein von dem Individuum

Abgesehen von der konzeptuellen Schwäche der Modellierung von Rollen als beigeordnete Instanzen gibt es noch ein paar kleinere Probleme, die nicht richtig gelöst sind. Damit die Played-by-Funktion (oder wie auch immer sie genannt wird) universell einsetzbar ist, müßte sie entweder

- auf den beiden jeweils größten aller natürlichen und Rollentypen deklariert sein, womit aber ihr deklarativer Charakter verloren ginge (denn es würde durch die Deklaration nicht mehr klar, welche Typen welche Rollen spielen), oder
- für jede vorgesehene Rolle-Typ-Kombination einmal überladen werden.

Letzteres wird häufig zum Anlaß genommen, auszuschließen, daß Individuen verschiedenen Typs dieselbe Rolle spielen (dann müßte der Wertebereich der Played-by-Funktion eine Vereinigung verschiedener Typen sein), eine Einschränkung, die einem hinreichend allgemeinen Rollenkonzept nicht gerecht wird.

Schwerwiegender noch ist das Problem, daß die Played-by-Funktion auf einer Ebene mit den Relationen steht, die der Modellierer (als Benutzer der Modellierungssprache) selbst deklariert. Sie ist also eine von diesen „vordefinierten Relationen“ wie die Teil-Ganzes-Beziehung, die zwar auf einer Ebene mit den „benutzerdefinierten“ stehen, deren Semantik aber fest vorgegeben ist und die der Modellierer so hinzunehmen hat. Warum das ein Problem ist, zeigen die folgenden eingeschobenen Überlegungen.

Worin besteht denn eigentlich der Unterschied zwischen den Deklarationen

$$\textit{employed-by}: \textit{Person} \rightarrow \textit{Party}\ ,$$
$$\textit{played-by}: \textit{Student} \rightarrow \textit{Person}$$

und

$$\textit{Person} <_{NR} \textit{Student}$$

und gar

$$<_{NR} \subseteq N \times R\ ?$$

Der wesentliche Unterschied zwischen den ersten beiden ist, daß die Semantik von *employed-by* zunächst unbestimmt, wohingegen die von *played-by* fest vorgegeben ist. Beide Funktionen müssen jedoch vom Benutzer im Rahmen der Modellierung explizit deklariert werden. Dabei ist allerdings *played-by* für jede

selbst verschiedene Identität beimißt (und damit das sog. Zählproblem löst; s. Abschnitt 3.3.3).

weitere Rollenspielerdeklaration zu überladen (denn genau in diesen Überladungen steckt ja die Information, welcher Typ welche Rollen spielt), während *employed-by* von vornherein so gewählt werden wird, daß keine (oder so wenig wie möglich) Überladungen notwendig sind. Ein Vergleich mit *has-a* (als vordefinierte Relation mit der Semantik einer Aggregation; s. Abschnitt 2.2.2) drängt sich hier auf; der mit *is-a* hinkt hingegen, da *is-a* eine Relation zweiter Ordnung ist (oder zumindest sein sollte).

Der Unterschied zwischen der zweiten und der dritten Deklaration ist subtiler: Während die Extensionen der zweiten, die die aktuellen Rollen-/Rollenspielerpaarungen enthält, wie die der ersten nicht deklariert werden kann, weil sie dynamisch ist, ist die dritte Deklaration selbst ein Element der Extension der vierten Deklaration; die Angabe dieser Extension gehört zum Umfang der Spezifikation eines Modells. Es bleibt also festzuhalten, daß bei Verwendung einer vordefinierten Funktion wie *played-by* ein Teil der Modellspezifikation in den Überladungen der Funktion steckt, während die Verwendung einer Metarelation wie $<_{NR}$ das Verhältnis von Rollen und Rollenspielertypen analog zur Typhierarchie deklariert wird.

Nun noch wie angekündigt zum objektorientierten Design. Eine „Rolleninstanz“, die ein Individuum in einer Rolle repräsentiert, ist, da ihre Existenz an der des zentralen, das Objekt der Realität repräsentierenden Instanz hängt und ihre Identität unauflösbar[85] mit derselben verbunden ist, gewissermaßen eine Instanz zweiter Klasse, für die es in den gängigen objektorientierten Programmiersprachen keine Entsprechung gibt. Rollen als beigeordnete Instanzen zu modellieren heißt daher, Rollen auf ein Muster aus bestehenden Sprachkonzepten zurückzuführen. Die Modellierung mit Rollen wird damit selbst zum Gegenstand der Modellierung, aber nicht im Sinne einer Metamodellierung, sondern in der Art der Spezifikation eines Entwurfsmusters. Nun wird aber gerade in Entwurfsmustern die Notwendigkeit der eigenständigen Existenz dieses Rollenkonzepts für die Modellierung besonders deutlich (Abschnitt 3.3.5), und die Spezifikation eines Rollenmusters, ohne dazu ein ursprüngliches Rollenkonzept zur Verfügung zu haben, gerät unweigerlich zu einer zirkulären. Tatsächlich beinhaltet die Auffassung von einem Individuum als einer Instanz als Träger der Identität und einer Instanz pro Rolle, die es spielt, implizit zwei Rollen, nämlich einmal die des Kerns oder Rollenspielers, und einmal die der Rolle. Wie dies in sich selbst ausgedrückt werden könnte, ist alles andere als offensichtlich [Steimann 1999b].

[85] Ausnahmen hiervon findet man in [Kristensen & Østerbye 1996; Wong et al. 1997]

Bei genauerer Betrachtung bleibt als einziges überzeugendes Argument für die Darstellung von Rollen als beigeordnete Instanzen die Tatsache, daß ein Objekt in jeder Rolle, die es spielt, seinen eigenen Zustand hat, und zwar selbst dann, wenn es eine Rolle mehrfach spielt. So kann z. B. ein Angestellter, wenn er mehrere Anstellungen hat, auch mehrere Gehälter und mehrere dienstliche Telefonnummern haben. Auf der anderen Seite sind *Gehalt* und *Telefonnummer* eigentlich gar nicht Attribute der Rolle *Angestellter*, sondern die von Konzepten wie *Anstellung* bzw. *Arbeitsplatz* [Steimann 1999b].

Grundsätzlich ist fraglich, ob die verschiedenen Auftreten eines Individuums in Rollen überhaupt als so unabhängig erachtet werden können, daß es die Darstellung durch verschiedene Instanzen rechtfertigt. So ist beispielsweise die Wochenarbeitszeit bei mehreren Anstellungen insgesamt limitiert, und die Tätigkeit (das Verhalten) des Angestellten in einer Anstellung bleibt nicht ohne Einfluß auf das in anderen, so daß die Repräsentation eines realen Objekts durch mehrere Modellobjekte zu einer als ausgesprochen künstlich anmutenden Koordination dieser Modellobjekte führen muß.

Kapitel 4

Rollen in der Modellierungspraxis

Während man früher den Begriff Modellierung vor allem mit der Datenmodellierung und damit mit der Modellierung von Datenbank- und Informationssystemen in Verbindung brachte, werden heute in zunehmendem Maße objektorientierte Softwareanalyse und -design durch einen durchgängigen objektorientierten (Software-)Modellierungsprozeß abgelöst. Anders als die Datenmodellierung befaßt sich die Softwaremodellierung aber nicht nur mit der Struktur der Information, sondern auch mit deren Verarbeitung. Bei objektorientierter Software kommt als Besonderheit hinzu, daß die Entitäten selbst für ihre Verarbeitung verantwortlich sind: Software als Zusammenspiel aktiver Komponenten, die – ihren realen Entsprechungen gleich – neben den sie kennzeichnenden Attributen auch über Funktionen verfügen, die wiederum, in sinnvoller Weise verknüpft, die Systemfunktionalität ausmachen.

Die objektorientierte Modellierung trägt dem Rechnung, indem sie die klassische Datenmodellierung um Notationen zur Modellierung von Funktionalität und Verhalten von Entitäten ergänzt [Wieringa 1998]. Die Tendenz bei den Modellierungswerkzeugen geht dahin, aus den Struktur- und Funktionsbeschreibungen direkt Programme oder zumindest Programmgerüste abzuleiten; die Modellierungssprache ist dann gewissermaßen die graphische Syntax einer Programmiersprache. Damit stellt sich natürlich unmittelbar die Frage, in welcher Form sich Modellierungskonstrukte, hier Rollen, in Programmen wiederfinden.

Dieses Kapitel zeigt auf, wie der im vorigen Kapiteln erarbeitete Rollenbegriff auf die objektorientierte Softwaremodellierung übertragen werden kann und in welcher Form sich Rollen in objektorientierter Software wiederfinden. Dazu wird eine Änderung der UNIFIED MODELING LANGUAGE (UML), deren Rollenbegriff im vorigen Kapitel bereits diskutiert wurde, vorgeschlagen, aus der sich die Implementierung von Rollen unmittelbar ergibt.

4.1 Übertragung der Rollendefinition auf UML

Wie bereits im letzten Kapitel bei der Betrachtung der verschiedenen Rollenbegriffe in der Literatur aufgezeigt, kennt UML seinen eigenen, der jedoch mit einigen Problemen behaftet ist. Historisch gewachsen ergeben sich diese Probleme u. a. aus dem doppelten Ursprung des Begriffs: dem (Rollen-) Namen eines Assoziationsendes, wie er der Datenmodellierung entstammt, und dem der Rolle in einer Kollaboration, wie sie in der rollenzentrierten Modellierung à la OORAM [Reenskaug et al. 1996] vorkommt. Der Versuch, diese beiden Ursprünge als zwei Varianten, nämlich die statische und die dynamische, desselben Begriffs hinzustellen [Rumbaugh et al. 1999, S. 414], erscheint bei genauerer Betrachtung eher plakativ.

Die bereits angesprochenen Probleme mit dem Rollenbegriff der gegenwärtigen Fassung von UML (Version 1.3) werden in Abschnitt 4.1.1 noch einmal detailliert. Interessanterweise lassen sich die aufgezeigten Defizite mit einer geeigneten Neufassung des Rollenbegriffs recht gut beheben. Dazu ist zunächst eine Änderung des Metamodells notwendig, die jedoch zugleich eine Reduktion der Anzahl von Modellierungskonzepten mit sich bringen und damit zu einer besseren Verständlichkeit und einer leichteren Anwendbarkeit führen kann.

In Abschnitt 4.1.2 werden die Änderungen am Metamodell beschrieben und die Vereinfachungen, die sich daraus ergeben. In Abschnitt 4.1.3 wird dann gezeigt, inwieweit es möglich ist, die alte Notation mit dem geänderten Metamodell beizubehalten, wie sich also die bisherige Notation mit dem neuen Metamodell vereinbaren läßt. Im Anschluß daran wird in Abschnitt 4.1.4 eine Änderung der UML-Notation selbst vorgeschlagen und deren Abbildung auf das geänderte Metamodell dargelegt. Es wird sich zeigen, daß die vorgeschlagenen Änderungen nicht unter allen Umständen Vorteile mit sich bringen. Abschließend wird in Abschnitt 4.1.5 noch der Zusammenhang zwischen der geänderten UML-Spezifikation und der Formalisierung in LODWICK aus dem vorigen Kapitel aufgezeigt; dieser Zusammenhang liegt eigentlich in der Luft, wird aber hintangestellt, damit das Kapitel auch isoliert gelesen werden kann. Die Zusammenhänge von UML-Modell und -Metamodell sowie die Beiträge der einzelnen Abschnitte sind in Abbildung 4.1 dargestellt.

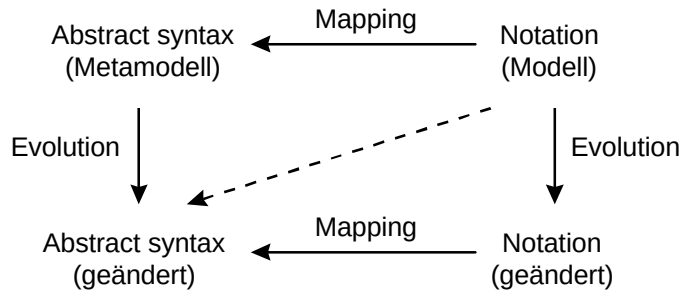

Abbildung 4.1: Zusammenhang von UML-Notation und Metamodell. Die Abbildung der ursprünglichen Notation in die ursprüngliche sog. abstrakte Syntax (syntaktischer Teil des Metamodells) ist in der UML-Dokumentation spezifiziert; der Übergang von der ursprünglichen zur geänderten Spezifikation sowie die Abbildung der ursprünglichen und geänderten Notation auf das geänderte Metamodell werden in den Abschnitten 4.1.2 bis 4.1.4 erläutert.

Anmerkung: Die Änderungen an UML, die in den Abschnitten 4.1.2 bis 4.1.4 vorgeschlagen werden, betreffen sowohl das Metamodell als auch die Diagrammtypen (weniger jedoch die Notation, die größtenteils erhalten bleiben kann). Um diese Abschnitte besser verstehen zu können, sollte man wissen, wie die Spezifikation von UML aufgebaut ist. Dazu kurz ein paar Hinweise.

Die UML-Spezifikation besteht aus der getrennten Angabe eines Metamodells und einer Notation (Abbildung 4.1). Die Notation unterscheidet verschiedene Diagrammtypen, mit denen unterschiedliche Aspekte eines Systems dargestellt werden können. Das Metamodell stellt mit der sog. abstrakten Syntax für die verschiedenen Diagrammtypen eine einheitliche Basis zur Verfügung, an der sowohl die Semantik der jeweiligen Notation also auch die Konsistenz der Inhalte verschiedener Diagramme zum selben Modell festgemacht wird. Die Diagramme werden mit Hilfe von Abbildungsregeln in Instanzen des Metamodells übertragen. Der Modellierer ist vor allem mit der Notation, der Syntax befaßt; wie bei anderen Sprachen auch, ist jedoch ein umfassendes Verständnis der Semantik Voraussetzung, um sie korrekt anwenden zu können.

4.1.1 Probleme des gegenwärtigen Rollenbegriffs in UML

Wie in Abschnitt 3.3.4 des vorigen Kapitels beschrieben, treten Rollen in UML vor allem entweder als Rollennamen oder als Classifier-, Assoziations- und Assoziationsenderollen auf. Zwischen Rollennamen und den Classifier-Rollen einer Kollaboration besteht ein gewisser Zusammenhang: Das, was in einem Klassendiagramm für das durch den Rollennamen bezeichnete Assoziationsende der Interface specifier ist, ist in einem Kollaborationsdiagramm für die Classifier-Rolle am Ende der zur Assoziation gehörigen Assoziationsrolle die Liste der Base classifier.

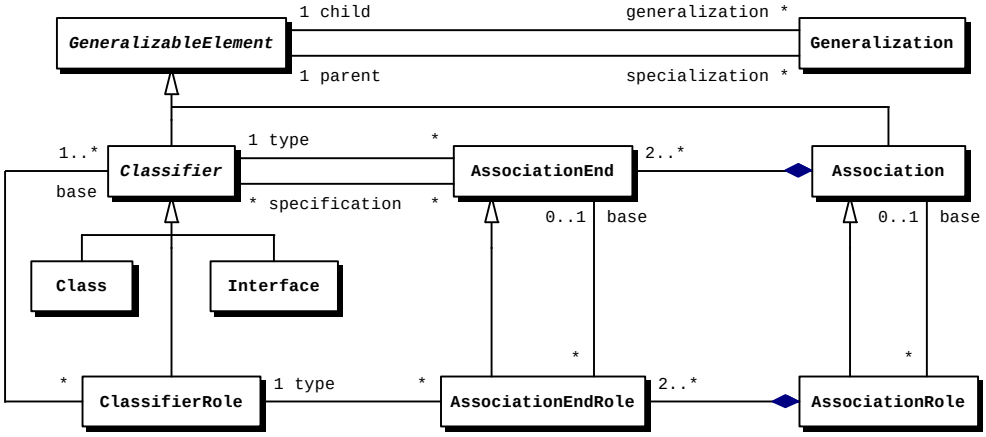

Abbildung 4.2: Ausschnitt des ursprünglichen UML-Metamodells (zusammengestellt aus den Abbildungen 2-6, 2-8 und 2-17 der UML-Spezifikation [OMG 1999]). Die Klasse *AssociationClass* als gemeinsame Unterklasse von *Association* und *Class* wurde nicht übernommen; s. dazu auch die Diskussion von Relationen als Typen in Abschnitt 2.2.2.

Das Metamodell in Abbildung 4.2 stellt den Zusammenhang dar. Man beachte, daß mit einer Assoziationsenderolle auf gleich vier Wegen Classifier verbunden werden können, nämlich einmal über den Pfad *base.type*, einmal über den Pfad *base.specification*, einmal über den Pfad *type.base* und nicht zuletzt (per Vererbung von *specification* von *AssociationEnd* auf *AssociationEndRole*) über *specification.base* (alles mit Ursprung *AssociationEndRole*). Bei Berücksichtigung der Kardinalitäten tut sich hier eine beachtliche Vielfalt auf, deren Konsistenthaltung für den Modellierer einige Probleme mit sich bringt. Außerdem können wegen der einheitlichen Verwendung des Konzepts Classifier Klassen und Interfaces selbst da gemischt auftreten, wo entweder eine Klasse oder ein Interface gebraucht wird. [Steimann 1999d]

Anmerkung: Tatsächlich entsteht ein Gutteil der Komplexität von UML dadurch, daß in der Spezifikation zumeist der allgemeine Begriff Classifier steht, der neben Klasse und Interface u. a. auch Use case umfaßt [Rumbaugh et al. 1999, S. 43]. Um die Generizität des UML-Metamodells, die in ihrem vollen Umfang nur schwer überschaubar ist, einzuschränken, wird in den folgenden Ausführungen der Umfang von Classifier auf Klasse und Interface beschränkt. Es bleiben jedoch trotzdem noch zahlreiche Probleme; z. B. können Klassen und Classifier-Rollen überall da gemischt auftreten, wo Classifier verlangt werden (z. B. an Assoziationsenden), und Classifier-Rollen und Assoziationsrollen können unabhängig von ihren Basen generalisiert werden.

Die Classifier-, Assoziations- und Assoziationsenderollen von UML sind Bestandteile des Kollaborationsdiagramms auf Spezifikationsebene und treten bei

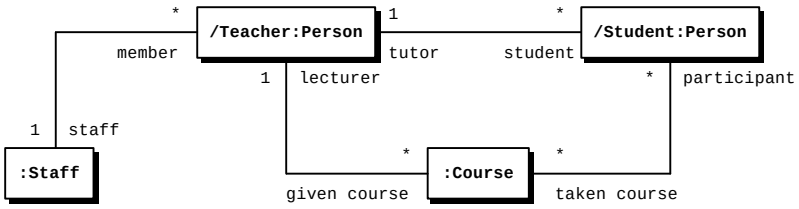

Abbildung 4.3: Ein Kollaborationsdiagramm mit zwei benannten und zwei unbenannten Classifier-Rollen (aus [Övergaard 1999, OMG 1999]). Man beachte, daß sich die Rollennamen (an den Assoziationsenden) von den Namen der Classifier-Rollen unterscheiden.

der Spezifikation einer Interaktion, also des dynamischen Zusammenspiels von Objekten zur Erfüllung einer Aufgabe, auf. Die Classifier-Rollen haben jedoch noch eine zweite, eine strukturbildende (und damit statische) Funktion: Sie erlauben die Aufteilung einer Klasse nach den Rollen, die ihre Instanzen spielen. Abbildung 4.3 zeigt ein solches Kollaborationsdiagramm. Die Classifier-Rollen *Teacher* und *Student* haben denselben Classifier, nämlich *Person*, zur Basis; die Aufteilung der Klasse *Person* in zwei Rollen erlaubt eine Differenzierung der Zusammenhänge, die in einem reinen Klassendiagramm nicht möglich wäre.

Tatsächlich läßt das dazugehörige Klassendiagramm in Abbildung 4.4 nicht erkennen, daß nur Lehrer Mitglieder von Staff sind und daß nur Lehrer Tutoren sein können [Övergaard 1999]. Diese Nebenbedingungen werden eben erst durch das Kollaborationsdiagramm in Abbildung 4.3 ausgedrückt. Da solche Nebenbedingungen jedoch statisch und von jeder Interaktion unabhängig sind, sollten sie auch davon unabhängig und nicht nur in einem Kollaborationsdiagramm spezifiziert werden können. Eine Aufteilung von *Person* in die Unterklassen *Teacher* und *Student* im Klassendiagramm hingegen würde, wie bereits in Abschnitt 3.1 dargelegt, die Realität nicht angemessen wiedergeben.

Kollaborationsdiagramme wie das in Abbildung 4.3 bringen zwar gegenüber einem Klassendiagramm einen Zugewinn an Ausdrucksstärke (es vermeidet beispielsweise den Eindruck, man könne von *Staff* über *member* und *taken course* zu *Course* gelangen), sie lassen aber dennoch Fragen offen. So geht aus

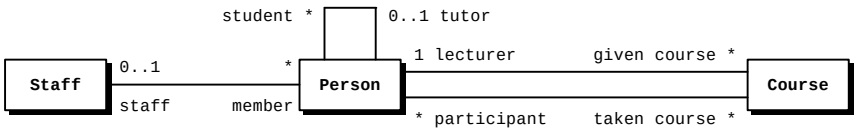

Abbildung 4.4: Klassendiagramm, in dem die Klassen und Assoziationen der Kollaboration aus Abbildung 4.3 eingeführt werden.

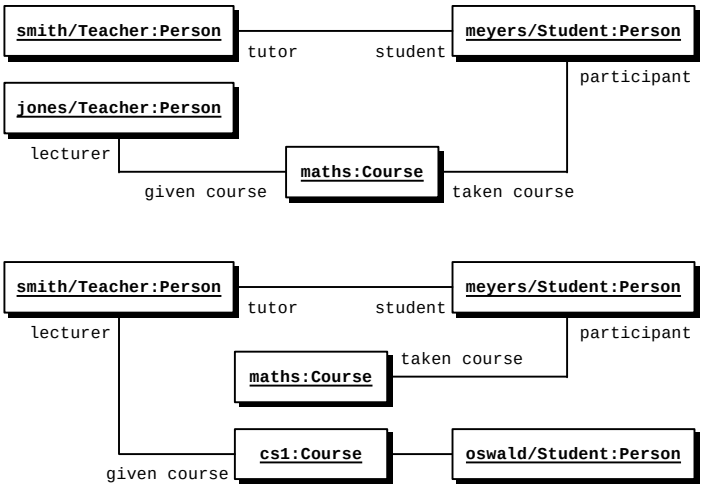

Abbildung 4.5: Kollaborationsdiagramme auf Instanzebene ohne Interaktionsannotation. Beide Diagramme legen nahe, daß ein Student nicht von derselben Person unterrichtet und betreut wird, einmal aus der Sicht des Studenten und einmal aus der Sicht des Lehrers.

Abbildung 4.3 wohl die Trennung von *Person* in *Teacher* und *Student* hervor, nicht jedoch, ob dieselbe Person sowohl Tutor als auch Dozent und dies auch noch (über den Umweg eines Kurses) ein und desselben Studenten sein kann. Die Zusammenführung der Rollennamen *tutor* und *lecturer* in einer Rolle *Teacher* erlaubt dies auf jeden Fall zunächst: Es ist nach Abbildung 4.3 sogar möglich, daß eine Person sich selbst betreut und/oder unterrichtet. UML sieht daher zusätzlich ein Kollaborationsdiagramm auf Instanzebene vor, in dem die Rollen wie in Abbildung 4.5 mit konkreten Objekten besetzt werden. Man beachte jedoch, daß auch solche Diagramme zunächst unabhängig von jedweder Interaktion sind; vielmehr handelt es sich eigentlich um Objektdiagramme (Szenarien in LODWICK und FREGE), in denen die Objekte mit Classifier-Rollen annotiert werden. Die ansonsten konsequent durchgezogene Trennung nach Struktur- (Statik) und Verhaltensspezifikationen (Dynamik) – zu letzteren zählen Kollaborationsdiagramme als Interaktionsdiagramme per definitionem – wird also mit der gegenwärtigen Doppelfunktion der Kollaborationsdiagramme durchbrochen.

4.1.2 Änderungen am Metamodell

Die erste und grundlegendste Änderung an UML, die hier vorgeschlagen wird, ist die Zusammenlegung des Interface- und des Rollenkonzepts. Interfaces füh-

ren in UML eher ein Schattendasein; sie entsprechen im wesentlichen den Interfaces von JAVA. Das heißt insbesondere, daß Interfaces zwar Operationen, aber keine Attribute deklarieren und daß sie nur an gerichteten Assoziationen und daran nur als Ziel teilhaben dürfen. Auf der anderen Seite spezifizieren gerade die Rollen von UML Interfaces: zum einen indirekt, indem sie Assoziationsenden benennen, die wiederum mit Interface specifiern behaftet sein können, zum andern direkt, indem sie als Classifier-Rollen die Eigenschaften, nicht jedoch die Klassen der Objekte, die diese Rollen spielen können, festlegen (obwohl Classifier-Rollen mit ihren Basis-Classifiern in der Regeln eine Klasse nennen; [OMG 1999, § 2.10.4, S. 2-113]; s. die Darlegung in Abschnitt 3.3.4). Tatsächlich sind sich Rollen und Interfaces in UML so ähnlich, daß die Zusammenlegung keinen prinzipiellen Verlust an Ausdrucksstärke mit sich bringt; die Einschränkungen für Interfaces, die für Rollen allgemein nicht akzeptabel sind, müssen jedoch fallengelassen werden.

Die zweite und augenfälligste Änderung, die vorgeschlagen wird, ist die, daß Assoziationen grundsätzlich nur noch Rollen verbinden. Der Rollenname als eigenständiges Konzept kann somit entfallen; statt dessen muß jede Rolle innerhalb einer Assoziation eindeutig sein, so daß mit der Rolle auch das Assoziationsende benannt wird. Für die UML-Notation bedeutet das, daß das Klassendiagramm, in dem die Assoziationen bisher dargestellt werden, und das Kollaborationsdiagramm auf Spezifikationsebene enger zusammenrücken – in beiden haben jetzt Assoziationen Rollen an ihren Enden, und die Klassen, die diese Rollen füllen, werden nur noch indirekt spezifiziert.

Das geänderte Metamodell ist in Abbildung 4.6 dargestellt. Man beachte die im Vergleich zu Abbildung 4.2 deutlich einfachere Struktur, die sich aus der in den folgenden Punkten spezifizierten Zusammenlegung von Modellelementen ergibt:

1. Die beiden Konzepte *Interface* und *ClassifierRole* werden zu einem namens *Role* zusammengeführt. *Class* und *Role* werden klar voneinander unterschieden. So können nur Klassen eigene Instanzen haben – Rollen sind generell nicht instanziierbar. Klassen und Rollen werden auch getrennt voneinander generalisiert; die gemeinsame Abstraktion *Classifier* verliert in diesem Zusammenhang ihre Bedeutung.
2. Die Pseudoattribute (Assoziationen) *type* und *specification* von *AssociationEnd* werden zu einer Assoziation *specifies* zusammengefaßt, die jedem Assoziationsende genau eine Rolle zuordnet. Diese Rolle spezifiziert direkt, welche Eigenschaften von seiten der Assoziation gefordert

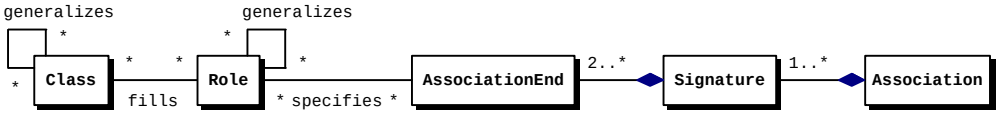

Abbildung 4.6: Das vereinfachte Metamodell. Die Überladung von Assoziationen (mehrfache Deklaration einer Relation mit gleichem Namen, aber unterschiedlichen Rollen) wird durch die Zuordnung mehrerer Signaturen zu einer Assoziation abgedeckt. Die Analogie zu LODWICK liegt auf der Hand, wird aber erst in Abschnitt 4.1.5 besprochen.

werden (vormals *specification*), und indirekt (über die Assoziation *fills* zwischen *Role* und *Class*, s. u.), Instanzen welcher Klassen an der Stelle auftreten dürfen (vormals *type*). Die Einschränkungen bzgl. der Interfaces werden fallengelassen. Das Attribut *name* eines Assoziationsendes (der Rollenname) wird entweder fallengelassen oder muß stets mit dem der damit verbundenen Rolle übereinstimmen.

3. Das Pseudoattribut *base* (von *ClassifierRole* zu *Classifier*) wird zur Assoziation *fills*, die Klassen mit Rollen verbindet. Dabei kann eine Klasse beliebig viele Rollen füllen und eine Rolle kann von beliebig vielen Klassen gefüllt werden. Die Interpretation von *fills* ist die von $<_{NR}$ aus LODWICK: Wenn eine Klasse eine Rolle füllt, bedeutet dies, daß alle Instanzen der Klasse überall da auftreten können, wo Instanzen der Rolle gefordert sind.
4. *AssociationRole* und *AssociationEndRole* gehen in *Association* bzw. *AssociationEnd* auf; die in den „Rollen“ enthaltene Information wird durch das Überladen von Assoziationen ersetzt. Damit ist die Classifier-Rolle die einzige der drei Kollaborationsrollen, die in das geänderte Metamodell übernommen wird. Dies ist jedoch kein Verlust: Die in der UML-Spezifikation genannten Gründe für die Einführung von *AssociationRole* und *AssociationEndRole* sind kaum plausibel; s. dazu die Besprechung von UML in Abschnitt 3.3.4.
5. Die Generalisierung von Assoziationen wird ebenfalls durch das Überladen von Assoziationen ersetzt. (Die Relation *generalizes* des Metamodells in Abbildung 4.2 ist ein Beispiel für eine überladene Relation.)

Es ist nicht ganz klar, ob die Generalisierung von Assoziationen in UML (abgesehen von der Tatsache, daß, genau wie bei Classifiern, Generalisierung und Spezialisierung sich anhand ihres Namens unterscheiden sollten) eine andere Semantik hat als die von Überladungen. Tatsächlich ist der einzige Hinweis in der UML-Spezifikation zur Generalisierung von Assoziationen eher dürftig:

„Generalization may be applied to associations as well as classes, although the notation may be messy because of the multiple lines. An association can be shown as an association class for the purpose of attaching generalization arrows.“ [OMG 1999, § 3.49.2, S. 3-80] Rumbaugh et al. [1999, S. 163] sind etwas ausführlicher; hier erfolgt zusätzlich der Hinweis, daß die Extension einer Spezialisierung einer Assoziation eine Teilmenge der der ursprünglichen Assoziation sein muß. Daß sich daraus eigentlich eine Äquivalenz von Assoziationsgeneralisierung und Überladung ergeben müßte, wird in [Steimann 2001] ausführlicher dargelegt.

4.1.3 Verwendung der ursprünglichen Notation mit dem geänderten Metamodell

UML hat in seinen aktuellen Versionen (1.1 und 1.3) einige Verbreitung gefunden – zahlreiche objektorientierte Modellierungswerkzeuge unterstützen es und in vielen Softwareentwicklungsprojekten wird es als Standard verwendet. Während eine Änderung des Metamodells nicht nur möglich (Änderungen hat es bereits innerhalb der Version 1 gegeben), sondern auch angezeigt erscheinen mag [Mellor 1999, S. 709], stellt eine umfassende Änderung der Notation ein sehr viel größeres praktisches Problem dar, weil dies nicht im Hintergrund geschehen kann, sondern alle UML-Artefakte unmittelbar betrifft. Es ist also durchaus lohnend, zu untersuchen, inwieweit die alte Notation bei Einsatz des geänderten Metamodells weiter verwendet, inwieweit also die alte Notation auf das neue Metamodell abgebildet werden kann (gestrichelte Linie in Abbildung 4.1).

Wenn man davon ausgeht, daß die Modellierung ohne Kenntnis des neuen Metamodells und der damit verbundenen veränderten Semantik erfolgt ist, wenn also insbesondere nicht bewußt Interfaces anstelle von Klassen an Assoziationsenden eingesetzt werden, dann führen die folgenden Abbildungsregeln zu einer rollenreichen, aber durchaus richtigen (im Sinne einer Modellierung mit Rollen) Interpretation.[86]

Klassendiagramm

Klassen (auch abstrakte) werden auf Instanzen von *Class*, Interfaces auf Instanzen von *Role* abgebildet. Assoziationsenden bleiben Assoziationsenden.

[86] Eine ausführlichere Darstellung der Abbildung der ursprünglichen Notation in das geänderte Metamodell findet man in [Steimann 2000d].

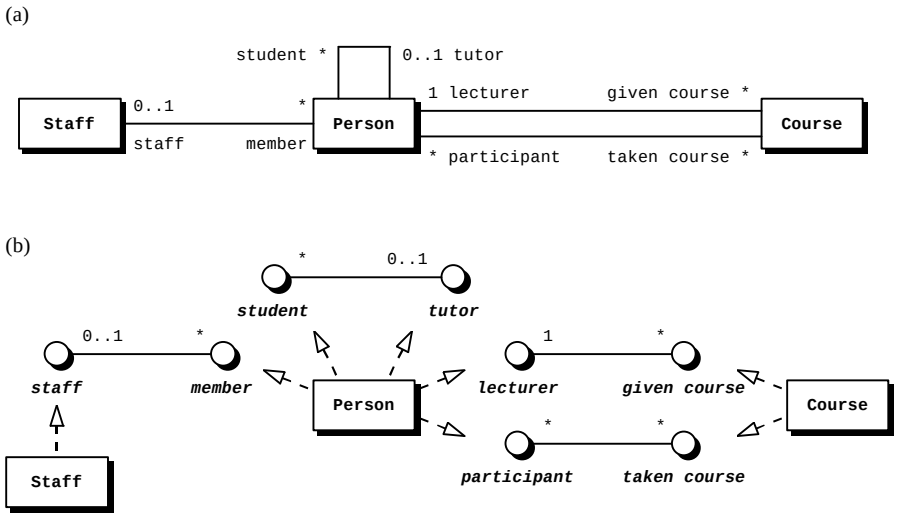

Abbildung 4.7: (a) herkömmliches Klassendiagramm und (b) Klassendiagramm mit expliziten Rollen. Auch (b) ist ein korrektes UML-Diagramm.

Allerdings wird der Classifier, der mit einem Assoziationsende verbunden ist und der ursprünglich auf das Pseudoattribut *type* abgebildet wurde, für den Fall, daß er eine Klasse und kein Interface ist, nicht mehr in direkten Zusammenhang mit dem Assoziationsende gebracht: Ihn vertritt eine Instanz von *Role*, die über *specifies* mit dem Assoziationsende verbunden ist und die er (als Instanz von *Class*) füllen muß. Falls vorhanden, trägt diese Instanz von *Role* den mit dem Assoziationsende verbundenen Namen, den (alten) Rollennamen (Abbildung 4.7).

Man mag es vielleicht nur selten sehen, aber jedes Assoziationsende eines UML-Klassendiagramms kann (zusätzlich oder alternativ zu einem Classifier) mit einem oder mehreren Interface specifiern versehen werden, die im alten Metamodell Werten des Pseudoattributs *specification* entsprechen. Die Interface specifier i_1, ..., i_n eines solchen Assoziationsendes mit Namen r (dem Rollennamen, gemeinsam dargestellt durch r:i_1, ..., i_n) werden zusammen auf eine Rolle r abgebildet, die eine (die größte) gemeinsame Unterrolle der Rollen (= Interfaces) i_1 bis i_n ist. Diese Rolle ist dieselbe wie die, die bei der Angabe eines Classifiers entsteht (s. o.). Wie allerdings Interface specifier und Classifier (bzw. die Pseudoattribute *type* und *specification*) in Einklang zu bringen sind, geht aus der UML-Spezifikation nicht hervor. Sinnvoll wäre, zu verlangen, daß wenn es sich bei dem Classifier um eine Klasse handelt, daß diese dann die Rolle r, die sich aus den Interface specifiern ableitet, füllen muß; aus Gründen

der Gleichbehandlung müßte dann ein Interface als Classifier Unterrolle von i_1 bis i_n sein und könnte mit *r* zusammenfallen.

Für jede Assoziation wird neben einer Instanz der Klasse *Association* auch eine der Klasse *Signature* angelegt, die auf die Assoziationsenden verweist. Man wird in UML in einem Klassendiagramm i. a. dieselbe Assoziation nicht mehrfach vorfinden; falls doch, so handelt es sich um eine Überladung, die einer bereits bestehenden Instanz von *Association* eine weitere von *Signature* zuordnet.

Kollaborationsdiagramm

Das UML-Kollaborationsdiagramm auf Spezifikationsebene enthält anstelle von Classifiern und Assoziationen Classifier-Rollen und Assoziationsrollen. Die Assoziationsenderollen fassen, entsprechend den Assoziationsenden, die Information zusammen, die den Assoziationsrollenenden zugeordnet sind. Das geänderte Metamodell kennt nur noch eine Sorte von Rollen, auf die die Classifier-Rollen abgebildet werden; Assoziationsrollen und Assoziationsenderollen werden wie ganz normale (dann überladene) Assoziationen und Assoziationsenden behandelt.

Jede Classifier-Rolle mit Namen *r* und Basis-Classifiern b_1, ..., b_n (in UML dargestellt durch /*r*:b_1, ..., b_n; vgl. Abbildung 4.3) wird auf eine Rolle *r* abgebildet. Diejenigen Basis-Classifier, die Interfaces sind, sind Rollen gleichgestellt; für die, die Klassen sind, müssen Rollen eingeführt werden, die davon abstrahieren. Die Rolle *r* muß dann eine (die größte gemeinsame) Unterrolle der Rollen sein, die mit den Basis-Classifiern b_1 bis b_n verbunden sind. Die Angabe der Basis einer Classifier-Rolle wird also in der Generalisierung (Spezifikation der Unterrollenbeziehung) aufgelöst.

Das UML-Kollaborationsdiagramm auf Instanzebene enthält Objekte (als Instanzen der Klassen) und deren Verknüpfungen (sog. Links). Es ist im wesentlichen eine alternative Darstellung des Sequenzdiagramms. Da die Instanzebene im Metamodell hier nicht behandelt wird, entfällt an dieser Stelle auch die Abbildung in dasselbe.

Konsistenthaltung

Assoziationsenderollen werden wie gesagt wie Assoziationsenden (eines Klassendiagramms) behandelt und dementsprechend in Rollen aufgelöst. Da Assoziationsenderollen per definitionem Einschränkungen sind, müssen die daraus hervorgehenden Rollen Unterrollen der zu den Assoziationsenden gehörenden

Rollen sein. Die Überladungen sind also automatisch kovariant (oder monoton), wie die folgende Betrachtung noch einmal zeigt.

Zuoberst in der Hierarchie stehen die Rollen, die den Assoziationsenden zugeordnet werden. Lagen an einem oder mehreren Assoziationsenden Interface specifier vor, so ergeben sich daraus bereits kleine Rollenhierarchien; die zur Assoziation gehörende Signatur verbindet dann die jeweils untersten Rollen dieser Hierarchien miteinander, nämlich die, die die Classifier vertreten.

Darunter kommen die Rollen, die aus den Assoziationsenderollen hervorgehen. Nimmt eine Assoziationsenderolle gegenüber dem Assoziationsende, zu dem sie gehört, keine Einschränkungen vor, werden auch keine neuen Rollen hinzugefügt; andernfalls sind die Rollen Unterrollen der Rollen, die den dazugehörigen Assoziationsenden entsprechen.

In UML können Assoziationsenden, die darauf basierenden Assoziationsende-

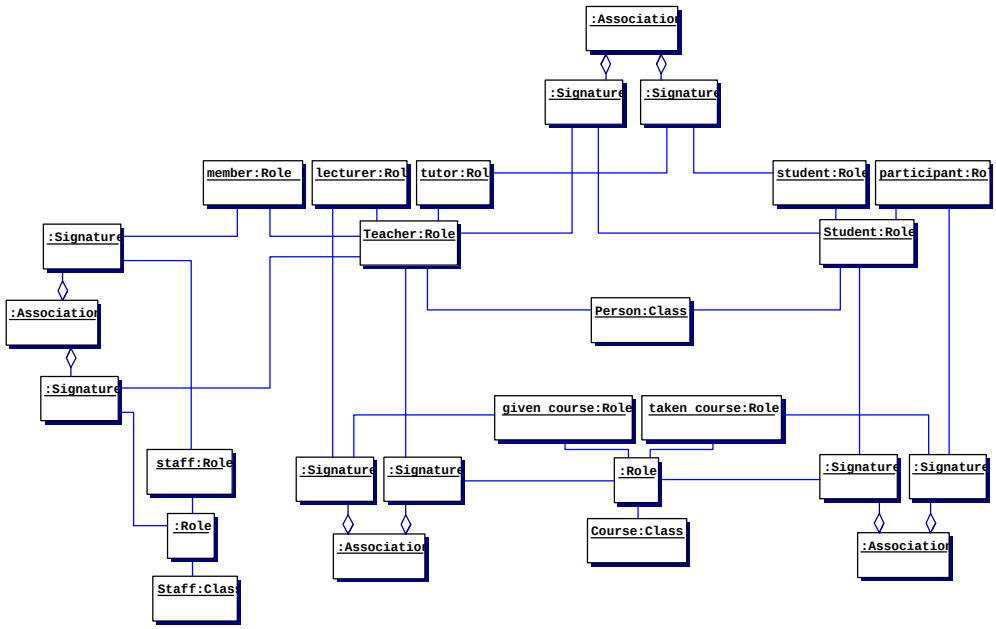

Abbildung 4.8: Klassen- und Kollaborationsdiagramm aus Abbildung 4.4 und Abbildung 4.3 als Instanz des geänderten Metamodells. Das Diagramm ist ein Objektdiagramm; die Kästen stehen für Instanzen der Metaklassen aus Abbildung 4.6. Die Beschriftung der Links fehlt – Oberrollen stehen über ihren Unterrollen, ansonsten ergibt sie sich eindeutig aus dem Metamodell. Die Assoziationsenden wurden weggelassen, da die Information, die sie enthalten, hier nicht dargestellt wird.

rollen und die mit diesen verbundenen Classifier-Rollen verschiedene Namen haben (Abbildung 4.3). Ein solcher Namenskonflikt (der im ursprünglichen UML keiner ist, weil sich hier lediglich die Rollennamen der Assoziationsenden und Classifier-Rollen gegenüberstehen) wird im geänderten Metamodell über die Rollenhierarchie und das Überladen von Assoziationen aufgelöst: Die den Assoziationsenden einer Assoziation eines Klassendiagramms zugeordneten Rollen werden auf Oberrollen der dazugehörigen Assoziationsenderollen eines Kollaborationsdiagramms abgebildet, die wiederum zu Oberrollen der Classifier-Rollen werden. Je nachdem, ob die ursprünglichen Assoziationsende- und Classifier-Rollen Einschränkungen gegenüber ihren Basen mit sich bringen, werden die Assoziationen überladen. Abbildung 4.8 zeigt eine Instanz des Metamodells aus Abbildung 4.6, die aus dem Klassendiagramm aus Abbildung 4.4 und aus dem Kollaborationsdiagramm aus Abbildung 4.3 hervorgegangen ist.

4.1.4 Änderungen an der Notation

Im vorigen Abschnitt wurde dargestellt, wie sich die Änderung des Metamodells in der üblichen UML-Notation berücksichtigen läßt, ohne daß dazu eine neue Notation oder gar andere Diagrammtypen vonnöten wären. Es ist dies sicherlich der in der Praxis zu bevorzugende Weg. An dieser Stelle sollen jedoch die theoretischen Argumente überwiegen und eine entsprechend motivierte Abwandlung der Notation vorgeschlagen werden, auch wenn diese in der Praxis kaum Chancen haben sollte. Als Kompromiß kommen aber auch Notationen wie die von Riehle [2000] in Frage; Abbildung 4.15 beispielsweise greift darauf zurück.

Wie in dem motivierenden Beispiel eingangs angedeutet, erfüllt das Kollaborationsdiagramm von UML derzeit neben der Spezifikation des (dynamischen) Zusammenspiels mehrerer Objekte zur Erfüllung einer bestimmten Funktion noch eine weitere Aufgabe: Es stellt eine differenziertere Sicht des (statischen) Klassendiagramms dar, indem es die Menge der Objekte eines Classifiers auf Basis der Beziehungen, in denen sie zu anderen stehen, aufteilt (vgl. die sogenannten qua-definierten Konzepte in Abschnitt 3.4.3). Da in der objektorientierten Modellierung aber üblicherweise klar zwischen Statik und Dynamik unterschieden wird und kein Grund vorliegt, weswegen damit gebrochen werden sollte, empfiehlt es sich auch hier, an dieser Trennung festzuhalten und den Zusammenhang von Klassen, Rollen und Assoziationen zunächst in einem statischen Kontext darzustellen.

Ein weiteres Ziel sollte es sein, Relationen graphisch nach Ordnungen getrennt darzustellen. Noch einmal deutlich machen kann man sich die Problematik am ursprünglichen Metamodell von UML (Abbildung 4.2): Während *Classifier-Role* eine Unterklasse von *Classifier* sein soll (und damit zwei Metaklassen in Beziehung gesetzt werden), drückt die zur Generalisierung parallele Relation *base* aus, daß zu jeder Instanz von *ClassifierRole* (also z. B. *Teacher*) eine oder mehrere Instanzen von *Classifier* (z. B. *Person*) gehören (einer Aussage, durch die nicht Metaklassen, sondern deren Instanzen, also Klassen, in Beziehung gesetzt werden). Es wird durch die Generalisierungen eben nicht ausgedrückt, daß jede Classifier-Rolle Unterklasse eines Classifiers ist; statt dessen wird das Verhältnis von konkreten Classifier-Rollen und konkreten Classifiern unabhängig von der Generalisierung durch die Relation *base* ausgedrückt.[87]

Es scheint also durchaus sinnvoll, die Darstellung der Klassen- und Rollenhierarchie (Relationen zweiter Ordnung) von der der Assoziationen (Relationen erster Ordnung) zu trennen. Dies wird durch die Unterscheidung des Klassendiagramms in ein Classifier-Diagramm und in ein Assoziationsdiagramm erzielt.

Classifier-Diagramm

Das Classifier-Diagramm, das hier wegen der gegebenen Einschränkung von *Classifier* ein Klassen- und Rollendiagramm ist, stellt Klassen und Rollen jeweils mit ihren Eigenschaften (einschließlich aller Attribute, jedoch *ohne* Assoziationen) sowie die hierarchische Ordnung derselben dar. Zur Darstellung der Rollen werden zwecks besserer Unterscheidung von Klassen Kreise verwendet, die in Form der Lollipop-Notation für Interfaces (die ja in Rollen aufgegangen sind) in UML bereits eingeführt sind [Fowler & Scott 1997; OMG 1999, § 3.28]. Klassen und Rollen werden jeweils getrennt voneinander generalisiert; für die Generalisierungen wird einheitlich der in UML übliche Generalisierungspfeil verwendet. Für die Verbindung von Klassen und Rollen wird der gestrichelte Generalisierungspfeil benutzt, der in UML für die Realisierung oder Implementierung steht. Da das Diagramm sonst keine grafisch dargestellte Information enthält, kann es stets von oben nach unten ausgerichtet werden, d. h., die Generalisierungen können stets über ihren Spezialisierungen stehen. Das Klassen- und Rollendiagramm des laufenden Beispiels ist in Abbildung 4.9

[87] Wenn man hier vielleicht einen Hinweis darauf vermutet, in UML seien (Classifier-)Rollen doch Unterklassen der Klassen, die sie füllen (eine Ansicht, die in der Literatur ja häufig vertreten wird; vgl. Abschnitt 3.4.2), dann ist man eben dem Irrtum aufgesessen, der gerade durch die mangelnde Trennung von Relationen verschiedener Ordnungen erst entsteht.

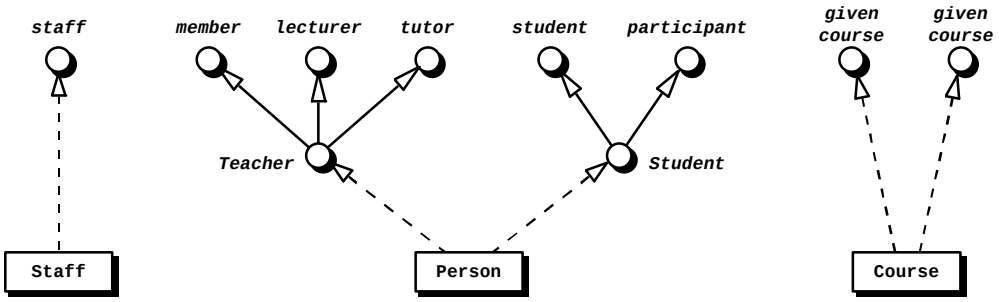

Abbildung 4.9: Classifier-Diagramm. Das Diagramm drückt neben der Definition der Klassen und Rollen auch die hierarchische Ordnung untereinander aus. Assoziationen werden nicht eingezeichnet; man beachte aber, daß die Attribute, die mit den Klassen und Rollen definiert werden, durchaus Klassen oder Rollen zum Wertebereich nehmen dürfen. Dies entspricht dann den navigierbaren gerichteten Assoziationen von UML, die zumeist durch den Rollennamen des gegenüberliegenden Endes bezeichnet werden („Pseudoattribute"). Daß Rollen teilweise klein-, teilweise großgeschrieben werden, liegt daran, daß die Bezeichner der ursprünglichen Diagramme beibehalten wurden. Klein steht für einen ehemaligen Rollenamen, Groß für eine ehemalige Classifier-Rolle. Somit ist *student* auch nicht gleich *Student*.

zu sehen. Unterklassen kommen darin nicht vor, der Fall, daß verschiedene Klassen dieselbe Rolle füllen, ebenfalls nicht.

Die Tatsache, daß in einem Klassen- und Relationsdiagramm Assoziationen nicht, Attribute aber sehr wohl vorkommen dürfen, mag befremdlich erscheinen, da in UML Assoziationen und Attribute bis zu einem gewissen Grad austauschbar sind. Ganz so ungewöhnlich ist das allerdings nicht: auch in einem Entity-Relationship-Diagramm wird zwischen Attributen und Relationen klar unterschieden (ein Umstand, der jedoch Gegenstand berechtigter Kritik ist [Halpin 1995]). Die Idee ist hier, daß Assoziationen die „großen" (oder zentralen) Beziehungen eines Modells sind, die eine eigenständige Bedeutung haben und die nicht nur einer Rolle (oder Klasse) zugeordnet werden können. Vgl. dazu auch die Einschränkung der Intensionen natürlicher Typen auf einstellige Prädikate in Abschnitt 2.3.1.

Das Classifier-Diagramm ist ein Diagramm auf Spezifikationsebene. Es gibt dazu kein sinnvolles Pendant auf Instanzebene: Sowohl *generalizes* als auch *fills* sind Relationen zweiter Ordnung, und die Pfeile des Diagramms stellen bereits Elemente (oder Instanzen) dieser Relationen dar – sie verbinden Instanzen der Metaklassen *Class* und *Role*. Die Abbildung der Notation in das Metamodell ist entsprechend einfach: Jedes Klassensymbol entspricht einer Instanz von *Class*,

jedes Rollensymbol einer von *Role*. Die Generalisierungspfeile mit ihren Enden entsprechen Tupeln der (überladenen) Relation *generalizes*, die Realisierungspfeile entsprechen Tupeln der Relation *fills*.

Assoziationsdiagramm

Das Assoziationsdiagramm ist ein neuer Diagrammtyp, der Relationen erster Ordnung, die Assoziationen, darstellt. Da Assoziationen nach dem neuen Metamodell nicht an Klassen, sondern ausschließlich an Rollen enden, enthält dieses Diagramm zunächst nur Rollen (wieder als Kreise dargestellt), und Linien, die für die Assoziationen stehen und die diese Rollen verbinden. Höher als zweistellige Relationen werden wie bisher in UML mit Rauten dargestellt. Wenn eine Rolle an mehreren Assoziationen beteiligt ist, wird sie nur einmal dargestellt; es entsteht so ein Netz von Assoziationen.

Überladene Assoziationen werden zunächst als getrennte Assoziationen mit gleichem Namen dargestellt, die sich aufgrund der Rollen an den Assoziationsenden unterscheiden. Wird eine Assoziation massiv überladen, so ist es sinnvoll, diese in einem separaten Diagramm darzustellen.

Die bevorzugte Darstellung der Vererbung von Assoziationen von Rollen auf ihre Unterrollen ist aus theoretischen Gesichtspunkten die Schachtelung von Rollensymbolen, d. h., Unterrollen in ihre Oberrollen einzuzeichnen, so wie es in Abbildung 4.10 gezeigt ist.[88] Dies hat sich jedoch in der Praxis nicht durchsetzen können, und so wird man auch in einem Assoziationsdiagramm Rollen mit ihren Oberrollen über einen Generalisierungspfeil verbinden wollen. Besser

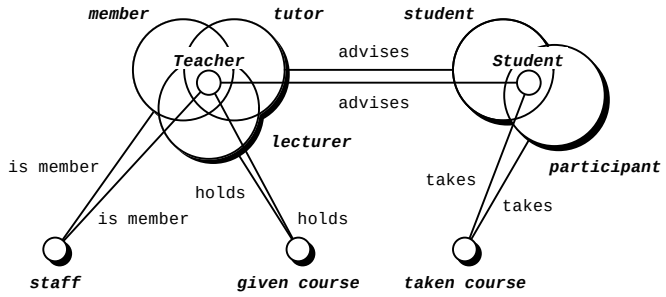

Abbildung 4.10: Assoziationsdiagramm. Unterrollen werden von ihren Oberrollen umschlossen, um das Subsumtionsverhältnis, in dem sie stehen, auszudrücken.

[88] Manche, von UML abweichende Modellierungsnotationen sehen genau das für Klassen vor, z. B. die in [Hay 1996].

ist dann jedoch, jede Überladung einer Assoziation, die eine Einschränkung darstellt (d. h., deren Enden jeweils mit Unterrollen besetzt sind), durch einen Generalisierungspfeil mit der Assoziation zu verbinden, die sie einschränkt. Beides bedeutet jedoch, daß wieder Relationen erster und zweiter Ordnung in einem Diagramm gemischt auftreten.

Die Abbildung eines Assoziationsdiagramms in eine Instanz des Metamodells ist wegen der Überladung von Assoziationen nicht ganz direkt: Für alle Assoziationen mit demselben Namen gibt es nur eine Instanz der Klasse *Association*, und für jede einzelne Assoziation mit diesen Namen (mit unterschiedlichen Rollen an den Enden) gibt es dazu eine eigene Instanz der Klasse *Signature*. Jedes Assoziationsende einer solchen einzelnen Assoziation wird in eine Instanz der Klasse *AssociationEnd* abgebildet, die einerseits Teil der Signatur ist, andererseits wiederum (über *specifies*) mit der dazugehörigen Rolle (als Instanz von *Role*) verbunden ist. Rollen und Rollenhierarchie eines Assoziationsdiagramms müssen nicht extra auf das Metamodell abgebildet werden; es wird lediglich verlangt, daß die Rollenhierarchie eines Assoziationsdiagramms mit der des Klassen- und Rollendiagramms konsistent ist. Nebenbedingungen für das Überladen, die sich in entsprechenden OCL-Ausdrücken für die Signaturen einer Assoziation (bzw. der Hierarchie der damit verbundenen Rollen) äußern würden, seien hier nicht behandelt.

Anmerkung: UML sieht für die Verbindung von Klassen und Interfaces neben der Lollipop-Notation und dem Realisierungspfeil noch die Uses-Abhängigkeit vor, die andeutet, daß eine Klasse auf ein andere zurückgreift, die zu diesem Zweck ein bestimmtes Interface realisieren muß. Diese Uses-Abhängigkeit ist zwar mit einer Assoziation verwandt, aber nicht identisch, wie sich unten zeigen wird.

Objektdiagramm

Das Assoziationsdiagramm unterscheidet grundsätzlich Rollen. Es ähnelt damit dem ursprünglichen Kollaborationsdiagramm auf Spezifikationsebene: Es läßt insbesondere offen, ob verschiedene Rollen auch von verschiedenen Instanzen gespielt werden müssen oder, wenn diese Rollen gemeinsame Unterrollen haben oder von derselben Klasse gefüllt werden, ob diese Rollen dann von derselben Instanz gespielt werden können.

In einem Objektdiagramm werden anders als in einem Klassen- oder Assoziationsdiagramm grundsätzlich Instanzen unterschieden. Welche Rollen diese Instanzen spielen, geht in einem herkömmlichen Objektdiagramm aus den Namen der Assoziationsenden (den Rollennamen) hervor, wenn diese denn eingetragen sind. Um die Rollen (und damit die Austauschbarkeit der Objekte) zu betonen,

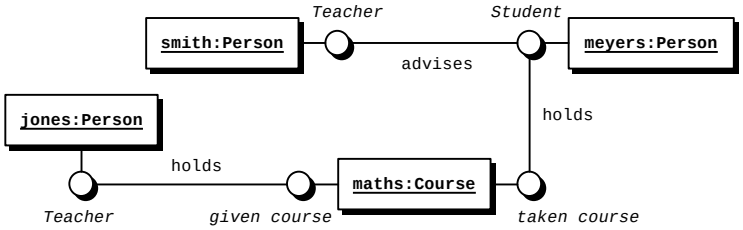

Abbildung 4.11: Objektdiagramm.

werden im geänderten Objektdiagramm die Rollen als Kreise aufgeführt. Die Verbindung der Instanzen mit ihren Rollen erfolgt sinnvollerweise wieder in Lollipop-Notation (Abbildung 4.11); es wurde aber auch schon eine andere, recht intuitive Notation (die an einen Hutständer erinnert) vorgeschlagen [Reenskaug et al. 1996, Abb. 1.10].

Kollaborationsdiagramm

UML unterscheidet zwischen einem Kollaborationsdiagramm auf Spezifikationsebene und einem auf Instanzebene. Beide sollen in erster Linie der Spezifikation von Interaktionen dienen. Wie in Abschnitt 4.1.1 bemerkt, enthält das Kollaborationsdiagramm auf Spezifikationsebene aber auch wesentliche strukturelle Aspekte, die ja in der geänderten Notation in das Assoziationsdiagramm ausgelagert wurden. Es bleibt dem Kollaborationsdiagramm, aufzuzeigen, welche Rollen (= Interfaces) von den beteiligten Klassen (Klassen als Stellvertreter für die kollaborierenden Instanzen) für die Interaktion erwartet werden. Ziel der Verwendung von Rollen ist es aber gerade, die Klassen, die die Rollenspieler liefern, als Träger der konkreten Implementierungen austauschbar zu halten.

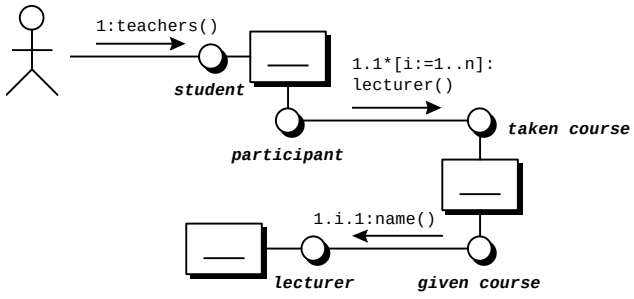

Abbildung 4.12: Kollaborationsdiagramm auf Instanzebene, in dem alle Objekte über die Rollen, die sie spielen, adressiert werden. Die Objekte müssen nicht anonym bleiben, zeigen aber insbesondere keine Klassenzugehörigkeit. Dies garantiert die in der UML-Spezifikation angesprochene maximale Flexibilität bei der Besetzung der Rollen in Kollaborationen.

Da zudem das Kollaborationsdiagramm auf Spezifikationsebene, was die Besetzung von Rollen durch Instanzen anbetrifft, nicht eindeutig ist (siehe oben), ist diese Variante des Kollaborationsdiagramms nicht mehr sinnvoll.

Es bleibt also das Kollaborationsdiagramm auf Instanzebene. In diesem werden sinnvollerweise unklassifizierte Objekte über die Rollen, die sie spielen, in Verbindung gesetzt, so wie in Abbildung 4.12 dargestellt. Die Objekte selbst haben dabei lediglich die Funktion, auf Existenz und Identität der Rollenspieler hinzuweisen, also auszudrücken, welche Funktionen in einer Interaktion von denselben und welche von verschiedenen Objekten übernommen werden. Die Interaktionen, die stattfinden, werden den Rollen zugeordnet bzw. den Assoziationen zwischen diesen. Soll eine assoziationsunabhängige Interaktion (z. B. auf der Basis von Attributen oder formalen Parametern) dargestellt werden, kann dazu, wie auch jetzt schon in UML, die Uses-Abhängigkeit (gestrichelter Pfeil) herangezogen werden (Abbildung 4.13).

Anmerkung: Attribute, die für eine Klasse, nicht jedoch eine ihrer Rollen vereinbart wurden, können im Kollaborationsdiagramm nicht herangezogen werden, da die Interaktionen klassenunabhängig bleiben sollen.

Darstellung des geänderten Metamodells in der geänderten Notation

Abbildung 4.6 zeigt die abstrakte Syntax des geänderten Metamodells in der ursprünglichen UML-Notation; sie ist wie die abstrakte Syntax des ursprünglichen Metamodells von UML selbst in einer Untermenge von UML dargestellt. Wenn die geänderte Notation die ursprüngliche ersetzen soll, dann sollte auch das geänderte Metamodell in ihr dargestellt werden können. Abbildung 4.14 zeigt eine solche Darstellung; die Aggregationen wurden hier als Attribute modelliert.

Man mag finden, daß diese Darstellung der aus Abbildung 4.6 in puncto Übersichtlichkeit unterlegen ist. Dies liegt vor allem daran, daß das Metamodell ein typisches Datenmodell ist und das Fehlen der Assoziationen im Classifier-Diagramm die für die Datenmodellierung zentralen Zusammenhänge zwischen Klassen nicht erkennen läßt. Im Assoziationsdiagramm werden diese Zusammenhänge auch nicht viel deutlicher: *fills*, *specifies* und *generalizes* sind gra-

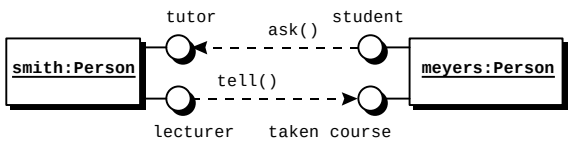

Abbildung 4.13: Kollaborationsdiagramm auf Instanzebene mit Uses-Abhängigkeiten.

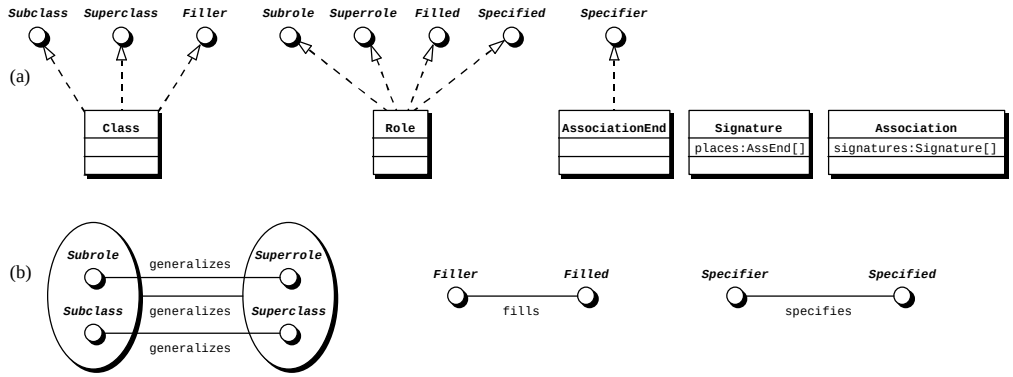

Abbildung 4.14: Das geänderte Metamodell in der geänderten Notation: (a) Classifier-Diagramm und (b) Assoziationsdiagramm. Die Beziehungen von *Association* zu *Signature* und von *Signature* zu *AssociationEnd* sind Aggregationen und deshalb nicht über Rollen und Assoziationen dargestellt.

phisch isoliert, obwohl sie über gemeinsame Rollenfüller in Verbindung stehen. Zudem sind im Assoziationsdiagramm die Zusammenhänge, die durch Aggregationen (Attribute) hergestellt werden, nicht wiederzufinden. Die Aufgabe der strikten Trennung in einem kombinierten Classifier- und Assoziationsdiagramm wie in Abbildung 4.15 schafft Klarheit, insbesondere dann, wenn auch Attribute (als Pfeile) eingezeichnet werden. Allerdings fehlt in diesem Diagrammtyp die Generalisierung von Rollen.

Die Vorteile der Trennung von Classifier- und Assoziationsdiagramm werden in einem Datenmodell wie dem Metamodell, das ja eine rein statische Angelegenheit ist, nicht immer deutlich. Vielmehr wird der Verlust an Verbundenheit zwischen den Klassen und damit des Eindrucks der transitiven Navigierbarkeit oft als Nachteil empfunden werden.[89] Dies mag einer der Gründe sein, warum sich die Beschränkung der Assoziationsenden auf Rollen, die ja von Elmasri et

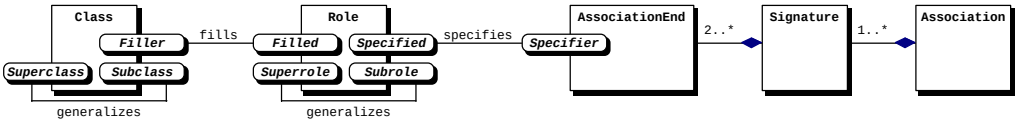

Abbildung 4.15: Geändertes Metamodell in einem kombinierten Klassen-/Rollen- und Assoziationsdiagramm; Notation nach [Riehle 2000].

[89] Jedoch ist dieser Eindruck häufig trügerisch: So ist es trotz Abbildung 4.4 beispielsweise nicht möglich, von *Staff* über *member* und *taken course* zu *Course* zu gelangen, da eben Lehrer keine Kurse nehmen, sondern geben. Daran vermag auch die Angabe von Assoziationsrichtungen nichts zu ändern.

al. [1985] und Chu und Zhang [1997] für die Datenmodellierung bereits vorgeschlagen wurde (s. Abschnitt 3.3.3), nicht durchsetzen konnte. Die Vorteile kommen erst dann zum Tragen, wenn nicht nur die Daten, sondern auch die Funktionalität eines Softwaresystems modelliert werden sollen. Nur dann erweist es sich als Vorteil, daß die großen Zusammenhänge (Assoziationen und die Kollaborationen, die sich entlang der Assoziationen entspannen) keine Klassen benennen, sondern Rollen und damit lediglich abstrakte Spezifikationen des Verhaltens und der Eigenschaften, die für die Erfüllung der Systemfunktionalität benötigt werden. Welche Klassen diese Rolle füllen, kann dazu zunächst offengelassen und von der Auswahl (vorgefertigter) Systemkomponenten abhängig gemacht werden.

Ein Vorteil der Einschränkung, daß Assoziationen an Rollen und nicht direkt an Klassen gebunden sind, wird dennoch auch hier deutlich: Wollte man genau diese Einschränkung aufheben, so bräuchte man Abbildung 4.14 nur um einen Realisierungspfeil von *Class* zu *Specified* zu ergänzen – Klassen könnten dann Rollen an den Assoziationsenden ersetzen.

4.1.5 Verhältnis zur Formalisierung

Das in Abschnitt 4.1.2 vorgestellte Metamodell ist nicht nur deutlich kompakter als das ursprüngliche der UML-Spezifikation, es entspricht auch in weiten Teilen der Spezifikation der Modellierungssprache LODWICK aus dem vorigen Kapitel. Tatsächlich besteht der Unterschied zwischen den beiden Formalisierungen vor allem in der abweichenden Handhabung von Assoziationen bzw. Relationen. So wird das Assoziationsende als eigenständiges Konzept in das Metamodell übernommen, um Nebenbedingungen wie Kardinalitäten und Navigierbarkeit unterbringen zu können. Der Vergleich des Metamodells mit LODWICK ergibt im einzelnen das folgende:

1. Die Metaklasse *Class* entspricht der Menge N von (natürlichen) Typen.
2. Die Metaklasse *Role* entspricht der Menge R von Rollen.
3. Die überladene Metarelation *generalizes* entspricht den beiden Relationen $\leq_{NN}$ und $\leq_{RR}$.
4. Die Metarelation *fills* entspricht der Relation $<_{NR}$.
5. Anders als im hiesigen Metamodell werden in LODWICK die Relationen nach Signaturen geordnet. Es entspricht aber dennoch jedes Paar (a, s), mit a Instanz von *Association* und s Instanz von *Signature*, einem Paar (p, w) aus LODWICK mit $p \in P_w$ und $w \in 2^R$, nur daß zu einem w in der

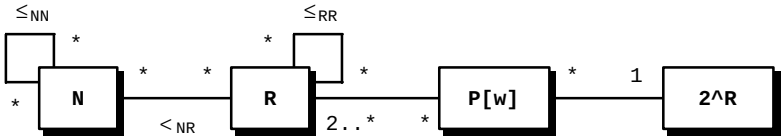

Abbildung 4.16: Metamodell LODWICKs gemäß den Definitionen aus Kapitel 3. *P*[*w*] ist eine parametrische (polymorphe) Klasse.

Regel verschiedene *p* gehören, während ein *s* immer eindeutig an ein *a* gebunden ist. Letzteres ist notwendig, weil die Verbindung von Assoziationsende zu Assoziation stets eindeutig sein muß.

6. Die Information, die in einem UML-Modell Assoziationsenden zugeordnet ist, ist in LODWICK zusammen mit der dazugehörigen Signatur Teil der Intension einer Relation. Auf die Form der Intension ist in den vorigen Kapiteln nicht weiter eingegangen worden; sie kann jedoch bezüglich der Extension mehr Bedingungen als nur die Kardinalitäten umfassen. Die Assoziationsenden finden sich jedenfalls nicht als eigenständige Modellierungselemente in LODWICK wieder.

Abbildung 4.16 stellt dem geänderten UML-Metamodell aus Abbildung 4.6 das Metamodell LODWICKs in UML-Notation gegenüber.

Das Beispielmodell der vorigen Abschnitte entspricht den folgenden Deklarationen in LODWICK:

$$\textit{Teacher} \leq_{RR} \textit{member},\ \textit{Teacher} \leq_{RR} \textit{lecturer},\ \textit{Teacher} \leq_{RR} \textit{tutor}$$
$$\textit{Student} \leq_{RR} \textit{student},\ \textit{Student} \leq_{RR} \textit{participant}$$

$$\textit{Staff} <_{NR} \textit{staff}$$
$$\textit{Person} <_{NR} \textit{Teacher},\ \textit{Person} <_{NR} \textit{Student}$$
$$\textit{Course} <_{NR} \textit{given course},\ \textit{Course} <_{NR} \textit{taken course}$$

$$\textit{smith}{:}\textit{Person},\ \textit{jones}{:}\textit{Person},\ \textit{meyers}{:}\textit{Person},\ \textit{oswald}{:}\textit{Person}$$
$$\textit{maths}{:}\textit{Course},\ \textit{cs1}{:}\textit{Course}$$

$$\textit{is member}{:}\ \textit{staff} \cdot \textit{member},\ \textit{is member}{:}\ \textit{staff} \cdot \textit{Teacher}$$
$$\textit{advises}{:}\ \textit{tutor} \cdot \textit{student},\ \textit{advises}{:}\ \textit{Teacher} \cdot \textit{Student}$$
$$\textit{holds}{:}\ \textit{lecturer} \cdot \textit{given course},\ \textit{holds}{:}\ \textit{Teacher} \cdot \textit{given course}$$
$$\textit{takes}{:}\ \textit{participant} \cdot \textit{taken course},\ \textit{takes}{:}\ \textit{Student} \cdot \textit{taken course}$$

$$[\textit{Teacher} \rightarrow \textit{smith},\ \textit{Student} \rightarrow \textit{meyers}]{:}\textit{advises}$$
$$[\textit{Student} \rightarrow \textit{meyers},\ \textit{taken course} \rightarrow \textit{maths}]{:}\textit{takes}$$
$$[\textit{Student} \rightarrow \textit{oswald},\ \textit{taken course} \rightarrow \textit{cs1}]{:}\textit{takes}$$

[*Teacher* → *jones*, *given course* → *maths*]:*holds*
[*Teacher* → *smith*, *given course* → *cs1*]:*holds*

Im Unterschied zur Instanz des Metamodells aus Abbildung 4.8 sind die anonymen Rollen, die sich aus dem Kollaborationsdiagramm von Abbildung 4.3 ergeben, nicht deklariert; entsprechend sind die Relationen anders überladen.

4.2 Bedeutung für die objektorientierte Programmierung

Während UML sowohl Rollen als auch Interfaces kennt, existiert in den gängigen objektorientierten Programmiersprachen wenn überhaupt, dann nur eins der beiden als Sprachkonstrukt: das Interface. In JAVA ist ein Interface eine Typspezifikation, die aus den Signaturen von Methoden besteht. Insbesondere macht ein Interface keinerlei Angaben zur Implementierung des Typs; es ist damit – wie eine abstrakte Klasse – nicht instanziierbar. Interfaces können dann auch in den Programmiersprachen, die kein explizites Interface-Konstrukt haben, durch abstrakte Klassen emuliert werden.

Die Idee, Interface und Implementierung eines Typs in einer Programmiersprache zu trennen, ist alt. CLU [Liskov et al. 1981] z. B. erlaubte so eine getrennte Übersetzung der sog. Cluster (Datentypen) eines Programms, ein Merkmal, das heute freilich selbstverständlich ist. Abstrakte Datentypen, die häufig in einem Atemzug mit der objektorientierten Programmierung genannt werden, geben mit ihrer Spezifikation nicht nur Signaturen, sondern auch deklarative Funktionsbeschreibungen vor und gehen damit einen wichtigen Schritt weiter. In der gängigen objektorientierten Programmierung bleiben davon jedoch allenfalls die Vor-/ Nachbedingungen und Invarianten à la EIFFEL [Meyer 1988] über.

Relativ neu dagegen ist die Idee, daß Interfaces jeweils nur eine partielle Spezifikation eines Typs sein können, daß ein Typ also mehrere Interfaces realisiert und vielleicht auch noch über Methoden verfügt, die Teil keines Interfaces sind. Wenn zudem verschiedene Klassen verschiedene, aber sich teilweise überlappende Mengen von Interfaces implementieren, dann bekommt die Trennung von Interface und Implementierung (Klasse) eine ganz neue Dimension: Sie ermöglicht die kontextabhängige Substituierbarkeit von Klassen [Steimann 2001]. Um dies in Sprachen ohne explizites Interface-Konstrukt (also mittels abstrakter Klassen) umzusetzen, benötigt man allerdings die Möglichkeit der Mehrfachtyphierarchie.

4.2.1 Der Zusammenhang von Rollen und Variablen

Was sind nun aber Rollen in der objektorientierten Programmierung? Der Abschnitt 4.1 schlägt für die Modellierung vor, Interfaces und Rollen gleichzusetzen. Interface-Deklarationen wären dann also Rollendeklarationen, und die Implements-Klausel entspräche der Rollenfüllerrelation. Instanzen von Rollen kann es damit keine geben, dafür aber Variablen, die eine Rolle zum Typ haben. Nun stehen in objektorientierten Programmen Variablen i. d. R. für Assoziationen (oder vielmehr deren Stellen) und müßten daher gemäß Abbildung 4.6 (und den Festlegungen von LODWICK) grundsätzlich Rollen, also Interfaces, zum Typ haben. Genau dies entspricht aber dem, was einige Autoren fordern: „Don't declare variables to be instances of particular concrete classes. Instead, commit only to an interface defined by an abstract class“ [Gamma et al. 1995, S. 18] und „Declare every variable and parameter with an abstract type written for that purpose“ [D'Souza & Wills 1999, S. 672].

Die Gleichsetzung von Rollen und Interfaces und die Forderung, Variablen mit Rollen als Typ zu deklarieren, ergeben eine besonders einfache Semantik von LODWICK. Demnach ist die dynamische Extension eines natürlichen Typs in einem objektorientierten Programm die Menge seiner Instanzen, die zum gegebenen Zeitpunkt existieren, und die einer Rolle die Menge aller Instanzen, die den Variablen des entsprechenden Typs oder dessen Subtypen zugewiesen sind. Der Zugriff auf Instanzen erfolgt damit ausschließlich über die Rollen, die sie spielen, und nur die Instanz selbst kann (über *self* oder *this*) auf ihre natürlichen Eigenschaften zugreifen.

4.2.2 Rollen als konzeptuelle Abstraktion von Interfaces

Während also Interfaces eine naheliegende Implementierung für Rollen sind, sind umgekehrt Rollen eine natürliche konzeptuelle Abstraktionen für die Interfaces einer Implementierung, für die es so recht sonst kein Gegenstück auf der Modellierungsebene zu geben scheint.[90] Wie sieht es aber aus mit den Interfaces in objektorientierten Programmen? Stehen sie wirklich für Rollen? Für JDK-Interfaces[91] wie *EventListener* und *Observer* (beide *java.util*) ist die Sa-

[90] Konkrete Klassen haben Spezies als konzeptuelle Abstraktion, abstrakte Genera. Attribute und Methoden dagegen sind üblicherweise Ausdruck von Relationen. Was sind also Interfaces?

[91] JDK = JAVA DEVELOPMENT KIT, hier in der Version 1.1; http://www.javasoft.com

che eindeutig: Beide sind Rollen des Model-View-Controller-Patterns und stehen dort an den Enden von Beziehungen. Doch sind auch Interfaces wie *Serializable* (*java.io*) oder *Cloneable* (*java.lang*) Rollen? Zunächst beschreiben diese Interfaces nur ein Merkmal, welches verschiedene Klassen aufweisen (oder auch nicht aufweisen) können, ohne daß dies eine Verwandtschaft der Klassen voraussetzt. Die Existenz einer Beziehung, in der Instanzen zu anderen stehen, scheint zunächst nicht prägend für diese Interfaces zu sein. Dennoch gibt es zu einem *Clonable* typischerweise auch einen, der klont, und zu einem *Serializable* für gewöhnlich eine Instanz, die die Serialisierung anstößt. *Clonable* und *Serializable* sind also durchaus auch Rollen, was sich im übrigen sprachlich ganz gut an der Endung -able ablesen läßt.[92]

Interfaces wie *Enumeration* (wiederum *java.util*) sind dagegen eher Interfaces im klassischen Sinne: Sie beschreiben ein Protokoll, also zulässige Folgen von Methodenaufrufen, deren genaue Sequenz und Ergebnis vom Zustand des Objektes, das das Protokoll realisiert, abhängt. Aber auch hier gehört zu jeder Enumeration sinnvollerweise eine Instanz, die über deren Elemente iteriert, so daß auch *Enumeration* im weiteren Sinne als Rolle *Enumerable* aufgefaßt werden kann.

Es bleibt die Gegenprobe: Sind typische JDK-Klassen keine Rollen? Für Klassen wie *String* oder *Vector*, die einfache Datenstrukturen implementieren, scheint das auf jeden Fall zu gelten. Zwar hat ein *Vector* Beziehung zu seinen Elementen, doch diese Beziehung ist die vom Ganzen mit seinen Teilen und daher nicht unbedingt Indikator einer Rolle. *String* wiederum ist eine Klasse, die wie *Person* viele Rollen hat: die eines Namen, die einer Adresse, die einer Definition etc. Genaugenommen bezeichnet jede Variable vom Typ *String* eine Rolle desselben[93], *String* selbst aber ist keine.

Andere Klassen wie *Reader* und *Writer* sind vom Namen her keine Klassen, sondern Rollen. Daß diese Klassen abstrakt sind, ist keine Erklärung, denn auch ihre konkreten Unterklassen heißen so. Tatsächlich sind *Reader* und *Writer* typische Hilfsklassen, also Klassen, die von anderen zur Erfüllung einer bestimmten Funktion gebraucht werden. Sie stehen damit für klassische Rollen

[92] Analog steht im Deutschen das durch die Endung -bar aus einem Verb gebildete Adjektiv zur Verfügung, um alternativ zum Partizip Passiv eine Rolle zu bezeichnen: das Klonbare vs. das Geklonte.

[93] Da dies jedoch zu einer Inflation von Rollen führen würde, wurde in der Änderung von UML in Abschnitt 4.1 zwischen Attributen und Assoziationen unterschieden und nur an Assoziationsenden Rollen verlangt.

einer Kollaboration. Allerdings haben diese Klassen nur diese eine Verwendung, so daß Klasse und Rolle hier als miteinander verschmolzen betrachtet werden können.

4.3 Anwendungsbeispiele

Zahlreiche Anwendungsbeispiele der Modellierung mit Rollen finden sich u. a. in [Reenskaug et al. 1996] sowie in [Riehle 2000]. Die folgenden beiden Beispiele aus der Praxis betonen insbesondere die Umsetzung der Rollen eines Modells als Interfaces in dem dazugehörigen objektorientierten Programm.

4.3.1 Das Composite pattern

Im Rahmen der Entwicklung eines Internet-basierten Lehr-/Lernservers wurde über die Einrichtung verschiedener Zugriffswege zum Unterrichtsmaterial nachgedacht. Neben der klassischen, sequentiellen Anordnung, die dem Aufbau eines Skriptes in Papierform, aber auch dem Ablauf der Vorlesung entspricht, sollen alternative Zugangswege wie ein Schlagwortverzeichnis, ein Glossar und eine Klassifikationen der Texteinheiten angeboten werden, mit denen sich Material gezielt auffinden läßt.

Zur Klassifikationen der Materialien sollen mehrstufige Standardklassifikationen wie die der ASSOCIATION OF THE COMPUTING MACHINERY (ACM)[94], aber auch andere beliebiger Schachtelungstiefe herangezogen werden können. Solche Klassifikationen sind typische Instanzen des Composite-Entwurfsmusters [Gamma et al. 1995], wie man es zum Beispiel in den baumartigen Dateisystemen vieler üblicher Betriebssysteme antrifft: Ein Kompositum umfaßt eine Anzahl Komponenten, die selbst entweder Komposita oder terminale Knoten sind. Im speziellen Fall der Klassifikation heißen die Komposita *Klassifikatoren* und die Komponenten *Einträge*. Beides sind Rollen der zweistelligen Relation *klassifizieren*.

Die Einträge einer Klassifikation sind zunächst Schlagwörter oder Terme. Wenn ein Term Subterme hat, dann spielt er neben der Rolle Eintrag auch noch die eines Klassifikators (für seine Subterme), wodurch die Rekursion entsteht.

[94] www.acm.org/class

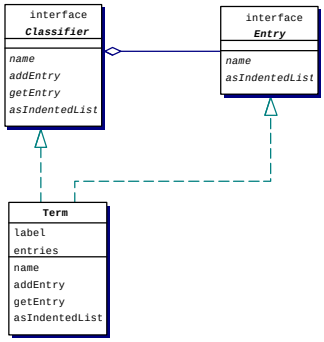

Abbildung 4.17: Unter einen Klassifikator fallen mehrere Einträge. Die Struktur einer Klassifikation ist rekursiv: Ein Eintrag kann selbst wieder Klassifikator sein. Man vergleiche dazu auch das ursprüngliche Composite pattern in Abbildung 4.21 am Ende dieses Unterabschnitts.

Schließlich ist auch ein Term, der keine Subterme hat, ein Klassifikator, nämlich der der Texte, die er klassifiziert; dazu gleich mehr.

Die beiden Rollen werden also als Interfaces modelliert und implementiert (Abbildung 4.17). Die Aggregation[95], die die Relation *klassifizieren* wiedergibt, ist ein Artefakt des Modells; sie kann zumindest in JAVA nicht als eine mit dem Interface deklarierte Instanzvariable realisiert werden. (Es wird aber der Zusammenhang zwischen *Classifier* und *Entry* auch durch die beiden Methoden *addEntry* und *getEntry* ausgedrückt.) Die eigentliche Realisierung der Aggregation muß von den Klassen übernommen werden, die die Rollen füllen – hier das Attribut *entries* der Klasse *Term*. Man ist vielleicht versucht, die Instanzva-

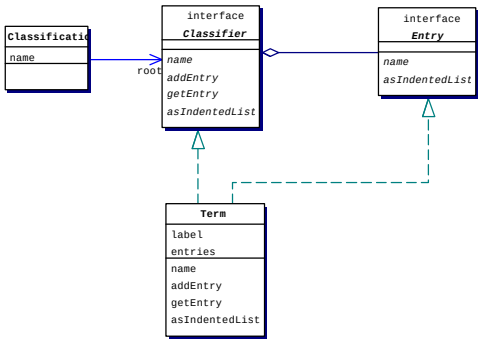

Abbildung 4.18: Einstieg in den Klassifikationsbaum über eine Instanz von *Classifier* namens *root*.

[95] Aggregationen können also durchaus auch Rollen haben.

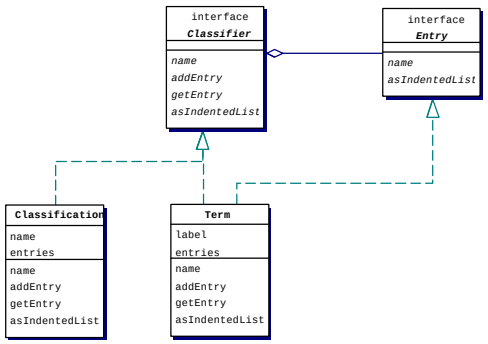

Abbildung 4.19: Eine Klassifikation spielt selbst die Rolle *Classifier*, ist aber kein Term.

riable *entries subterms* zu nennen; da unter einen Klassifikationsterm jedoch nicht nur dessen Subterme fallen, wäre das eine schlechte Wahl.

Eine Klassifikation hat also Einträge, die Klassifikatoren sein und somit selbst wieder Einträge haben können. Man wird die Klassifikation daher vielleicht an einem Attribut *root* vom Typ *Classifier* aufhängen wollen, das den Einstieg in die jeweilige Klassifikation (eine Instanz der Klasse *Classification*) bietet. Dies ist in Abbildung 4.18 dargestellt.

Da das Attribut *root* als Assoziation modelliert wurde, mag man sich berechtigterweise fragen, ob es sich bei *root* nicht eigentlich um eine Rolle handeln müßte, die gemäß den Abbildungsregeln aus Abschnitt 4.1.3 eine Oberrolle von *Classifier* sein müßte oder eine zusätzliche direkte Rolle von *Term*. Dies erscheint jedoch zum einen vergleichsweise viel Aufwand für eine simple Sache, zum anderen läßt es einen anderen Aspekt unberücksichtigt: Die Instanz, die die Rolle *root* füllt, muß nach Abbildung 4.18 eine Instanz von *Term* sein, aber welchem konkreten Term entspricht eine ganze Klassifikationshierarchie wie beispielsweise die ACM-Klassifikation? Bei näherer Betrachtung stellt sich heraus, daß eine Klassifikation selbst die Funktion eines Klassifikators hat, *Classification* also eine Klasse ist, die wie *Term* die Rolle *Classifier* füllt, aber eben nicht *Term* ist.[96] Das ist in Abbildung 4.19 dargestellt.

Wie man sieht, ist die konsequente Anwendung des Grundsatzes, daß an Assoziationsenden Rollen stehen müssen, durchaus dazu geeignet, Schwachstellen oder sogar Fehler eines Modells aufzuzeigen. Der offensichtliche Vorteil ist aber, daß sich Instanzen beliebiger Klassen, die das Interface *Entry* implemen-

[96] *Classification* hat z. B. das Attribut *name*, *Term label*. Weitere Unterschiede zwischen den beiden ergeben sich aus Verwendungen der Klassen, die hier nicht angeführt sind.

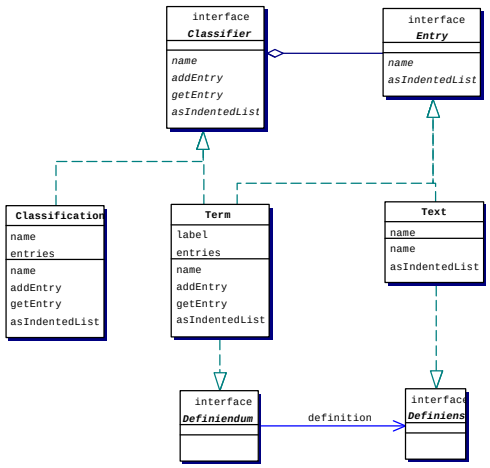

Abbildung 4.20: Einträge in einer Klassifikation sind neben Klassifikationstermen auch die zu klassifizierenden Elemente wie Text, Audio, Video etc. Die Klassen müssen lediglich die Rolle eines Eintrags füllen können, also *Entry* implementieren. Instanzen der Klassen, die eine Klassifikation ausmachen, können aber auch noch andere Aufgaben erfüllen; sie können beispielsweise die Einträge eines Glossars bilden.

tieren, als Einträge für eine Klassifikation qualifizieren. So können z. B. Texte, die in vielerlei anderen Zusammenhängen auftreten und die mannigfaltige Funktionen erfüllen, wie in Abbildung 4.20 gezeigt beinahe ohne jeden Aufwand und vor allem ohne (wie bei einem Einordnen in eine Klassenhierarchie klassifizierbarer Elemente) per Vererbung unerwünschte Nebeneffekte befürchten zu müssen, in den Klassifkationsbaum eingefügt werden.

Des weiteren erlaubt die konsequente Verwendung von Interfaces (Rollen), für einen bestimmten Zweck eingeführte Klassen vielseitig wiederzuverwenden. Terme beispielsweise treten nicht nur in Klassifikationen auf, sondern auch in Glossaren, wo sie als Definiendum durch einen Text, das Definiens, definiert werden. Beides sind zusätzliche Rollen der bereits bestehenden Klassen *Term* bzw. *Text*, die sie um weitere Funktionen anreichern. Abbildung 4.20 zeigt die dazugehörige Erweiterung des Modells. Entsprechend können Terme durch das Hinzufügen einer weiteren Rolle auch als Stichworte in einem Stichwortverzeichnis fungieren.

Es scheint also, als wäre die Modellierung mit Rollen davon geprägt, daß für besondere Zwecke oder nachträglich hinzukommende Anforderungen neue Rollen, nicht neue Klassen eingeführt werden und daß die rollenfüllenden Klassen dann um die für die neuen Funktionen notwendigen Eigenschaften ergänzt

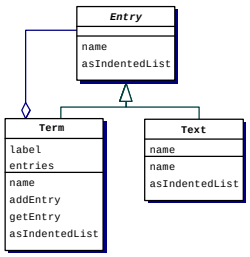

Abbildung 4.21: Das Composite-Pattern aus [Gamma et al. 1995], instanziiert für das obige Beispiel.

werden. Dies unterscheidet sie aber fundamental von der herkömmlichen objektorientierten Modellierung und vor allem von der Implementierung, wonach für neue Funktionen neue (Sub)Klassen eines bestehenden Modells oder Programms gebildet werden sollen.

Abschließend sei noch der Vergleich mit dem ursprünglichen Composite pattern [Gamma et al. 1995] in Abbildung 4.21 angeführt. Etwas unglücklich daran ist, daß sich Teil und Ganzes der Aggregation nicht (wie in Abbildung 4.17) gleichberechtigt gegenüberstehen, sondern daß das Ganze dem Teil untergeordnet ist. Außerdem sind weder Terme noch Texte ihrem Wesen (Genus) nach Einträge. Es steht hier eben die Vererbung bzw. Typsubsumtion im Mittelpunkt und nicht, wie in der Modellierung mit Rollen, die Relation mit ihren unmittelbar Beteiligten.

4.3.2 Die Partnerrollen eines Finanzdienstleisters

Ein Finanzdienstleister unterhält als Unternehmen zahlreiche Beziehungen, nicht nur zu seinen Kunden, sondern auch zu seinen Mitarbeitern, zu Subunternehmern etc. Zusammenfassend kann man diese alle als Partner des Dienstleisters bezeichnen, wobei ein Partner entweder eine Person oder ein Personenzusammenschluß ist.

Jeder dieser Partner steht nun in mindestens einer, meistens aber mehreren bestimmten Beziehungen zum Finanzdienstleister. So ist ein Finanzierungsnehmer häufig auch Kontoinhaber und ein Mitarbeiter (Partner im Anstellungsverhältnis) meistens auch Kunde (er nimmt eine Finanzdienstleistung seines Arbeitgebers in Anspruch). Jede solche Beziehung definiert eine (eigene) Rolle, die ein Partner spielen kann. Ein einfaches Modell, das das darstellt, ist in Abbildung 4.22 zu sehen.

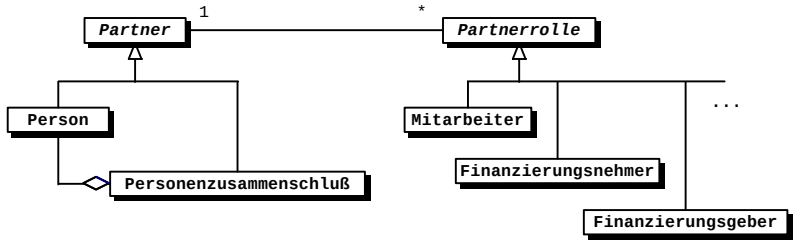

Abbildung 4.22: Die Partner eines Finanzdienstleisters und ihrer Rollen. Der Finanzdienstleister selbst und die Beziehungen, in denen er zu seinen Partnern steht, sind implizit; letztere stecken in den Rollen.

Genaugenommen ist, wenn man einen Rollenbegriff wie den von LODWICK ansetzt, *Partner* selbst eine Rolle, wenn auch eine recht abstrakte. Um Partner zu sein, muß ein Objekt nämlich in einer Partnerschaftsbeziehung zu einem anderen stehen, und im gegebenen Fall steht ja schließlich jeder Partner in Partnerschaftsbeziehung zum Finanzdienstleister, der selbst der andere Partner ist. Wenn man den Finanzdienstleister, aus dessen Sicht das Modell erstellt wird, und die Beziehung zu seinen Partnern als implizit gegeben ansieht, dann kann man allerdings Partner auch als natürlichen Typ ansehen (und die Partnerrollen als seine Rollen), denn außer Partnern gibt es keine Personen oder Personenzusammenschlüsse im Modell.

Was dem Modell aber zunächst zu fehlen scheint, ist die Darstellung der verschiedenen Beziehungen, die der Finanzdienstleister zu den unterschiedlichen Partnerrollen unterhält, die diese Partnerrolle wesentlich prägen und die nicht zuletzt die Basis für die Interaktionen zwischen dem Finanzdienstleister und seinen Partnern in diesen Rollen darstellen. Man kann jedoch leicht argumentieren, daß diese Beziehungen implizit in den Klassen stecken, die die Partnerrollen repräsentieren. Das würde dann die These von Henderson-Sellers [1998] belegen, nämlich daß reifizierte (d. h. als Klassen dargestellte) Assoziationen immer Rollen sind (s. Abschnitt 3.3.4). Ein Partnermodell, das die Reifizierung rückgängig macht, d. h., das diese Rollen expliziert, ist in Abbildung 4.23 dargestellt. Man beachte, daß auch der Finanzdienstleister verschiedene Rollen annehmen kann, daß er insbesondere auch selbst Finanzdienstnehmer eines Finanzdienstleisterpartners sein kann.

Auf größeren Widerstand stößt allerdings die Konsequenz, daß nach diesem Modell einzelne Typen, hier *Partner*, *Person* und *Personenzusammenschluß*, eine Vielzahl von Rollen, nämlich alle Partnerrollen, füllen sollen. Auf die Implementierung übertragen heißt das, daß die dazugehörigen Klassen alle Inter-

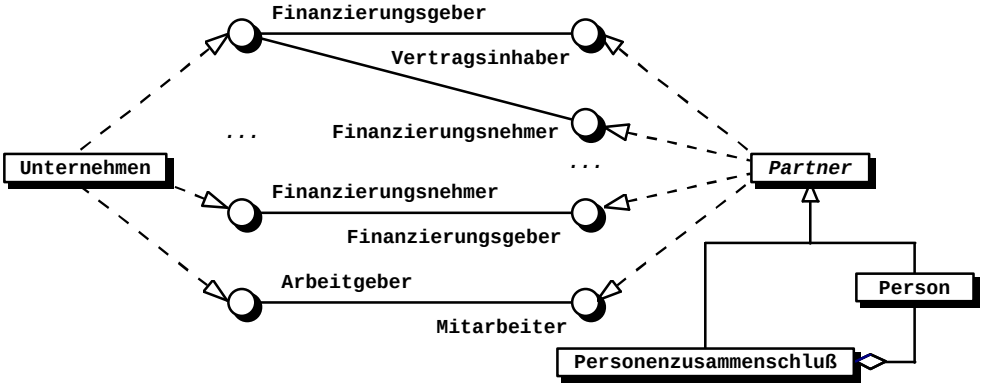

Abbildung 4.23: Die Partnerrollen als Rollen der natürlichen Typen *Unternehmen* und *Person*. *Unternehmen* ist der Typ des Finanzdienstleisters, aus dessen Sicht das Modell erstellt wird. Er ist deshalb *Person* nicht untergeordnet.

faces realisieren müssen. Das führt zwangsläufig zu recht umfangreichen (und deswegen scheinbar unhandlichen) Klassendefinitionen. Die Frage, die man hier allerdings stellen muß, ist, ob irgend etwas einfacher wird, wenn man, wie bei der Modellierung von Rollen als beigeordnete Instanzen üblich, den Code zur Implementierung der Rollen auf mehrere Klassen aufteilt, oder ob hier nicht vielleicht nur die Unsitte, eine Klasse mit einer Datei (und damit mit einer atomaren Manipulationseinheit) gleichzusetzen, den Blick verstellt. Natürlich kann man die Implementierung einer Klasse in logische Einheiten aufteilen (z. B. für jede Rolle eine) und diese getrennt pflegen. Nicht alle Programmiersprachen unterstützen jedoch die Bildung verschiedener Name spaces innerhalb einer Klasse, wodurch die Abgrenzung dieser Einheiten zu einem Problem werden kann. Auf der anderen Seite ist es immer dann, wenn Rollen in ihrem Verhalten nicht unabhängig voneinander sind, sogar einfacher, diese Abhängigkeit umzusetzen, wenn die Verhaltens- und Zustandsspezifikationen nicht über mehrere Klassen verteilt sind.

Die zweite große Frage, die sich aufdrängt, ist, wie die verschiedenen Zustände einer Person, die ein und dieselbe Rolle mehrfach spielt, voneinander getrennt werden sollen. Wie z. B. soll man die verschiedenen Kontonummern einer Person, die mehrere Konten innehat, voneinander trennen, wenn sie die Rolle *Kontoinhaber* nur einmal implementieren kann? Die Antwort ist einfach: Zur Beziehung, die eine Person als Kontoinhaber zu ihrem Finanzdienstleister hat, gehört immer auch ein Konto, und die Kontonummer ist Eigenschaft des Kontos, nicht der Rolle *Kontoinhaber*. Eine Eigenschaft *Kreditwürdigkeit* dagegen

ist dann eine sinnvolle Eigenschaft der Rolle, wenn der Kontoinhaber nur eine Kreditwürdigkeit bei seinem Finanzdienstleister hat, und zwar unabhängig davon, ob er ein oder mehrere Konten hat. Wäre die Eigenschaft *Kreditwürdigkeit* an eine Instanz der Klasse *Kontoinhaber* gebunden, so müßte die Kreditwürdigkeit des Kontoinhabers mit mehreren Konten erst ermittelt werden oder alle Attribute müßten den gleichen Wert haben.

Kapitel 5

Schluß

Die in den vorangegangenen Kapiteln vorgestellte Modellierungssprache LODWICK bringt als wesentliche Neuerung gegenüber anderen, vergleichbaren Modellierungssprachen ein elementar und konzeptuell eigenständig angelegtes Rollenkonzept. Zwar ist eine gewisse Ähnlichkeit von Rollen und Genera in LODWICK durchaus erkennbar, jedoch sind Rollen im Gegensatz zu Genera immer über Relationen definiert, und die Klassifikation der Individuen, die Rollen (über das Rollenspielen) nach sich ziehen, ist ihrem Charakter nach eine dynamische Mehrfachklassifikation.

5.1 Zusammenfassung

Die vorgelegte Definition eines Rollenbegriffs stellt Rollen neben natürliche Typen und verbindet beide über eine neue Relation mit spezieller Semantik. Diese Relation, die Rollenfüllerrelation, gleicht der Subtypenrelation insofern als

- sie eine Relation zweiter Ordnung ist,
- die statische Extension einer Rolle im Regelfall genau die Vereinigung der statischen Extensionen der Typen ist, die sie füllen, und
- sich der absolute Teil der Intension einer Rolle auf alle Typen vererbt, die diese Rolle füllen.

Jedoch unterscheidet sie sich von der Subtypenrelation, indem

- die dynamische Extension einer Rolle die meiste Zeit eine Teilmenge der Vereinigung der dynamischen Extensionen der Typen, die sie füllen, ist und
- nicht die vollständige Intension einer Rolle – nämlich der relative Teil nicht – auf die Typen, die sie füllen, übertragen wird.

Dabei deckt der relative Teil der Intension einer Rolle ihre Abhängigkeit von den Relationen ab, an deren Stellen sie auftritt, also insbesondere, an welchen Beziehungen Individuen teilhaben müssen, damit sie unter die Rolle fallen (der Extension der Rolle zugerechnet werden). Dieser relative Teil vererbt sich von Rollen zu ihren Unterrollen, aber eben nicht von Rollen zu den sie füllenden Typen, denn die sind per definitionem von Relationen unabhängig. Der absolute Teil einer Rollenintension hingegen vererbt sich auch über die Rollenfüllerrelation hinweg – er entspricht der Intension eines natürlichen Typs und ist eine partielle Spezifikation der Individuen, die eine Rolle spielen.

Die vorgelegte Definition von Rollentypen einschließlich ihres Verhältnisses zu natürlichen Typen unterscheidet sich damit grundsätzlich

- von der Darstellung von Rollen als Spezialisierungen oder Generalisierungen natürlicher Typen, die eine Möglichkeit zur Objektmigration voraussetzen und zudem unabhängig von Relationen sind, und

- von der Darstellung von Rollen als beigeordnete Instanzen, die ein Individuum der Realität als Verbund von Individuen im Modell repräsentieren.

Insbesondere wird der für den Rollenbegriff charakteristischen Mehrfachklassifikation in dieser Arbeit dadurch Rechnung getragen, daß ein Individuum (indirekt) Instanz mehrerer Typen sein kann, nämlich neben seiner Spezies mit deren Genera auch noch all der Rollen, die diese füllen, und der dynamischen Klassifikation dadurch, daß die Extension einer Rolle davon abhängt, ob die Individuen in den zu der Rolle gehörenden Beziehungen stehen. Dabei ist auch im Modell ein Individuum in einer Rolle stets das Individuum selbst, und nicht eine zusätzliche Instanz.

Das Bemerkenswerte dieser Rollendefinition schließlich ist, daß sie nicht nur viele Aspekte des semantisch so reichen Konzepts auf natürliche Art und Weise abdeckt, sondern daß sie auch von der Metamodellierung über die Modellierung bis hin zur Implementierung eines Modells durchgängig anwendbar ist. Der folgende Abschnitt soll die Eigenschaften noch einmal im einzelnen herausstellen.

5.2 Bewertung

Die Definition des Rollenbegriffs, die LODWICK beinhaltet, ist in vielen die Modellierung umfassenden und sie berührenden Disziplinen einheitlich und durchgängig anwendbar. Insbesondere gilt:

1. Sie deckt einen bestehenden sprachlichen und konzeptuellen Rollenbegriff auf natürliche Weise ab.
2. Sie ist ontologisch begründet.
3. Sie ist leicht formalisierbar.
4. Sie fügt sich nahtlos in die Konzepte und Prinzipien der Datenmodellierung ein und erweitert diese um ein häufig fehlendes Strukturelement.
5. Sie gestattet eine Neuordnung der objektorientierten Modellierung und dabei insbesondere eine klarere Trennung zwischen Struktur- und Verhaltensaspekten eines Modells. Darüber hinaus erweitert sie den Polymorphiebegriff der Objektorientierung um eine der eigentlichen Wortbedeutung entsprechende Dimension.
6. Sie läßt sich direkt in objektorientiertes Design und Software umsetzen und harmoniert dabei sowohl mit produktivitätssteigernden Ansätzen wie Entwurfsmustern und Frameworks als auch mit anerkannten Programmierpraktiken.

Wohlgemerkt: All soll eine einzige Definition leisten. Wie sie das tut, wird im folgenden noch einmal Punkt für Punkt dargelegt.

Natürlichkeit

Rollen sind keine Besonderheit der Modellierung – sie kommen in den verschiedensten Konzeptualisierungen vor und finden auch in der unmittelbarsten Äußerung menschlichen Denkens, der natürlichen Sprache, ihren Niederschlag.

Fillmores Kasusrollen [Dirven & Radden 1987], die semantischen, thematischen oder θ-Rollen Handelnder, Gegen-Handelnder, Objekt etc. sind abstrakte Rollen von Prädikaten, die in einem Satz oder anderen sprachlichen Ausdruck inhaltlich mit konkreten Entitäten besetzt werden, die diese Rollen spielen. Legt

man eine aristotelische Klassifikation zugrunde, d. h., geht man davon aus, daß jede Entität Individuum einer Spezies ist und daß sich Spezies zu Genera zusammenfassen lassen, dann kann ein Sinnaufzählungslexikon diesen abstrakten Rollen Spezies und Genera gegenüberstellen, die die Rollen füllen (sog. Selektionsbeschränkungen). Und wenn die Rollen Fillmores zu abstrakt sind, um durch eine allgemeingültige Zuordnung von Rollen zu Rollenfüllern sinnvolle von unsinnigen Sätzen abzugrenzen[97], dann wird man die Abstraktion weniger weit treiben und konkretere Rollen, wie sie etwa durch grammatikalische Variationen des Prädikats (z. B. Lodwicks „murtherer“ und „murthered“ [Lodwick 1647]) entstehen, verwenden.

Wenn sich also die (abstrakten) Rollen eines Prädikats und die dazugehörigen Selektionsbeschränkungen soweit annähern, daß die Zuordnung universell gültig, d. h. auch auf andere Auftreten derselben Rollen anwendbar ist, dann wird die Gegenüberstellung von Spezies/Genera und den Rollen, die sie füllen, vom Prädikat unabhängig und entspricht genau der Deklaration der Relation $<_{NR}$ in LODWICK. Mit der Deklaration der Elemente dieser Rollenfüllerrelation werden die linguistischen Konzepte der semantischen Rolle und der Selektionsbeschränkung (in Form der Angabe von Genera) also gewissermaßen zusammengeführt. Daß es in der linguistischen Praxis so dann doch nicht geht, liegt nicht an einer unzureichenden Auffassung von Rollen, sondern daran, daß sich natürliche Sprache nicht in das starre Korsetts eines Sinnaufzählungslexikon zwängen läßt (s. z. B. [Pustejovsky 1995]).

In anderen als natürlichsprachlichen Konzeptualisierungen wird ebenfalls gern zwischen Rollen und den Typen, aus denen die Rollen gefüllt werden, unterschieden. Sowas konzeptuelle Graphen [1984] z. B. haben einen Rollenbegriff, der wie der LODWICKs an Relationen hängt, der jedoch Rollen als Subtypen natürlicher Typen auffaßt und auch sonst nicht klar zwischen Rollen und Typen trennt. Tatsächlich aber ist die Unterscheidung zwischen Rollen und natürlichen Typen fundamental ontologischer Natur [Chandrasekaran et al. 1999], und die Frage, ob und warum es überhaupt ein eigenständiges Rollenkonzept für die Modellierung geben sollte, eine ontologische.

Ontologische Begründung

Der Rollenbegriff von LODWICK scheint also durchaus ein natürlicher zu sein. Er ist darüber hinaus aber auch ontologisch begründet. Die beiden wesentlichen

[97] das ist sicher der Fall

Merkmale einer solchen Grundlegung hat Guarino [1992] genannt: Es sind dies die die Identität eines Individuums betreffende (semantische) Rigidität und die die Notwendigkeit des Bestehens einer Beziehung betreffende Fundierung. Beide werden durch die Definition des Rollenbegriffs in Kapitel 3 auf einfache Weise abgedeckt [Steimann 2000b]; es gibt damit ein objektives Kriterium dafür, was in einem Modell als Rolle und was als natürlicher Typ dargestellt werden sollte.

Es existieren natürlich auch andere ontologische Fassungen des Rollenbegriffs (z. B. [Guarino et al. 1994; Sowa 2000]), die ebenfalls ihre Vorzüge haben. Dies ist nicht verwunderlich, denn auch Ontologien sind nicht unabhängig von ihrem Zweck, und die Wahl bzw. Festlegung einer Ontologie wird immer von der beabsichtigten Verwendung abhängen. Daß die hier getroffene Definition des Rollenbegriffs für die Modellierung aber eine besonders natürliche ist, mag man daran ablesen, daß sie sich ohne Probleme auf die Metaebene (die Ebene der Metamodellierung) heben läßt (Abschnitt 3.2.8); für manche andere gilt das nicht.

Leichte Formalisierbarkeit

Anders als in vielen Ontologien ist in der Modellierungssprache LODWICK *Rolle* keine sog. Top-level-Kategorie, von denen die Rollen eines Modells (also *Angestellter*, *Student* etc.) per Spezialisierung abgeleitet werden. *Rolle* ist nicht die allgemeinste aller Rollen; vielmehr sind in LODWICK alle Rollen Instanzen vom Metatyp *Rolle*[98] (bzw. Elemente von *R*). Entsprechendes gilt für Genera und Spezies[99]. Daß dies sinnvoll ist, wird schon daran deutlich, daß die Begriffe Genus und Spezies ja die Knoten des Baums des Porphyrios [Sowa 1992] klassifizieren, also insbesondere nicht selbst zu dessen Knoten gehören können. Ob und wie man den Zusammenhang von Genus, Spezies und Rolle selbst wieder in einem Modell (dem Metamodell) klärt, insbesondere ob man diese Konzepte in eine Taxonomie wie die Sowas oder Guarinos einordnet, ist für die Modellierung selbst unerheblich; ich beschränke mich hier darauf, sie als Elemente der Metasprache zu deklarieren, die Verwendung derselben in einem Modell durch die Syntax zu klären und verbanne weitere ontologische Betrachtungen (außer vielleicht der Feststellung, daß es sich bei allen dreien um Typen handelt) in das Reich der ontologischen Begründung.

[98] *Rolle* kann dann selbst keine Rolle sein, denn Rollen sind nicht instanziierbar.

[99] Da eine Spezies per definitionem keine Subtypen haben kann, kann *Spezies* entweder nicht die allgemeinste aller Spezies sein oder muß selbst ein Genus sein.

Formal äußert sich die Einführung des Rollenbegriffs in zwei Veränderungen gegenüber der rollenlosen Modellierung:

1. in der Einführung einer von der der natürlichen Typen disjunkten Menge von Rollen, die in einer Rollenhierarchie angeordnet sind und deren Elemente selbst keine Instanzen haben dürfen, die aber über eine Rollenfüllerrelation gezielt mit natürlichen Typen in Verbindung gebracht werden, deren Individuen die jeweilige Rolle spielen können, und
2. in der Bedingung, daß Relationen nur auf Rollen deklariert werden dürfen.

Dadurch wird insbesondere gewährleistet, daß Rollen nur im Kontext von Beziehungen eingenommen werden können und daß ein rollenspielendes Individuum im Modell immer nur durch eine Instanz repräsentiert wird, und zwar unabhängig davon, wie viele Rollen es gerade spielt.

Zugegebenermaßen ist die Ausstattung der Sprachen FREGE und LODWICK nicht sonderlich üppig – selbst wenn man den statischen Aspekt als angemessen berücksichtigt ansieht, ist die dynamische Betrachtung doch die einfachste aller möglichen. Auf der anderen Seite macht gerade diese Betrachtung keinerlei Annahmen außer der primitiven, daß sich nämlich Dynamik über eine Menge möglicher Folgen von Szenarien oder Zuständen des Modells äußert. Die Formalisierung bleibt damit offen für alle möglichen Formen zeitdiskreter Modellierung. Wesentlich für diese Arbeit ist, daß der dynamische Aspekt der Rollendefinition durch diese Betrachtung hinreichend unterstützt wird, und das ist, wie Kapitel 3 gezeigt hat, sicher der Fall.

Verträglichkeit mit der Datenmodellierung

Anders als in anderen Ansätzen zur Rollenmodellierung und vor allem anders als in Datenmodellen ohne explizites Rollenkonzept muß in LODWICK ein Individuum, das mehrere Rollen spielt, nicht durch mehrere Objekte (Instanzen der Typen, die die Rollen repräsentieren) vertreten werden. Dadurch können erhebliche Probleme in der Datenmodellierung vermieden werden. Insbesondere wird damit eine der ursprünglichen Forderungen an ein Rollenkonzept in der Datenmodellierung, nämlich das folgende Dilemma zu lösen, erfüllt:

> *„[...] most conventional file records and relational file n-tuples are role oriented. These files typically deal with employees, customers, patients, or students, all of which are role types. This role orientation is in contrast with integrated database theory which has taught that each record should represent all aspects of some one entity in the real*

> *world. This difference in viewpoint has caused a great deal of confusion. The reason for the confusion is understood when it is realized that neither the roles of the real world nor the entities of the real world are a subset of the other.“* [Bachman & Daya 1977]

Ein anderer wichtiger Beitrag von LODWICK für die Datenmodellierung ist, daß das Verhältnis von Rollentypen und natürlichen Typen richtiggestellt wird. Anders als häufig angenommen, sind Rollen keine Subtypen der natürlichen Typen, deren Instanzen die Rollen spielen (und zwar nicht einmal dann, wenn man nur die Extensionen der Typen im Auge hat und das gleichzeitige Spielen mehrere Rollen einer Instanz sowie den dynamischen Rollenwechsel per dynamischer und Mehrfachklassifikation zuläßt) [Steimann 1999b; Steimann 2001]; statisch betrachtet sind es sogar eher Supertypen. Wenn das gegen die Intuition oder Beobachtung ist, dann liegt das daran, daß zu einem bestimmten Zeitpunkt (oder in einer möglichen Welt) tatsächlich fast immer nur ein Teil der Instanzen eines Typs eine Rolle spielt, die dynamische (oder modale) Extension der Rolle also nur eine Teilmenge der des Typs ist, aus der die Rolle ihre Individuen rekrutiert. A priori ist jedoch nicht feststellbar, welche Individuen das sein werden, und so gilt die Teilmengenbeziehung insbesondere für die statische Modellierung nicht, denn sonst stünden nicht alle Instanzen eines Typs gleichartig und -berechtigt nebeneinander.

Neuordnung der objektorientierten Modellierung

Die Statik der objektorientierten Modellierung basiert im wesentlichen auf zwei Strukturierungsmöglichkeiten: der Klassenhierarchie und den Relationsdeklarationen. Dazu kommen bei einer operationalen Betrachtung der statischen Abhängigkeiten die an Interaktionen ausgerichteten Kollaborationen, in denen nicht Klassen, sondern Rollen die maßgeblichen Strukturierungselemente sind. Auch wenn Methoden à la OORAM [Reenskaug et al. 1996; Riehle & Gross 1998] eine Synthese von Kollaborationen zu Klassendiagrammen und damit eine Zuordnung von Rollen zu Klassen vorsehen, bleibt dabei ein ganz wesentlicher Punkt unberücksichtigt: Rollen sind nicht nur über Interaktionen identifizierbar, sondern auch über statische, interaktionsferne Beziehungen, wie u. a. das Beispiel aus der UML-Spezifikation [OMG 1999; Övergaard 1999] unfreiwillig zeigt [Steimann 2000d; 2001]. Entsprechend weist die Definition von Rollen in LODWICK diese als (statische) Strukturelemente aus, die gleichberechtigt zwischen Klassen und Relationen stehen und deshalb (wie in UML nur die Rollennamen) unabhängig von jeder Interaktion stets auch Bestandteil von Klassendiagrammen sind. Kollaborationsdiagramme können sich deswegen auf

die Darstellung von Interaktionen konzentrieren, was ihrer eigentlichen Aufgabe entspricht.

Die ursprüngliche Einführung des Rollenbegriffs für das Netzwerkmodell wartete mit einem interessanten Nebeneffekt auf, der ein praktisches Problem löste. Die Elemente eines Records können nämlich im Netzwerkmodell durchaus Entitäten verschiedener Typen sein. Beim Iterieren durch diese Elemente, die alle in derselben Beziehung zu ihrem Owner stehen, stellt sich dann das Problem, daß Unterprogramme in Abhängigkeit vom Typ einer Entität aufgerufen werden müssen, auch wenn es sich im Grunde um dieselben Funktionen handelt. Durch die Einführung von Rollentypen, die alle Entitäten mit einer einheitlichen Schnittstelle versehen, konnte dies umgangen werden, eine Eigenschaft, die heute die objektorientierte Programmierung unter dem Schlagwort Polymorphismus für sich beansprucht.

Tatsächlich wartet auch LODWICK mit einem Polymorphiebegriff auf, der durch die Definition von Rollen entsteht und der sich von dem der Generalisierungs- und Abstraktionshierarchien fundamental unterscheidet. Die mögliche Polymorphie von Objekten (nicht von Funktionen oder Operationen) wird damit zu einer eigenen, von der Generalisierung (oder Klassenhierarchie) unabhängigen Dimension der Modellierung, die sich in der Rollenfüllerrelation ausdrückt [Steimann 1999b].

Direkte Umsetzbarkeit

Zwar sollte sich die Modellierung als Instrument vor allem der Analysephase eines Softwareprojekts weit genug von Implementationsaspekten fernhalten, doch die immer wieder versprochene Nahtlosigkeit des Übergangs gerade von objektorientierter Analyse und Design (und damit von Modell) zu Software verlangt, daß sich die Modellierungskonzepte zumindest ohne größere Transformationen in die objektorientierten Programmiersprachen umsetzen lassen. Wenn man also ein Rollenkonzept für die Modellierung einführt, dann sollte dies so gestaltet sein, daß es sich möglichst direkt auf die vorhandenen Konstrukte objektorientierter Programmiersprachen übertragen läßt.

Die Fassung des Rollenbegriffs in LODWICK erlaubt es, Rollen mit Interfaces gleichzusetzen, wodurch eine direkte Umsetzung von Rollen im Modell in ein Programm (ohne Umweg beispielsweise über ein Entwurfsmuster) möglich ist. Dabei sprechen gegen die Verwendung von Interfaces nach der Definition JAVAs und UMLs (nicht aber der von CORBA) die Restriktionen, denen sie dort unterliegen. Voraussetzung für die Verwendung von Klassen zur Spezifi-

kation von Interfaces ist jedoch die Möglichkeit der Mehrfachvererbung, damit eine Klasse neben ihrem Genus (Oberklasse, von der sie Struktur und Implementierung erbt) auch noch die Rollen deklarieren kann, die sie füllt. Außerdem ist dann klarzustellen, welche der Oberklassen einer Klasse ihre Rollen sind und welche ihre Genera, eine Unterscheidung, die syntaktisch nicht gegeben ist.

In einem objektorientierten Programm wird der temporäre Charakter des Rollenspielens durch die Bindung einer Instanz an eine oder mehrere Variablen ausgedrückt: Die Instanz spielt die mit der Variable verbundene Rolle und all deren Oberrollen genau so lang, wie sie an die Variable gebunden ist. Man beachte, daß so jede Instanz jede Rolle, für die sie deklariert ist, gleichzeitig und beliebig oft spielen kann – es ist dies genau die Umsetzung der minimalen Standarddefinition des relativen Teils der Intension von Rollen in LODWICK. In Abwandlung Quines bekannten Ausspruchs „to be is to be the value of a variable“ [1939] gilt also hier:

To play a role is to be the value of a variable.

5.3 Ausblick

Ziel meiner Arbeit war es, einen über viele Teilgebiete der Modellierung einheitlich anwendbaren Rollenbegriff zu präsentieren, der so grundsätzlich angelegt ist, daß er ohne Probleme in die Riege der fundamentalen Modellierungskonzepte Klasse, Relation und Vererbung aufgenommen werden kann. Das ist, wie ich meine, gelungen. Ob und wieweit dieser Rollenbegriff angenommen und in die Modellierungspraxis eingeführt werden wird, wird sich zeigen müssen. Von entscheidender Bedeutung scheint mir dabei zu sein, ob die bisherige UML-Rollendefinition durch eine der hier ausgearbeiteten entsprechende ersetzt werden wird.

Dennoch bleibt noch einiges zu klären, nicht nur in Bezug auf UML. Die folgenden Abschnitte stellen einen kurzen Überblick über die wesentlichen offenen Punkte dar.

UML

Die in Kapitel 4 vorgeschlagene Änderung des UML-Metamodells ist an Klassen als Classifier ausgerichtet. In UML zählen aber nicht nur Klassen, sondern so verschiedene Konzepte wie Actors, Use cases, Subsystems und Components zu den Classifiern. Es ist also zu untersuchen, inwieweit sich der Rollenbegriff auch auf diese Arten von Classifiern sinnvoll ausdehnen läßt. Die Tatsache, daß sie alle über Assoziationen in Verbindung gesetzt werden können, spricht zunächst dafür. Doch ob beispielsweise Überladungen (und damit ggf. auch Unterrollen) für Beziehungen zwischen Systemkomponenten sinnvoll sind, ist zumindest nicht offensichtlich.

Andere Modellierungskonzepte von UML wie das Generalisieren von Assoziationen und die genaueren Umstände der sog. Projektion im Kontext von Kollaborationen sind in der UML-Spezifikation nicht genau genug beschrieben, um deren Ersetzbarkeit durch Überladung in allen Fällen zu klären. Die spezielle Rolle von Assoziationsklassen und ihr Verhältnis zu Rollen blieb hier ebenfalls unberücksichtigt, was mit der unter theoretischen Gesichtspunkten fragwürdigen Stellung als Zwitter zwischen Relationen und Klassen begründet wird.

Formulierung der Intensionen

Die Arbeit befaßt sich zwar mit statischen und dynamischen Aspekten der Modellierung, doch wie insbesondere das dynamische Modell genau spezifiziert werden soll, bleibt offen. Wegen dessen enormer Komplexität wird ein Formalismus mit einer gewissen Ausdrucksstärke vonnöten sein, um die Intensionen der Spezies-, der Genus-, der Rollen- und der Relationssymbole formulieren zu können. Andere Arbeiten gehen hier erheblich weiter, z. B. [Jungclaus 1993; Kappel & Schrefl 1996; Andersen 1997; Snoeck et al. 1999], doch deren Rollenbegriffe erfüllen nicht die hier diskutierten allgemeinen Anforderungen an ein solches Konzept. Es wäre also zu untersuchen, inwieweit sich die in dieser Arbeit getroffene Rollendefinition auch auf konkretere Spezifikationsformen von Modellen übertragen läßt.

Eingang in die Programmierpraxis

Zwar raten verschiedene Autoren, für die Variablen eines Programms keine Klassen, sondern nur Interfaces (ggf. durch abstrakte Klassen repräsentiert) als Typen anzugeben, doch der damit verbundene Mehraufwand bei Entwurf und Codierung wirkt abschreckend. Insbesondere muß bei jeder neu eingeführten Variable geprüft werden, ob die damit verbundene Rolle schon in (Form eines entsprechenden Interfaces) eingeführt wurde oder vielleicht erst noch zu definieren ist. Hier ist auf jeden Fall ein Umdenken der Designer und Implementierer erforderlich, mit all den damit verbundenen Schwierigkeiten.

Schön wäre es, die Nachfolgerin von JAVA in der Gunst der Software-Gemeinde hätte ein Sprachkonstrukt mit Namen Rolle. Dieses würde im wesentlichen die Funktion der JAVA-Interfaces übernehmen, sollte aber nicht der m. E. unnötigen Einschränkung, daß sie keine Attribute haben dürfen und somit nicht für die direkte Umsetzung von gerichteten Assoziationen in Pointer taugen, unterliegen. Da Interfaces als Interfaces konzeptuell praktisch ohne Bedeutung, Rollen aber eine sehr natürliche konzeptuelle Abstraktion sind, würde dies vielleicht zur einer breiteren Akzeptanz des Interface- bzw. dann Rollenkonzeptes in der objektorientierten Programmierung führen. Noch schöner wäre es, diese Sprache verzichtete bei der Auflösung überladener Methodendeklarationen auf Laufzeitoptimierungen und übertriebene Vorsicht bei der statischen Typprüfung zugunsten einer unter theoretischen Gesichtspunkten wünschenswerten Gleichbehandlung aller Parameter. Dann nämlich ließe sich die Überladung von Relationen, wie sie auf Modellierungsebene vorkommt, auch

direkt in ein Programm übernehmen, ohne unliebsame Überraschungen wie in Abschnitt 2.3.4 beschrieben fürchten zu müssen.

Literatur

[Abadi & Cardelli 1996] M Abadi, L Cardelli *A Theory of Objects* (Springer, New York 1996).

[Abiteboul & Hull 1987] S Abiteboul, R Hull „IFO: A formal semantic database model“ *ACM Transactions on Database Systems* 12:4 (1987) 525–565.

[Aitchison et al. 1997] J Aitchison, A Gilchrist, D Bawden *Thesaurus construction and use: a practical manual* 2. Ausgabe (Aslib, London 1997).

[Aït-Kaci & Nasr 1986] H Aït-Kaci , R Nasr „Login: a logic programming language with built-in inheritance“ *Journal of Logic Programming* 3 (1986) 185–215.

[Al-Ahmad & Steegmans 1999] W Al-Ahmad, E Steegmans „Improving support for specialization inheritance“ *Journal of Object-Oriented Programming* 11:8 (1999) 29–36.

[Albano et al. 1993] A Albano, R Bergamini, G Ghelli, R Orsini „An object data model with roles“ in: R Agrawal, S Baker, D Bell (eds) *Proceedings of the 19th International Conference on Very Large Databases* (Dublin, Morgan Kaufmann 1993) 39–51.

[Andersen 1997] EP Andersen *Conceptual Modeling of Objects: A Role Modeling Approach* Dissertation (Universität Oslo, 1997).

[Armstrong & Mitchell 1994] JM Armstrong, RJ Mitchell „Uses and abuses of inheritance“ *Software Engineering Journal* 9:1 (1994) 19–26.

[Bachman & Daya 1977] CW Bachman, M Daya „The role concept in data models“ in: *Proceedings of the 3rd International Conference on Very Large Databases* (1977) 464–476.

[Bachman 1980] CW Bachman „The role data model approach to data structures“ in: SM Deen, P Hammersley *Proceedings of the International Conference on Data Bases*, University of Aberdeen, July 1980 (Heyden & Son, 1980) 1–18.

[Bachman 1989] CW Bachman „A personal chronicle: Creating better information systems, with some guiding principles“ *IEEE Transactions on Knowledge and Data Engineering* 1:1 (1989) 17–32.

[Bäumer et al. 1997] D Bäumer, D Riehle, W Siberski, M Wulf „The role object pattern“ *Proceedings of the 1997 Conference on Pattern Languages of Programs (PLoP ´97)* (1997).

[Becker & Walter 1977] H Becker, H Walter *Formale Sprachen* (Vieweg, Braunschweig 1977).

[Bläsius et al. 1989] KH Bläsius, U Hedtstück, CR Rollinger (Hrsg) *Sorts and Types in Artificial Intelligence* Lecture Notes in Artificial Intelligence 418 (Springer 1989).

[Bock & Odell 1998] C Bock, JJ Odell „A more complete model of relations and their implementation: roles“ *Journal of Object-Oriented Programming* 11:2 (1998) 51–54.

[Booch 1994] G Booch *Object-Oriented Analysis and Design with Applications* (Addison-Wesley, Menlo Park 1994).

[Brachman & Schmolze 1985] R Brachman, J Schmolze „An overview of the KL-One knowledge representation scheme“ *Cognitive Science* 9:2 (1985) 171–216.

[Bühler 1934] K Bühler *Sprachtheorie: Die Darstellungsfunktion der Sprache* (Gustav Fischer, Jena 1934).

[Buschmann 1998] F Buschmann „Falsche Annahmen (Teil 2)“ *OBJEKTspektrum* 4 (1998) 84–85.

[Cardelli & Wegner 1985] L Cardelli, P Wegner „On understanding types, data abstracting and polymorphism“ *ACM Computing Surveys* 17:4 (1985) 471–522.

[Carnap 1947] R Carnap *Meaning and Necessity* deutsche Übersetzung: *Bedeutung und Notwendigkeit* (Springer, Wien 1972)

[Carnap 1954] R Carnap *Einführung in die symbolische Logik* 3. Auflage (Springer, Wien 1968)

[Carpenter 1992] B Carpenter *The Logic of Typed Feature Structures: with Applications to Unification Grammars, Logic Programs and Constraint Resolution* (Cambridge University Press 1992).

[Chandrasekaran et al. 1999] B Chandrasekaran, JR Josephson, VR Benjamins „What are ontologies, and why do we need them?“ *IEEE Intelligent Systems* (Januar/Februar 1999) 20–26.

[Chen 1976] PP Chen „The entity-relationship model: Towards a unified view of data“ *ACM Transactions on Database Systems* 1:1 (1976) 9–36.

[Chu & Zhang 1997] WW Chu, G Zhang „Associations and roles in object-oriented modeling“ in: DW Embley, RC Goldstein (Hrsg) *Proceedings of the 16th International Conference on Conceptual Modeling: ER '97* (Springer-Verlag, Berlin 1997) 257–270.

[Coad & Mayfield 1999] P Coad, M Mayfield *JAVA Design: Building Better Apps and Applets* 2. Ausgabe (Yourdon Press, Upper Saddle River 1999).

[Codd 1970] EF Codd „A relational model of data for large shared data banks“ *Communications of the ACM* 13:6 (1970) 377–387.

[Codd 1979] EF Codd „Extending the database relational model to capture more meaning“ *ACM Transactions on Database Systems* 4 (1979) 397–434.

[Computerwoche 1991] „SNI setzt große Hoffnungen in Esprit-Projekt ‘Ithaca’“ *Computerwoche* 36 (1991).

[D’Souza & Wills 1999] DF D’Souza, AC Wills *Objects, Components and Frameworks with UML: The CATALYSIS Approach* (Addison-Wesley, Reading 1999).

[Dirven & Radden 1987] R Dirven, G Radden *Fillmore’s Case Grammar: A Reader* (Julius Groos Verlag, Heidelberg 1987).

[Eco 1994] U Eco *Die Suche nach der vollkommenen Sprache* (Verlag C. H. Beck, München 1994).

[Ehrich et al. 1989] HD Ehrich, M Gogolla, UW Lipeck *Algebraische Spezifikation abstrakter Datentypen* (B.G. Teubner, Stuttgart 1989).

[Ehrich et al. 1993] HD Ehrich, R Jungclaus, G Denker „Object roles and phases“ in: UW Lipeck, G Koschorreck (Hrsg) *Proceedings of the International Workshop on Information Systems — Correctness and Reusability (IS-CORE'93)* Informatik-Berichte 1/93 (Universität Hannover, Institut für Informatik, Hannover 1993) 114–121.

[Elmasri et al. 1985] R Elmasri, J Weeldreyer, A Hevner „The category concept: an extension to the entity relationship model“ *Data & Knowledge Engineering* 1:1 (1985) 75–116.

[Elmasri et al. 1991] R Elmasri, I EI-Assal, V Kouramajian „Semantics of temporal data in an extended ER model“ in H Kangassalo (Hrsg) *Entity-Relationship Approach: The Core of Conceptual Modelling (ER `90)* (1991) 239–254.

[Elmasri & Navathe 1994] R Elmasri, SB Navathe *Fundamentals of Database Systems* 2. Ausgabe (Benjamin Cummings, Redwood City 1994).

[Essink & Erhart 1991] LJB Essink, WJ Erhart „Object modelling and system dynamics in the conceptualization stages of information systems development“ in: F van Assche, B Moulin, C Rolland (Hrsg) *Proceedings of the IFIP TC8/WG8.1 Working Conference on the Object Oriented Approach in Information Systems* (North Holland, 1991) 89–116.

[Falkenberg 1976] E Falkenberg „Concepts for modelling information“ in: GM Nijssen (Hrsg) *Proceedings of the IFIP Conference on Modelling in Data Base Management Systems* (North-Holland, Amsterdam 1976) 95–109.

[Fillmore 1971] CJ Fillmore „Types of lexical information“ in: DD Steinberg, LA Jakobovits *Semantics: An Interdisciplinary Reader in Philosophy, Linguistics and Psychology* (Cambridge University Press 1971) 370–392.

[Firesmith & Henderson-Sellers 1998] DG Firesmith, B Henderson-Sellers „Upgrading OML to version 1.1 part 2: additional concepts and notations“ *Journal of Object-Oriented Programming* 11:5 (1998) 61–67.

[Fowler & Scott 1997] M Fowler, K Scott *UML Distilled* (Addison-Wesley, Reading 1997).

[Fowler 1997a] M Fowler *Analysis Patterns: Reusable Object Models* (Addison-Wesley, Menlo Park 1997).

[Fowler 1997b] M Fowler „Dealing with roles“ Supplement zu

[Freeman 1981] M Freeman „The QUA link“ in: JG Schmolze, RJ Brachman *Proceedings of the* 1981 *KL-one Workshop* Report No. 4842 (Bolt Beranek and Newman Inc., 1982) 55–65.

[Gamma et al. 1995] E Gamma, R Helm, R Johnson, J Vlissides *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley 1995).

[Gamma 1996] E Gamma „The extension objects pattern“ *Proceedings of the 1996 Conference on Pattern Languages of Programs (PLoP ´96)* (1996).

[Gardies 1991] JL Gardies „Intension/Extension“ in: H Burkhardt, B Smith (Hrsg) *Handbook of Metaphysics and Ontology* (Philosophia-Verlag, Muenchen 1991).

[Glubrecht et al. 1983] JM Glubrecht, A Oberschelp, G Todt *Klassenlogik* (Bibliographisches Institut, Mannheim 1983).

[Gogolla 1994] M Gogolla *An Extended Entity-Relationship Model: Fundamentals and Pragmatics* Lecture Notes in Computer Science 767 (Springer-Verlag, Berlin 1994).

[Goguen & Meseguer 1992] JA Goguen, J Meseguer „Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations“ *Theoretical Computer Science* 105:2 (1992) 217–273.

[Goldberg & Robson 1983] A Goldberg, D Robson *Smalltalk-80: the language and its implementation* (Addison-Wesley, Reading 1983).

[Gosling et al. 1996] J Gosling, B Joy, G Steele *The Java Language Specification* (Addison-Wesley, Reading 1996).

[Gottlob et al. 1996] G Gottlob, M Schreffl, B Röck „Extending object-oriented systems with roles“ *ACM Transactions on Information Systems* 14:3 (1996) 268–296.

[Guarino 1992] N Guarino „Concepts, attributes and arbitrary relations: some linguistic and ontological criteria for structuring knowledge bases“ *Data & Knowledge Engineering* 8 (1992) 249–261.

[Guarino et al. 1994] N Guarino, M Carrara, P Giaretta „An ontology of meta-level categories“ in: J Doyle, E Sandewall, P Torasso *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning* (Morgan Kaufmann, San Francisco 1994) 270–280.

[Hainaut 1996] JL Hainaut „Specification preservation in schema transformations: application to semantics and statistics“ *Data & Knowledge Engineering* 19: 2 (1996) 99–134.

[Hainaut et al. 1997] JL Hainaut, JM Hick, J Henrard, D Roland „Understanding the implementation of is-a relations“ in: B Thalheim (Hrsg) *15th International Conference on Conceptual Modeling: ER '96* (Springer, Berlin 1997) 42–57.

[Halbert & O'Brien 1987] DC Halbert, PD O'Brien „Using types and inheritance in object-oriented programming“ *IEEE Software* 4:5 (1987) 71–79.

[Halpin 1995] TA Halpin *Conceptual Schema and Relational Database Design* (Prentice-Hall, Sidney 1995).

[Halpin & Proper 1995] TA Halpin, HA Proper „Subtyping and polymorphism in object-role modelling“ *Data & Knowledge Engineering* 15 (1995) 251–281.

[Hammer & McLeod 1981] M Hammer, D McLeod „Databse description with SDM: A semantic database model“ *ACM Transactions on Database Systems* 6:3 (1981) 351–386.

[Hay 1996] DC Hay *Data Model Patterns: Conventions of Thought* (Dorset Haous Publishing, New York 1996).

[Henderson-Sellers 1998] B Henderson-Sellers „Towards the formalization of relationships for object modelling” in: C Mingins, R Duke, B Meyer (Hrsg) *Technology of Object-Oriented Languages and Systems Proceedings (TOOLS 25)* (IEEE Computer Society, Los Alamitos 1998) 267–283.

[Henderson-Sellers et al. 1998] B Henderson-Sellers, A Simons, H Younessi *The OPEN Toolbox of Techniques* (Addison Wesley Longman, Harlow 1998).

[Herbrand 1930] J Herbrand in: W Goldfarb (Hrsg) *Logical Writings* (Reidel, Dordrecht 1971).

[Hitz & Kappel 1999] M Hitz, G Kappel *UML@Work: von der Analyse zur Realisierung* (dpunkt-Verlag, Heidelberg 1999).

[Hogg et al. 1992] J Hogg, D Lea, A Wills, D deChampeaux, R Holt „The Geneva convention on the treatment of object aliasing“ *OOPS Messenger* (1992).

[Hürsch 1994] WL Hürsch *Should Superclasses be Abstract?* in: M Tokoro, R Pareschi (Hrsg) *Proceedings of the 8th European Conference on Object-Oriented Programming (ECOOP '94)* Lecture Notes in Computer Science 821 (Springer 1994) 12–31.

[Husserl 1901] E Husserl *Logische Untersuchungen* 2. Band, 1. Teil, 3. Auflage (Max Niemeyer, Halle a. d. Saale 1922).

[ISO 1987] ISO 704 *Principles and Methods of Terminology* (International Organization for Standardization, Genève 1990).

[ISO 1990a] ISO 1087 *Terminology — Vocabulary* (International Organization for Standardization, Genève 1990).

[ISO 1990b] ISO/IEC 10027 *Information Technology — Information Resource Dictionary System (IRDS) Framework* (International Organization for Standardization/International Electrotechnical Commission, Genève 1990).

[ISO 1993a] ISO *Information Technology – Open Systems Interconnection – Structure of Management Information: Management Information Model* (ISO/IEC 1993); auch als: CCITT Recommendation X.720 (ITU).

[ISO 1993b] ISO *Information Technology – Open Systems Interconnection – Structure of Management Information: Guidelines for the Definition of Managed Objects* (ISO/IEC 1993); auch als: CCITT Recommendation X.722 (ITU).

[Jackson & Cameron 1983] MA Jackson, JR Cameron *System Development* (Prentice Hall, Englewood Cliffs 1983).

[Jungclaus et al. 1991] R Jungclaus, G Saake, T Hartmann, C Sernadas *Object-Oriented Specification of Information Systems: The TROLL Language* Informatik Berichte 91-04 (TU Braunschweig, Braunschweig 1991).

[Jungclaus 1993] R Jungclaus *Modeling of Dynamic Object Systems — A Logic-Based Approach* Dissertation (Verlag Vieweg, Wiesbaden 1993).

[Kappel & Schrefl 1989] G Kappel, M Schrefl „A behavior integrated entity-relationship approach for the design of object-oriented databases“ in: C Batini (Hrsg) *Proc. of the 7th International Conference on Entity-Relationship Approach* (Elsevier, Amsterdam 1989) 311–328.

[Kappel & Schrefl 1991] G Kappel, M Schrefl „Object/behaviour diagrams“ in: *Proceedings of the 7th International Conference on Data Engineering* (IEEE Computer Society Press, Los Alamitos 1991) 530–539.

[Kappel & Schrefl 1996] G Kappel, M Schrefl *Objektorientierte Informationssysteme: Konzepte, Darstellungsmittel, Methoden* (Springer, Wien 1996).

[Kappel et al. 1998] G Kappel, W Retschitzegger, W Schwinger „A comparison of role mechanisms in object-oriented modeling“ in: K Pohl, A Schärr, G Vossen (Hrsg) *Modellierung '98* Bericht Nr. 6/98-I (Angewandte Mathematik und Informatik, Universität Münster 1998) 105–109.

[Kent 1978] W Kent *Data and Reality: Basic Assumptions in Data Processing Reconsidered* (North-Holland, Amsterdam 1978).

[Khoshafian & Copeland 1986] SN Khoshafian, GP Copeland „Object identity“ in: *OOPSLA SIGPLAN Notices* 22:12 (1986) 406–416; auch in: S Zdonik, D Maier (eds) *Readings in Object-Oriented Database Systems* (Morgan Kaufmann, San Mateo 1990) 37–46.

[Kilian 1991] MF Kilian „A note on type composition and reusability“ *OOPS Messenger* 2:3 (1991) 24–32.

[Kristensen 1995] BB Kristensen „Object-oriented modeling with roles“ in: J Murphy, B Stone (Hrsg) *OOIS'95: Proceedings of the International Conference on Object Oriented Information Systems* (Springer-Verlag, Berlin 1996) 57–71.

[Kristensen & Østerbye 1996] BB Kristensen, K Østerbye „Roles: conceptual abstraction theory and practical language issues“ *Theory And Practice of Object Systems* 2:3 (1996) 143–160.

[Kutzler & Lichtenberger 1983] B Kutzler, F Lichtenberger *Bibliography on Abstract Data Types* (Springer, Berlin 1983).

[Lakoff 1987] G Lakoff *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind* (The Chicago University Press, Chicago 1987).

[Li & Lochovsky 1998] Q Li, FH Lochovsky „ADOME: An advanced object modeling environment“ *IEEE Transactions on Knowledge and Data Engineering* 10:2 (1998) 255–276.

[Liao 1996] SY Liao „Perspective application: an efficient tool to identify roles and subclasses in object-oriented modeling“ *Report on Object Analysis and Design* 2:5 (1996) 40–43.

[Lieberman 1986] H Lieberman „Using prototypical objects to implement shared behavior in object oriented systems“ *ACM SIGPLAN Notices* 21:11 (1986) 214–223.

[Liskov et al. 1981] B Liskov et al. *CLU: Reference Manual* Springer LNCS 114 (Springer, Berlin 1981).

[Lodwick 1647] *A Common Writing* abgedruckt in: V Salmon *The Works of Francis Lodwick* (Longman, London 1972).

[Maier et al. 1985] D Maier, D Rozenshtein, J Stein „Representing roles in universal scheme interfaces” *IEEE Transactions on Software Engineering* 11:7 (1985) 644–652.

[Martin & Odell 1992] J Martin, JJ Odell *Object-Oriented Analysis and Design* (Prentice-Hall, Englewood Cliffs 1992).

[Maughan & Durnota 1995] G Maughan, B Durnota „MON: An object relationship model incorporating roles, classification, publicity and assertions“ *Proceedings of the 1994 International Conference on Object Oriented Information Systems OOIS'94* (Springer-Verlag, London 1995) 166–180.

[Mellor 1999] S Mellor„Advanced methods and tools for a precise UML“ in: R France, B Rumpe (Hrsg) *«UML» ´99 – The Unified Modeling Language* 2nd International Conference, Springer LNCS 1723 (Springer, Berlin 1999) 99–115.

[Mendelzon et al. 1994] AO Mendelzon, T Milo, E Waller „Object migration“ *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems PODS* (1994) 232–242.

[Meseguer & Goguen 1993] J Meseguer, JA Goguen „Order-sorted algebra solves the constructor-selector, multiple representation, and coercion problems“ *Information and Computation* 103 (1993) 114–158.

[Meyer 1988] B Meyer *Object-oriented Sofware Construction* (Prentice-Hall, Englewood Cliffs 1988).

[Meyers 1977] *Meyers enzyklopädisches Lexikon* (Bibliographisches Institut, Mannheim 1977).

[Mrva 1999] M Mrva „Role-centered design for evolution“ in: *Proceedings of the 1999 IEEE Conference on Engineering of Computer-Based Systems (IEEE, 1999)*.

[Mylopoulos et al. 1990] J Mylopoulos, A Borgida, M Jarke, M Koubarakis *Telos: A Language for Representing Knowledge about Information Systems* MIP - 9011 (Fakultät für Mathematik und Informatik, Universität Passau 1990).

[Nijssen & Halpin 1989] GM Nijssen, TA Halpin Conceptual Schema and Relational Database Design: a Fact Oriented Approach (Prentice Hall, New York 1989).

[Norrie 1994] *Proc. of the 12th International Conference on the Entity-Relationship Approach: ER '93* (Springer-Verlag, Berlin 1993) 390–401.

[Norrie et al. 1996] MC Norrie, A Steiner, A Würgler, M Wunderli „A model for classification structures with evolution control“ in: B Thalheim *15th International Conference on Conceptual Modeling Proceedings: ER '96* (Springer-Verlag, Berlin 1996) 456–471.

[Oberschelp 1962] A Oberschelp „Untersuchungen zur mehrsortigen Quantorenlogik“ *Mathematische Annalen* 145 (1962) 297–333.

[Odell 1992] JJ Odell „Dynamic and multiple classification“ *Journal of Object-Oriented Programming* 8:4 (1992) 45–48.

[Ogden & Richards 1923] CK Ogden, IA Richards *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism* 10. Ausgabe, Erstausgabe von 1923 (Routledge & Kegan Paul Ltd, London 1972).

[OMG 1994] *Relationship Service Specification* (OMG TC Document 94.5.5, 1994).

[OMG 1999] OMG *Unified Modeling Language Specification* Version 1.3 (http://www.omg.org, Juni 1999).

[Orfali et al. 1998] R Orfali, D Harkey, J Edwards *Instant CORBA* (Addison Wesley Longmans GmbH, Bonn 1998).

[Övergaard 1999] G Övergaard „A formal approach to collaborations in the Unified Modeling Language“ in: R France, B Rumpe (Hrsg) *«UML» ´99 – The Unified Modeling Language* 2nd International Conference, Springer LNCS 1723 (Springer, Berlin 1999) 99–115.

[Papazoglou 1991] MP Papazoglou „Roles: A methodology for representting multifaceted objects“ in: D Karagiannis (Hrsg) *Proceedings of the International Conference on Database and Expert Systems Applications (DEXA 91)* (Berlin, Springer-Verlag 1991) 7–12.

[Papazoglou 1995] MP Papazoglou „Unraveling the semantics of conceptual schemas“ *Communications of the ACM* 38:9 (1995) 80–94.

[Pernici 1990] B Pernici „Objects with roles“ in: FH Lochovsky, RB Allen (Hrsg) *Proceedings of the Conference on Office Information Systems* (SIGOIS Bulletin 11:2/3, ACM Press, New York 1990) 205–215.

[Pustejovsky 1995] J Pustejovsky *The Generative Lexicon* (MIT Press, Cambridge 1995).

[Quine 1939] WVO Quine „Designation and existence“ *The Journal of Philsophy* 36 (1939).

[Reenskaug et al. 1992] T Reenskaug, EP Andersen, AJ Berre, A Hurlen, A Landmark, OA Lehne, E Nordhagen, E Nêss-Ulseth, G Oftedal, AL Skaar, P Stenslet „OORASS: seamless support for the creation and Maintenance of object oriented systems“ *Journal of Object-Oriented Programming* 5:6 (1992) 27–41.

[Reenskaug et al. 1996] T Reenskaug, P Wold, OA Lehne *Working with Objects — The OOram Software Engineering Method* (Manning, Greenwich 1996).

[Reimer 1985] U Reimer „A representation construct for roles“ *Data & Knowledge Engineering* 1:3 (1985) 233–251.

[Reimer 1991] U Reimer *Einführung in die Wissensrepräsentation* (Teubner, Stuttgart 1991).

[Renouf & Henderson-Sellers 1995] DW Renouf, B Henderson-Sellers „Incorporating roles into MOSES“ in: C Mingins, B Meyer *Proceedings of the 15th International Conference on Technology of Object-Oriented Languages and Systems: Tools 15* (Prentice Hall, 1995) 71–82.

[Richardson & Schwarz 1991] J Richardson, P Schwartz „Aspects: Extending objects to support multiple, independent roles“ in: J Clifford, R King (eds) *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data* (SIGMOD Record 20:2, ACM Press, 1991) 298–307.

[Riehle & Gross 1998] D Riehle, T Gross. „Role model based framework design and integration“ in: *Proceedings of the 1998 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA ’98)* (1998) 117–133.

[Riehle 2000] D Riehle *Framework Design: A Role Modeling Approach* Dissertation (ETH, Zürich 2000).

[Rumbaugh et al. 1991] J Rumbaugh, M Blaha, W Premerlani, F Eddy, W Lorensen *Object-Oriented Modeling and Design* (Prentice Hall, Englewood Cliffs 1991).

[Rumbaugh et al. 1998] J Rumbaugh, I Jacobsen, G Booch *The Unified Modeling Language Reference Manual* (Addison-Wesley, Reading 1998).

[Russell 1922] B Russell Einleitung zu: L Wittgenstein *Tractatus Logico-Philosophicus* Erstausgabe 1922 (Routledge, London 1998).

[Ryant 1997] I Ryant „Why inheritance means extra trouble“ *Communications of the ACM* 40:10 (1997) 118–119.

[Salmon 1972] V Salmon *The Works of Francis Lodwick* (Longman, London 1972).

[Schank 1975] RC Schank *Conceptual Information Processing* (North-Holland, Amsterdam 1975).

[Schmidt 1938] A Schmidt „Über deduktive Theorien mit mehreren Sorten von Grunddingen“ *Mathematische Annalen* 123 (1962) 485–506.

[Schoenfeld 1996] A Schoenfeld „Domain specific patterns: conversions, persons and roles, and documents and roles“ *Proceedings of the 1996 Conference on Pattern Languages of Programs (PLoP ´96)* (1996).

[Scholz 1941] H Scholz „Gottlob Frege“ in: H Hermes, F Kambartel, J Ritter *Mathesis Universalis* (Benno Schwalbe & Co., Basel 1961) 268–278.

[Scholz 1942] H Scholz „Leibniz“ in: H Hermes, F Kambartel, J Ritter *Mathesis Universalis* (Benno Schwalbe & Co., Basel 1961) 128–151.

[Scholz 1959] H Scholz *Abriss der Geschichte der Logik* (Verlag Karl Alber, Freiburg 1959).

[Schrefl 1991] M Schrefl „Behavior modeling by stepwise refining behavior diagrams“ in: H Kangassalo (Hrsg) *Entity-Relationship Approach: The Core of Conceptual Modelling (ER `90)* (1991) 119–134.

[Sciore 1989] E Sciore „Object specialization“ *ACM Transactions on Infomation Systems* 7:2 (1989) 103–122.

[Shieber 1986] SM Shieber *An Introduction to Unification-Based Approaches to Grammar* CSLI Lecture Notes 4 (Ventura Hall, Stanford 1986).

[Smith & Smith 1977] JM Smith, DCP Smith „Database abstractions: aggregation and generalization“ *ACM Transactions on Database Systems* 2:2 (1977) 105–133.

[Smolka & Aït-Kaci 1989] G Smolka, H Aït-Kaci „Inheritance hierarchies: semantics and unification“ *Symbolic Computation* 7:3 (1989) 343–370.

[Snoeck & Dedene 1996] M Snoeck, G Dedene „Generalization/specialization and role in object oriented conceptual modeling“ *Data & Knowledge Engineering* 19:2 (1996) 171–195.

[Snoeck et al. 1999] M Snoeck, G Dedene, M Verhelst, AM Depuyat *Object-oriented Enterprise Modelling with MERODE* (University Press Leuven, 1999).

[Sowa 1984] JF Sowa *Conceptual Structures: Information Processing in Mind and Machine* (Addison-Wesley 1984).

[Sowa 1988] JF Sowa „Using a lexicon of canonical graphs in a semantic interpreter“ in: MW Evens (Hrsg) *Relational Models of the Lexicon: Representing Knowledge in Semantic Networks* (Cambridge University Press, Cambridge 1988) 113–137.

[Sowa 1992] JF Sowa „Semantic networks“ in: SC Shapiro (Hrsg) *Encyclopedia of Artificial Intelligence* 2. Ausgabe (John Wiley, New York 1992) 1493–1511.

[Sowa 2000] JF Sowa *Knowledge Representation: Logical, Philosophical, and Computational Foundations* (Brooks/Cole, Pacific Grove 2000).

[Steimann 1996] F Steimann *Canonical Conceptual Graphs: Problems and Mines of Solutions* unveröffentlichtes Manuskript (Institut für Medizinische Informatik, Universität Hildesheim, 1996).

[Steimann 1998] F Steimann „Dependency parsing for medical language and concept representation“ *Artificial Intelligence in Medicine* (1998) 77–86.

[Steimann et al. 1999] F Steimann, P Fröhlich, W Nejdl „Model-based diagnosis for open systems fault management“ *AI Communications* 12:1&2 (1999) 5–17.

[Steimann & Nejdl 1999] F Steimann, W Nejdl „Modellierung und Ontologie“ Technischer Bericht (www.kbs.uni-hannover.de, 1999).

[Steimann 1999a] F Steimann „Letter to the editor“ *Journal of Object-Oriented Programming* 12:2 (1999) 8–9.

[Steimann 1999b] F Steimann „On the representation of roles in object-oriented and conceptual modelling“ *Data & Knowledge Engineering* im Druck.

[Steimann 2000a] F Steimann „Abstract class hierarchies, factories, and stable designs“ *Communications of the ACM* 43:4 (2000) 109–111.

[Steimann 2000b] F Steimann „Eine Grundlegung des Rollenbegriffs für die objektorientierte Modellierung (mit dem Vorschlag einer Änderung von UML)“ in: J Ebert, U Frank (Hrsg) *Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik: Modellierung 2000* (Fölbach-Verlag, Koblenz 2000) 55–69.

[Steimann 2000c] F Steimann „The family pattern“ *Journal of Object-Oriented Programming* (2000) im Druck.

[Steimann 2000d] F Steimann „A radical revision of UML’s role concept“ Technischer Bericht (www.kbs.uni-hannover.de, 2000).

[Steimann 2001] F Steimann „Role = Interface: a merger of concepts“ *Journal of Object-Oriented Programming* (2001) im Druck.

[Su 1991] J Su „Dynamic constraints and object migration“ in: GM Lohman, A Sernadas, R Camps (eds) *Proceedings of the 17th International Conference on Very Large Databases* (VLDB Endowment Press, Sarattoga 1991) 233–242.

[Taivalsaari 1993] A Taivalsaari „OOP with modes“ JOOP 6:3 (1993) 25–32.

[ter Hoftsede & van der Weide 1993] AHM ter Hofstede, ThP van der Weide „Expressiveness in conceptual data modelling“ *Data & Knowledge Engineering* 10:1 (1993) 65–100.

[Tesnière 1965] Tesnière *Éléments de syntaxe structurale* 2nd edition (Librairie C Klincksieck, Paris 1965).

[VanHilst & Notkin 1996] M VanHilst, D Notkin „Using role components to implement collaboration-based designs“ in: *Proc. of the 1996 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA ’96) SIGPLAN Notices* 31:10 (1996) 359–369.

[von Kutschera 1989] F von Kutschera *Gottlob Frege: Eine Einführung in sein Werk* (de Gruyter, Berlin 1989).

[Wagner 1989] CF Wagner „Implementing abstraction hierarchies“ in: C Batini (Hrsg) *Proc. of the 7th International Conference on Entity-Relationship Approach* (Elsevier, Amsterdam 1989) .

[Wegner & Zdonik 1988] P Wegner, B Zdonik „Inheritance as an incremental modification mechanism or what like is and isn’t like“ in S Gjessing, K Nygaard (Hrsg) *ECOOP ’88: European Conference on Object-Oriented Programming* Proceedings (Springer-Verlag, Berlin 1988) 55–77.

[Wegner 1987] P Wegner „The object-oriented classification paradigm“ in: P Shriver, P Wegner (eds) *Research Directions in Object-Oriented Programming* (MIT Press 1987) 479–560.

[Whitehead & Russell 1910] AN Whitehead, B Russell *Principia Mathematica: Vorwort und Einleitungen* Originalausgabe von 1910, deutsche Übersetzung 1932 von H Mokre (Suhrkamp, Frankfurt 1994).

[Wieringa et al. 1993] RJ Wieringa, R Jungclaus, P Hartel, T Hartmann, G Saake „OMTROLL – object modeling in TROLL“ in: UW Lipeck, G Koschorreck (Hrsg) *Proceedings of the International Workshop on Information Systems — Correctness and Reusability (IS-CORE’93)* Informatik-Berichte 1/93 (Universität Hannover, Institut für Informatik, Hannover 1993).

[Wieringa et al. 1994] R Wieringa, W de Jonge, P Spruit „Roles and dynamic subclasses: a modal logic approach“ in: *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)* (1994).

[Wieringa et al. 1995] R Wieringa, W de Jonge, P Spruit „Using dynamic classes and role classes to model object migration“ *Theory and Practice of Object Systems* 1:1 (1995) 61–83.

[Wieringa 1998] R Wieringa „A survey of structured and object-oriented software specification methods and techniques“ *ACM Computing Surveys* 30:4 (1998) 459–527.

[Winkler 1992] JFH Winkler „Objectivism: ‘class’ considered harmful“ *Communications of the ACM* 35:8 (1992) 128–130.

[Wirth 1988] N Wirth „Type extensions“ *ACM Transactions of Programming Languages and Sytems (TOPLAS)* 10:2 (1988) 204–214.

[Wittgenstein 1922] L Wittgenstein *Tractatus Logico-Philosophicus* Erstausgabe 1922 (Routledge, London 1998).

[Wong et al. 1997] RK Wong, HL Chau, FH Lochovsky „Dynamic knowledge representation in DOOR“ in: X Wu, J Tsai, N Pissinou, K Makki (Hrsg) *Proceedings of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop* (IEEE Computer Society, Los Alamitos 1997) 89–96.

Personenverzeichnis

Stichwortverzeichnis

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

ZEITSCHRIFTENVERÖFFENTLICHUNGEN DES AUTORS

1. F Steimann, KP Adlassnig „Clinical monitoring with fuzzy automata“ *Fuzzy Sets & Systems* 61:1 (1994) 37–42.
2. F Steimann, M Hayde, B Panzenböck, KP Adlassnig, A Pollak „Fuzzy support for serodiagnosis: the ONSET program“ *IEEE Engineering in Medicine and Biology Magazine* 13:5 (1994) 705–709.
3. F Steimann, C Brzoska „Dependency Unification Grammar for PROLOG“ *Computational Linguistics* 21:1 (1995) 95–102.
4. F Steimann „The interpretation of time-varying data with DIAMON-1“ *Artificial Intelligence in Medicine* 8:4 (1996) 343–358.
5. F Steimann, M Hayde „A method to derive the time of onset of infection from serological findings“ *Methods of Information in Medicine* 36:1 (1997) 51–58.
6. F Steimann „Fuzzy set theory in medicine“ (Editorial) *Artificial Intelligence in Medicine* 11:1 (1997) 1–7.
7. F Steimann „Dependency parsing for medical language and concept representation“ *Artificial Intelligence in Medicine* (1998) 77–86.
8. LI Kuncheva, F Steimann „Fuzzy diagnosis“ (Editorial) *Artificial Intelligence in Medicine* 16:2 (1999) 121–128.
9. F Steimann, P Fröhlich, W Nejdl „Model-based diagnosis for open systems fault management“ *AI Communications* 12:1&2 (1999) 5–17.
10. F Steimann „Letter to the Editor“ *Journal of Object-Oriented Programming* 12:2 (1999) 8–9.
11. F Steimann „Abstract class hierarchies, factories, and stable designs“ *Communications of the ACM* 43:4 (2000) 109–111.
12. F Steimann „The family pattern“ *Journal of Object-Oriented Programming* (2000) im Druck.
13. F Steimann „On the representation of roles in object-oriented and conceptual modelling“ *Data & Knowledge Engineering* im Druck.
14. F Steimann „Role = Interface: a merger of concepts“ *Journal of Object-Oriented Programming* (2001) im Druck.
15. F Steimann „Survey on the use and usefulness of fuzzy sets in medical AI“ Festschrift in zwei Heften anläßlich des 80. Geburtstages von LA Zadeh *Artificial Intelligence in Medicine* (2001) in Vorbereitung.