Metadaten-Management für kooperative Anwendungen

Dem Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades
Doktorin der Naturwissenschaften
genehmigte Dissertation

von

Ulrike Steffens

aus Reinbek

2004

1. Gutachter: Prof. Dr. Florian Matthes
2. Gutachter: Prof. Dr. Helmut Weberpals
3. Gutachter: Prof. Dr. Friedrich H. Vogt

Tag der mündlichen Prüfung: 9. Dezember 2004

Diese Arbeit ist meinem Großvater, Herrn Heinz Heyer, gewidmet,
den ich sehr vermisse.

Danksagung

An dieser Stelle möchte ich mich bei allen Menschen bedanken, die mich während meiner Promotion unterstützt haben. Insbesondere danke ich Herrn Prof. Dr. Florian Matthes für die Betreuung, die mir zuteil geworden ist, und Herrn Prof. Dr. Helmut Weberpals für das Interesse an meiner Arbeit.

Darüber hinaus danke ich meinen Kollegen am Arbeitsbereich Softwaresysteme der Technischen Universität Hamburg-Harburg, allen voran Herrn Prof. Dr. Joachim W. Schmidt dafür, dass ich am Arbeitsbereich promovieren konnte. Ein spezieller Dank gilt auch Dr. Claudia Niederée. Unsere gemeinsame wissenschaftliche Arbeit habe ich ebenso schätzen gelernt wie die freundschaftlichen Unterhaltungen. Außerdem danke ich meinen Kollegen Thomas Büchner, Hartmut Gau, Ulrike Hantschmann, Patrick Hupe, Rainer Marrone, Thomas Rahmlow, Dr. Hans-Werner Sehring, Thomas Sidow, Michael Skusa, Dr. Holm Wegner und Axel Wienberg für die gute Zusammenarbeit sowie Rolando Armuelles, Rudolf Bartel, Jörk Behrends, Célio Carreto, Hendry Chandra, Tobias Näth, Alexander Trapp, Siripong Treetasanatavorn, Marc Vollmann, Ernst August Wieden und Lan Zhang, die durch ihre Studien- und Diplomarbeiten ebenfalls zum Gelingen meiner Promotion beigetragen haben.

Ich danke auch meinen Kollegen im Bereich "Betriebliches Informations- und Wissensmanagement" sowie Birgit Novy und Carina Sandmann im OFFIS-Institut Oldenburg für die Unterstützung in der Zeit vor meiner Disputation.

Meinen Freunden danke ich vor allen Dingen für die Geduld, die sie während der gesamten Zeit mit mir hatten, und für die vielen aufmunternden Worte. Mein Dank gilt hier besonders Dr. Stephanie Assmann, Gabriele Günter, Sengül Gül, Katrin Nossack, Martina Pauleit und Dr. Gerald Schröder.

Außerdem danke ich Arslan Brömme für die Diskussionen, den Zuspruch und die guten Momente, an die ich mich immer gern erinnern werde.

Mein ganz besonderer Dank gilt aber meiner Familie, die mir stets Mut gemacht hat und bei der ich immer einen Platz zum Arbeiten und auch zum Abschalten gefunden habe, allen voran meinen Eltern und meinen Großeltern.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Seit Mitte der 1990er Jahre ermöglicht das Internet und insbesondere das WorldWideWeb den Austausch von digitalen Ressourcen und die global verteilte Nutzung von Informationsdiensten [MNSS99], was in zunehmendem Maße einer breiten Allgemeinheit sowohl im Privat- als auch im Arbeitsleben zugute kommt.

Im Arbeitsleben bietet die Nutzung von Internet-Technologie die Grundlage für die verteilte synchrone und asynchrone Zusammenarbeit verschiedener Kooperationspartner [Joh88]. Das WorldWideWeb bildet dabei im Ganzen und in Teilen eine Vielzahl virtueller Arbeitsräume, die jeweils mit gemeinsamen digitalen Ressourcen ausgestattet sind und daher auch als *gemeinsame Informationsräume* bezeichnet werden. Allgemeiner lässt sich das Konzept des gemeinsamen Informationsraums, auch außerhalb des Internet, auf alle verteilten Informationssysteme, die einen Ablageort für kooperativ genutzte Ressourcen darstellen, ausdehnen. Hierbei sind abgeschlossene Informationsräume, die nur einer ausgewählten Gruppe von Benutzern zugänglich sind, wie das Intranet einer Organisation, ebenso vertreten wie offene Informationsräume, in denen beliebige Benutzer Arbeitsressourcen ablegen und finden können.

Das WorldWideWeb selbst kann ebenfalls als ein großer offener Informationsraum angesehen werden. Diese Betrachtungsweise legt allerdings auch eines der Hauptprobleme offen, die insbesondere bei der asynchronen Zusammenarbeit in gemeinsamen Informationsräumen auftreten: Der Nutzer sieht sich hier einer Vielzahl von verschiedenen Ressourcen gegenüber, die potentiell für die Lösung seiner individuellen Arbeitsaufgabe zur Verfügung stehen, deren tatsächliche inhaltliche Relevanz und Qualität er aber ohne Aufwand nicht einzuschätzen vermag.

Verschiedene Forschungs- und Anwendungsgebiete der Informatik haben das beschriebene Problem bereits frühzeitig ausgemacht und unabhängig voneinander ähnliche Lösungsansätze hervorgebracht, die einheitlich auf der Grundidee beruhen, Ressourcen durch zusätzliche Informationen, sogenannte *Metadaten*, näher zu beschreiben, so dass ein potentieller Nutzer zügig zu einer qualitativen Einschätzung einer Ressource gelangen kann.

Der Umgang mit Metadaten hängt stark von dem jeweiligen Kontext ab, in dem sie verwendet werden. So existieren Szenarien, in denen Kooperationspartner *eng und regelmäßig* zusammenarbeiten und dabei immer wieder auf die gleiche Art von Metadaten zurückgreifen, so dass die Festlegung eines einheitlichen Schemas für diese Metadaten vorteilhaft für die Steuerung der Kooperationsprozesse eingesetzt werden kann. Als Beispiel für einen entsprechenden gemeinsamen Informationsraum sei hier eine digitale Bibliothek [Arm00] genannt, in der einem festen Benutzerkreis durch Bibliothekare bibliographische Informationen zum Bibliotheksbestand zur Verfügung gestellt werden. In anderen Szenarien wiederum arbeiten Kooperationspartner eher unregelmäßig oder sogar zufällig zusammen und erfassen die dabei verwendeten Metadaten in einer Form, die ihren jeweils eigenen Qualitätsansprüchen genügt. Ein Beispiel für einen Informationsraum, der diese Art der nur *lose gekoppelten Kooperation* unterstützt, ist die Homepage eines Wissenschaftlers, der verschiedene Publikationen zu seinem Thema gelesen hat und Informationen zu diesen Publikationen im WorldWideWeb in semistrukturierter oder unstrukturierter Form für interessierte Internet-Nutzer zur Verfügung stellt.

Die existierenden Modelle, die zur Repräsentation von Metadaten verwendet werden, weisen in Abhängigkeit des entsprechenden Kooperationskontextes erhebliche Unterschiede auf. Die verfolgten Ansätze reichen von einer strukturierten und streng schematisierten Verwaltung von Metadaten in klassischen Datenbanksystemen [LS87] über die Standardisierung von festen Metadatenschemata für bestimmte Anwendungsgebiete [WIC97, Tea01] und die gemeinsame Verwaltung von Ressourcen und Metadaten in semistrukturierten Dokumenten [ABS00] bis hin zu einer vollständig flexiblen Nutzung von Metadaten im Bereich des Semantic Web [BLHL01, RDF04b] und sind zumeist wohlverstanden.

Die Tatsache, dass einerseits in unterschiedlichen Kooperationskontexten Metadaten zu denselben Ressourcen existieren und dass sich andererseits Informationsräume insbesondere auch im WorldWideWeb zunehmend Benutzergruppen öffnen, die zuvor nicht oder nicht direkt an der entsprechenden Kooperation teilhatten, eröffnet über die Verwaltung von Metadaten in einzelnen bestimmten Datenmodellen hinaus die folgenden neuen kooperativen Nutzungsmöglichkeiten für Metadaten:

- Nutzung von Metadaten aus einem Kooperationskontext in einem anderen Kooperationskontext, auch wenn in diesen Kontexten einerseits enge und andererseits lose gekoppelte Formen der Zusammenarbeit verfolgt werden
- Konstruktion von gemeinsamen Informationsräumen, die auf einem Ressourcenbestand unterschiedliche Formen der Zusammenarbeit gleichermaßen begünstigen

Um eine solche integrierte Nutzung von Metadaten unterstützen zu können, wird ein Gesamtkonzept für die Zusammenarbeit auf der Grundlage von Metadaten in verschiedenen Arbeitskontexten benötigt, für dessen Entwurf insbesondere die folgenden Fragestellungen geklärt werden müssen:

- Welche strukturellen und semantischen Gemeinsamkeiten, die als Basis für eine integrierte Nutzung dienen können, weisen Metadaten in Informationsräumen mit unterschiedlichen Kooperationskontexten auf?
- Welche strukturellen und semantischen Unterschiede, die die integrierte Nutzung behindern könnten, bestehen zwischen Metadaten in Informationsräumen mit unterschiedlichen Kooperationskontexten?
- Welche strukturellen, semantischen und algorithmischen Konzepte können zum Einsatz gebracht werden, um die durch diese Unterschiede entstehenden Nachteile auszugleichen?

Das Ziel der vorliegenden Dissertation ist es, sich diesen Fragestellungen zu widmen, um hierdurch zu einer konzeptuellen Grundlage zu gelangen, die einerseits den Austausch bereits existierender Metadaten auch dann unterstützt, wenn diese auf der Grundlage verschiedener Formen der Kooperation entstanden sind, und die andererseits die Konstruktion gemeinsamer Informationsräume ermöglicht, in denen Kooperationsformen von einer lose gekoppelten bis hin zu einer engen, schematisierten Zusammenarbeit gleichermaßen praktikabel sind.

1.2 Verwandte Themengebiete

Zur Überbrückung der Unterschiede zwischen verschiedenen Kontexten existieren im Bereich der Datenmodelle und der Datenmodellierung Ansätze auf verschiedenen Abstraktionsniveaus.

Der direkteste Ansatz besteht hier in der Vermeidung von Unterschieden, indem für die Beschreibung von Ressourcen, die in bestimmten Anwendungsgebieten angesiedelt sind, von vornherein feste Metadatenstandards eingeführt und verwendet werden. Beispiele hierfür sind z.B. der Dublin Core [WIC97] zur einheitlichen Beschreibung von Internet-Ressourcen und der SCORM-Standard [Tea01] zur Beschreibung von E-Learning-Inhalten.

Im Bereich der Datenmodelle von Datenbanksystemen existieren Ansätze zur Integration von konkreten Datenbankschemata, die in verschiedenen Arbeitskontexten entstanden sind [SL90]. Die heterogenen Strukturen und Werte müssen hierbei zusammengeführt werden [HM93]. Ontologien, die insbesondere auch die Integration von Metadaten aus verschiedenen Kontexten unterstützen (vgl. Abschnitt 4.3), stellen auch im Datenbankbereich ein geeignetes Werkzeug dar [HG02].

Über die Ansätze auf der Schemaebene hinaus führen Ansätze zur Metamodellierung, deren Ziel es ist, ein Metamodell zu finden, in das unterschiedliche Modelle, die teilweise gleiche Phänomene beschreiben, umgewandelt werden können [KK02]. Metamodelle kommen zur Zeit insbesondere im Bereich der Unternehmensmodellierung zum Einsatz, wo regelmäßig unterschiedliche Modelle für unterschiedliche Arbeitsaufgaben zum Einsatz kommen [SZ92, JvBA$^+$03].

Im Bereich der Ressourcebeschreibungen stellen auch RDF und RDF Schema spezielle Metamodelle dar [PH01], die sich die Open World Assumption zu

eigen machen und somit insbesondere für die lose gekoppelte Kooperation auf der Grundlage von Metadaten geeignet sind, während eine enge, schematisierte Zusammenarbeit kaum unterstützt wird.

Die Annäherung von Arbeitskontexten kann jedoch nicht nur durch geeignete Datenmodelle erreicht werden, sondern auch durch Algorithmen, die Metadaten tolerant interpretieren. Die Best-Match-Algorithmen des Information Retrieval sind ein Beispiel für eine solche tolerante Vorgehensweise bei der Interpretation [TC92].

1.3 Zur Gliederung der Arbeit

Als eine erste Grundlage werden die Arbeit in gemeinsamen Informationsräumen sowie die generelle Verwendung von Metadaten getrennt voneinander betrachtet, bevor der Einsatz von Metadaten speziell in Informationsräumen beschrieben wird (vgl. Kapitel 2). Hierbei werden zum einen typische Vorgehensweisen beim Umgang mit Metadaten herausgearbeitet, zum anderen wird der allgemeine Begriff der Metadaten anhand derjenigen Aspekte des Informationsraums, die durch sie beschrieben sind, weiter unterteilt.

Im Anschluss werden verschiedene existierende Datenmodelle im Einzelnen bezüglich ihrer Eignung für die Repräsentation kooperativ genutzter Metadaten untersucht (vgl. Kapitel 3). Hierzu wird sowohl betrachtet, welche der in Kapitel 2 aufgeführten Vorgehensweisen durch das jeweilige Modell unterstützt werden, als auch, welche der in Kapitel 2 diskutierten Arten von Metadaten dargestellt werden können. Als eine erste Mindestanforderung lässt sich formulieren, dass ein integrierendes Metadatenmodell die hier als geeignet identifizierten Konzepte umfassen sollte.

Dieses integrierende Metadatenmodell wird im weiteren Verlauf der Arbeit entwickelt (vgl. Kapitel 4). Es führt die in Kapitel 3 erarbeiteten Konzepte zusammen und ergänzt sie, so dass ein allgemeines Modell entsteht, das die konzeptuelle Grundlage für die flexible Wahl verschiedener Kooperationsszenarien und Vorgehensweisen im Umgang mit Metadaten sowohl in einem einzigen allgemeinen als auch in einzelnen spezialisierten Informationsräumen darstellt.

Tolerante Algorithmen werden in gemeinsamen Informationsräumen eingesetzt, um Metadaten, die innerhalb eines Kontexts erfasst wurden, auf einen anderen Kontext abzubilden, damit diese dort besser verstanden werden können. Kapitel 5 beschreibt zunächst die Verfahren der Fuzzy-Logik als eine mögliche Grundlage zur Realisierung toleranter Algorithmen, bevor es die Anbindung toleranter Algorithmen an das in Kapitel 4 entwickelte Metadatenmodell mit Hilfe von Fuzzy-Prädikaten erläutert und Nutzungsmöglichkeiten für diese Algorithmen aufzeigt.

Kapitel 6 stellt zwei konkrete Systeme zur Verwaltung kooperativ genutzter Metadaten vor, deren Datenmodelle jeweils Spezialisierungen des integrierenden Metadatenmodells aus Kapitel 4 darstellen, die sich aus dem durch das jeweilige System unterstützten speziellen Kooperationsszenarium ergeben. Zugleich wird hier aufgezeigt, wie tolerante Algorithmen im praktischen Einsatz Mehrdeutigkeiten entschärfen, die zwischen Kooperationspartnern bezüglich der gemeinsa-

men Metadaten bestehen.

Die Arbeit schließt mit einer Zusammenfassung der Ergebnisse und einem Ausblick auf weitere mit dem Thema der Arbeit verbundene Fragestellungen (vgl. Kapitel 7).

Kapitel 2

Kooperative Metadatennutzung

Gemeinsame Informationsräume dienen der Unterstützung sowohl der synchronen als auch der asynchronen Gruppenarbeit. Ihre Aufgabe bei der asynchronen Gruppenarbeit ist dabei primär die eines Materialspeichers für kooperativ genutzte digitale Ressourcen. Da eine direkte Kommunikation zwischen den Kooperationspartnern bei der asynchronen Gruppenarbeit oft entfällt, wird die Verständigung über die genaue Bedeutung sowie die Nutzungsmöglichkeiten für gemeinsame Ressourcen hier erschwert. Die Bereitstellung von Metadaten, die die vorhandenen Ressourcen beschreiben, ist ein Lösungsansatz, um dieses Problem zu entschärfen.

In diesem Kapitel wird zunächst die Arbeit in gemeinsamen Informationsräumen erläutert (vgl. Abschnitt 2.1). Im Anschluss wird ein Überblick über die allgemeine Verwendung von Metadaten gegeben (vgl. Abschnitt 2.2), bevor ihr Einsatz zur Unterstützung der asynchronen Zusammenarbeit in gemeinsamen Informationsräumen beschrieben wird (vgl. Abschnitt 2.3).

2.1 Gemeinsame Informationsräume

Computer-Unterstützung führte in den letzten Jahrzehnten zu Veränderungen in der Arbeitswelt, insbesondere auch im Bereich der Gruppenarbeit. Die gemeinsame Arbeit zur gleichen Zeit am gleichen Ort ist nur noch eine von vielen Facetten der Kooperation. Wachsende Rechen- und Speicherkapazitäten sowie ubiquitäre Netzwerkverbindungen ermöglichen es Kooperationspartnern in der ganzen Welt, zu unterschiedlichen Zeiten ihr gemeinsames Arbeitspensum zu bewältigen.

Das Forschungsgebiet *CSCW (***C***omputer* **S***upported* **C***ooperative* **W***ork)* untersucht die Auswirkungen und Perspektiven der Computerunterstützung für die Zusammenarbeit von Personen [BS91]. Hierbei gilt es, die Besonderheiten menschlicher Kooperation zu berücksichtigen, so dass Computer-Werkzeuge die Zusammenarbeit ihrer Benutzer nicht behindern, sondern die Arbeitssituation im optimalen Fall sogar verbessern. Die Funktionalität von CSCW-Werkzeugen ist dabei so vielfältig wie die Mechanismen, die Menschen im Kontext ih-

Raum / Zeit	gleiche Zeit	unterschiedliche Zeit
gleicher Ort	• Sitzungsunterstüzung • Systeme zur gemeinsamen Bearbeitung komplexer Aufgaben	• Roomware • Workflowmanagement • Projektmanagement
unterschiedlicher Ort	• Telekonferenzsysteme • Chat-Systeme • **Gruppenwerkzeuge in gemeinsamen Informationsräumen**	• Newsgroups • Bulletin-Boards • **Materialspeicher in gemeinsamen Informationsräumen**

Abbildung 2.1: CSCW-Klassifizierung nach Raum und Zeit

rer Zusammenarbeit einsetzen. Einfache Kommunikationssysteme, wie E-Mail-Systeme, fallen ebenso unter den Oberbegriff CSCW wie auch Gruppeneditoren oder Work-Flow-Managementsysteme. Einzelne dieser Werkzeuge können dabei als Komponenten von umfassenderen CSCW-Systemen auftreten.

In der Literatur existieren verschiedene Ansätze, um das Gebiet des CSCW weiter zu unterteilen und die daraus entstandenen Werkzeuge und Systeme entsprechend einzuordnen. Eine Möglichkeit zur Einordnung von CSCW-Systemen bietet beispielsweise der in [TSMB95] vorgestellte Ansatz. Ausgangspunkt ist die Beobachtung, dass CSCW-Systeme bei der Unterstützung von Zusammenarbeit drei verschiedene Kernaufgaben erfüllen können, indem sie erstens die *Kommunikation* und damit den Austausch von Nachrichten zwischen den Kooperationspartnern ermöglichen oder indem sie zweitens die *Kooperation* und damit die unmittelbare Zusammenarbeit der Partner bei der Lösung einer Aufgabe fördern oder indem sie drittens die *Koordination des Arbeitsprozesses* steuern, wobei reale Abläufe im System abgebildet werden.

Das wohl am weitesten verbreitete Klassifizierungsschema für CSCW-Komponenten ist jedoch die *Raum-Zeit-Matrix* (vgl. Abb. 2.1), in der verschiedene Arten der Zusammenarbeit zum einen danach unterschieden werden, ob die Kooperationspartner zu gleichen oder unterschiedlichen Zeiten arbeiten, und zum anderen danach, ob sie sich dabei an gleichen oder unterschiedlichen Orten befinden [Joh88].

Gemeinsame Informationsräume bilden eine spezielle Klasse von CSCW-Systemen. Sie unterstützen sowohl die synchrone als auch die asynchrone Zusammenarbeit, indem sie der Metapher physikalischer Gruppenarbeitsräume folgen [RG96]. Für die synchrone Zusammenarbeit stellen sie Werkzeuge wie gemeinsame Whiteboards, Chat-Werkzeuge oder personengebundene Telepointer zur Verfügung (vgl. Abb. 2.1, unten links), während die asynchrone Zusammenarbeit dadurch unterstützt wird, dass die von der Gruppe gemeinsam verwendeten oder angefertigten Ressourcen, wie aufgabenspezifische Software, Projektpläne oder Dokumente, persistent im gemeinsamen Informationsraum verbleiben und dort auch durch Kooperationspartner, die alleine an der Grup-

penaufgabe arbeiten, jederzeit genutzt werden können [ABK01] (vgl. Abb. 2.1, unten rechts).

Die Verbreitung des WorldWideWeb macht es zu einer Plattform für zahlreiche kooperative Informationsräume. Aufgrund der weltweiten Nutzung des Web und seiner einfachen Mechanismen zur Bereitstellung und zum Abruf von Informationen steht dabei insbesondere seine asynchrone Nutzung als kooperativer Materialspeicher im Vordergrund. Die Zusammenarbeit findet dabei zum einen in *privaten Informationsräumen* statt, die bestimmten Gruppen vorbehalten sind und mit Hilfe Web-basierter CSCW-Systeme [BHST95, RMS+01] errichtet werden. Zum anderen stellen *öffentliche Informationsräume* u.U. recht unterschiedliche Materialen für unterschiedliche Aufgaben und unterschiedliche Arten der Zusammenarbeit frei zugänglich zur Verfügung (vgl. z.B. [BB97]). Insbesondere kann das WorldWideWeb, das eine Vielzahl an Informationen [MNSS99] und Diensten [NSH02] bereithält, in seiner Gesamtheit selbst als prominentester öffentlicher Informationsraum betrachtet werden. Vor diesem Hintergrund sollen im Folgenden die Anforderungen diskutiert werden, die sich bei der asynchronen Nutzung gemeinsamer Informationsräume stellen.

Sowohl in privaten als insbesondere auch in öffentlichen Informationsräumen besteht das Problem, dass verschiedene Kooperationspartner *unterschiedlichen Arbeitskontexten* entstammen können. Der Arbeitskontext prägt sich dabei durch die persönlichen Erfahrungen des Akteurs, durch seine Arbeitsaufgabe, durch die Technologie, die er für seine Tätigkeit verwendet, und durch seine soziale und formale Einbettung in bestimmte Organisationen [Lea74]. Je nach Arbeitskontext gestalten Akteure die Ressourcen, die sie in einem Informationsraum ablegen, unterschiedlich, so dass eine spätere Interpretation durch Akteure mit anderem Arbeitskontext ein schwierige Aufgabe ist.

Hinzu kommt, dass nicht immer alle Kooperationspartner in einem gemeinsamen Informationsraum von vornherein bekannt sind. Im WorldWideWeb beispielsweise kann man oft sogar überhaupt nicht absehen, wer welche Inhalte wann betrachten und interpretieren wird. Sind anonyme Partner an einer Kooperation beteiligt, so entfällt die Möglichkeit, außerhalb des gemeinsamen Informationsraums miteinander zu kommunizieren, um beispielsweise Rückfrage bezüglich unklarer Ressourcen zu halten oder vorab gemeinsame Regeln zur Gestaltung der im Informationsraum abzulegenden Ressourcen zu vereinbaren. Die gemeinsamen Ressourcen selbst bilden in diesem Fall den einzigen Kommunikationskanal zwischen den anonymen Kooperationspartnern.

Insbesondere in Informationsräumen, in denen eine große Zahl gemeinsamer Ressourcen abgelegt ist, ist das Wiederauffinden von einzelnen bestimmten Ressourcen keine triviale Aufgabe. Hier helfen zum einen *Suchwerkzeuge* dabei, Ressourcen anhand bestimmter Suchkriterien ausfindig zu machen. Zum anderen bieten *Klassifikationssysteme* die Möglichkeit, durch den Informationsraum zu navigieren und sich so der gesuchten Ressource schrittweise zu nähern.

Ressourcen in gemeinsamen Informationsräumen können nicht nur zahlreich sein, sondern sind auch raschen und häufigen Änderungen unterworfen. Die Möglichkeit, relevante Änderungen im Informationsraum verfolgen und bei der Bearbeitung eigener Aufgaben darauf reagieren zu können, ist daher ebenfalls eine Anforderung für dessen kooperative Nutzung. *Awareness-Dienste*, die Ko-

operationspartner über Änderungen informieren, ohne diese mit zu viel Information zu überfluten [DB92], bieten hier einen Lösungsansatz.

Die Vollständigkeit und Qualität von in Informationsräumen bereitgestellten Ressourcen ist Schwankungen unterworfen. Um die Eignung von Ressourcen für die individuelle Arbeitsaufgabe einschätzen zu können, ist es hilfreich, sowohl ihren Erfasser als auch dessen Arbeitskontext zu kennen. [BB97] fordern aus diesem Grunde beispielsweise, dass Ressourcen in einem gemeinsamen Informationsraum möglichst immer ihr Urheber zugeordnet sein sollte. *Empfehlungsdienste* [BD99] leisten einen weiteren Beitrag zur Ermittlung der Qualität von Ressourcen im kooperativen Kontext, indem Akteure hier auf Beurteilungen ihnen bekannter und als kompetent eingeschätzter Kooperationspartner zurückgreifen können, um sich ein Bild über bestimmte Ressourcen machen zu können.

Aus den beschriebenen Anforderungen an gemeinsame Informationsräume ergibt sich, dass die alleinige Ablage der Arbeitsressourcen im Materialspeicher nicht ausreichend ist, um eine für alle Kooperationspartner gleichermaßen zufriedenstellende asynchrone Zusammenarbeit zu gewährleisten.

Ein Ansatz, um die Kooperation in gemeinsamen Informationsräumen zu erleichtern, lässt sich gewinnen, indem man auf die eingangs erwähnte Metapher zurückgreift und noch einmal physikalische Gruppenarbeitsräume betrachtet. Auch hier besteht das Problem, dass der Umgang mit den im Raum vorhandenen Ressourcen sich bei Abwesenheit anderer Kooperationspartner schwierig gestaltet. So ist vielleicht unklar, wie eine im Raum vorhandene Maschine bedient wird, jemand hat das Schriftstück, das man in der letzten Woche bearbeitet hat, an einen unbekannten Platz gelegt, und jede Fotokopie, die man anfertigen möchte, verursacht einen Papierstau. Kooperationspartner, die sich einen Gruppenarbeitsraum teilen, entschärfen diese Probleme, indem sie neben den Arbeitsressourcen selbst Informationen über diese Ressourcen hinterlassen, die anderen Gruppenmitgliedern die Nutzung erleichtern. So liegt ein Handbuch für die Benutzung der Maschine bereit, der Ordner, in dem sich das vermisste Schriftstück befindet, ist aussagekräftig beschriftet, und am Fotokopierer ist ein Stück Papier mit der Aufschrift „defekt“ befestigt.

Analog lassen sich in gemeinsamen Informationsräumen die dort vorhandenen Ressourcen durch Informationen darüber ergänzen. Diese ergänzenden Informationen werden dann ebenfalls im Informationsraum abgelegt. So kann beispielsweise die Interpretation von Ressourcen vereinfacht werden, die einem fremden Arbeitskontext entstammen, indem zusätzliche Erläuterungen gegeben werden, die dem interpretierenden Kooperationspartner eine Vorstellung von dem fremden Kontext und der Bedeutung der Ressource vermitteln. Die Qualität einer Ressource kann mit Hilfe von Beschreibungen ihrer Herkunft [BKT01] oder direkt damit verbundenen Kommentaren eingeschätzt werden. Zudem können Software-Werkzeuge in Informationsräumen, wie Suchwerkzeuge, Empfehlungs- oder Awareness-Dienste, Informationen über Ressourcen generieren und sie für ihre spezifischen Aufgaben austauschen und verwenden.

Informationen, die, wie die ergänzenden Informationen in einem gemeinsamen Informationsraum, andere Informationen beschreiben, werden als *Metadaten* bezeichnet. Der folgende Abschnitt 2.2 gibt zunächst einen Überblick über

die allgemeine Verwendung von Metadaten zur Beschreibung von Ressourcen, bevor Abschnitt 2.3 Szenarien für die Nutzung von Metadaten in gemeinsamen Informationsräumen aufzeigt und ein abstraktes Bild der durch Metadaten gegebenen Möglichkeiten zur Beschreibung von kooperativ genutzten Ressourcen zeichnet.

2.2 Metadaten

Der Begriff „Metadaten“ wird häufig und in vielfältiger Weise über seine wortwörtliche Bedeutung als „Daten über Daten“ hinaus verwendet. So kommen z.B. [LMSP95] zu folgender Auffassung:

> Our feeling is that at this point ”metadata” as a descriptive term has become so debased by overuse (and means so many different things in different communities and contexts) that it is now virtually meaningless without extensive qualification; ...

Da sich jedoch treffendere Ersatzbegriffe wie „Metainformationen“ [Smi96] oder „Ressourcebeschreibungen“ [RDF04b] nie vollständig durchsetzen konnten und die Bezeichnung „Metadaten“ die mit Abstand am häufigsten gewählte für diejenige Klasse von Informationen ist, die das Thema dieser Arbeit sind, soll auch hier dieser viel diskutierte Begriff abweichend von seiner ursprünglichen Bedeutung verwendet werden. Eine geeignete Begriffsbestimmung hierfür findet sich in [DH98]:

> Metadata is data associated with objects which relieves their potential users of having full advance knowledge of their existence or characteristics. It supports a variety of operations. A user could either be a program or a person.

Metadaten werden also sowohl von Personen als auch von Programmen verwendet, die in dieser Arbeit beide einheitlich als *Akteure* bezeichnet werden. Akteure sehen sich Objekten gegenüber gestellt, die sie potentiell zu einem beliebigen Zweck einsetzen können und die somit Ressourcen darstellen. Oft ist es Akteuren jedoch nicht möglich, ein vollständiges Vorwissen über die Eigenschaften einer Ressource zu haben oder sich diese Kenntnisse durch eingehende Untersuchung der Ressource selbst zu erarbeiten. Metadaten, die zu einem früheren Zeitpunkt vom betreffenden Akteur selbst oder einem anderen Akteur entweder direkt aus der Ressource abgeleitet oder als Ergänzung zu dieser erfasst worden sind, beschreiben die Eigenschaften der Ressource in komprimierter Form, die der Aufgabe des Akteurs angemessen ist, ihm ein schnelles Verständnis der Beschaffenheit der Ressource ermöglicht und dadurch die Abwicklung seiner Tätigkeit beschleunigt.

Die Erfassung von Metadaten birgt neben dem eigentlichen Umgang mit der Ressource selbst einen zusätzlichen Arbeitsaufwand, dessen Nutzen zudem erst zu Tage tritt, wenn die Metadaten eingesetzt werden, um die weitere Verwendung der Ressource zu erleichtern. Zwischen der Erfassung und dem Einsatz

von Metadaten können demnach große Zeitspannen liegen, und es ist darüber hinaus oft nicht absehbar, ob erfasste Metadaten überhaupt genutzt werden. Um die Erfassung von Metadaten zu motivieren, gilt es zu beachten, dass diese in engem Zusammenhang mit der eigentlichen Arbeit des Erfassers steht, damit er die Ressource möglichst keinen zusätzlichen Betrachtungen unterziehen muss, um die Metadaten zu ermitteln. Darüber hinaus sollten alle Möglichkeiten ausgeschöpft werden, bestimmte Metadaten automatisch zu erfassen.

Mit Hilfe von Metadaten kann der Umgang mit Ressourcen in verschiedenen Tätigkeitsfeldern unterstützt und dokumentiert werden. Die inhaltliche Beschreibung von Ressourcen erscheint beispielsweise in der Einteilung von Metadaten nach ihrer Funktion in [GS98] nur als eine aus einer Reihe von möglichen Aufgaben, für die Metadaten genutzt werden können. Metadaten lassen sich danach wie folgt untergliedern:

Metadaten zur Ressourcebeschreibung wie bibliographische Katalogeinträge, domänenspezifische Indexe, Hyperlinks zwischen verschiedenen Ressourcen, Benutzerannotationen etc.

Metadaten zur Verwaltung von Ressourcen wie Informationen über die Beschaffung von Ressourcen, Urheberrechte, rechtliche und organisatorische Zugangsbeschränkungen, Versionskontrolle, Aufbewahrungsort nichtdigitaler Ressourcen etc.

Metadaten zur Erhaltung von Ressourcen wie Dokumentation des physikalischen Zustands von Ressourcen, Dokumentation von Aktionen zur Erhaltung, wie des Anlegens von Backup-Kopien von Dateien, etc.

Technische Metadaten wie Dokumentation von Hardware- und Software-Ressourcen, Dateiformate, Größe von Dateien, Software-Anforderungen bei Nutzung und Darstellung von Ressourcen, Antwortzeiten, etc.

Metadaten über die Nutzung von Ressourcen wie Verwendungshistorie, Zugriffsstatistiken, Information über die Wiederverwendung von Ressourcen z.B. in Zitaten oder als Fragment einer umfangreicheren Ressource etc.

Die vorliegende Einteilung entstammt der Informationswissenschaft. Hier ist beispielsweise die Erhaltung von Ressourcen ein wichtiges Thema, während sie in anderen Anwendungsgebieten u.U. gar nicht als eigenes Tätigkeitsfeld betrachtet wird und stattdessen z.B. unter die Kategorie der Verwaltung von Ressourcen fallen könnte. Desgleichen erscheinen hier Kategorien nicht, die in anderen Domänen eine wichtige Rolle spielen, wie betriebswirtschaftliche Metadaten, die z.B. die Kosten der Anschaffung, Lagerung oder Nutzung einer Ressource erfassen. Daraus wird deutlich, dass für verschiedene Anwendungsgebiete ebenso verschiedene Metadaten benötigt werden, auch wenn sich die eingesetzten Ressourcen selbst von Gebiet zu Gebiet nicht unterscheiden. Die Funktion von Metadaten lässt sich daher nicht vollständig benennen oder gar begrenzen.

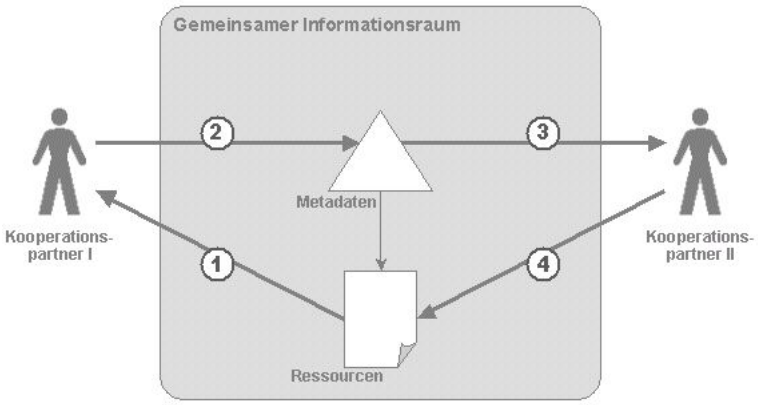

Abbildung 2.2: Beschreibung von Ressourcen im gemeinsamen Informationsraum durch Metadaten

Ein weiterer Aspekt der obigen Einteilung ist, dass ihre Kategorien nicht überschneidungsfrei sind. Als einfaches Beispiel dazu kann das Dateiformat einer Ressource dienen, das z.B. in die Gruppe der technischen Metadaten fällt, aber auch bei der Erhaltung von Ressourcen eine Rolle spielen kann. Daraus lässt sich schließen, dass bei der Erfassung von Metadaten nicht immer von vornherein klar ist, für welche Aufgabe diese später verwendet werden.

2.3 Metadaten in gemeinsamen Informationsräumen

Für die asynchrone kooperative Arbeit in gemeinsamen Informationsräumen sind Metadaten von besonderer Bedeutung, da hier häufig keine direkte Kommunikation zwischen den Kooperationspartnern stattfindet und die Metadaten neben den Ressourcen selbst einen weiteren indirekten Kommunikationskanal darstellen, über den der Kontext, in dem eine gemeinsame Ressource erstellt und verwendet wird, erläutert und möglicherweise sogar geklärt werden kann.

Im Folgenden werden nun zunächst mögliche Vorgehensweisen bei der Nutzung von Metadaten in gemeinsamen Informationsräumen diskutiert, bevor ein Überblick darüber gegeben wird, welche Aspekte der Ressourcen im Informationsraum durch Metadaten beschrieben werden können. Abschließend wird der Einsatz *toleranter Algorithmen* als eine mögliche Strategie bei der Nutzung kooperativer Metadaten erläutert.

2.3.1 Vorgehensweisen bei der Nutzung von Metadaten

Da Informationsräume die technischen Voraussetzungen zur Speicherung von digitalen Informationen in unterschiedlichen Formaten bereits mitbringen, können Metadaten hier ebenso wie die durch sie beschriebenen Ressourcen abgelegt werden (vgl. Abb. 2.2). Der Metadatenerfasser (Kooperationspartner I) nutzt dabei seine Kenntnis der Ressource (1), um die dazugehörigen Metadaten zu erstellen und im Informationsraum zu speichern (2). Sein Partner (Kooperati-

onspartner II) betrachtet zunächst die Metadaten (3), um auf ihrer Grundlage eine Entscheidung über die Verwendung der Ressource zu treffen (4).

Im Rahmen dieses allgemeinen Verwendungsschemas für Metadaten in gemeinsamen Informationsräumen gibt es eine Reihe von Vorgehensweisen, die abhängig von der Art der beschriebenen Ressourcen sowie den Arbeitskontexten der Kooperationspartner und ihrem gemeinsamen Kooperationskontext geeignet sein können, um die Zusammenarbeit auf der Grundlage von Metadaten in einem gemeinsamen Informationsraum zu unterstützen. Diese Vorgehensweisen werden im Folgenden im Einzelnen erläutert:

Vorgehensweise–1: Beschreibung von Ressourcen außerhalb des gemeinsamen Informationsraums

In Informationsräumen ohne Metadaten sind ausschließlich Ressourcen, die per Kopie oder digitaler Referenz im Informationsraum abgelegt sind, Gegenstand der unterstützten Kooperation. Mit Hilfe von Metadaten können darüber hinaus jedoch auch Ressourcen, die selbst außerhalb des Informationsraums bleiben, beschrieben und dadurch für die Zusammenarbeit herangezogen werden (vgl. Abb. 2.3). Voraussetzung hierfür ist, dass der Metadatenerfasser seine Kenntnis über die Ressource außerhalb des Informationsraums erwerben kann (1). Er beschreibt dann die Ressource innerhalb des Informationsraums durch Metadaten (2), die sein Kooperationspartner auf zweierlei Weise nutzen kann (3), indem er entweder die Metadaten als digitales Surrogat der Ressource zur alleinigen Grundlage seiner Arbeit macht oder sie wie im allgemeinen Verwendungsschema als Entscheidungsunterstützung einsetzt, um außerhalb des Informationsraums Zugriff auf die Ressource selbst zu nehmen (4).

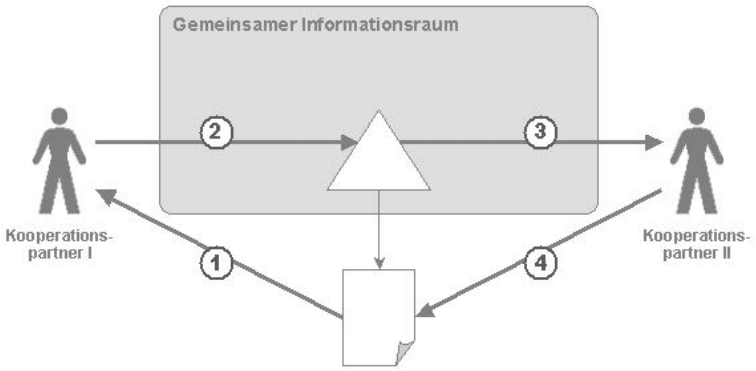

Abbildung 2.3: Beschreibung von Ressourcen außerhalb des gemeinsamen Informationsraums

Durch die beschriebene Vorgehensweise lassen sich insbesondere Ressourcen zum Gegenstand der Kooperation machen, die nicht oder nur schlecht in Informationsräumen gespeichert werden können.

Hierzu zählen beispielsweise nicht-digitale Ressourcen wie mechanische Werkzeuge, Immobilien, Bücher oder Kunstwerke [SSW01], kostenpflichtige Ressourcen, deren Beschaffung zu Kooperationszwecken nicht möglich ist, Ressourcen, die einem besonderen rechtlichen Schutz unterliegen, sowie Ressourcen, deren Umfang die Kapazitäten des Informationsraums für Speicherung und Zugriff überschreitet.

Vorgehensweise–2: Ressourcebeschreibung durch multiple Metadaten

Die Möglichkeit der Erfassung mehrerer unterschiedlicher Metadateninstanzen zu ein und derselben Ressource kann die Kooperation in gemeinsamen Informationsräumen unterstützen. Hierbei können ein oder, wie Abbildung 2.4, mehrere Akteure verschiedene Metadaten für eine Ressource anlegen (2a, 2b). Kooperationspartnern steht dann die Aggregation dieser Metadaten zur Verfügung, so dass sie sich in einem Schritt möglichst vollständig über die Ressource informieren können (3).

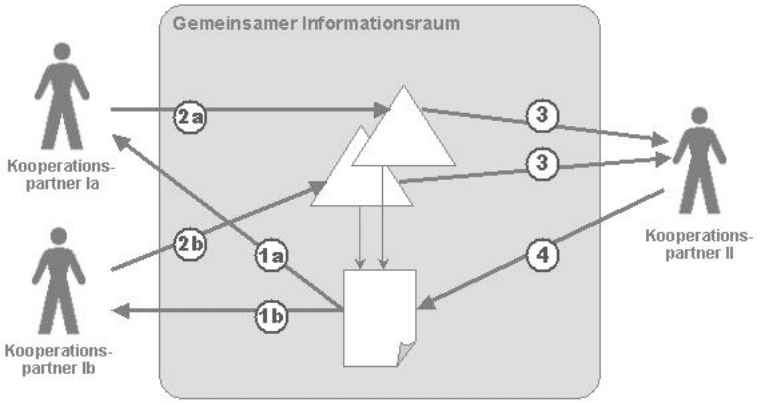

Abbildung 2.4: Ressourcebeschreibung durch multiple Metadaten

Bei nur einer zulässigen Metadateninstanz pro Ressource trägt der Erfasser unter Umständen einen Mehraufwand, wenn er seine eigenen Metadaten in die Struktur der bereits existierenden Instanz einpassen muss. Hat er hingegen die Möglichkeit, eine eigene Metadateninstanz anzulegen, so kann er dieses entlang seines eigenen Arbeitskontexts tun, wodurch die Erfassung vereinfacht wird.

Akteure aus unterschiedlichen Arbeitskontexten teilen zudem nicht immer ihre Auffassung von einer Ressource. Die Verwendung mehrerer Metadateninstanzen unterstützt die Erfassung und Übermittlung widersprüchlicher Beschreibungen zu einer Ressource und wird damit der Rolle von Metadaten als Kommunikationskanal im gemeinsamen Informationsraum gerecht.

Weiterhin kann die Verteilung von Metadaten zu einer Ressource über mehrere Instanzen genutzt werden, um den Metadatenbestand zu strukturieren, indem z.B. für jede Kooperationsaufgabe oder jede Arbeitsgruppe eine eigene Instanz eingerichtet wird. Akteure können sich dann auf die für sie jeweils relevanten Metadaten konzentrieren und alle weiteren für die Ressource erfassten Metadaten ausblenden.

Vorgehensweise–3: Strukturierung von Metadaten

Ressourcen weisen meist eine Reihe von *Eigenschaften* auf, die im Rahmen einer Kooperation relevant sind. Das Verständnis für die Existenz dieser Eigenschaften führt oft dazu, dass Metadaten in *strukturierter Form* erfasst werden. In der Metadatenstruktur werden die Eigenschaften der Ressource getrennt voneinander beschrieben und diese Einzelbeschreibungen so zusammengefügt, dass die Metadaten nicht nur die einzelnen Eigenschaften, sondern auch etwaige Zusammenhänge zwischen ihnen widerspiegeln (vgl. Abb. 2.5).

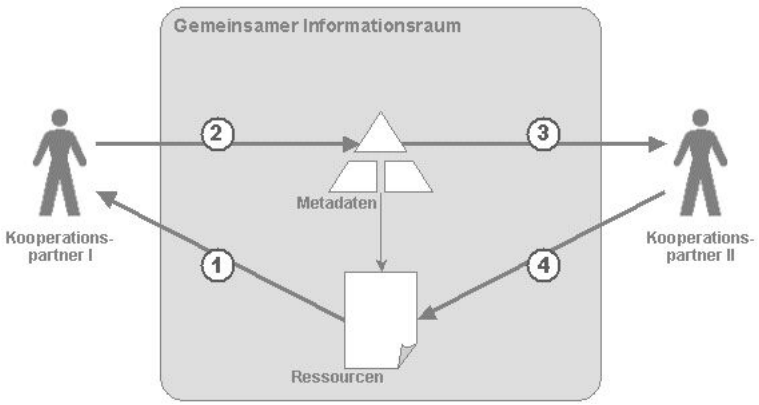

Abbildung 2.5: Strukturierung von Metadaten

Die Strukturierung von Metadaten birgt zusätzliche Informationen, die bei der Interpretation verwendet werden, um einzelne Bestandteile von Metadaten gezielt auffinden und auswerten zu können. Nicht alle kooperativen Tätigkeiten bilden jedoch eine geeignete Basis für die Erfassung strukturierter Metadaten. Die Forderung nach einer Struktur kann dem Arbeitskontext des Erfassers zuwider laufen, so dass die Erfassung am Ende einen übermäßigen Aufwand für ihn darstellt. Ein Beispiel für einen Arbeitskontext, in dem die Strukturierung von Metadaten nicht immer eine willkommene Vorgehensweise ist, ist die kreative Arbeit mit Ressourcen, bei der die Kooperationspartner die Flexibilität benötigen, die Ressource mit beliebigen Annotationen versehen zu können [Mar00].

Vorgehensweise–4: Freiwillige Bereitstellung von Zusatzinformationen für Metadaten

Durch die Strukturierung von Metadaten werden zusätzliche Informationen erzeugt, die die Interpretation von Metadaten unterstützen können (vgl. Vorgehensweise–3). Doch auch die Struktur, die ein Akteur bei der Erfassung der Metadaten wählt, ist von seinem Arbeitskontext geprägt, so dass die unterschiedlichen Arbeitskontexte von Kooperationspartnern einer eindeutigen Interpretation der Metadaten entgegenstehen können. Das folgende einfache Beispiel, das dreimal dieselbe Information durch jeweils unterschiedlich aufgebaute Metadaten beschreibt, kann dieses verdeutlichen:

```
<Erstellungsdatum>
  <Tag>01</Tag> <Monat>01</Monat> <Jahr>2003</Jahr>
</Erstellungsdatum>

<Date>03-01-01</Date>

<Beschreibung>
  Das Dokument wurde am ersten Januar des Jahres 2003
  erstellt.
</Beschreibung>
```

Insbesondere Softwaresysteme benötigen zur korrekten Interpretation von Metadaten Informationen über die Metadatenstruktur und den Aufbau der beschreibenden Werte. So ist die Bezeichnung, hinter der sich im obigen Beispiel das Erstellungsdatum verbirgt, ebenso eine für die Interpretation relevante Information, wie die Schachtelungstiefe der Beschreibungen und das Format, in dem das Datum erfasst ist. Personen besitzen zwar die Fähigkeit, in vielen Fällen durch Intuition und persönliche Erfahrung die Bedeutung von Metadaten auch ohne zusätzliche Informationen erfassen zu können, doch auch ihnen sind Grenzen gesetzt, wie bei der Interpretation des zweiten Datums im Beispiel, bei der auch ein menschlicher Akteur nicht sicher sein kann, um was für ein Datum es sich handelt.

Informationen, die nicht der direkten Beschreibung einer Ressource dienen, sondern ein Hilfsmittel für die korrekte Interpretation von Metadaten darstellen, werden in der vorliegenden Arbeit als *Zusatzinformationen*[1] bezeichnet.

Zusatzinformationen können, wie oben erläutert, die Struktur von Metadaten beschreiben und damit einen Teil des Arbeitskontexts des Metadatenerfassers offenlegen. Eine weiterer Weg, einem interpretierenden Akteur diesen Arbeitskontext nahezubringen, ist die

[1] Der Begriff der „Zusatzinformationen“ ist hierbei klar vom Begriff der „Meta-Metadaten“ abzugrenzen, bei denen es sich um beliebige Beschreibungen einzelner konkreter Metadaten handelt.

Darlegung von Klassifikationssystemen, die diesen Kontext semantisch repräsentieren (vgl. Abschnitt 2.3.2). Beispiele für die Verwendung von Klassifikationssystemen sind die Ablage von Ressourcen in der Verzeichnishierarchie einer Organisation oder die Klassifikation von Ressourcen in einem Bibliothekssystem [Bli35]. Beschreibungen der in einem Arbeitskontext existierenden Klassifikationssysteme stellen ebenfalls Zusatzinformationen für die Interpretation von Metadaten dar.

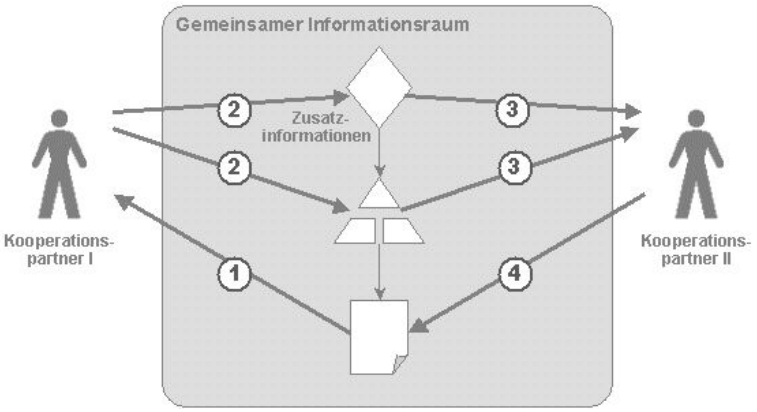

Abbildung 2.6: Freiwillige Bereitstellung von Zusatzinformationen für Metadaten

Ebenso wie Ressourcen im gemeinsamen Informationsraum durch ihre Metadaten direkt ergänzt werden, können auch die entsprechenden Zusatzinformationen für Metadaten im gemeinsamen Informationsraum abgelegt werden (vgl. Abb. 2.6, (2)). Kooperationspartner greifen dann über den Informationsraum auf die Zusatzinformationen zu und verwenden sie bei der Interpretation der Metadaten (3).

Vorgehensweise–5: Wiederverwendung von Zusatzinformationen für verschiedene Metadaten

Metadatenerfasser, die viele Ressourcen beschreiben, machen es sich oft zur Gewohnheit, dieses immer wieder nach einem bestimmten Muster zu tun. In diesem Fall lassen sich einmalig einheitliche Zusatzinformationen formulieren, die eine Reihe von Metadaten für verschiedene Ressourcen ergänzen (vgl. Abb. 2.7). Die Kenntnis der Ressourcen (1) erleichtert dabei die Festlegung der Zusatzinformationen (2). Metadaten können dann unter Bezugnahme auf die Zusatzinformationen zu beliebigen Zeitpunkten erfasst werden (3,4). Die Interpretation einer entsprechenden Metadateninstanz erfolgt durch den Kooperationspartner jeweils mit Hilfe derselben Zusatzinformationen (5).

Die *Wiederverwendung von Zusatzinformationen* für verschiedene Metadaten reduziert einerseits den Erfassungsaufwand. Andererseits

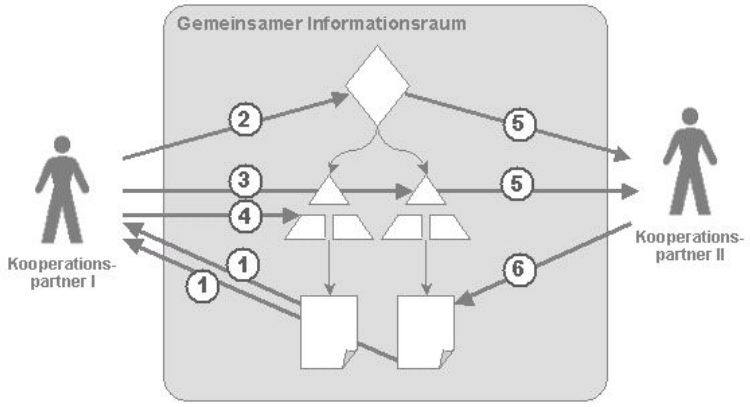

Abbildung 2.7: Wiederverwendung von Zusatzinformationen für verschiedene Metadaten

erhält der Metadatenbestand eines Erfassers eine Ordnung, die sowohl die Interpretation als auch die Erfassung selbst vereinfacht.

Vorgehensweise–6: Zusammenfassung von allgemeinen Zusatzinformationen in einem gemeinsamen Schema

Eine konsequente Weiterführung der Wiederverwendung von Zusatzinformationen (vgl. Szenarium–5) stellt die Vereinbarung von gemeinsamen Zusatzinformationen zwischen Kooperationspartnern dar. Diese einigen sich zunächst auf eine Reihe von Zusatzinformationen, denen die für die Zusammenarbeit erfassten Metadaten entsprechen müssen (1). Der Metadatenerfasser richtet sich bei der Erstellung der Metadaten nach diesen Vorgaben (3), so dass sich sein Kooperationspartner bei der Interpretation darauf verlassen kann, dass sie eingehalten sind (4).

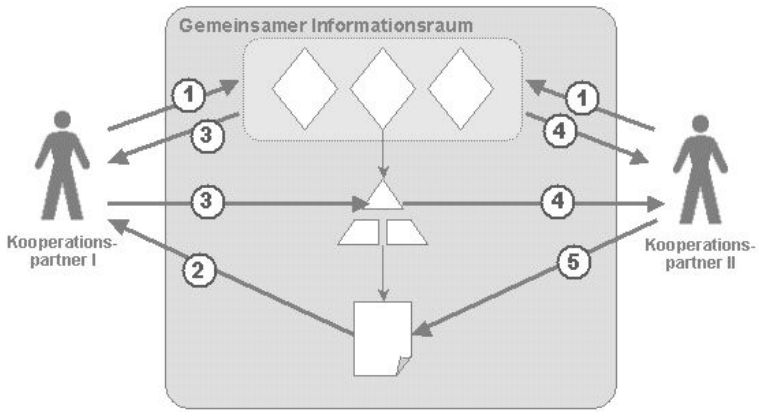

Abbildung 2.8: Zusammenfassung von allgemeinen Zusatzinformationen in einem gemeinsamen Schema

Neben den Vorteilen der Wiederverwendung einzelner Zusatzinformationen besitzt die Festlegung eines gemeinsamen Schemas den zusätzlichen Nutzen, dass dieses für den gesamten Verlauf der Kooperation Bestand hat und daher auch durch Software unterstützt werden kann [RMSS03]. So müssen Interpretationsalgorithmen nicht bei jeder Interpretation neue Zusatzinformationen auswerten, sondern können hinsichtlich des Schemas optimiert werden. Für die Erfassung können Eingabeschnittstellen zur Verfügung gestellt werden, die die Erfassung von Metadaten unterbinden, die nicht dem Kooperationsschema entsprechen.

Die Vereinbarung eines Schemas setzt jedoch die Existenz eines entsprechenden Kommunikationskanals zwischen den Kooperationspartnern voraus, der bei der asynchronen Nutzung gemeinsamer Informationsräume nicht immer gegeben ist.

Die Festlegung von Schemata ist häufig auch Gegenstand internationaler Standardisierungsvorhaben für Metadaten verschiedener Anwendungsfelder. Das MARC-Format [Fur00], das Vereinbarungen für den Austausch von digitalen Katalogeinträgen zwischen Bibliotheken enthält, und der Dublin-Core-Standard [WIC97, Lag01] für Metadaten zur Beschreibung von Informationsressourcen im Internet sind typische Beispiele für solche Standards.

Bei den aufgeführten Vorgehensweisen werden Metadaten und Zusatzinformationen eingesetzt, um die indirekte Kommunikation zwischen Kooperationspartnern über den gemeinsamen Informationsraum zu verbessern. Zugleich beinhalten sie Strategien, um den durch die Erfassung von Metadaten und Zusatzinformationen entstehenden Mehraufwand erträglich zu machen und zu begrenzen.

2.3.2 Beschreibungsaspekte von Ressourcen

Nachdem im vorigen Abschnitt betrachtet wurde, auf welche Weise Metadaten und auch Zusatzinformationen zu diesen Metadaten in gemeinsamen Informationsräumen erfasst und genutzt werden können, soll nun im Detail erläutert werden, welche Eigenschaften der im Informationsraum abgelegten Ressourcen und welche weiteren Aspekte durch Metadaten und Zusatzinformationen beschrieben und dadurch dem Kooperationspartner übermittelt werden können.

Zunächst besitzt eine Ressource eine Reihe inhärenter Eigenschaften, die aus ihr selbst abgeleitet werden können. Inhärente Eigenschaften sind z.B. der Titel eines Artikels, die Version eines Software-Werkzeugs oder das Erstellungsdatum eines Memorandums. Gerade innerhalb von Informationsräumen sind Ressourcen jedoch darüber hinaus von einem Kontext umgeben. Zum einen stehen sie miteinander in Beziehung. So kann eine Ressource z.B. eine andere referenzieren, sie kann ein Teil einer anderen Ressource sein, eine Alternative zu einer anderen Ressource darstellen usw. Zum anderen sind Ressourcen in Klassifikationssysteme ihrer Anwendungsdomäne(n) eingeordnet.

Klassifikationssysteme unterteilen die Menge der Ressourcen in nicht notwendigerweise disjunkte Teilmengen, indem sie sie nach anwendungsspezifischen Kriterien bestimmten Klassen[2] zuordnen. Eine Klasse drückt eine semantische Zusammengehörigkeit der ihr zugeordneten Ressourcen aus. Wie die Klassen gewählt werden, hängt wiederum von den durch sie unterstützten Aufgaben ab. So lässt sich beispielsweise ein von allen Mitarbeitern eines Unternehmens verwendeter Verzeichnisbaum als Klassifikationssystem auffassen und in einem gemeinsamen Informationsraum als solches übernehmen, wobei die einzelnen Verzeichnisse die Klassen des Klassifikationssystems darstellen. In Bibliotheken [NSS+98] und im Wissensmanagement [Weg02a] werden häufig Begriffsindexe verwendet. Jeder Begriff repräsentiert einen semantischen Aspekt der unterstützten Anwendungsdomäne. Ressourcen, die einem Begriff zugeordnet sind, stehen inhaltlich mit diesem in Zusammenhang. Begriffssysteme stellen ebenfalls geeignete Klassifikationssysteme für gemeinsame Informationsräume dar.

Wie die gewählten Beispiele bereits andeuten, können die Klassen eines Klassifikationssystems untereinander ebenfalls wieder in Beziehung gesetzt werden, wie Unter- und Oberverzeichnisse in Verzeichnisbäumen oder Begriffe und Querverweise darauf in einem Index. Das entstehende Netz aus Klassen gekoppelt mit der Tatsache, dass eine Klasse einen Zugriffspunkt auf alle Ressourcen darstellt, die semantisch zu ihr gehören, macht Klassifikationssysteme zu Leitsystemen, über die Kooperationspartner im gemeinsamen Informationsraum zu für sie inhaltlich relevanten Ressourcen geführt werden [RMS+01].

Metadaten beschreiben in gemeinsamen Informationsräumen alle Aspekte der Ressourcen selbst, also sowohl deren inhärente Eigenschaften, als auch ihre Beziehungen untereinander sowie ihre Zuordnung zu den Klassen der im Informationsraum repräsentierten Klassifikationssysteme (vgl. Abb. 2.9)

Zudem können Zusatzinformationen zur Verfügung gestellt werden, die Aspekte beschreiben, die nicht direkt mit bestimmten Ressourcen des Informationsraums in Verbindung gebracht werden können, deren Kenntnis jedoch die Interpretation der Metadaten unterstützen kann. Hierzu zählen zum einen Typinformationen über die Struktur der Metadaten und die zur Beschreibung verwendeten Werte. Zum anderen stellen auch Informationen über die Klassen der Klassifikationssysteme eines Informationsraums und die Beziehungen zwischen diesen Klassen allgemeine Zusatzinformationen dar.

Insgesamt ergeben sich mit den Ressourcen selbst, den Klassifikationssystemen der Anwendungsdomänen, den Metadaten und den Zusatzinformationen vier verschiedene Arten von Informationen, die in einem gemeinsamen Informationsraum abgelegt werden können. Die Ressourcen und die Klassifikationssysteme ergeben sich dabei direkt aus den durch den Informationsraum unterstützten Arbeitsaufgaben (vgl. Abb. 2.9, links), während Metadaten und Zusatzinformationen durch ihre beschreibende Funktion einen Mehrwert

[2]Der Begriff der Klasse bezeichnet im Rahmen dieser Arbeit eine Menge von Ressourcen, die nach semantischen Gesichtspunkten zusammengefasst werden, wie es beispielsweise bei der Verwendung von Beschreibungslogiken geschieht (vgl. Abschnitt 3.6), und ist abzugrenzen vom Begriff der Klasse als Typ zur Festlegung von Wertstrukturen, wie er z.B. in der objektorientierten Programmierung Verwendung findet (vgl. Abschnitt 3.2, Punkt OODM–1).

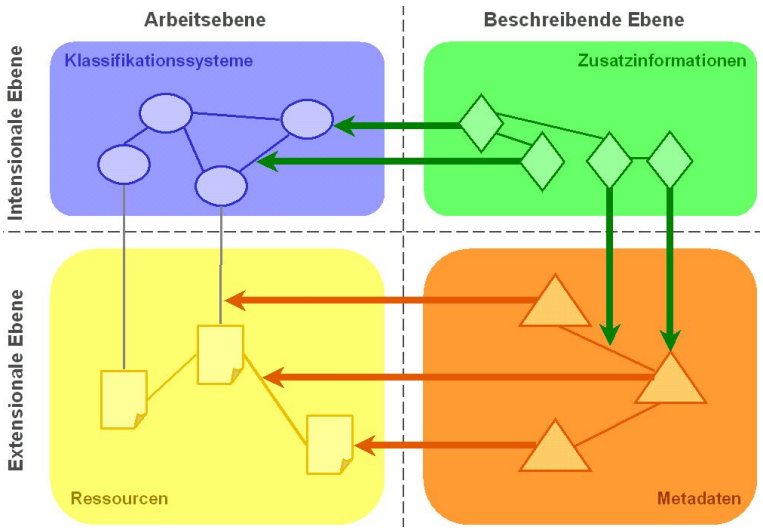

Abbildung 2.9: Beschreibungsstruktur zwischen Ressourcen, Klassifikationssystemen, Metadaten und Zusatzinformationen

in Ergänzung zur Arbeitsebene darstellen (vgl. Abb. 2.9, rechts). Bei Ressourcen und ihren Metadaten handelt es sich um konkrete Einzelobjekte, die in bestimmten Arbeitsschritten im Rahmen bestimmter Aufgaben entstehen und Verwendung finden, und damit um *extensionale* Informationen (vgl. Abb. 2.9, unten). Klassifikationssysteme und auch Zusatzinformationen haben einen *intensionalen* Charakter, da sie Erläuterungen und Vorgaben für die Nutzung ganzer Mengen von Einzelobjekten beinhalten (vgl. Abb. 2.9, oben).

Vor dem Hintergrund der in Abbildung 2.9 vorgenommenen Unterteilung gemeinsamer Informationen in eine extensionale und eine intensionale Ebene soll nun zum Abschluss kurz auf die Rolle von Meta-Metadaten eingegangen werden. Bei der Erfassung von Meta-Metadaten werden einzelne Metadateninstanzen selbst wiederum als Ressourcen aufgefasst (vgl. Abb. 2.10, 4) und durch eigene Metadaten beschrieben (7). Ein Beispiel für eine Meta-Metadateninstanz ist die Information, dass die Aussage, dass ein Dokument den Titel „abc“ hat (Metadateninstanz), vom Metadatenerfasser „xyz“ gemacht worden ist (Meta-Metadateninstanz).

Sowohl Metadaten als auch Meta-Metadaten sind auf der extensionalen Ebene eines Informationsraums angesiedelt. Sie unterstützen jedoch zumeist unterschiedliche Kooperationsaufgaben, die in Abbildung 2.10 durch die Verwendung unterschiedlicher Farben repräsentiert sind. Die Arbeit mit Meta-Metadaten unterscheidet sich dabei in ihrer Komplexität nicht von der Arbeit mit Metadaten, so dass auf allen Metaebenen dieselben Beschreibungsmechanismen zum Einsatz kommen sollten.

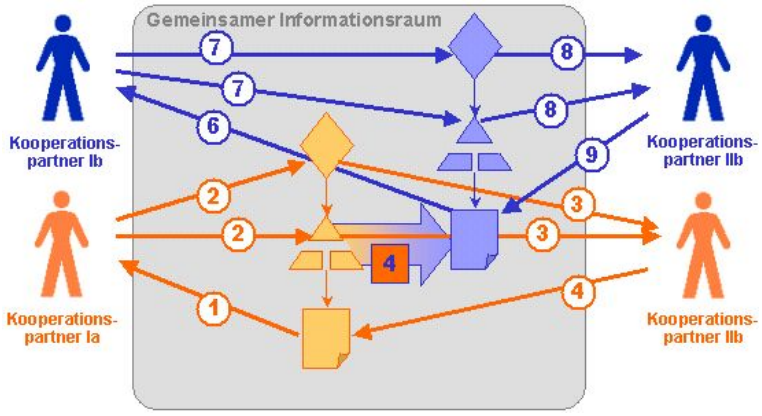

Abbildung 2.10: Beschreibung von Metadaten durch Meta-Metadaten

2.3.3 Tolerante Algorithmen zur kooperativen Nutzung

In den vorangehenden Abschnitten wurden verschiedene Vorgehensweisen für die kooperative Nutzung von Metadaten und Zusatzinformationen in gemeinsamen Informationsräumen beschrieben und aufgezeigt, welche Eigenschaften von Ressourcen und welche weiteren Aspekte durch Metadaten und Zusatzinformationen beschrieben und dadurch dem Kooperationspartner übermittelt werden können.

Jedoch lässt sich, auch wenn Ressourcen in gemeinsamen Informationsräumen durch umfangreiche und detaillierte Metadaten und Zusatzinformationen ergänzt werden, nicht garantieren, dass das Verständnis von der Ressource auf Seiten des Metadatenerfassers einem Kooperationspartner über den Kommunikationskanal der Metadaten vollständig übermittelt werden kann. Der Grund hierfür liegt darin, dass sich beide Partner in verschiedenen Kontexten befinden, die ihr jeweiliges Verständnis prägen [SW49]. Um diese Verständnisunterschiede zu überwinden, kann die Interpretation von Metadaten in gemeinsamen Informationsräumen mit Hilfe toleranter Algorithmen erfolgen.

Ein toleranter Algorithmus (vgl. Abb. 2.11) greift auf die vom Metadatenerfasser zur Verfügung gestellten Metadaten und Zusatzinformationen (2) zu, die dessen durch seinen Arbeitskontext geprägtes Verständnis von einer Ressource (1) repräsentieren. Darüber hinaus verfügt der tolerante Algorithmus jedoch auch über Informationen bezüglich des Arbeitskontexts eines interpretierenden Akteurs. Diese werden genutzt, um die im Informationsraum abgelegten Metadaten und Zusatzinformationen auf eine Repräsentation abzubilden, die dem Arbeitskontext des interpretierenden Akteurs näher ist und daher von ihm besser verstanden werden kann (4). Diese dem Interpretationskontext nahe Repräsentation, mit deren Hilfe der interpretierende Akteur seine Entscheidungen hinsichtlich der Nutzung der beschriebenen Ressource trifft (5), kann transient sein oder ihrerseits persistent als weiterer Metadateneintrag zur Ressource im Informationsraum abgelegt werden (6).

Tolerante Algorithmen fungieren somit bei der Kooperation auf der Grund-

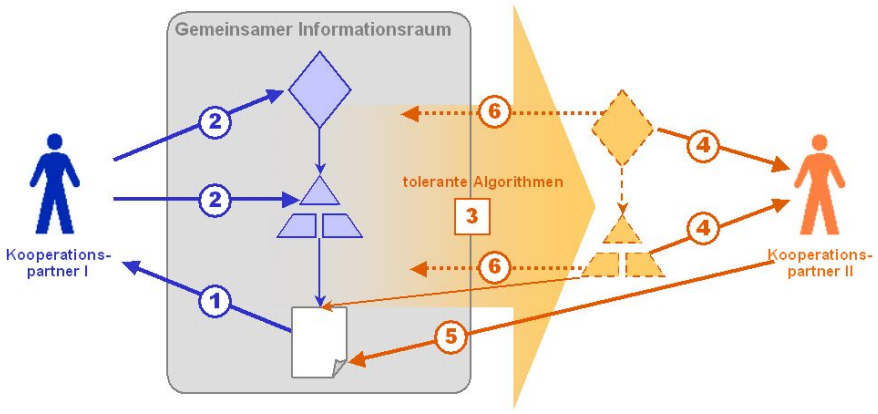

Abbildung 2.11: Tolerante Algorithmen als begleitende Strategie für die kooperative Nutzung von Metadaten

lage von Metadaten als eine Art „semantischer Übersetzungsdienst“ zwischen dem erfassenden und dem interpretierenden Akteur und können somit deren Kommunikation hinsichtlich der gemeinsam verwendeten Ressourcen weiter verbessern.

2.4 Zusammenfassung

Gemeinsame Informationsräume werden bei der synchronen und bei der asynchronen Kooperation eingesetzt und stellen Kooperationspartnern sowohl die für die Zusammenarbeit notwendigen Werkzeuge als auch einen Ablageort für gemeinsam verwendete Ressourcen zur Verfügung. Das WorldWideWeb als Pool für digitale Materialien und Dienste stellt eine weltweite Plattform für gemeinsame Informationsräume dar, in denen die asynchrone Kooperation zumeist im Vordergrund steht.

Bei der asynchronen Nutzung entfällt eine unmittelbare Verständigung der Kooperationspartner über die gemeinsam verwendeten Ressourcen. Dennoch müssen Akteure im gemeinsamen Informationsraum in die Lage versetzt werden, auch in großen Materialsammlungen einzelne, für sie relevante Ressourcen aufzufinden, ein Verständnis für Ressourcen zu entwickeln, die aus einem fremden Arbeitskontext stammen und daher u.U. ungewohnte Eigenschaften besitzen, die Qualität fremder Ressourcen einzuschätzen und an ihnen vorgenommene Änderungen zu verfolgen.

Die Erfassung von Metadaten, die kooperativ genutzte Ressourcen beschreiben, und ihre Ablage im Informationsraum gemeinsam mit den beschriebenen Ressourcen schafft einen zusätzlichen Kommunikationskanal zwischen Kooperationspartnern, die sich nun nicht mehr nur direkt über die Ressourcen, sondern auch über deren Beschreibungen miteinander verständigen können. Auf diese Weise kann den Anforderungen an asynchron genutzte Informationsräume entsprochen werden, indem die Kluft zwischen den unterschiedlichen Arbeitskontexten der Akteure mit Hilfe von Beschreibungen verringert wird und Diensten

wie Suchwerkzeugen oder Awareness-Werkzeugen Daten zur Verfügung gestellt werden, auf denen sie schnell und korrekt arbeiten können.

Für die kooperative Nutzung von Metadaten gibt es eine Reihe von Vorgehensweisen, die je nach Kooperationskontext unterschiedlich gut geeignet sind, um die Arbeit und Zusammenarbeit der Akteure zu unterstützen und den durch die Erfassung von Metadaten entstehenden zusätzlichen Arbeitsaufwand gering zu halten. Gemeinsame Informationsräume, die beliebige Kooperationsszenarien zulassen, sollten die Gesamtheit dieser Vorgehensweisen ermöglichen, so dass Kooperationspartner die für ihren Kontext passenden Vorgehensweisen jeweils wählen und individuell verfolgen können. Für gemeinsame Informationsräume, die von vornherein für ein bestimmtes Kooperationsszenarium ausgelegt sind, können die diesem Szenarium entsprechenden Vorgehensweisen gezielt unterstützt und ihre Verwendung beispielsweise durch die Bereitstellung optimierter Datenstrukturen oder die Gestaltung von Benutzerschnittstellen entsprechend gefördert werden.

Insgesamt werden somit in asynchron genutzten Informationsräumen sowohl Ressourcen als auch Metadaten abgelegt. Die Verwendung der Ressourcen kann zusätzlich durch die Repräsentation von Klassifikationssystemen erleichtert werden, die die jeweilige Anwendungsdomäne repräsentieren, während die Nutzung von Metadaten durch die Bereitstellung von allgemeinen Zusatzinformationen unterstützt werden kann, so dass sich insgesamt eine Vierteilung des Informationsraums in Ressourcen, Metadaten, Klassifikationssysteme und Zusatzinformationen ergibt.

Existierende Datenmodelle und Systeme, die zur Repräsentation und kooperativen Verwendung von Metadaten geeignet sind, unterstützen zumeist ein einzelnes Kooperationsszenarium, das zugleich mit bestimmten Vorgehensweisen und bestimmten Anteilen der Information, die in diesem Kapitel als für eine kooperative Verwendung adäquat identifiziert wurde, verbunden ist, während es andere Vorgehensweisen und Informationen nicht berücksichtigt. Im folgenden Kapitel 3 werden Datenmodelle vorgestellt, die geeignet sind, um Metadaten für bestimmte Kooperationsszenarien zu repräsentieren, und ihre zugrundeliegenden Konzepte erläutert, bevor diese Konzepte in Kapitel 4 in dem integrierenden Metadatenmodell KooMet, das eine Reihe unterschiedlicher Kooperationsszenarien abbilden kann, zusammengefasst und ergänzt werden.

Obwohl Metadaten einen neuen Kommunikationskanal zwischen erfassendem und interpretierendem Akteur in einem gemeinsamen Informationsraum eröffnen, reicht ihre alleinige Nutzung oft nicht aus, um die existierende Verständniskluft zwischen den Parteien zu schließen. Tolerante Algorithmen, die zwischen dem Arbeitskontext des Erfassers und des Nutzers vermitteln, stellen einen weiteren Schritt dar, um das Verständnis erfasster Metadaten zu vertiefen. Kapitel 5 beschreibt Fuzzy-Prädikate als ein Beispiel für tolerante Algorithmen, die im Rahmen des integrierenden Metadatenmodells KooMet zum Einsatz gebracht werden können.

Kapitel 3

Existierende Metadatenmodelle

Für die Repräsentation von Metadaten in bestimmten Kooperationskontexten existiert bereits eine Reihe geeigneter Datenmodelle. Diese unterstützen zumeist ein einzelnes oder einige wenige Kooperationsszenarien und eignen sich daher jeweils zur Realisierung einer Teilmenge der in Abschnitt 2.3.1 beschriebenen Vorgehensweisen bei der Verwendung von Metadaten in asynchron genutzten gemeinsamen Informationsräumen.

Dieses Kapitel stellt das relationale Datenmodell (vgl. Abschnitt 3.1), das objektorientierte Datenmodell (vgl. Abschnitt 3.2), die Datenmodelle in XML (vgl. Abschnitt 3.3) und im Information Retrieval (vgl. Abschnitt 3.4) und die Konzepte des Resource Description Framework (vgl. Abschnitt 3.5) sowie der Beschreibungslogiken (vgl. Abschnitt 3.6) jeweils im Einzelnen vor dem Hintergrund ihrer Eignung für die Repräsentation kooperativ genutzter Metadaten vor, indem es erläutert, welche Konzepte eines Modells welche der vorgestellten Vorgehensweisen (vgl. Abschnitt 2.3.1) unterstützen können und für die Darstellung welcher Art von Metadaten und Zusatzinformationen (vgl. Abschnitt 2.3.2) diese Konzepte geeignet sind. Für das integrierende Metadatenmodell KooMet (vgl. Kapitel 4), das eine möglichst breite Palette unterschiedlicher und u.U. auch gegensätzlicher Vorgehensweisen bei der Zusammenarbeit auf der Grundlage von Metadaten ermöglichen soll, wird gefordert, dass es die im Verlauf dieses Kapitels als geeignet identifizierten Konzepte umfasst und angemessen zusammenführt.

Bei der Untersuchung der einzelnen Modelle werden jeweils nur Konzepte aufgeführt, die in vorangehenden Abschnitten noch nicht beschrieben worden sind. Daher berücksichtigt die Beschreibung eines Modells u.U. bestimmte Konzepte nicht mehr, obwohl sie einen wichtigen Teil dieses Modells darstellen. Insgesamt ergibt sich so über die Grenzen der Abschnitte dieses Kapitels hinweg eine Auflistung von Konzepten, die einander schrittweise ergänzen und erweitern.

3.1 Das relationale Datenmodell

Das relationale Datenmodell wurde 1970 von Codd vorgestellt [Cod70]. Es zeichnet sich insbesondere durch eine mengenorientierte Verarbeitung von Daten aus, die die satzorientierte Verarbeitung der bis dahin gebräuchlichen Datenmodelle ablöste. Ein Großteil der heute existierenden kommerziell erfolgreichen Datenbanksysteme realisiert das relationale Datenmodell. Der Zugriff auf die gespeicherten Informationen erfolgt dabei zumeist über die Sprache SQL [ISO87], die auf dem relationalen Datenmodell aufbaut.

Das relationale Datenmodell weist eine Reihe charakteristischer Konzepte auf, die teilweise die Vorgehensweisen bei der Nutzung von Metadaten in gemeinsamen Informationensräumen (vgl. Abschnitt 2.3.1) unterstützen können, teilweise hierfür aber auch nicht oder nicht in vollem Umfang geeignet sind. Die folgende Aufstellung identifiziert die Konzepte des relationalen Datenmodells und ordnet sie entsprechend ein:

RDM–1: Strukturierung durch Attribute

Die Beschreibung eines Objekts der Anwendungswelt erfolgt im relationalen Datenmodell durch eine Reihe von Attribut-Wert-Paaren. Jedes Attribut repräsentiert für alle Objekte, die es beschreibt, einen bestimmten semantischen Aspekt. Vor diesem Hintergrund werden Attribute in relationalen Datenbanken verwendet, um zum einen bei Anfragen passende Objektbeschreibungen anhand ihrer Attributwerte auszuwählen (*Selektion*) und zum anderen Anfrageresultate auf bestimmte Aspekte der gefundenen Objekte zu beschränken (*Projektion*). In der folgenden SQL-Anfrage werden beispielsweise Beschreibungen von Personen anhand des dort verzeichneten Alters ausgewählt und das Ergebnis auf die Nachnamen innerhalb der ausgewählten Personenbeschreibungen beschränkt:

```
select p.nachname from Personen p where p.alter < 25
```

Da Attribute bestimmte Eigenschaften von Objekten benennen, ermöglichen sie also zum einen einen direkten Zugang zu diesen Eigenschaften und zum anderen den Vergleich verschiedener Objekte anhand dieser Eigenschaften. Beide Funktionen sind auch bei der kooperativen Arbeit mit Metadaten nutzbringend einsetzbar, indem Attribute zur Strukturierung von Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–3) verwendet werden. Sind die Metadaten einer Ressource in Attribut-Wert-Paare unterteilt, so kann ein interpretierender Akteur zum einen relevante Beschreibungen und damit indirekt auch die durch sie beschriebenen Ressourcen selbst auffinden, indem er die entsprechenden Eigenschaften dieser Ressourcen in einer Anfrage angibt. Zugleich hat er die Möglichkeit, gezielt den Teil einer Beschreibung einzusehen, der für ihn wichtig ist.

Neben der Benennung von Eigenschaften können Attribute auch mit weiteren für alle Eigenschaften gültigen Zusatzinformationen (vgl. Abschnitt 2.3.2) verknüpft werden und dadurch verschiedene

kooperative Dienste unterstützen. So kann z.B. die Zugangskontrolle innerhalb von gemeinsamen Informationsräumen über Attribute gesteuert werden, ebenso wie die einheitliche personalisierte Gestaltung von Benutzerschnittstellen [HT99] oder das Versenden von Änderungsmitteilungen [DB92].

RDM–2: Zusammenfassung von Attributen in Records

Das relationale Datenmodell fasst alle Attribut-Wert-Paare, die ein Objekt beschreiben, zusammen. Die daraus entstehende Struktur wird als *Tupel* bezeichnet. Ein Tupel ist im relationalen Modell zugleich die Beschreibung und der eindeutige digitale Repräsentant des beschriebenen Objekts. In der Datenbank entspricht es diesem Objekt, ohne dass es einen direkten Verweis darauf enthält.

Die Gesamtheit aller gespeicherten Tupel wird als *Datenbankinstanz* bezeichnet. Es gilt die Annahme, dass eine Datenbankinstanz den Zustand der entsprechenden Anwendungswelt vollständig repräsentiert und ein durch ein Tupel beschriebenes Objekt keine weiteren für die Anwendung relevanten Eigenschaften außer den durch die Attribut-Wert-Paare erfassten besitzt (*Closed-World-Assumption*).

Auch außerhalb des relationalen Datenmodells, beispielsweise in verschiedenen Programmiersprachen [Wat90], sind Strukturen anzutreffen, die in ihrem Aufbau Tupeln entsprechen. Allgemein werden Strukturen, die eine Reihe von Attribut-Wert-Paaren zusammenfassen, auch als *Records* bezeichnet. Diese Bezeichnung wird auch im weiteren Verlauf dieser Arbeit verwendet. Andere Begriffe, die in den weiteren vorgestellten Modellen Record-artige Strukturen bezeichnen, werden dabei jeweils auf den Begriff des Records abgebildet.

In kooperativen Metadatenanwendungen bietet die Zusammenfassung von Attribut-Wert-Paaren in Records zunächst eine zusätzliche Möglichkeit zur Strukturierung der Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–3), die jedoch so, wie sie im relationalen Datenmodell umgesetzt wird, nicht flexibel genug ist.

Zum einen läuft die Zusammenfassung aller Attribut-Wert-Paare in einem einzigen Record einer Ressourcebeschreibung durch multiple Metadaten zuwider (vgl. Abschnitt 2.3.1, Vorgehensweise–2). Darüber hinaus werden Metadaten häufig in Arbeitsprozessen verwendet, in denen nach der Interpretation der Metadaten der Zugriff auf die beschriebene Ressource selbst erfolgt. Eine Ressource und ihre Metadaten sind daher unabhängige Phänomene, deren Repräsentation in einem Metadatenmodell gleichermaßen wichtig ist. Das Konzept eines einzelnen Records als Repräsentant für eine Ressource, die selbst innerhalb des Datenmodells nicht in Erscheinung tritt, lässt sich daher im Bereich kooperativer Metadaten nicht halten.

Gleiches gilt für die Closed-World-Assumption. Anders als bei klas-

sischen Anwendungen des relationalen Datenmodells existiert bei der kooperativen Nutzung von Metadaten keine in sich abgeschlossene Anwendungswelt. Der Nutzen von Metadaten liegt häufig gerade darin, dass sie, einmal erfasst, für verschiedene Arbeitsaufgaben verwendet werden können. Um jedoch neue Arbeitsaufgaben vollständig durch Metadaten unterstützen zu können, müssen existierende Records anders als im relationalen Datenmodell dynamisch um neue Attribut-Wert-Paare erweitert werden können.

RDM–3: Typisierung auf Record- und auf Wertebene

Objekte, die von derselben Art sind, wie alle Personen oder alle Dokumente, werden im relationalen Datenmodell durch Tupel beschrieben, die alle aus denselben Attributen zusammengesetzt sind. Jedes Attribut kann dabei in einem Tupel nur einmal vorkommen. Die Attributmenge eines Tupels wird auch als *Tupeltyp* bezeichnet. Jedes Tupel in einer relationalen Datenbankinstanz besitzt einen Tupeltyp. Tupel gleichen Typs werden in einer Tabelle, die auch als *Relation* bezeichnet wird und dem Datenmodell seinen Namen gibt, zusammengefasst. Konkrete Relationen werden dabei häufig nach der Art der Objekte benannt, die in ihnen beschrieben sind, also z.B. als Personen-Relation oder als Dokument-Relation.

Ebenso, wie der Tupeltyp für Objekte einer Art vorgegeben ist, ist auch der Typ für die Werte eines Attributs einheitlich festgelegt. Das relationale Modell erlaubt dabei für Attribute ausschließlich Typen, die einfache Werte wie Zahlen oder Zeichenketten umfassen. Komplexe Typen wie Mengen oder geschachtelte Tupel sind nicht vorgesehen.

Bei der Erfassung von Metadaten führt das Fehlen komplexer Typen für Attributwerte zum einen dazu, dass Metadaten, die logisch innerhalb eines komplexen Wertes zusammengefasst werden sollten, über unzusammenhängende einfache Werte verteilt werden müssen. Zum anderen werden Erfasser dazu verleitet, komplexe Werte in Form von Zeichenketten zu formulieren. Diese enthalten dann zwar implizit Informationen über die Struktur des ehemals komplexen Wertes, diese kann jedoch bei der Interpretation der Zeichenkette nicht unbedingt ermittelt werden. Allgemein kann die ausschließliche Verwendung einfacher Attributwerte somit dazu führen, dass bei der Strukturierung von Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–3) die Eigenschaften der Ressource durch die Struktur der Metadaten nicht adäquat abgebildet und dadurch dem Kooperationspartner nicht komplett übermittelt werden können.

Durch die vollständige und einheitliche Typisierung von Tupeln und Attributwerten im relationalen Datenmodell wird sowohl die Optimierung von Speicherungsstrukturen und die Verwendung von Indexen [BM72] als auch eine effiziente mengenorientierte Datenverarbeitung ermöglicht. Bei der kooperativen Nutzung von Metada-

ten können Zusatzinformationen entsprechend in einem detaillierten Schema zusammengefasst werden (vgl. Abschnitt 2.3.1, Vorgehensweise–6), um die aufgezählten Vorteile auch für das Metadaten-Management zu erreichen.

Daneben machen jedoch unterschiedliche Arbeitskontexte auch andere Vorgehensweisen im Zusammenhang mit der Typisierung von Metadaten erforderlich, wie die flexible Erfassung multipler Metadaten für eine Ressource (vgl. Vorgehensweise–2) oder die freiwillige Bereitstellung oder Auslassung von Zusatzinformationen durch einen Akteur, der individuell strukturierte Metadaten erfasst hat (vgl. Vorgehensweise–4). Diese Vorgehensweisen werden durch das Typkonzept des relationalen Modells nicht unterstützt. Ansätze zur Schemaevolution in relationalen Datenbanken [RCR93] greifen zwar das Problem auf, dass sich Anforderungen an ein Datenbankschema im Laufe der Zeit verändern können und schrittweise darin aufgenommen werden müssen, dienen aber zum einen nicht der Betrachtung von Sprüngen zwischen den Schemata verschiedener Arbeitskontexte, die u.U. kaum Gemeinsamkeiten aufweisen. Zum anderen vernachlässigen diese Ansätze die Tatsache, dass sich auch innerhalb desselben Arbeitskontexts insbesondere bei der kreativen Arbeit mit Ressourcen eine neue Schemaversion selten ad hoc ergibt, sondern sich über eine längere Zeitspanne entwickelt. Die entsprechenden Metadaten müssen dann zwischenzeitlich ohne ein festes Schema verwaltet werden können [WEG+02b].

Ein weiterer Unterschied zwischen dem Typkonzept des relationalen Datenmodells und der Typisierung kooperativ genutzter Metadaten besteht darin, dass ersteres, ebenso wie typisierte Programmiersprachen, Typangaben als *einschränkende* Bedingungen begreift, so dass Werte von vornherein nur angelegt werden können, wenn sie dem geforderten Typ entsprechen. Hierdurch wird sichergestellt, dass Software, die auf gespeicherte Werte zugreift, diese stets in einer bestimmten Form vorfindet und verarbeiten kann *(Typsicherheit)*. Für Metadaten lassen sich solche Einschränkungen wiederum nur in Arbeitskontexten durchsetzen, die eine direkte Kontrolle der erfassten Metadaten erlauben, beispielsweise in geschlossenen Informationsräumen, in denen Eingaben durch ein entsprechendes Softwaresystem überprüft werden. In großen offenen Informationsräumen, die viele Arbeitskontexte nebeneinander unterstützen, wie dem WorldWideWeb, ist eine solche Kontrolle in der Regel jedoch nicht gegeben. Hier kommt der Angabe von Typen keine einschränkende, sondern eine *beschreibende* Bedeutung zu. Metadatenerfasser geben interpretierenden Akteuren mit Hilfe von Typinformationen Hinweise auf die Beschaffenheit der Metadaten. Der interpretierende Akteur muss sich darauf verlassen, dass die vorgefundenen Typangaben korrekt sind, zugleich muss er sich jedoch bewusst sein, dass diese Angaben nicht auf Korrektheit geprüft sind und damit sehr

wohl fehlerhaft sein können. Software, die mit dieser Art von Metadaten arbeitet, muss entsprechend robust ausgelegt sein und mit Typfehlern umgehen können.

Auf die Vorteile der Typisierung sollte beim Austausch von Metadaten in gemeinsamen Informationsräumen insgesamt nicht verzichtet werden. Die durch das relationale Datenmodell vorgesehene Möglichkeit, einschränkende Typinformationen in einem einheitlichen Schema zu festzulegen, sollte jedoch zugunsten des Konzepts der freiwilligen Bereitstellung von beschreibenden Typinformationen (vgl. Abschnitt 2.3.1, Vorgehensweise–4) sowohl für einzelne Werte als auch in Schemata gelockert werden. In interpretierenden Anwendungen kann dann auf vorhandene Typinformationen zurückgegriffen werden, während für die Interpretation von Werten ohne Typinformationen ein möglicher Typ entweder implizit angenommen oder anhand der Metadatensyntax ermittelt wird. Obwohl durch diesen kombinierten Ansatz aus freiwilligen und schematisierten Typinformationen in vielen Fällen die effiziente Speicherung und Verarbeitung von Metadaten erschwert wird, vereinfacht er dennoch die Interpretation, indem er den Aufwand für die Ermittlung von Typen reduziert.

RDM–4: Festlegung von Default-Werten für Attribute

Neben dem Typ für die Werte eines Attributs kann im relationalen Datenmodell auch ein Default-Wert festgelegt werden, der dem Attribut in allen Tupeln zugewiesen wird, für die der Erfasser nicht explizit einen anderen Wert angibt. Das folgende Beispiel zeigt die Definition einer Angestellte-Relation in SQL, in der jedem Angestellten die Abteilung 'EDV' zugewiesen wird, falls für das Abteilung-Attribut in einem Tupel keine anderen Angaben gemacht werden:

```
create table Angestellte (
  Nachname varchar(50),
  Vorname varchar(50),
  Abteilung varchar(30) default 'EDV')
```

Die Definition von Default-Werten kann die Erfassung von Metadaten insbesondere in großen Informationsräumen vereinfachen, in denen sich Eigenschaften für viele Ressourcen gleichen, da sie den Erfasser von der Aufgabe befreien, diese Eigenschaften für jede Ressource erfassen zu müssen, und dennoch gewährleisten, dass die entsprechende Information für Kooperationspartner zur Verfügung steht.

RDM–5: Referenzierung von Records über Schlüssel

Im relationalen Datenmodell sind Relationen mathematische Mengen aus Tupeln. Daraus folgt, dass es innerhalb einer Relation keine zwei identischen Tupel geben kann und somit jedes Tupel über die

Belegung einiger oder aller seiner Attribute innerhalb der Relation eindeutig identifiziert werden kann. Eine Menge von Attributen, über die jedes Tupel einer Relation identifiziert werden kann, wird im relationalen Modell als *Schlüssel* bezeichnet. In der folgenden Personen-Relation ist beispielsweise die Menge {Vorname, Nachname, Geburtstag} ein möglicher Schlüssel:

Nachname	**Vorname**	**Geburtstag**	**Gehalt**
Meier	Karl	24.12.1958	2.000 EUR
Meier	Karl	03.10.1962	2.000 EUR
Meier	Kurt	04.07.1961	2.000 EUR

Dadurch, dass der Schlüssel Tupel eindeutig identifiziert, kann er in anderen Relationen verwendet werden, um auf diese Tupel Bezug zu nehmen. In dieser Funktion wird er als *Fremdschlüssel* bezeichnet. In der folgenden Relation werden Personen über den oben festgelegten Schlüssel als Leiter einer Abteilung referenziert:

AbtName	**LeiterNachn**	**LeiterVorn**	**LeiterGeb**	**...**
Buchhaltung	Meier	Karl	24.12.1958	...
EDV	Meier	Karl	03.10.1962	...
...	...	...	...	...

Die Referenzierung von bereits erfassten Informationen ermöglicht deren Wiederverwendung in verschiedenen Kontexten und vermeidet redundante Kopien, die einerseits einen erhöhten Erfassungsaufwand mit sich bringen und andererseits Quellen für mögliche Inkonsistenzen darstellen.

Bei der kooperativen Nutzung von Metadaten verringert die Referenzierung bereits existierender Metadaten den Erfassungsaufwand. Je häufiger auf vorhandene Metadaten verwiesen wird, desto geringer ist zudem die Wahrscheinlichkeit dafür, dass widersprüchliche Metadaten zu einer Ressource erfasst werden. Die Interpretation von Metadaten wird dadurch vereinfacht.

Das Konzept der Referenzierung lässt sich somit für die Strukturierung von Metadaten einsetzen (vgl. Abschnitt 2.3.1, Vorgehensweise–3). Die Möglichkeit, Records als Ganzes zu referenzieren, ist dabei jedoch nicht ausreichend. Zusätzlich müssen auch kleinere Bestandteile einer Metadatenstruktur referenzierbar sein, wie einzelne Werte, die für sich genommen bereits umfangreich oder kompliziert zu erfassen sein können und deren Wiederverwendung sich daher auszahlen kann.

Die Verwendung von Schlüsseln als Referenzen ist für kooperativ genutzte Metadaten nicht geeignet, da hier nicht sichergestellt werden kann, dass zwei unterschiedliche Ressourcen nicht dieselbe Beschreibung besitzen. Nur eine Trennung zwischen der Identität und der

Beschreibung von realen Objekten, wie sie beispielsweise im objektorientierten Datenmodell vollzogen ist (vgl. Abschnitt 3.2, Punkt OODM–2), kann eindeutige Referenzen innerhalb eines Metadatenbestandes gewährleisten.

RDM–6: Beziehungen als eigenständige Entitäten

Über die Festlegung der Eigenschaften für einzelne Anwendungsobjekte hinaus können die Relationen des relationalen Datenmodells auch genutzt werden, um beliebige n-stellige *Beziehungen* zwischen verschiedenen Objekten darzustellen. Hierzu werden die Schlüssel für die Tupel der in Beziehung stehenden Objekte in einem neuen Tupel zusammengefasst. Darüber hinaus kann dieses Beziehungstupel mit weiteren Attributen ergänzt sein, die die Beziehung näher beschreiben. Die folgende Relation z.B. beschreibt ternäre Beziehungen, in denen Professoren, referenziert durch ihren Nachnamen und ihren Arbeitsbereich, Studenten, referenziert durch ihre Matrikelnummer, in einer Lehrveranstaltung, referenziert durch eine Kursnummer, unterrichten. Die Beziehungen sind ergänzt durch die Information, in welchem Jahr der entsprechende Kurs abgehalten wurde und welche Note der Student darin erzielt hat:

Prof	**Bereich**	**Matr**	**KursNr**	**Jahr**	**Note**
...	...	...	...	...	...
Schulz	Softwaresysteme	42878	18.9543	1999	5,0
Meier	Softwaresysteme	42878	18.9543	2000	3,0
Schulz	Verteilte Systeme	42878	18.9927	1999	3,3
...	...	...	...	...	...

Die Tupel in einer solchen Relation repräsentieren Beziehungen als gleichberechtigte Entitäten, die durch Attribute beschrieben und in anderen Relationen referenziert werden können.

In gemeinsamen Informationsräumen stehen Ressourcen in vielfältigen Beziehungen zueinander (vgl. Abschnitt 2.3.2), die bei der strukturierten Erfassung von Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–3) in Abgrenzung zu den inhärenten Eigenschaften einer einzelnen Ressource durch eigene Entitäten beschrieben werden können.

Zusätzlich zu der oben erwähnten Referenzierbarkeit und Beschreibbarkeit der Beziehungen ergeben sich daraus weitere Vorteile. So ist bei der Interpretation der Metadaten die Auswertung eines einzigen Metadatenkonstrukts, nämlich der Beziehung selbst, ausreichend, um auf die Beziehung schließen zu können. Wird die Beziehung zwischen Ressourcen dagegen dargestellt, indem im Attributwert einer Ressourcebeschreibung eine andere referenziert wird, so müssen bei der Interpretation von Metadaten unter Umständen erst die Beschreibungen aller beteiligten Entitäten durchlaufen werden, bis die

> Beziehung zwischen ihnen zu Tage tritt. Darüber hinaus besteht hier ein semantisches Ungleichgewicht zwischen den an der Beziehung beteiligten Objekten, während diese innerhalb von eigenständigen Beziehungsentitäten einander gleichberechtigt gegenüber stehen.

Das relationale Datenmodell beinhaltet somit eine Reihe von Konzepten, die sich auf den Einsatz von Metadaten in gemeinsamen Informationsräumen anwenden lassen. Konzepte zur Strukturierung, wie die Unterteilung von Informationen in Attribut-Wert-Paare und die Zusammenfassung dieser Paare in Records, ermöglichen strukturiertes Arbeiten und vergrößern die Übersicht sowohl bei der Erfassung als auch bei der Interpretation von Metadaten. Die Record-Strukturen des relationalen Datenmodells sind jedoch zu starr und beschränken die Flexibilität des Metadatenerfassers.

Die Repräsentation von Beziehungen als eigenständige Entitäten ist ein weiteres Strukturierungsmittel des relationalen Datenmodells. Beziehungen können für Metadaten genutzt werden, um Informationen über Zusammenhänge sowohl zwischen Ressourcen als auch zwischen ihren Beschreibungen an zentraler Stelle zur Verfügung zu stellen, wiederum zu beschreiben und zu referenzieren.

Typangaben zu Werten und Records stellen auch in gemeinsamen Informationsräumen wichtige Zusatzinformationen dar, die die Interpretation von Metadaten vereinfachen können. Hierbei muss jedoch sowohl die Vorgehensweise der freiwilligen Bereitstellung von Typinformationen als auch die Vorgehensweise der Zusammenfassung von Typinformationen in einem Schema ermöglicht werden (vgl. Abschnitt 2.3.1). Die vollständige Typisierung im relationalen Datenmodell ermöglicht nur die zweite Vorgehensweise. Darüber hinaus besitzen Typinformationen in gemeinsamen Informationsräumen beschreibenden Charakter und unterscheiden sich damit von den als Einschränkungen interpretierten Typangaben im relationalen Datenmodell.

Sowohl die Möglichkeit zur Formulierung von Default-Werten als auch zur Referenzierung von Informationen sind Konzepte, die die Erfassung von Metadaten vereinfachen können. Die Verwendung von Schlüsselwerten ist jedoch für die Referenzierung von Metadaten-Records nicht geeignet. Darüber hinaus muss die Referenzierung nicht nur ganzer Records, sondern auch anderer Metadatenbestandteile ermöglicht werden.

3.2 Das objektorientierte Datenmodell

Das objektorientierte Paradigma postuliert eine Abkehr von einer prozessorientierten Sicht auf Software hin zu einer datenorientierten Sicht, die sich die Tatsache zu Nutze macht, dass Daten zumeist den stabilen Teil eines Systems ausmachen und daher eine solidere Basis für die Software-Entwicklung bilden als die Funktionen des Systems [Mey88]. Auf dieser konzeptuellen Grundlage entwickelten sich ab der zweiten Hälfte der 60er Jahre zunächst objektorientierte Programmiersprachen [DN66, GR83], die später durch objektorientierte Analyse- und Design-Methoden [RBP+91, Boo94] und objektorientierte Datenbankmodelle ergänzt [ABW+90, CB97] wurden.

Für das objektorientierte Datenmodell nennen [ABW+90] eine Reihe von Charakteristika, von denen die wichtigsten im Folgenden aufgeführt und im Einzelnen darauf hin untersucht werden, ob sie zusätzlich zu den bereits als geeignet identifizierten Eigenschaften des relationalen Modells ebenfalls Relevanz für die Verwendung in kooperativen Metadatenanwendungen besitzen:

OODM–1: Komplexe Werte in Form von Kollektionen und Records

Im relationalen Datenmodell können Attribute nur durch einfache Werte belegt werden (vgl. Abschnitt 3.1, Punkt RDM–3). Das objektorientierte Datenmodell dagegen lässt auch komplexe Werte in Form von Records, Mengen oder Listen zu, die es erlauben, Phänomene der Anwendungswelt direkt im Datenmodell abzubilden.

Records sind im objektorientierten Datenmodell in Form von *Objekten* realisiert, die dem Datenmodell zugleich seinen Namen geben. Neben einer Reihe von Attribut-Wert-Paaren, die seinen Zustand repräsentieren, verfügt ein Objekt auch über ein eigenes Verhalten. Dieses soll hier zunächst vernachlässigt werden und wird später in Punkt OODM–3 diskutiert.

Ein Objekt kann als Wert in Attribut-Wert-Paaren anderer Objekte auftreten, so dass sich hier eine Schachtelung von Objekten und damit von Record-Strukturen ergibt. So wird im folgenden Beispiel, das in der Programmiersprache Java formuliert ist, festgelegt, dass ein Person-Objekt über eine Reihe von Attributen verfügt, wovon eines wiederum durch ein Objekt belegt ist, das seinerseits die strukturierte Beschreibung der Adresse der Person enthält:

```
class Person {
  String vorname;
  String nachname;
  int geburtsjahr;
  Adresse anschrift;
}

class Adresse {
  String strasse;
  int hausnr;
  int plz;
  String ort;
}
```

Die Verwendung eines konkreten Objekts zur Belegung eines Attributs vollzieht sich in der Objektorientierung über die Referenzierung dieses Objekts, das jeweils eine eigene Identität besitzt (vgl. Punkt OODM–2).

Neben Objekten können auch Kollektionen wie Mengen oder Listen als Werte in einem Attribut-Wert-Paar eingesetzt werden. Dabei

dienen Mengen der Darstellung von ungeordneten Sammlungen unterschiedlicher Elemente, wie von Orten, an denen eine Ressource verfügbar ist, oder von Software-Produkten, mit denen sich ein digitales Dokument öffnen lässt. Listen erweitern diesen Kollektionsbegriff, indem sie zusätzlich eine Ordnung auf Kollektionselementen definieren, wie in der Autorenliste eines Dokuments.

Für die Realisierung von Kollektionswerten existieren in konkreten Ausprägungen objektorientierter Modelle unterschiedliche Ansätze. So sieht der Object Database Standard ODMG 2.0 vor, dass alle Elemente einer Kollektion vom gleichen Typ und Kollektionen somit homogen sein müssen [CB97]. Kollektionselemente in der Programmiersprache Java hingegen sind stets vom Wurzeltyp Object und können damit praktisch jedem Subtyp von Object angehören. Hierdurch können in Java auch heterogene Kollektionen erstellt werden [Eck98].

Zur konkreten Erzeugung und Verwendung von komplexen Werten sowohl in Form von geschachtelten Objekten als auch von Kollektionen stehen entsprechende Konstruktoren und Operatoren zur Verfügung. Diese sind orthogonal einsetzbar, wodurch auch Kollektionen und Objekte erzeugt und bearbeitet werden können, die ihrerseits wiederum komplexe Werte enthalten.

Die oben aufgeführten Beispiele für komplexe Werte machen bereits deutlich, dass diese auch im Rahmen der Metadatenerfassung auftreten können. Die Verwendung komplexer Werte in Metadaten unterstützt dabei eine Strukturierung, die dem Arbeitskontext des Erfassers entspricht (vgl. Abschnitt 2.3.1, Vorgehensweise–3).

Die Bestimmung eines einheitlichen Typs für Kollektionselemente, wie sie in [CB97] vorgesehen ist, kann als Maßnahme zur Unterstützung der Interpretation dienen. Zugleich muss die Struktur der Metadaten jedoch flexibel an die unterschiedlichen Arbeitskontexte von Metadatenerfassern anpassbar sein. Daher kann die Angabe eines Typs für alle Kollektionselemente hier nur eine Option sein, neben der ebenso die Möglichkeit zur Erstellung untypisierter und heterogener Kollektionen existiert.

OODM–2: Eindeutige Identifikation von Records

Im relationalen Datenmodell werden Tupel über einen Schlüssel referenziert. Dieser umfasst eine Kombination von Attributwerten, die für jedes Tupel eindeutig sein müssen (vgl. Abschnitt 3.1, Punkt RDM–5). Das objektorientierte Datenmodell hingegen weist jedem Objekt einen eindeutigen Identifikator zu und realisiert dadurch den Unterschied zwischen gleichen und identischen Objekten. Nicht identische Objekte können im objektorientierten Datenmodell durch dieselben Werte beschrieben werden und bleiben über ihre Identifikatoren dennoch unterscheidbar.

Während der Einsatz von Schlüsseln für die Erfassung von Metadaten nicht geeignet ist (vgl. Abschnitt 3.1), kann die Verwendung von Identifikatoren die Erfassung strukturierter Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–3) vereinfachen. Zum einen lassen sich beliebige Bestandteile einer Metadatenstruktur mit Identifikatoren ausstatten, so dass sie nicht mehrfach erfasst werden müssen, sondern referenziert und dadurch wiederverwendet werden können. Diese Vorgehensweise bietet sich insbesondere für umfangreiche Bestandteile einer Metadatenstruktur an. Da eine Wiederverwendung auch und gerade von Zusatzinformationen, die die Struktur und den Anwendungskontext von Metadaten näher beschreiben, die Erfassung von Metadaten ebenfalls erleichtert (vgl. Abschnitt 2.3.1, Vorgehensweise–5), sollten auch diese Zusatzinformationen mit einer eigenen Identität versehen werden können.

Darüber hinaus lassen sich durch Identifikatoren auch die durch Metadaten beschriebenen Ressourcen referenzieren. Hierdurch wird eine getrennte Speicherung von Metadaten und Ressource erlaubt und die Beschreibung von Ressourcen, die sich außerhalb des gemeinsamen Informationsraums befinden (vgl. Abschnitt 2.3.1, Vorgehensweise–1), überhaupt erst ermöglicht.

Die kooperative Nutzung von Metadaten verläuft häufig über die Grenzen verschiedener Systeme hinweg, so dass Identifikatoren, anders als in den meisten objektorientierten Systemen, nicht automatisch vergeben, sondern von den Kooperationspartnern den entsprechenden Entitäten zugewiesen werden müssen. Da ein Metadatenbestand oft aus einer Vielzahl einzelner Bestandteile zusammengesetzt ist (vgl. z.B. Abschnitt 2.3.1, Vorgehensweise–2), kann dieses eine mühsame oder sogar unlösbare Aufgabe darstellen. Darüber hinaus muss ein Akteur einen Überblick über alle existierenden Identifikatoren besitzen, um sie in seinen Referenzen verwenden zu können. Dieses kann für große oder detaillierte Bestände ebenfalls nicht als selbstverständlich vorausgesetzt werden, so dass ein gemeinsamer Informationsraum weitere Methoden zur Referenzierung von einzelnen Metadatenkonstrukten vorsehen muss.

OODM–3: Kapselung

Ein Objekt dient im objektorientierten Datenmodell als Kapsel, die ihrer Umgebung über eine Schnittstelle einen Satz von Operationen zur Verfügung stellt. Innerhalb des Objekts ist zum einen sein Zustand in Form von Attribut-Wert-Paaren und zum anderen die Implementation der Operationen verborgen. Der Zustand des Objekts kann nur über die Schnittstelle abgerufen oder verändert werden. Programme, die das Objekt verwenden, sind daher unabhängig von der jeweiligen Implementation einer Operation, auch wenn diese sich ändern sollte.

Das Konzept der Kapselung beruht darauf, dass die Definition ei-

nes Objekttyps nicht nur die Festlegung der Datenstruktur der entsprechenden Objekte, sondern auch die Angabe der für die Objekte gültigen Operationen umfasst. Die Definition von Objekttypen findet im Rahmen eines Software-Entwicklungsprozesses statt, an dem Akteure teilhaben, die die notwendigen Kenntnisse für die Festlegung und Realisierung von Operationen besitzen.

Die Typisierung von Metadaten findet jedoch auf einer anderen Ebene statt. Hier definieren Metadatenerfasser Typen als Zusatzinformationen, um die von ihnen erfassten Werte näher zu beschreiben und damit die Interpretation der Metadaten zu unterstützen (vgl. Abschnitt 2.3.1, Vorgehensweise–4). Es kann nicht davon ausgegangen werden, dass ein Metadatenerfasser in der Lage ist, Operationen für einen Typ anzugeben. Daher beschränkt sich die Definition eines Metadatentyps auf die Beschreibung der Struktur und der Ausprägungen für mögliche Werte dieses Typs. Die Implementierung von Operationen für die Werte verlagert sich dagegen in die Realisierung von Interpretationsalgorithmen hinein (vgl. Kapitel 5), die auf die durch den Typ vorgegebenen Zusatzinformationen über die Werte zurückgreifen. Aufgrund dieser Verlagerung lässt sich das Konzept der Kapselung bei der Kooperation auf der Grundlage von Metadaten nicht anwenden.

OODM–4: Ableitung neuer Objekttypen

Das objektorientierte Datenmodell setzt Objekttypen innerhalb einer *Typhierarchie* zueinander in Beziehung, in der sich Subtypen durch *Vererbungsmechanismen* von ihren Supertypen ableiten. Programmierer können diese Vererbungsmechanismen nutzen, um schrittweise ein eigenes, für ihre Anwendungsdomäne geeignetes Typsystem zu konstruieren. Der Programmcode der Supertypen braucht auf diese Weise nicht kopiert zu werden, sondern wird innerhalb der Subtypen wiederverwendet. Hierdurch werden ähnliche Anwendungsobjekte durch ähnliche Datenstrukturen abgebildet und weisen ein ähnliches Verhalten auf. Objekte des Subtyps können in objektorientierten Programmen an Stelle ihres Supertyps verwendet werden (*Substituierbarkeit*). In der Nutzung von vordefinierten und benutzerdefinierten Typen gibt es keinerlei Unterschiede. Objektorientierte Programmiersprachen bieten für beide Arten von Typen dieselben Konstrukte an.

[ABW+90] unterscheiden vier verschiedene Ansätze zur Vererbung: Bei der *Substitutionsvererbung* wird ein Subtyp definiert, indem zusätzliche Operationen für Objekte dieses Typs bereitgestellt werden. Die *Inklusionsvererbung* sieht einen Typ als Subtyp eines anderen Typs an, wenn alle Objekte des Subtyps auch im Supertyp enthalten sind. Bei der *Einschränkungsvererbung* werden Bedingungen formuliert, die für die Objekte des Subtyps gelten müssen. So bilden beispielsweise Personen im Alter zwischen 13 und 19 den Sub-

typ Teenager des Supertyps Person. Die *Spezialisierungsvererbung* schließlich definiert für den Subtyp zusätzliche Attribute.

Typhierarchien können den Mehraufwand begrenzen, der in gemeinsamen Informationsräumen bei der Erfassung von Zusatzinformationen zu Metadaten entsteht (vgl. Abschnitt 2.3.1, Vorgehensweise–4), indem sie die Wiederverwendung bereits erfasster Typinformationen für Records und Beziehungen ermöglichen. Zum anderen machen sie Zusammenhänge zwischen verschiedenen Typen deutlich, die bei der Interpretation von Metadaten genutzt werden und insbesondere die Implementierung von Interpretationsalgorithmen vereinfachen können. Vor diesem Hintergrund ermöglicht die Substituierbarkeit von Werten eines Supertyps durch Werte eines Subtyps die direkte Weitergabe von Metadaten, auch wenn diese nicht exakt auf die Verwendung beim entsprechenden Kooperationspartner zugeschnitten sind. So kann beispielsweise ein detaillierter Record mit zahlreichen Attribut-Wert-Paaren auch von Interpretationsalgorithmen verarbeitet werden, die nur auf wenige Attribut-Wert-Paare zugreifen.

Da für die Definition von Typen innerhalb von Metadaten die Festlegung möglicher Operationen für die entsprechenden Werte entfällt (vgl. Punkt OODM–3), kann das Konzept der Substitutionsvererbung hier nicht angewendet werden, die anderen drei Formen der Vererbung sind jedoch Kandidaten für den Aufbau von Typhierarchien für Metadaten.

Bei der Erfassung von Zusatzinformationen für Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–4) besteht oft der Bedarf, auch den Aufbau einfacher Werte in einem Typ detailliert zu beschreiben, wie z.B. bei Personennamen oder Ortsbezeichnungen. Hierfür bedarf es jedoch einer Erweiterung der im objektorientierten Modell enthaltenen Vererbungsmechanismen um Methoden zur Ableitung neuer aus vorhandenen einfachen Typen anhand des Aufbaus der in ihnen enthaltenen Werte.

Das objektorientierte Datenmodell umfasst somit eine Reihe zusätzlicher Konzepte, die sich für eine Verwendung beim Austausch von Metadaten in gemeinsamen Informationsräumen nutzen lassen. Die Möglichkeit, Attributen komplexe Werte zuzuordnen, erlaubt die intuitive Erfassung von Metadaten entsprechend dem Arbeitskontext des Erfassers.

Identifikatoren eigenen sich sowohl zur Referenzierung von Ressourcen, die nicht gemeinsam mit ihren Metadaten abgelegt sind, als auch von einzelnen Metadatenbestandteilen. Da Identifikatoren für Metadaten häufig manuell erfasst werden müssen, kann eine reine Verwendung von Identifikatoren insbesondere für große Metadatenbestände jedoch arbeitsaufwendig und unübersichtlich werden.

Der Aufbau von Typhierachien begrenzt den Mehraufwand bei der Erfassung von Zusatzinformationen zu Metadaten. Die Substituierbarkeit innerhalb der Typhierachie kann die Interpretation von Metadaten, die in einem fremden

Arbeitskontext erfasst wurden, vereinfachen. Typhierachien sind ein Konzept, das im Metadatenkontext auch für die Definition einfacher Typen geeignet ist.

Die Festlegung von Operationen auf Metadaten ist aufgrund mangelnder Kenntnisse der Mehrheit der Metadatenerfasser für die Kooperation in gemeinsamen Informationsräumen nicht durchführbar. Objektorientierte Konzepte wie Kapselung und Substitutionsvererbung sind daher nicht anwendbar.

3.3 XML und ergänzende Standards

Die Spezifikation der *Extensible Markup Language*, kurz *XML*, wurde 1998 vom WorldWideWeb Consortium veröffentlicht [XML00], um die Kluft zwischen *SGML* [ISO86] auf der einen und HTML [HTM99] auf der anderen Seite durch eine Sprache zu überbrücken, die sich sowohl für den Publishing-Bereich als auch für den Austausch von Dokumenten und Nutzdaten im WorldWideWeb eignet [Bos97].

XML strukturiert Informationen innerhalb von Dokumenten, die sowohl mit einfachen Texteditoren als auch mit speziellen XML-Werkzeugen erstellt werden können. Die Strukturierung erfolgt mit Hilfe von Markierungen (*Tags*), die die Bestandteile der Struktur umschließen. Hierdurch können XML-Daten sowohl innerhalb von Textdokumenten als auch separat davon erfasst werden.

Die Tatsache, dass XML selbst keine festen Tags für Anwendungen definiert, findet in seiner Einordnung als *Metasprache* Ausdruck. Zur Definition anwendungsspezifischer Schemata existieren verschiedene *Schemabeschreibungssprachen*, wie *DTD*s und *XML Schema* [XML01a]. In XML werden Schemata für die strukturelle Validierung und Verarbeitung von XML-Dokumenten verwendet. Darüber hinaus können sie zur Dokumentation von Anwendungsdaten dienen und im Kooperationskontext verbindliche Vorgaben liefern, anhand derer die beteiligten Partner ihre XML-Dokumente erstellen [Wal02].

Mittlerweile wird XML nicht nur durch Schemabeschreibungssprachen, sondern auch durch weitere Standards ergänzt, die sich auf bestimmte Nutzungsaspekte von XML beziehen und seine Einsatzfähigkeit verbessern. So unterstützt die *XPath*-Spezifikation [XPa99] die Formulierung von Pfadausdrücken, über die auf einzelne Bestandteile innerhalb von XML-Strukturen zugegriffen werden kann. Die Funktionalität von XPath machen sich weitere Spezifikationen zu Nutze, wie *XPointer* zur Navigation durch verschiedene XML-Dokumente [XPT02], *XQL* (*XML Query Language*) zur Suche in XML-Daten sowie *XSL* (*XML Stylesheet Language*) zur Formatierung von XML-Dokumenten [XSL01]. Daneben werden mit *XLink* (*XML Linking Language*) Mechanismen zur Referenzierung von XML-Dokumenten und deren Bestandteilen zur Verfügung gestellt [XLi01]. Diese führen über die in HTML enthaltenen Mechanismen zur Referenzierung hinaus, indem z.B. bidirektionale und multiple Referenzen, Rollen und eine eigenständige Verlinkung abseits des Dokumentkontexts ermöglicht werden.

Insbesondere die Möglichkeit der Strukturierung von Daten innerhalb von Textdokumenten und des Austauschs von XML-Dokumenten über das WorldWideWeb als großem gemeinsamen Informationsraum macht XML und seine

ergänzenden Standards auch für die kooperative Nutzung von Metadaten relevant, so dass ihre diesbezüglichen Charakteristika im Folgenden untersucht werden sollen:

XML–1: Flexible Zusammenfassung von Attributen[1] in Records

Ein XML-Dokument besteht aus einem einzelnen Wurzelelement, das weitere *XML-Elemente* umfassen kann. Allgemein kann jedem XML-Element ein Wert zugeordnet sein, der entweder ein Literal und damit nicht weiter zerlegbar ist oder der wiederum aus einer Reihe von XML-Elementen besteht. XML-Elemente können somit in beliebiger Tiefe geschachtelt werden.

XML-Elemente, die ihrerseits weitere XML-Elemente enthalten, lassen sich als Records auffassen. Die geschachtelten Elemente entsprechen dabei den Attribut-Wert-Paaren in einem Record. Das Attribut lässt sich auf den Namen des XML-Elements abbilden, während der Wert zwischen den beiden Element-Tags erscheint.

Im relationalen und im objektorientierten Datenmodell tritt ein Attribut innerhalb eines Records genau dann ein einziges Mal auf, wenn es im entsprechenden Record-Typ aufgeführt ist. XML hingegen bietet bei der Zusammenfassung von XML-Unterelementen eine größere Flexibilität. So ist es in einem XML-Dokument, falls es nicht anders durch ein Schema vorgegeben ist, zulässig, beliebige XML-Elemente zusammenzufassen, wobei sich auch XML-Elemente mit demselben Namen auf derselben Ebene wiederholen dürfen. Dieses kommt der Zusammenfassung beliebiger, auch sich wiederholender Attribute in einem Record gleich. So stellt die folgende Beschreibung für ein Buch für sich genommen ein korrektes XML-Dokument dar:

```
<?XML version="1.0" encoding="UTF-8" ?>
<Buch>
  <Kommentar>Ich finde das Buch langweilig</Kommentar>
  <Titel>Ein ganz wichtiges Buch</Titel>
  <Autor>V. Verfasser</Autor>
  <Kommentar>Ist ein ganz prima Buch</Kommentar>
</Buch>
```

Durch Attribut-Wert-Paare, die sich für ein Attribut wiederholen, kommt hierbei die gleiche Semantik zum Ausdruck wie durch ein einzelnes Attribut, das in einem Attribut-Wert-Paar mit einer Kollektion belegt ist. Dennoch unterscheiden sich beide Möglichkeiten

[1] XML definiert zusätzlich zu dem hier verwendeten Begriff eines Attributs als semantischem Aspekt zur Beschreibung einer Ressource den spezielleren Begriff des *XML-Attributs* als Ergänzung eines XML-Elements. Um Unklarheiten zu vermeiden, werden XML-Attribute stets genau als solche bezeichnet, wenn auf sie Bezug genommen wird. Tritt der Begriff „Attribut“ wie hier ohne das Präfix „XML“ auf, so handelt es sich um den Begriff in seiner allgemeineren Bedeutung.

durch die Vorgehensweise bei der Erfassung. So ist beim Anlegen eines Kollektionswerts für ein Attribut von vornherein klar, dass in der Kollektion mehrere Elemente enthalten sein können, während sich wiederholende Attribut-Wert-Paare nach und nach zu einem Record hinzugefügt werden können, ohne dass dieses vorab voraussehbar ist. Auf diese Weise stellen sich wiederholende Attribute gerade für die kooperative Arbeit mit Metadaten eine interessante Alternative dar, die insbesondere auch die asynchrone Pflege von Metadaten-Records durch verschiedene Akteure unterstützt.

Insgesamt eröffnet die Flexibilität in der Zusammenfassung von Attributen bei der Strukturierung kooperativ genutzter Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–3) allen Kooperationspartnern die Möglichkeit, Attribut-Wert-Paare dem eigenen Arbeitskontext entsprechend zu erfassen und dabei weder auf für sie relevante Paare verzichten noch irrelevante Paare erfassen zu müssen.

Bereits vorhandene Metadaten verschiedener Akteure lassen sich bruchlos integrieren (vgl. Abschnitt 2.3.1, Vorgehensweise–2), ohne dass hierbei schon ein Aufwand für eine strukturelle Konvertierung oder Zusammenführung entsteht, wie er bei fest vorgegebenen Record-Strukturen notwendig wäre. Daneben wird die spontane Erfassung von Metadaten, wie von Kommentaren, Kritik oder Hinweisen auf Fehler in einer Ressource, unterstützt. Insgesamt motiviert die flexible Zusammenfassung von Attributen somit die Erfassung eines vielfältigen Spektrums unterschiedlicher Metadaten, die bei anderen Kooperationspartnern neue Möglichkeiten im Umgang mit der beschriebenen Ressource aufzeigen können.

XML–2: Variable Vergabe von Typen

Während im relationalen und im objektorientierten Modell sowohl einfache als auch komplexe Werte durchgängig typisiert sind und ihr Aufbau somit vorgeschrieben ist, können XML-Dokumente ganz ohne Typisierung auskommen. So sind auch XML-Dokumente ohne Angabe eines dazugehörigen Schemas zulässig.

XML und verschiedene seiner begleitenden Standards beinhalten darüber hinaus Mechanismen, die es erlauben, einem XML-Dokument als Ganzem oder auch Teilen davon Typinformationen hinzuzufügen. So kann wie im folgenden Beispiel im Kopf eines XML-Dokuments die Angabe eines Schemas auftreten, dem das Dokument entspricht:

```
<?XML version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Buecher SYSTEM "../DTDs/buecher.dtd">
```

Alle XML-Elemente, die in diesem Dokument erscheinen, müssen im angegebenen Schema definiert sein und in ihrer Struktur den dortigen Definitionen genügen.

Der XML-Namespace-Standard [XML99] bietet zusätzlich die Möglichkeit, Typinformationen aus verschiedenen Schemata auf Ebene der XML-Elemente in ein XML-Dokument einzubringen. Jedem Schema wird dabei im öffnenden Tag des XML-Elements ein Präfix zugewiesen. Geschachtelte XML-Elemente, deren Namen dieses Präfix vorangestellt ist, müssen den Typinformationen des jeweiligen Schemas entsprechen. Das folgende Beispiel weist dem XML-Element Bestellung Typinformationen aus den Schemata bestellung.dtd und buecher.dtd zu:

```
<Bestellung xmlns="file:/DTDs/bestellung.dtd"
            xmlns:buch="file:/DTDs/buecher.dtd">

  <BestNr>12345</BestNr>
  <Artikel>
    <buch:Buch>
      <buch:Titel>Ein ganz wichtiges Buch</buch:Titel>
      <buch:Autor>V. Verfasser</buch:Autor>
    </buch:Buch>
  </Artikel>
</Bestellung>
```

XML-Elemente, die in buecher.dtd definiert sind, tragen das Präfix buch. Bei dem Schema bestellung.dtd, das ohne Präfix angeben ist, handelt es sich um ein Default-Schema, dem alle Elemente ohne Präfix genügen müssen. Default-Schemata erleichtern die Erfassung von XML-Daten.

Ein Schema gilt immer für das XML-Element, in dem es angeben wird, sowie in allen darin geschachtelten Elementen. Um die Erfassung nicht typisierter XML-Elemente auf Schachtelungsebenen unterhalb eines typisierten Elements zu ermöglichen, enthalten Schemabeschreibungssprachen spezielle Wildcards, die für beliebige XML-Elemente stehen. So wird im folgenden Beispiel in XML Schema durch das any-Element angegeben, dass das XML-Element Sonstiges eine Reihe beliebiger Unterelemente enthalten kann:

```
<xsd:element name="Sonstiges">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Mit den beschriebenen Mechanismen lassen sich in XML die Vollständigkeit und die Granularität von Typinformationen frei variieren. Typisierung wird somit nicht erzwungen, sondern Typen werden dann angegeben, wenn es angemessen ist.

Ähnlich wie im relationalen und im objektorientierten Datenmodell können auch Schemata für XML-Daten zur Speicher- und Anfrageoptimierung herangezogen werden [ABS00]. Für XML-Dokumente ohne ein zugeordnetes Schema existieren hier ergänzend Verfahren, mit deren Hilfe sich aus einer Menge von XML-Dokumenten ein entsprechendes Schema ableiten lässt [AMN94], sowie Optimierungsalgorithmen, die ohne Schema allein auf Grundlage der allgemeinen strukturellen Eigenschaften von XML arbeiten [MW99].

Um Schemata für die Speicherung und Verarbeitung von XML-Daten heranziehen zu können, wird im Rahmen einer strukturellen *Validierung* überprüft, ob die entsprechenden XML-Dokumente den angegebenen Schemata entsprechen. In diesem Zusammenhang lassen sich Typinformationen in XML als Bedingungen auffassen, die die Erfassung von Daten einschränken (vgl. hierzu auch RDM–3). Da sich in XML tief verschachtelte Datenstrukturen aufbauen lassen, dienen vorgehaltene Typinformationen interpretierenden Akteuren jedoch zusätzlich als Grundlage für die Navigation durch die Daten und für die effektive Anfrageformulierung [GW97]. In diesem Sinne haftet XML-Schemata auch ein beschreibender Charakter an, der der Verwendung von Typinformationen für kooperativ genutzte Metadaten entspricht.

Allgemein ist das Konzept der variablen Vergabe von Typen eine der Grundlagen für die Nutzung von Metadaten in gemeinsamen Informationsräumen, da es unterschiedliche Formen der Kooperation gleichermaßen unterstützt. So ist eine enge, detailliert geplante Zusammenarbeit auf der Basis eines strikten gemeinsamen Schemas (vgl. Abschnitt 2.3.1, Vorgehensweise–6) ebenso möglich wie freiere Formen der Zusammenarbeit, in denen Metadaten nur dann mit Typinformationen ausgestattet werden, wenn es dem Erfasser adäquat erscheint (vgl. Vorgehensweise–4).

XML–3: Definition der möglichen Kombinationen von Attributen in Record-Typen

Die in Punkt XML-1 beschriebene Flexibilität bei der Kombination von Unterelementen innerhalb von XML-Elementen führt dazu, dass sich bei der Typisierung von XML-Elementen eine Reihe von Struktureigenschaften definieren lassen, die über die bloße Zusammenfassung der Unterelemente hinausgeht. So lassen sich dadurch, dass ein XML-Element grundsätzlich beliebige Unterelemente beinhalten kann, in einem XML-Element-Typ alternative Unterelemente festlegen, von denen jeweils nur eines im Oberelement auftritt. Die entsprechenden XML-Elemente ähneln somit varianten Records, wie sie auch in verschiedenen Programmiersprachen verwendet werden (vgl. z.B. [Wir71]). Des weiteren können in XML-Element-Typen Quantitäten definiert werden, so dass bestimmte Unterelemente innerhalb eines XML-Elements in einer bestimmten Mindest- oder

Höchstanzahl enthalten sind.

XML wird nicht nur zur Repräsentation von Daten, sondern im Publishing-Bereich auch zur Strukturierung von Dokumenten insgesamt eingesetzt. [KM03] beispielsweise unterscheiden vor diesem Hintergrund zwischen datenzentrierten und dokumentzentrierten XML-Dokumenten. In dokumentzentrierten XML-Dokumenten spielt die Reihenfolge der enthaltenen XML-Elemente häufig eine Rolle. Diese lässt sich daher bei der Typisierung von XML-Elementen ebenfalls festlegen. XML weicht hier von traditionellen Datenmodellen ab, in denen der Reihenfolge von Attribut-Wert-Paaren in Records keine Bedeutung beigemessen wird.

XML-Schemabeschreibungssprachen wie DTDs und XML Schema umfassen demnach Konzepte zur Definition der Struktur in XML-Element-Typen, die über die des relationalen und des objektorientierten Modells für Record-Typen hinausgehen. Die Ordnung zwischen Unterelementen und Gruppen von alternativen Unterelementen werden hier mit Hilfe von unterschiedlichen *Modellgruppen* (*model groups*) definiert. Eine Modellgruppe fasst dabei eine Reihe von Unterelementen zusammen und drückt aus, dass diese Elemente entweder in der aufgeführten Reihenfolge oder alternativ zueinander im umschließenden XML-Element auftreten. Der folgende Auszug aus einer DTD besagt z.B., dass in einem Buch-Element die XML-Elemente Titel und Autor in genau dieser Reihenfolge auftreten müssen und dass ein Kommentar-Element entweder eine Anmerkung, eine Kritik oder einen Fehler beinhaltet:

```
<!ELEMENT Buch (Titel,Autor)>
<!ELEMENT Kommentar (Anmerkung | Kritik | Fehler)>
```

In XML Schema werden in anderer Syntax die gleichen Modellgruppen zur Verfügung gestellt wie in DTDs, ergänzt durch eine weitere Gruppe, die als *ALL-Gruppe* bezeichnet wird und durch die eine Struktur für Records definiert werden kann, wie sie aus dem relationalen und dem objektorientierten Datenmodell bekannt ist (vgl. Abschnitt 3.1, Punkt RDM–3). In XML-Elementen, die der ALL-Gruppe entsprechen, kommt somit ein aufgeführtes Unterelement jeweils genau einmal vor, wobei die Unterelemente in einer beliebigen Reihenfolge erscheinen. Im folgenden Beispiel wird eine ALL-Gruppe verwendet, um festzulegen, dass ein Buch-Element jeweils genau ein Titel-Element und ein Autor-Element in beliebiger Reihenfolge zusammenfasst:

```
<xsd:complexType name="Buch">
  <xsd:all>
    <xsd:element name="Titel" type="Titel"/>
```

```
    <xsd:element name="Autor" type="Autor"/>
  </xsd:all>
</xsd:complexType>
```

Zur Quantifizierung von Unterelementen in einem XML-Element stellen DTDs die beiden allgemeinen Indikatoren „∗“ und „+“ bereit, die aussagen, dass ein XML-Element beliebig viele bzw. ein oder mehr Unterelemente mit gleichem Namen besitzen kann. Fehlen diese Indikatoren, so tritt das entsprechende Element genau einmal innerhalb des umschließenden XML-Elements auf. Der folgende Auszug aus einer DTD besagt beispielsweise, dass sich ein Buch-Element aus genau einem Titel-Element gefolgt von beliebig vielen Kapitel-Elementen zusammensetzt:

```
<!ELEMENT Buch (Titel,Kapitel*)>
```

In XML Schema lässt sich darüber hinaus die genaue Zahl der Unterelemente eines XML-Elements angeben. So drückt das folgende Beispiel aus, dass ein Buch-Element aus genau einem Titel-Element gefolgt von drei oder mehr Kapitel-Elementen besteht:

```
<xsd:complexType name="Buch">
  <xsd:sequence>
    <xsd:element name="Titel" type="Titel"/>
    <xsd:element name="Kapitel" type="Kapitel"
         minOccurs="3" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Für die Strukturierung von Metadaten in gemeinsamen Informationsräumen (vgl. Abschnitt 2.3.1, Vorgehensweise–3) wurde die flexible Record-Struktur als ein Mittel identifiziert, um der Vielfältigkeit der durch die Kooperation verbundenen Arbeitskontexte gerecht zu werden. Bei der schrittweisen Erweiterung von flexiblen Records durch einzelne Attribut-Wert-Paare kann es dabei vorkommen, dass für ein Attribut mehrere Paare zu einem Record hinzugefügt werden. Die so entstandene Struktur gleicht semantisch der Verwendung eines Kollektionswerts für das entsprechende Attribut (vgl. Punkt XML–1).

Die Festlegung eines Record-Typs beschreibt den Aufbau der dazugehörigen Records. Wird hier für ein Attribut von vornherein erkannt, dass es mit mehreren Werten belegt werden kann, so kann ein Kollektionstyp für den entsprechenden Attributwert vorgesehen werden. Innerhalb dieses Kollektionstyps kann neben der Anzahl der möglichen Werte für das Attribut auch deren Typ und die genaue Semantik der Kollektion festgelegt werden. Die Möglichkeit, auch

Quantitäten für sich wiederholende Attribute festzulegen, bedeutet an dieser Stelle eine zweite Ausdrucksmöglichkeit für denselben Sachverhalt und sollte zu Gunsten der Einfachheit bei der Typisierung von Metadaten-Records entfallen.

Ebenso erscheint das Konzept einer Reihenfolge für Attribute in einem Record als ungeeignet für die Typisierung von Metadaten-Records, da einerseits jedes Attribut-Wert-Paar als eine von den anderen Paaren unabhängige Beschreibungseinheit für eine Ressource gedeutet wird und da andererseits gerade in Szenarien der asynchronen und räumlich verteilten Kooperation, wie sie in gemeinsamen Informationsräumen hauptsächlich anzutreffen sind, eine Reihenfolge für Attribute bei der Metadatenerfassung nur schwierig durchzusetzen ist.

Die Festlegung alternativer Attribute hingegen lässt sich als Zusatzinformation zu Metadaten-Records verwenden und vereinfacht insbesondere die Typisierung von Records, die Ressourcen beschreiben, die viele gleiche Eigenschaften aufweisen und sich nur durch wenige Attribute unterscheiden.

XML–4: Ableitung neuer einfacher Typen

Während in XML selbst und auch in DTDs einfache Datentypen hauptsächlich auf verschiedene Arten von Zeichenketten beschränkt sind, besitzt XML Schema ein umfassendes Typsystem für einfache Datentypen [XML01b]. Einfache Typen werden hier weiter unterteilt in *atomare Typen*, *Listentypen* und *Vereinigungstypen*.

Atomare Typen beinhalten Werte, die nicht weiter unterteilbar sind. XML Schema bietet eine Reihe von vordefinierten atomaren Typen an. Hierzu gehören Typen für Zeichenketten, spezielle XML-Typen, wie ID und IDREF, Typen für Zeitangaben, numerische Typen sowie Wahrheitswerte (vgl. Abb. 3.1).

Der Wertebereich von Listentypen umfasst Listen von durch Leerzeichen getrennten atomaren Werten. Da Listen nicht durch XML-Markup, sondern durch Leerzeichen strukturiert werden, werden sie in XML Schema als einfache Werte betrachtet.

Ein Vereinigungstyp fasst verschiedene einfache Typen zusammen, so dass sein Wertebereich aus der Vereinigungsmenge der Wertebereiche dieser Typen besteht.

Neben der Bildung von Vereinigungstypen bietet XML Schema noch eine weitere Möglichkeit, um bereits existierende einfache Typen als Grundlage für die Definition neuer Typen zu verwenden, indem *Einschränkungen* für die existierenden Typen formuliert werden können. Die Einschränkung eines einfachen Typs kann durch eine oder mehrere *Facetten* beschrieben werden, die jeweils den Wertebereich des Typs auf eine bestimmte Weise weiter begrenzen. XML

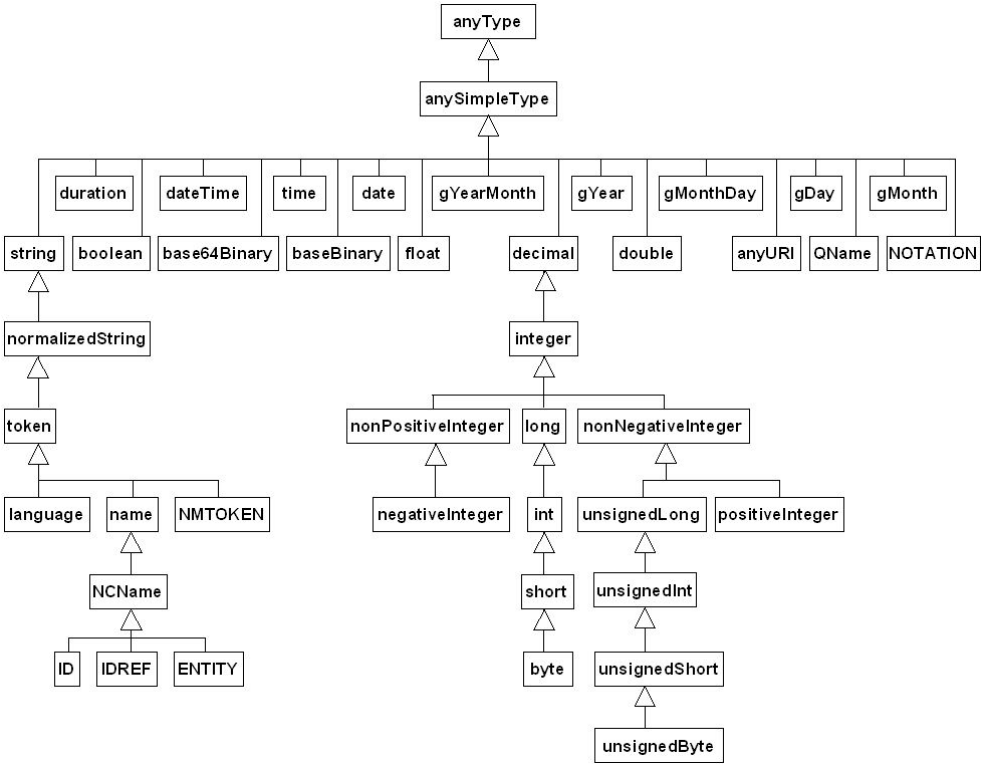

Abbildung 3.1: Hierarchie der in XML Schema vordefinierten einfachen Typen [XML01b]

Schema stellt Facetten zur Verfügung, die die Länge von Wertepräsentationen einschränken, Facetten, die die Größe von zugelassenen Werten näher bestimmen, Facetten, die die Repräsentation von zugelassenen numerischen Werten beschreiben, sowie jeweils eine Facette, die die Normalisierung von Leerzeichen in den Repräsentationen beschreibt, eine, die gültige Werte aufzählt, und eine, die ein bestimmtes Muster für zugelassene Werte mit Hilfe von regulären Ausdrücken beschreibt. Im folgenden Beispiel wird ein Typ für Bewertungen definiert, der den in XML Schema vordefinierten atomaren Typ integer auf die Zahlen zwischen 1 und 6 einschränkt:

```
<xsd:simpleType name="Bewertungstyp">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="6"/>
  </xsd:restriction>
</xsd:simpleType>
```

Die Ableitung neuer einfacher Typen sowohl durch die Bildung von Vereinigungstypen als auch durch die Formulierung von Einschränkungen unterstützt die Erfassung detaillierter Zusatzinformationen

in gemeinsamen Informationsräumen (vgl. Abschnitt 2.3.1, Vorgehensweise–3), indem zum einen der Erfassungsaufwand für neue einfache Typen verringert wird und zum anderen durch die Offenlegung von Zusammenhängen zwischen verschiedenen einfachen Typen die Interpretation von Metadaten begünstigt wird.

Bei der Nutzung von einschränkenden Facetten kann ebenso wie bei komplexen Typen die Substituierbarkeit des Basistyps durch den abgeleiteten Typ genutzt werden, um Interpretationsalgorithmen zu optimieren. Vereingungstypen vereinfachen bei der Kooperation auf der Grundlage multipler Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–2) eine spätere Zusammenfassung verschiedener Repräsentationen derselben Eigenschaft. So lässt sich der oben definierte Bewertungstyp in einem Vereinigungstyp um die Werte gut, mittel und schlecht erweitern, wenn diese in einem anderen Arbeitskontext anstelle der Zahlen 1 bis 6 verwendet werden:

```
<xsd:simpleType name="KoopBewertungstyp">
  <xsd:union memberTypes="Bewertungstyp">
    <xsd:simpleType>
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="gut"/>
        <xsd:enumeration value="mittel"/>
        <xsd:enumeration value="schlecht"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:union>
</xsd:simpleType>
```

Die Definition von Listentypen als einfache Typen in XML Schema hingegen besitzt den Nachteil, dass die Aufteilung einer Liste in ihre Unterelemente nur implizit über Leerzeichen repräsentiert ist. Während in XML einfache Listentypen insbesondere für die Typisierung von XML-Attributen verwendet werden, ist für das integrierende Metadatenmodell KooMet das Konzept komplexer Kollektionstypen (vgl. Abschnitt 3.2, Punkt OODM–1) vorzuziehen, deren Werte explizit strukturiert sind.

XML–5: Referenzierung einzelner Strukturbestandteile über Pfadausdrücke

Mit Hilfe des XPath-Standards [XPa99] können XML-Elemente, XML-Attribute oder Textstellen in einem XML-Dokument referenziert werden, auch wenn sie nicht über einen eigenen Identifikator verfügen. Hierzu stellt XPath eine Reihe von Ausdrücken zur Verfügung, die die referenzierten Bestandteile anhand der Struktur des XML-Dokuments und ihrer Position darin identifizieren.

XPath verwendet Pfadausdrücke, um durch die geschachtelten XML-Elemente zu navigieren. So referenziert der Pfadausdruck

```
Buch/Vorwort/Autor
```

das XML-Element Autor in der folgenden XML-Datenstruktur, indem er Schritte entlang der Hierarchie der XML-Elemente angibt, die durchlaufen werden müssen, um auf das referenzierte Element zu stoßen:

```
<Buch>
  <Vorwort>
    <Text>Das vorliegende Werk ...</Text>
    <Autor>W. Wichtig</Autor>
  </Vorwort>
  ...
</Buch>
```

Da die Schachtelung von XML-Elementen beliebig sein kann, sofern sie nicht durch ein Schema eingeschränkt ist (vgl. Punkt XML-1), ist nicht immer von vornherein klar, in welchem XML-Element ein Unterelement auftritt und auf welcher Ebene der Gesamtstruktur es sich befindet. Um XML-Elemente in verschiedenen Kontexten referenzieren zu können, stellt XPath die Platzhalter $*$ und // zur Verfügung. Der Platzhalter $*$ steht dabei für ein beliebiges XML-Element. Der Platzhalter // steht für das Ausgangselement und alle in beliebiger Tiefe darin geschachtelten XML-Elemente. Der erste der folgenden beiden Ausdrücke referenziert somit den Titel der Einleitung und den Titel des Kapitels in der darauf folgenden XML-Struktur, während sich der zweite Ausdruck auf alle Titel-Elemente bezieht:

```
Buch/*/Titel
```

```
Buch//Titel
```

```
<Buch>
  <Titel>XML-Grundlagen<Titel>
  <Einleitung>
    <Titel>Einleitung und Motivation</Titel>
    <Text>...</Text>
  </Einleitung>
  <Kapitel>
    <Titel>Grundlagen</Titel>
    <Abschnitt>
      <Titel>SGML</Titel>
      <Text>...</Text>
    </Abschnitt>
    ...
  </Kapitel>
  ...
</Buch>
```

Des weiteren lassen sich in XPath Prädikate formulieren, die XML-Elemente anhand ihrer Position innerhalb des umschließenden Elements oder anhand ihres Element-Inhalts auswählen. Der erste der folgenden zwei Ausdrücke bezieht sich auf das dritte Kapitel-Element einer Struktur, während der zweite Text-Elemente referenziert, die die Zeichenkette „XPath“ enthalten:

```
Kapitel[3]
Text[contains(.,"XPath")]
```

Die ausschließliche Verwendung von Identifikatoren zur Referenzierung von Bestandteilen strukturierter Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–3) kann sowohl die Erfassung als auch die Verknüpfung und die Interpretation von Metadaten verkomplizieren (vgl. Abschnitt 3.2, Punkte OODM–2). Pfadausdrücke sind dazu geeignet, Metadatenwerte anhand der Metadatenstruktur selbst zu referenzieren, auch wenn diese nicht vollständig bekannt ist. Sie stellen damit eine Alternative zur Referenzierung von Metadaten über Identifikatoren dar. Metadaten sollten daher in einer Form strukturiert sein, die die Verwendung von Pfadausdrücken zur Referenzierung ermöglicht.

XML–6: Unterscheidung zwischen Metadaten und ihren Meta-Metadaten

Ein XML-Element kann durch beliebig viele *XML-Attribute* beschrieben sein, die innerhalb des öffnenden Element-Tags aufgeführt sind und jeweils einen atomaren Wert besitzen. Dabei darf ein XML-Attribut in der Beschreibung eines XML-Elements nur jeweils einmal vorkommen. Im folgenden Beispiel gibt das XML-Attribut xml:lang die Sprache an, in der der Inhalt des XML-Elements Kommentar verfasst ist:

```
<Kommentar xml:lang="en">
  This is an English comment
</Kommentar>
```

Die Verwendung von XML-Attributen zur Erfassung von Metainformationen für XML-Elemente hat den Vorteil, dass bei der Visualisierung von XML-Dokumenten im allgemeinen nur die Informationen zwischen den Element-Tags dargestellt werden. Die Werte der XML-Attribute sind somit nicht sichtbar, können aber von einer interpretierenden Anwendung trotzdem ausgewertet und verwendet werden.

Im Rahmen dieser Arbeit lassen sich die Inhalte von XML-Elementen als Metadaten auffassen, während es sich bei den Werten von XML-Attributen um Meta-Metadaten handelt (vgl. Abschnitt 2.3.2).

> Die Verarbeitung von Metadaten und ihren Meta-Metadaten erfolgt häufig in unterschiedlichen Arbeitskontexten. Eine logische Trennung von Metadaten und Meta-Metadaten ist notwendig, um zu gewährleisten, dass diese ihren jeweiligen Kontexten zugeordnet werden können, und vereinfacht dadurch die Interpretation sowohl durch Algorithmen als auch durch Personen. Insbesondere verhindert sie, dass sich Metadaten und Meta-Metadaten zyklisch aufeinander beziehen und dadurch ihre Bedeutung vollständig einbüßen (für ein Beispiel vgl. Abschnitt 3.5, Punkt RDF–3).
>
> In XML wird die Trennung von Metadaten und Meta-Metadaten über ihre unterschiedliche Darstellung durch XML-Elemente einerseits und durch XML-Attribute andererseits erreicht. XML-Attribute können jedoch nur mit einfachen Werten belegt werden und dürfen auch nicht mehrfach in einem Element-Tag erscheinen. Die Darstellungsmöglichkeiten für Meta-Metadaten sind also im Vergleich zu denen für Metadaten eingeschränkt. Zudem beschränkt sich die Erfassung von Meta-Metadaten auf nur eine Ebene, da für XML-Attribute keine weiteren Metadaten erfasst werden können.
>
> Einschränkungen dieser Form sind für ein integrierendes Metadatenmodell nicht adäquat, da hier davon ausgegangen werden muss, dass die Anforderungen des Arbeitskontexts, in dem die Meta-Metadaten verwendet werden, ebenso hoch sind wie die Anforderungen des Arbeitskontexts der Metadaten auf der Ebene darunter. Daher sollten für die Erfassung von Metadaten und Meta-Metadaten auf beliebigen Ebenen einheitlich dieselben Konzepte zur Verfügung stehen, so dass die Arbeit mit Meta-Metadaten nicht künstlich behindert wird.

XML und seine ergänzenden Standards beinhalten einige zusätzliche Konzepte, die für die Kooperation auf der Grundlage von Metadaten in gemeinsamen Informationsräumen geeignet sind. So erlaubt die flexible Zusammenfassung von Attribut-Wert-Paaren in Records eine dem Arbeitskontext des jeweiligen Akteurs angemessene Metadatenerfassung und erhöht die Interoperabilität zwischen Metadaten aus verschiedenen Arbeitskontexten.

Auch für die Typisierung von Metadaten stellt XML zusätzliche Konzepte bereit. Die variable Vergabe von Typinformationen in unterschiedlicher Granularität, wie sie für XML durch die Verwendung von Namensräumen *(Namespaces)* realisiert ist, ermöglicht verschiedene Vorgehensweisen bei der Vergabe von Zusatzinformationen in gemeinsamen Informationsräumen (vgl. Abschnitt 2.3.1). Die Festlegung alternativer Attribute in Record-Typen vereinfacht die Erfassung und die Interpretation von Metadaten, ebenso wie die Ableitung neuer einfacher Typen durch Facetten und Vereinigung ihrer Wertemenge, wie sie in XML Schema vorgesehen ist. Einfache Listentypen hingegen scheinen speziell auf die Nutzung in XML-Attributen zugeschnitten zu sein. Hier ist die Verwendung komplexer Typen zur Darstellung von Kollektionen vorzuziehen.

Für die Referenzierung von Metadatenbestandteilen stellen Pfadausdrücke,

wie sie mit XPath definiert sind, eine geeignete Ergänzung zur Verwendung von Identifikatoren dar.

Die Unterscheidung von XML-Elementen und sie beschreibenden XML-Attributen schließlich ist ein Beispiel für die konzeptuelle Trennung von Metadaten und ihren Meta-Metadaten, die erforderlich ist, um die damit verbundenen Kooperationskontexte zu unterscheiden. Die durch XML-Attribute gegebene Einschränkung der Formulierungsmöglichkeiten für Meta-Metadaten entspricht jedoch nicht in jedem Fall der Komplexität von Aufgaben, denen Meta-Metadaten zugrunde liegen.

3.4 Datenmodelle im Information Retrieval

Information Retrieval [vR75, SM84] umfasst die Vorbereitung und Auwertung von Suchanfragen auf Sammlungen von Dokumenten, wobei der Inhalt der Dokumente die Grundlage bildet. Der Fokus lag hier ursprünglich auf Textdokumenten, wurde aber mittlerweile auch auf Bild-, Audio- und Videodokumente ausgeweitet. Die Anfänge der Information Retrieval Forschung reichen bis in die 1960er Jahre zurück. Aufgrund der Einführung der automatischen Textverarbeitung und des Personal Computers sowie der Verringerung der Kosten für digitalen Speicherplatz stieg die Zahl der verfügbaren digitalen Dokumente in den 70er und 80er Jahren drastisch an, wodurch Information Retrieval Systeme zu kommerzieller Relevanz verholfen wurde. Mittlerweile steht mit dem WorldWideWeb eine globale Sammlung unzähliger digitaler Dokumente zur Verfügung. Auch hier verwenden Suchmaschinen [BP98] Information Retrieval Technologie, um den Informationsbedarfen von Internet-Nutzern nachkommen zu können.

Bereits früh wurde im Information Retrieval die Bedeutung der automatischen Indexierung von Dokumenten erkannt. Die Dokumente werden hier vorverarbeitet, indem automatisch ein Index über die in den Dokumentinhalten repräsentierten Begriffe erzeugt wird. Die Auswertung von Suchanfragen erfolgt anhand dieses Indexes, ohne dass dabei auf die indizierten Dokumente zugegriffen werden muss.

Da Retrieval-Indexe zum einen eine spezielle Art von Metadaten darstellen und zum anderen einen ersten Schritt in Richtung einer Beschreibung von Anwendungsdomänen mit Hilfe von Klassen (vgl. Abschnitt 2.3.2) verkörpern, werden im Folgenden diejenigen Konzepte des Information Retrieval beschrieben, die für die asynchrone Kooperation in gemeinsamen Informationsräumen Relevanz besitzen:

IR–1: Begriffliche Klassifikation von Ressourcen

> Im Information Retrieval erfolgt bei der automatischen Indexierung eine Zuordnung von Dokumenten zu einem *Begriff*, wodurch zum Ausdruck kommt, dass die Dokumente in inhaltlichem Zusammenhang zu diesem Begriff stehen. Die Zuordnung wird anhand bestimmter Kriterien vorgenommen, die für jedes Dokument des Bestands untersucht werden. Ein einfaches Kriterium für die Indexierung eines Dokuments unter einem Begriff ist beispielsweise eine

hohe Vorkommenshäufigkeit des Begriffs innerhalb des Dokuments. Bei der Auswertung von Anfragen dient ein Begriff im Index als Einstiegspunkt, über den alle ihm zugeordneten Dokumente direkt aufgefunden werden können.

Da der Retrieval-Index semantische Aspekte der in ihm angeordneten Dokumente repräsentiert, stellt er eine Art Vokabular der in den Dokumenten beschriebenen Anwendungsdomäne dar und kann damit als einfaches Klassifikationssystem (vgl. Abschnitt 2.3.2) betrachtet werden, in dem Begriffe als Klassen unverbunden nebeneinander stehen. Hierbei wird bereits eine Funktion solcher Klassen in einem Klassifikationssystem deutlich. Sie dienen als Einstiegspunkte, über die Akteure, die mit der Semantik der Anwendungsdomäne vertraut sind, auf gesuchte Informationen zugreifen können.

Die Kooperation in gemeinsamen Informationsräumen kann durch Einstiegspunkte in Form von Klassen ebenfalls unterstützt werden, da Kooperationspartner über diese Klassen entlang des eigenen Arbeitskontexts direkt zu benötigten Ressourcen geführt werden. Die Einschränkung von Klassen auf Begriffe, die wie im Information Retrieval den Inhalt der Ressourcen beschreiben, erscheint im Kontext beliebiger Kooperationsszenarien jedoch als unangemessen. Vielmehr müssen Klassen zur Unterteilung des Ressourcenbestands anhand von beliebigen, dem jeweiligen Szenarium entsprechenden Aspekten verwendet werden können.

IR–2: Gewichtete Klassifikation

Einige Retrieval-Indexe enthalten für jede Zuordnung eines Dokuments zu einem Begriff zusätzlich eine *Gewichtungsinformation*. Diese drückt aus, wie stark der inhaltliche Zusammenhang von Dokument und Begriff ist, und wird bei der Auswertung von Anfragen in Retrieval-Modellen mit bestmöglicher Übereinstimmung *(best-match models)* [TC92] wie dem Vektorraummodell oder dem probabilistischen Modell verwendet. Diese Modelle basieren auf der Annahme, dass es in der Relevanz, die ein Dokument für eine Anfrage besitzt, beliebige Abstufungen gibt, und ordnen daher im Anfrageergebnis die Dokumente in eine *Relevanzrangfolge* (relevance ranking) ein.

Übertragen in den Kontext gemeinsamer Informationsräume stellt die Gewichtung eine zusätzliche *Beschreibung der Zuordnung* von Ressourcen zu Klassen und damit eine spezielle Art von Metadaten dar (vgl. Abschnitt 2.3.2). Neben der Gewichtung sind hier weitere Informationen bezüglich dieser Zuordnung denkbar, die ebenfalls für eine Interpretation von Metadaten und Ressourcen verwendet werden können, wie eine Benennung der Zuordnung, der Erfasser der Zuordnung, ein Kommentar, der die Zuordnung begründet o.ä. Insgesamt sollten daher Metadaten, die Zuordnungen von Ressourcen zu Klassen der Anwendungsdomäne beschreiben, nicht nur die

bloße Existenz der Zuordnungen feststellen, sondern darüber hinaus auch Informationen, die die Zuordnungen selbst betreffen, aufnehmen können.

Retrieval-Indexe sind spezielle Datenstrukturen für eine einzige Anwendungsklasse. Darüber hinaus sind sie jedoch auch ein Beispiel für den Einsatz von Metadaten, die mit Hilfe von Klassen Ressourcen in eine Anwendungsdomäne einordnen und dadurch die Grundlage zum Auffinden inhaltlich bedeutsamer Ressourcen bilden. Gewichtungen innerhalb des Indexes erhöhen die Qualität der Anfrageergebnisse. Überträgt man dieses Beispiel auf die kooperative Nutzung von Metadaten für unterschiedliche Anwendungsklassen, so kann eine Verallgemeinerung des Konzepts, Ressourcen Klassen zuzuordnen und diese Zuordnungen zusätzlich zu beschreiben, das Auffinden geeigneter Ressourcen und ihre optimale Verwendung unterstützen.

3.5 RDF und RDF Schema

Das *Resource Description Framework RDF* ist 1999 vom WorldWideWeb Consortium (W3C) erstmals spezifiziert und speziell für die Beschreibung von Ressourcen entwickelt worden [RDF04b]. Es umfasst ein Datenmodell, das die Struktur von RDF-Metadaten beschreibt und auf speziellen Graphstrukturen basiert [RDF04a]. Für dieses Modell gibt es darüber hinaus eine Syntax, die die Darstellung von RDF-Metadaten in XML (vgl. hierzu auch Abschnitt 3.3) und damit ihren Austausch im Internet ermöglicht [RDF04e]. Das Datenmodell wird durch die Beschreibungssprache *RDF Schema* ergänzt, die es erlaubt, Bestandteile von RDF-Strukturen durch Zusatzinformationen näher zu beschreiben [RDF04d].

Eines der Hauptziele bei der Entwicklung von RDF war die Interoperabilität von RDF-Ressourcebeschreibungen im Sinne ihrer Austauschbarkeit zwischen verschiedenen Akteuren. Aus diesem Grunde sind einige der Konzepte in RDF für die Verwendung von Metadaten in gemeinsamen Informationsräumen besonders relevant. Diese Konzepte werden im folgenden aufgeführt und erläutert:

RDF–1: Explizite Zuordnung von Records zu Ressourcen

Datenmodelle wie das relationale oder das objektorientierte Datenmodell repräsentieren Daten für beliebige Anwendungen. Die Beschreibung eines Objekts der realen Welt ist dabei zugleich auch der digitale Repräsentant des Objekts, ohne dass ein expliziter Bezug zu diesem Objekt besteht (vgl. Abschnitt 3.1, Punkt RDM–2). RDF hingegen ist ein Modell, das speziell die Beschreibung von Ressourcen durch Metadaten unterstützt. In einer RDF-Beschreibung (Description), die ihrer Struktur nach wiederum als Record angesehen werden kann, kann daher die durch sie beschriebene Ressource explizit referenziert werden. So wird die folgende, aus einem einzelnen Attribut-Wert-Paar bestehende RDF-Beschreibung der Res-

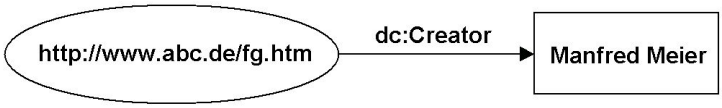

Abbildung 3.2: Graphische Darstellung einer RDF-Aussage

source http://www.abc.de/fg.htm zugeordnet[2]:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/metadata/dublin_core#">

<rdf:Description about = "http://www.abc.de/fg.htm">
  <dc:Creator> Manfred Meier </dc:Creator>
</rdf:Description>

</rdf:RDF>
```

Die einzelnen Attribut-Wert-Paare werden dabei als *Aussagen* (*Statements*) aufgefasst, die aus einem Subjekt in Form der beschriebenen Ressource, einem Prädikat in Form des Attributs und einem Objekt in Form des Attributwerts bestehen. Diese Aussagen lassen sich auch in ein graphisches Modell umsetzen, in dem Subjekte und Objekte durch Knoten sowie Prädikate durch Kanten zwischen diesen Knoten repräsentiert werden. Abbildung 3.2 zeigt die graphische Darstellung der RDF-Aussage, die im vorangegangenen Beispiel getroffen wurde.

Metadaten werden in Informationsräumen häufig gemeinsam mit der Ressource, die sie beschreiben, bearbeitet (vgl. Abschnitt 2.3.1), so dass es erforderlich ist, direkt von den Metadaten auf die beschriebene Ressource schließen zu können, damit ein Akteur jederzeit von der Bearbeitung der Metadaten in die Bearbeitung der Ressource selbst wechseln kann. Die explizite Zuordnung von Records zu der entsprechenden Ressource ist daher ein grundlegender Bestandteil der Datenstrukturen innerhalb eines gemeinsamen Informationsraums.

RDF–2: Multiple Beschreibungen für eine Ressource

In Modellen, in denen reale Ressourcen durch Records repräsentiert werden (vgl. Abschnitt 3.1, Punkt RDM–2), wird jeder Ressource genau ein Record zugeordnet. Da RDF eine Trennung zwischen

[2]In diesem Beispiel sind die XML-Namespaces explizit definiert. Sofern in den folgenden Beispielen die Festlegung der Namensräume fehlt, gilt die Annahme, dass das Präfix rdf den RDF-Namensraum „http://www.w3.org/1999/02/22-rdf-syntax-ns#“ repräsentiert, während das Präfix rdfs für den Namensraum „http://www.w3.org/2000/01/rdf-schema#“ von RDF Schema steht.

der beschriebenen Ressource und deren Beschreibung vollzieht (vgl. Punkt RDF–1), eröffnet sich hier zugleich die Möglichkeit, mehrere RDF-Beschreibungen für ein und dieselbe Ressource zu erfassen. In RDF ist daher die Anzahl der RDF-Beschreibungen für eine Ressource nicht begrenzt.

Die Möglichkeit, multiple Beschreibungen für eine Ressource erfassen zu können, entspricht dem Vorgehen bei der kooperativen Arbeit in gemeinsamen Informationsräumen (vgl. Abschnitt 2.3.1, Vorgehensweise–2), das den Akteuren die an ihren Arbeitskontext angepasste Erfassung von Metadaten ermöglicht und den Austausch sich ergänzender Metadaten zwischen verschiedenen Kooperationspartnern unterstützt.

RDF–3: Anbindung von Meta-Metadaten durch Reifikation

Meta-Metadaten werden in RDF in Form von Aussagen über andere Aussagen erfasst. Damit eine Aussage das Subjekt einer anderen Aussage werden kann, muss sie zunächst selbst in eine Ressource umgewandelt werden. Dieser Vorgang wird auch als *Reifikation* bezeichnet. Hierbei werden in einer RDF-Beschreibung zunächst Subjekt, Prädikat und Objekt der ursprünglichen Aussage als Attribut-Wert-Paare zusammengefasst und mit dem Typ Statement versehen. Die dazugehörigen Metaaussagen werden dann in derselben RDF-Beschreibung mit erfasst. Das folgende Beispiel enthält somit die Information, dass die Aussage, dass Manfred Meier der Autor von fg.htm ist, von Paula Paulsen erfasst wurde:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:abc="http://www.abc.de/schema">

<rdf:Description>
  <rdf:subject resource="http://www.abc.de/fg.htm"/>
  <rdf:predicate resource="http://www.abc.de/schema/Autor"/>
  <rdf:object>Manfred Meier</rdfs:object>
  <rdf:type resource=
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"/>
  <abc:Erfasser>Paula Paulsen</abc:Erfasser>
</rdf:Description>

</rdf:RDF>
```

Die Reifikation einer Aussage entspricht nicht der Aussage selbst. Die Interpretation des obigen Beispiels ergibt also nicht automatisch, dass Manfred Meier der Autor von fg.htm ist, wohl aber, dass Paula Paulsen eine entsprechende Aussage getroffen hat.

Durch die Reifikation von Aussagen wird in RDF einerseits erreicht, dass für die Beschreibung von Aussagen dieselben RDF-Konzepte

zur Verfügung stehen wie für alle anderen Ressourcen auch. Anders als in XML (vgl. Abschnitt 3.3, Punkt XML–6) wird hier somit für Meta-Metadaten die gleiche Menge von Vorgehensweisen unterstützt wie für Metadaten. Die Unterscheidung zwischen einer RDF-Aussage und ihrer Reifikation erscheint andererseits als künstliche Barriere bei der Erfassung von Metadaten, da sie eine doppelte Angabe derselben Informationen notwendig macht und dadurch einen zusätzlichen Arbeitsaufwand mit sich bringt. Der Ansatz, Metadaten mit Identifikatoren zu versehen und dann durch Meta-Metadaten zu beschreiben, entschärft dieses Problem.

Anders als in XML (vgl. Abschnitt 3.3, Punkt XML–6) existiert in RDF kein Konzept zur logischen Trennung verschiedener Ebenen von Metadaten und Meta-Metadaten. Hierdurch kommt es zu einer Vermengung der durch die verschiedenen Metaebenen repräsentierten Arbeitskontexte, die die Interpretation sowohl für menschliche als auch für Software-Akteure erschwert (vgl. Abschnitt 2.3.2). Insbesondere können in RDF Metadaten bzw. Meta-Metadaten erfasst werden, die einander zyklisch referenzieren und denen sich keine klare Semantik zuordnen lässt, wie das folgende Beispiel zeigt:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:abc="http://www.abc.de/schema">


<rdf:Description ID="123">
  <rdf:type resource=
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"/>
  <rdf:subject resource="#456"/>
  <rdf:predicate resource="http://www.abc.de/schema/Erfasser"/>
  <rdf:object>Manfred Meier</rdfs:object>
</rdf:Description>

<rdf:Description ID="456">
  <rdf:type resource=
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"/>
  <rdf:subject resource="#123"/>
  <rdf:predicate resource="http://www.abc.de/schema/Erfasser"/>
  <rdf:object>Paula Paulsen</rdfs:object>
</rdf:Description>
</rdf:RDF>
```

Um nicht interpretierbare Zyklen auszuschließen und verschiedene Arbeitskontexte voneinander unterscheiden zu können, ist für die Nutzung von Metadaten in gemeinsamen Informationsräumen ein Konzept zur logischen Trennung verschiedener Ebenen von Metadaten und Meta-Metadaten erforderlich.

RDF–4: Klassifikation als ordnendes Konzept

RDF Schema [RDF04c] bietet einen Rahmen für die semantische Einordnung von RDF-Metadaten in ihren Anwendungskontext. Ein grundlegendes Konzept hierbei ist die Definition von *Klassen*, die semantische Konzepte der Anwendungsdomäne repräsentieren und denen RDF Beschreibungen und die dazugehörigen Ressourcen zugeordnet werden.

Auch Klassen werden in RDF als Ressourcen betrachtet, so dass ihre Definition durch RDF-Beschreibungen erfolgt, die durch die type-Eigenschaft der vordefinierten RDF-Schema-Klasse Class zugeordnet sind:

```
<rdf:Description ID="Publikation">
  <rdf:type resource=
    "http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

Eine Ressource kann dann wiederum unter Verwendung der type-Eigenschaft einer oder mehreren beliebigen Klassen zugeordnet werden:

```
<rdf:Description about = "http://www.abc.de/fg.htm">
  <rdf:type resource="#Publikation"/>
  <dc:Creator> Manfred Meier </dc:Creator>
</rdf:Description>
```

Das Beispiel zeigt bereits, dass RDF-Klassen keinerlei Struktur für die ihnen zugeordneten RDF-Beschreibungen festlegen. Für die Ressource fg.htm können hier unabhängig von der Klasse beliebige Attribut-Wert-Paare erfasst werden. Während Record-Typen (vgl. Abschnitt 3.1, RDM–3) den Aufbau ihrer Instanzen beschreiben, kommt RDF-Klassen somit eine andere Funktion zu [Nor94]. Ihnen können RDF-Beschreibungen ungeachtet ihrer Struktur zugeordnet und die durch sie beschriebenen Ressourcen (vgl. Punkt RDF–1) semantisch gruppiert werden. Die Kriterien zur Bildung von Klassen leiten sich allein aus der jeweiligen Anwendungsdomäne her und sind die Basis anwendungsspezifischer Klassifikationssysteme, die beliebig gestaltet sein und formal, beispielsweise mit Hilfe von Beschreibungslogiken (vgl. Abschnitt 3.6), oder frei, beispielsweise spontan und dynamisch durch Akteure oder Gruppen von Akteuren, konstruiert werden können. Durch die Möglichkeit, Klassen zu definieren, stellt RDF Schema eine Schnittstelle zu diesen anwendungsspezifischen Klassifikationssystemen her.

Die Einordnung in die Klassen einer Anwendungsdomäne kann Aufschluss über die Qualität und die Verwendungsmöglichkeiten von Ressourcen im gemeinsamen Informationsraum geben. Die Beschreibung von Klassen, wie sie durch RDF Schema realisiert wird, stellt

damit eine wichtige Zusatzinformation bei der kooperativen Verwendung von Metadaten dar (vgl. Abschnitt 2.3.1, Vorgehensweise–4, sowie Abschnitt 2.3.2).

RDF–5: Beziehungen zwischen Klassen innerhalb von Klassifikationssystemen

RDF Schema bietet die Möglichkeit, Klassen in Spezialisierungshierarchien anzuordnen, indem über die subClassOf-Eigenschaft für jede Klasse beliebig viele Superklassen angegeben werden können. Im folgenden Beispiel wird der Klasse Artikel die Klasse Publikation als Superklasse zugewiesen:

```
<rdf:Description ID="Artikel">
  <rdf:type resource=
    "http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Publikation"/>
</rdf:Description>
```

Der Aufbau zyklischer Subklassenbeziehungen ist nicht erlaubt. Instanzen einer Klasse gelten automatisch auch als Instanzen aller direkten und transitiven Superklassen dieser Klasse und können in RDF-Eigenschaften mit entsprechend eingeschränkten Definitions- oder Wertebereichen (vgl. Punkt RDF–6) verwendet werden.

Durch die Spezialisierungsbeziehung werden verschiedene Klassen der entsprechenden Anwendungsdomäne (vgl. Punkt RDF–4) miteinander verbunden, so dass ein Klassifikationssystem entsteht, das sowohl den Einstieg in als auch die Navigation durch die Klassen der Anwendungsdomäne ermöglicht (vgl. Abschnitt 2.3.2). Zugleich stellt die Subklassenbildung ein Konzept dar, auf das beim Aufbau semantischer Klassifikationssysteme mit Hilfe von Beschreibungslogiken (vgl. Abschnitt 3.6) häufig zurückgegriffen wird, so dass hier ein weiterer Teil der Schnittstelle zwischen RDF, RDF Schema und darauf aufsetzenden semantischen Beschreibungen von Anwendungsdomänen zu Tage tritt.

Neben der Spezialisierungsbeziehung sind in einem Klassifikationssystem beliebige weitere Beziehungen zwischen den Klassen vorstellbar, wie Querverweise, Beziehungen, die Alternativen oder Gegensätze ausdrücken, Teil-Ganzes-Beziehungen, usw. [Pep00]. Solche Beziehungen zwischen Klassen lassen sich als neue RDF-Eigenschaften ebenso definieren wie die subClassOf-Eigenschaft in RDF Schema und können dann zur Beschreibung von Klassifikationssystemen verwendet werden. Das folgende Beispiel definiert eine Teil-Ganzes-Beziehung in der RDF-Eigenschaft istTeilVon und verbindet mit Hilfe dieser Eigenschaft die Klassen Kapitel und Buch:

```
<rdf:Description ID="istTeilVon">
  <rdf:type resource=
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource=
    "http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:range rdf:resource=
    "http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description ID="Kapitel">
  <rdf:type rdf:resource=
    "http://www.w3.org/2000/01/rdf-schema#Class"/>
  <istTeilVon rdf:resource="#Buch"/>
</rdf:Description>
```

Während die Bedeutung der in RDF Schema definierten subClassOf-Eigenschaft und die damit verbundenen Konsequenzen, wie die Zugehörigkeit von Instanzen zu den entsprechenden Klassen und Superklassen, bekannt und daher in gängigen RDF-Werkzeugen implementiert sind, müssen für selbst definierte Beziehungen zwischen RDF-Klassen eigene Interpretationsroutinen programmiert werden, die die Information entsprechend umsetzen. Zu diesem Zweck muss die formale Semantik der Beziehungen sowohl dem Metadatenerfasser als auch dem interpretierenden Akteur bekannt sein, da sie nicht über die RDF-Beschreibung einer Beziehung übermittelt wird. Die Möglichkeit einer Beschreibung der formalen Semantik für Klassifikationssysteme direkt innerhalb des gemeinsamen Informationsraums, die über die Angabe von Spezialisierungsbeziehungen hinaus führt, ist eine Anforderung, die von RDF Schema allein nicht erfüllt wird.

Insgesamt unterstützt die Verbindung von verschiedenen Klassen über Beziehungen beliebiger Art den Aufbau von Klassifikationssystemen und stellt einen weiteren Schritt zur semantischen Beschreibung von Anwendungsdomänen dar (vgl. Abschnitt 2.3.2). Die Erfassung der Beziehungen zwischen Klassen liefert somit eine Zusatzinformation (vgl. Abschnitt 2.3.1, Vorgehensweise–4), die in gemeinsamen Informationsräumen zur Interpretation von Metadaten und im Fall von Spezialisierungsbeziehungen darüber hinaus auch zur Ableitung neuer, bisher nur implizit vorhandener Metadaten, wie der Zugehörigkeit einer Ressource zu einer Klasse, verwendet werden kann (vgl. Punkt DL–2).

RDF–6: Semantische Beschreibung von Attributen

RDF-Eigenschaften entsprechen vom Konzept her Attributen in Metadaten-Records. Attribute sind in RDF jedoch nicht nur ein Mit-

tel zur Strukturierung von Records, sondern können darüber hinaus selbst mit Hilfe von Zusatzinformationen in Form von RDF-Schema-Konstrukten beschrieben werden. Solche Zusatzinformationen übermitteln die Semantik eines Attributs und geben somit zusätzlichen Aufschluss über Ressourcen, in deren Beschreibungen dieses Attribut zum Einsatz kommt (vgl. Punkt RDM–1).

RDF-Eigenschaften können in RDF Schema auf zwei verschiedene Weisen semantisch beschrieben werden. So lässt sich einerseits die Bedeutung von RDF-Eigenschaften über die Einschränkung ihrer Definitions- und Wertebereiche darstellen. Andererseits kann eine Spezialisierungshierarchie zwischen RDF-Eigenschaften aufgebaut werden.

Sowohl der Definitions- als auch der Wertebereich einer RDF-Eigenschaft können auf bestimmte RDF-Klassen eingeschränkt werden. Hierdurch wird der Bedeutungszusammenhang, in dem die Eigenschaft verwendet wird, genauer beschrieben. Zur Festlegung des Definitionsbereichs einer RDF-Eigenschaft wird die domain-Eigenschaft verwendet, der Wertebereich ist durch die range-Eigenschaft beschrieben. Beide Konstrukte sind in RDF Schema definiert.

Das folgende Beispiel legt für die RDF-Eigenschaft Elternteil fest, dass sie nur in RDF-Beschreibungen der Klasse Person erscheint und dort auch nur mit Ressourcen der Klasse Person belegt werden kann:

```
<rdf:Description ID="Elternteil">
  <rdf:type resource=
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain resource="#Person"/>
  <rdfs:range resource="#Person"/>
</rdf:Description>
```

Eine RDF-Eigenschaft kann mehrere domain- und mehrere range-Eigenschaften besitzen. In einem solchen Fall wird als Definitions- bzw. als Wertebereich die Schnittmenge der jeweils aufgeführten Klassen angenommen.

Die range-Eigenschaft wird in RDF nicht nur zur semantischen Beschreibung von RDF-Eigenschaften verwendet, sondern wie im folgenden Beispiel auch zu deren Typisierung mit einfachen Typen:

```
<rdf:Description ID="Geburtsdatum">
  <rdf:type resource=
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain resource="#Person"/>
  <rdfs:range resource="&xsd;date"/>
</rdf:Description>
```

Durch diese Möglichkeit werden die Konzepte des *Typs*, der die syntaktische Form eines Werts beschreibt (vgl. Punkt XML–2), und der *Klasse*, die eine Ressource einem Konzept der Anwendungsdomäne zuordnet (vgl. Punkt RDF–4), vermischt. Da sich Klassifikation und Typisierung auch hinsichtlich der Interpretation der jeweiligen Zusatzinformationen unterscheiden, fehlt hier die notwendige Trennung beider Konzepte, die die Unterscheidung zwischen ihnen bereits bei der Metadatenerfassung und nicht erst bei der Interpretation ermöglicht.

Neben der Festlegung von Definitions- und Wertebereichen stellt die Erfassung von Spezialisierungsbeziehungen eine weitere Möglichkeit zur semantischen Beschreibung von RDF-Eigenschaften dar. Eine Spezialisierungsbeziehung zwischen zwei RDF-Eigenschaften wird mit Hilfe der subPropertyOf-Eigenschaft zum Ausdruck gebracht. So besagt das folgende Beispiel, dass die Mutter-Eigenschaft eine speziellere Form der Elternteil-Eigenschaft ist:

```
<rdf:Description ID="Mutter">
  <rdf:type resource=
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:subPropertyOf rdf:resource="#Elternteil"/>
</rdf:Description>
```

Die Festlegung der Klassen, deren Instanzen durch ein Attribut verbunden werden können, sowie der Aufbau von Attributhierarchien verdeutlichen den Bedeutungszusammenhang, in dem Attribute verwendet werden, und stellen somit Zusatzinformationen (vgl. Abschnitt 2.3.1, Vorgehensweise–4) dar, durch die Metadaten semantisch ergänzt werden können. Diese Zusatzinformationen unterstützen die Interpretation von Metadaten sowohl durch menschliche als auch durch Software-Akteure.

Als Datenmodell, das speziell für die Erfassung von interoperablen Metadaten entwickelt wurde, stellt RDF den expliziten Bezug zwischen der Beschreibung und der beschriebenen Ressource her. Es wird damit der zentralen Rolle von Ressourcen in gemeinsamen Informationsräumen gerecht und ermöglicht die Beschreibung einer einzigen Ressource durch multiple Metadaten.

Durch Reifikation können in RDF einzelne Attribut-Wert-Paare durch Meta-Metadaten beschrieben werden. Da die beschriebenen Attribut-Wert-Paare wiederum als Ressourcen angesehen werden, stehen für die Erfassung von Meta-Metadaten dieselben Beschreibungsmittel wie für die Erfassung von Metadaten zur Verfügung. Eine Trennung zwischen verschiedenen Metaebenen ist jedoch nicht vorgesehen, was zu einer Vermischung verschiedener Arbeitskontexte und unzulässigen zyklischen Beschreibungen führen kann.

Durch RDF-Klassen können Ressourcen nach beliebigen Aspekten der Anwendungsdomäne gruppiert werden, wodurch eine Verallgemeinerung des Konzepts der Begriffe in Retrieval-Indexen (vgl. Abschnitt 3.4) erreicht wird. Durch

den Aufbau von Spezialisierungsbeziehungen zwischen Klassen entstehen Klassifikationssysteme, durch die Akteure zu gesuchten Ressourcen navigieren können. Neben der Spezialisierung sind weitere Beziehungen zwischen Klassen denkbar, die in RDF und RDF Schema nicht vorab definiert sind, aber mit den vorhandenen Konzepten realisiert werden können.

Die Festlegung von Klassen als Definitions- und Wertebereiche von Attributen sowie der Aufbau von Attributhierarchien stellt Attribute und damit die durch sie beschriebenen Ressourcen in den Bedeutungszusammenhang der jeweiligen Anwendungsdomäne, so dass beide Konzepte für die Erfassung semantischer Zusatzinformationen in gemeinsamen Informationsräumen geeignet sind.

3.6 Beschreibungslogiken

Beschreibungslogiken (Description Logics) stellen eine Methode der *Wissensrepräsentation* [Sow00] dar. Sie entstanden aus dem Bedarf heraus, netzwerkbasierte und Frame-basierte Modelle zur Wissensrepräsentation [Qui67, Min81] mit einer formalen Semantik auszustatten. Hierdurch können Wissensrepräsentationen eindeutig interpretiert werden und darüber hinaus als explizite Grundlage dienen, um weiteres implizit vorhandenes Wissen mit Hilfe von Inferenzalgorithmen abzuleiten.

Die *Wissensbasis*, in der die Anwendungsdomäne mit Hilfe einer Beschreibungslogik dargestellt ist, besteht aus einer *TBox* und einer *ABox*. Die TBox beinhaltet eine *Terminologie* in Form sowohl von Klassen, die bestimmte Individuen der Anwendungswelt gruppieren, als auch von Rollen, die binäre Beziehungen zwischen Individuen darstellen. Die ABox umfasst eine Reihe von *Zusicherungen* (*Assertions*), die für einzelne benannte Individuen bezüglich der TBox getroffen werden.

Als Wurzelsprache der Beschreibungslogiken gilt die im System KL-ONE [BS85] realisierte Beschreibungslogik, die in den späten 1970er Jahren entstand. In den 1980er Jahren war die Feststellung, dass die Komplexität der Inferenzalgorithmen mit der Ausdrucksmächtigkeit einer Beschreibungslogik zunimmt [BL84], der Anlass für eine Reihe von Untersuchungen, die mit Hilfe der Komplexitätstheorie Einordnungen für einzelne Sprachkonstrukte der Beschreibungslogiken trafen und die die Grundlage für verschiedene Klassen von Beschreibungslogiken mit unterschiedlicher Komplexität bildeten.

Für die angestrebte Erweiterung des WorldWideWeb zu einem Semantic Web, über das nicht nur Personen, sondern auch Software-Agenten Informationen austauschen, die sie korrekt interpretieren können, stellen Beschreibungslogiken ein potentielles Werkzeug dar [Fen00]. Mittlerweile haben sich daher auch XML-basierte Beschreibungssprachen etabliert, wie DAML+OIL [Hor02], das als eine der Grundlagen für die Entwicklung einer Ontologiebeschreibungssprache durch das WorldWideWeb Consortium [BvHH$^+$04] gilt.

Beschreibungslogiken ergänzen die bisher für die Verwendung in gemeinsamen Informationsräumen untersuchten Konzepte insbesondere hinsichtlich der Bereitstellung von formalen Grundlagen für die Definition von Klassen der An-

wendungsdomäne und ihren semantischen Zusammenhängen sowie hinsichtlich des Umgangs mit unvollständigen Informationen:

DL–1: Formale Definition von Klassen, Attributen und Beziehungen

Beschreibungslogiken repräsentieren die Semantik einer Anwendungsdomäne, indem sie Klassen von in der Domäne befindlichen Individuen sowie die Beziehungen zwischen diesen Individuen formal beschreiben. Die Konzepte zur Beschreibung von Beziehungen lassen sich dabei sowohl auf Beziehungen, die Individuen gleichberechtigt miteinander verbinden (vgl. Punkt RDM–6), als auch auf Beziehungen, die über die Belegung eines Attributs zwischen zwei Ressourcen hergestellt werden (vgl. Punkt RDM–1 und OODM–2), anwenden.

Zur formalen Beschreibung von Klassen stellen Beschreibungslogiken eine Reihe von *Konstruktoren* bereit, die neue Klassen auf der Grundlage bereits existierender Klassen und Beziehungsarten definieren und in der TBox zum Einsatz kommen. Im Folgenden werden verschiedene Konstruktoren für Klassen vorgestellt. Dabei sei angemerkt, dass nicht jeder dieser Konstruktoren in jeder Beschreibungslogik enthalten ist, da die Kombination der Konstruktoren entscheidenden Einfluss auf die Komplexität der Inferenzalgorihtmen für die jeweilige Beschreibungslogik (vgl. DL–2) hat. Die Semantik eines Konstruktors wird formal mit Hilfe von *Interpretationen* definiert. Eine Interpretation besteht aus einer nicht-leeren Menge von Individuen $\Delta^{\mathcal{I}}$ und einer Interpretationsfunktion, die jeder atomaren Klasse A eine Menge $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ zuordnet und jeder Beziehungsart R eine binäre Relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Zunächst können mit Hilfe von Konstruktoren die Mengen der Individuen von existierenden Klassen mengentheoretisch zu neuen Klassen kombiniert werden. Dabei entspricht die *Konjunktion* $C \sqcap D$ zweier Klassen der Schnittmengenbildung, die *Disjunktion* $C \sqcup D$ der Vereinigung und die *Negation* $\neg C$ der Komplementbildung hinsichtlich der Interpretationsdomäne $\Delta^{\mathcal{I}}$:

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$$

Darüber hinaus lassen sich Klassen definieren, indem formal festgelegt wird, welche Arten von Beziehungen ihre Individuen eingehen können. Über eine *Existenzrestriktion* $\exists R.C$ wird vorgegeben, dass für jedes Individuum der definierten Klasse ein Individuum in der Klasse C existiert, mit dem es in einer Beziehung R steht:

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.\langle a, b\rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$$

Eine *Wertrestriktion* $\forall R.C$ legt fest, dass alle Individuen, mit denen ein Individuum aus der definierten Klasse in der Beziehung R steht, stets in der Klasse C enthalten sind:

$$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b.\langle a,b\rangle \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\}$$

Konjunktion, Disjunktion, Negation sowie Existenz- und Wertrestriktion bilden zusammen die Menge der Konstruktoren der Beschreibungslogik $\mathcal{ALC}$ *(Attributive Concept Description Language with Complements)* [SSS91]. $\mathcal{ALC}$ ist die Grundlage für eine Reihe weiterer Beschreibungslogiken, die durch die Hinzunahme bestimmter Konstruktoren oder Axiome entstehen [BN03]. So lässt sich $\mathcal{ALC}$ mit Hilfe von *Anzahlrestriktionen* zu $\mathcal{ALCN}$ ergänzen. Anzahlrestriktionen $(\geq nR)$ bzw. $(\leq nR)$ sagen aus, dass die Individuen der definierten Klasse mit mindestens bzw. höchstens n Individuen in der Beziehung R stehen:

$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b.\langle a,b\rangle \in R^{\mathcal{I}}\}| \geq n\}$$
$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b.\langle a,b\rangle \in R^{\mathcal{I}}\}| \leq n\}$$

Anzahlrestriktionen lassen sich erweitern zu *qualifizierten Anzahlrestriktionen* $(\geq nR.C)$ bzw. $(\leq nR.C)$, durch die formuliert werden kann, dass die Individuen der definierten Klasse mit mindestens bzw. höchstens n Individuen der Klasse C in der Beziehung R stehen:

$$(\geq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b.\langle a,b\rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}| \geq n\}$$
$$(\leq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b.\langle a,b\rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}| \leq n\}$$

Die Konstruktoren einer Beschreibungslogik lassen sich untereinander kombinieren, so dass die Semantik einer Klasse genau beschrieben werden kann. Das folgende Beispiel etwa zeigt die Definition der Klasse MutterOhneTochter, in der Konjunktion, Negation und Wertrestriktion verbunden sind:

$$\mathsf{MutterOhneTochter} = \mathsf{Mutter} \sqcap \forall \mathsf{hat\text{-}Kind}.\neg\mathsf{Frau}$$

Die universelle Klasse, die alle Individuen der Interpretationsdomäne enthält, und die leere Klasse, die kein Individuum enthält, werden häufig bei der Beschreibung von Klassen verwendet, so dass für sie die Abkürzungen $\top$ und $\bot$ existieren, die sich mit den vorgestellten Konstruktoren wie folgt definieren lassen:

$$\top = C \sqcup \neg C$$
$$\bot = C \sqcap \neg C$$

Neben Klassen können in einer TBox auch neue Beziehungsarten definiert werden, indem existierende Beziehungsarten invertiert werden oder indem ihre transitive Hülle gebildet wird:

$$(R^{-})^{\mathcal{I}} = \{\langle a,b\rangle \mid \langle b,a\rangle \in R^{\mathcal{I}}\}$$
$$(R^{*})^{\mathcal{I}} = (R^{\mathcal{I}})^{*}$$

In der TBox einer Wissensbasis werden Klassen und Beziehungsarten nicht nur definiert, sondern auch miteinander in Zusammenhang gebracht, so dass über diese Zusammenhänge weitere semantische Aspekte der Anwendungsdomäne beschrieben werden. Zusammenhänge zwischen Klassen und Zusammenhänge zwischen Beziehungsarten werden mit Hilfe von Axiomen dargestellt, in denen die

Subsumptionsbeziehung $\sqsubseteq$ zwischen Klassen und zwischen Beziehungsarten zum Ausdruck gebracht werden kann:

$C_1 \sqsubseteq C_2 \quad gdw. \quad {C_1}^{\mathcal{I}} \subseteq {C_2}^{\mathcal{I}}$

$R_1 \sqsubseteq R_2 \quad gdw. \quad {R_1}^{\mathcal{I}} \subseteq {R_2}^{\mathcal{I}}$

Mit Hilfe von Axiomen in Verbindung mit Konstruktoren lassen sich insbesondere auch Gegebenheiten darstellen, die bei der Modellierung von Anwendungsdomänen häufig auftreten, wie die Disjunktheit von Klassen, Überdeckungen oder auch die Einschränkung des Definitions- und Wertebereichs von Beziehungsarten, so dass hier die Ausdrucksmöglichkeiten spezieller Konstrukte, wie der domain- und range-Eigenschaften in RDF Schema (vgl. Punkt RDF–6) in einem allgemeineren Satz von Konstrukten mit beinhaltet sind.

Während die TBox die abstrakten Konzepte der Anwendungsdomäne formal definiert, benennt die ABox konkrete Instanzen, ordnet diese den Klassen der TBox zu und verknüpft sie in Beziehungen. Im Hinblick auf das Metadaten-Management in gemeinsamen Informationsräumen lässt sich die Schnittstelle zwischen ABox und TBox abbilden auf die Schnittstelle zwischen konkreten Ressourcen und ihren Metadaten auf der einen und Zusatzinformationen hinsichtlich vorhandener Klassifikationssysteme auf der anderen Seite (vgl. Abschnitt 2.3.2).

Die Konzepte, die in Beschreibungslogiken zur formalen Beschreibung von Klassen verwendet werden, weisen an einigen Stellen Ähnlichkeiten zu Konzepten der Typisierung auf. So erinnert die Formulierung von Existenz- und Wertrestriktionen beispielsweise an die Zusammenfassung von bestimmten Attributen in Record-Typen. Trotz dieser Ähnlichkeiten handelt es sich bei der formalen Beschreibung der *Semantik einer Anwendungsdomäne* durch Beschreibungslogiken um einen anderen Vorgang als bei der Beschreibung des *Aufbaus von Werten* in Typsystemen. Dieses wird insbesondere dadurch deutlich, dass Beschreibungslogiken die *Open World Assumption* zugrunde liegt, die besagt, dass über Annahmen, die anhand der Wissensbasis nicht beweisbar sind, keine endgültige Aussage getroffen werden kann. Typsysteme wie das des relationalen oder des objektorientierten Datenmodells (vgl.Abschnitte 3.1 und 3.2) hingegen basieren auf der *Closed World Assumption*, in der vorausgesetzt wird, dass alles, was sich anhand der gegebenen Informationen nicht beweisen lässt, falsch ist. So lässt sich in einer TBox beispielsweise formulieren, dass alle Instanzen der Klasse Elternteil Personen sind, die mindestens ein Kind haben, das wiederum eine Person ist:

$\mathsf{Elternteil} = \mathsf{Person} \sqcap \exists \mathsf{hatKind}.\mathsf{Person}$

In der ABox kann nun eine Instanz als Elternteil klassifiziert werden, ohne dass ihr konkret eine Person über die hatKind-Beziehung zugeordnet wird. Hier wird davon ausgegangen, dass ein Kind für die Instanz existieren kann, auch wenn es in der Wissensbasis nicht

beschrieben ist. Formuliert man hingegen einen Record-Typ Eltern-Typ, für den man fordert, dass seine Werte ein Attribut-Wert-Paar für das Attribut Kind enthalten, so muss in allen Records dieses Typs das Attribut entsprechend erfasst sein, da sich interpretierende Akteure und Anwendungen auf seine Existenz verlassen und die Interpretation nur so fehlerfrei ablaufen kann.

Die bei der Typisierung erforderlichen Einschränkungen lassen sich mit einfachen Beschreibungslogiken somit nicht formulieren. Autoepistemische Beschreibungslogiken [DNR97], deren formale Semantik nicht wie oben beschrieben auf der Grundlage *einer* möglichen Interpretation, sondern stattdessen auf der Grundlage *aller* möglichen Interpretationen definiert ist, bieten das Potential, um entsprechende Beschreibungen zu erstellen.

Für die kooperative Nutzung von Metadaten in gemeinsamen Informationsräumen sollten Typisierung und Klassifikation grundsätzlich als zwei unabhängige Konzepte betrachtet werden, so dass einerseits Widersprüche bei der Interpretation von Typen und Klassen vermieden werden und andererseits den unterschiedlichen Betrachtungsebenen beider Konzepte Rechnung getragen wird.

DL–2: Inferenz von Klassifikationsinformationen

Beschreibungslogiken basieren auf der Annahme, dass nicht alle Informationen über die dargestellte Anwendungsdomäne vollständig in der Wissensbasis abgelegt sein müssen. Da die Sprachkonstrukte der Beschreibungslogiken jedoch auf einer formalen Semantik beruhen, ist es möglich, durch logisches Schließen aus explizit vorhandenen Informationen weitere implizite Informationen abzuleiten. Hierfür stellen Systeme, die Beschreibungslogiken realisieren [HM01], eine Reihe von Inferenzalgorithmen zur Verfügung.

Einer dieser Inferenzdienste ist mit dem Test einer Klasse C_0 auf ihre *Konsistenz* gegeben. Hierbei wird überprüft, ob es Individuen geben kann, die zu C_0 gehören. Ein zur Konsistenzprüfung eingesetztes Verfahren ist der *Tableau-Algorithmus* [SSS91], der einen Baum konstruiert, dessen Knoten mögliche Individuen der Anwendungsdomäne darstellen. Als Wurzel des Baums wird eine Instanz der zu überprüfenden Klasse C_0 angenommen. Ausgehend von der Wurzel wird der Baum schrittweise aufgebaut, indem für mögliche Beziehungen eines Individuums Kanten zu neuen Knoten, die die Zielindividuen der Beziehungen repräsentieren, angefügt werden. Jedem Knoten des Baums werden dabei in jedem Schritt Mengen von Klassen zugewiesen, in denen sich das jeweilige Individuum gemäß der TBox befinden muss. Der Algorithmus terminiert, wenn der Baum sich nicht mehr verändert oder wenn Widersprüche in der Klassenmenge eines Knotens auftreten, wie etwa $\{C, \neg C\}$. Im Falle eines solchen Widerspruchs ist bewiesen, dass C_0 inkonsistent ist, also keine Instanzen besitzen kann.

Der Konsistenztest lässt sich ausweiten zum *Kohärenztest*, der alle inkonsistenten Klassen einer TBox auffindet. Der Kohärenztest wird häufig eingesetzt, um Modellierungsfehler ausfindig zu machen, da die Beschreibung von Klassen, die leer sein müssen, zumeist nicht in der Absicht des Modellierers liegt.

Der *Subsumptionstest* ist ein weiterer Inferenzdienst und überprüft, ob eine Klasse C von einer Klasse D subsumiert wird, ob also gilt $C \sqsubseteq D$ (vgl. DL–1). Er wird häufig eingesetzt, um *Taxonomien* aufzustellen, die jede Klasse mit ihren speziellsten Super- und mit ihren allgemeinsten Subklassen verbinden und somit zu einer weiteren Strukturierung vorhandenen Domänenwissens beitragen.

Als letzter Inferenzdienst sei hier noch der *Instanztest* aufgeführt, der für ein bestimmtes Individuum a überprüft, ob es eine Instanz der Klasse C darstellt. Der Instanztest kann bei der kooperativen Arbeit auf der Grundlage von Metadaten dazu verwendet werden, nicht erfasste Klassen von Ressourcen zu ermitteln, um diese bei der Interpretation zur Verfügung zu stellen.

Inferenzdienste lassen sich nicht unabhängig voneinander betrachten. Häufig existiert ein logischer Zusammenhang zwischen ihnen, und sie lassen sich teilweise aufeinander zurückführen. Die Komplexität eines Inferenzproblems ist von der Menge der Konstruktoren und Axiome einer Beschreibungslogik abhängig. Schon die Hinzunahme eines einzelnen Operators kann ein Inferenzproblem in eine andere Komplexitätsklasse verschieben [Don03]. Die Wahl einer konkreten Beschreibungslogik zur Modellierung einer Anwendungsdomäne hängt somit nicht nur davon ab, welche Zusammenhänge in der Wissensbasis repräsentiert werden sollen und welche Konstrukte hierfür nötig sind, sondern auch davon, welche Inferenzdienste für die Wissensbasis angeboten werden sollen und welche Komplexität sie für die gewählte Beschreibungslogik besitzen.

Für die kooperative Nutzung von Metadaten in gemeinsamen Informationsräumen kann demnach keine allgemein gültige Empfehlung für eine bestimmte Beschreibungslogik zur Beschreibung der Anwendungsdomäne gegeben werden. Die Struktur von Metadaten sollte jedoch geeignet sein, um diese mit Beschreibungslogiken zu verbinden. Die bereits vorgestellten Konzepte der Klasse (vgl. Punkt RDF–4) und des Attributs (vgl. Punkt RDM–1) als Beziehungsart eignen sich hier als Schnittstelle.

Beschreibungslogiken stellen eine Reihe weiterer Methoden zur Verfügung, um in Klassifikationssystemen Anwendungsdomänen semantisch zu beschreiben. So können zu einer Klasse gehörige Ressourcen anhand ihrer Beziehungen zu Ressourcen anderer Klassen formal beschrieben und existierende zu neuen Klassen kombiniert werden. Trotz struktureller Ähnlichkeiten unterscheidet sich das Konzept der Klassifikation dabei grundlegend vom Konzept der Typisierung.

Die Beschreibungslogiken zugrunde liegende Open-World-Assumption entlastet Metadatenerfasser von der Aufgabe, Metadaten und Zusatzinformationen vollständig erfassen zu müssen. Inferenzalgorithmen sind in der Lage, fehlende Informationen zu ergänzen, und daher insbesondere auch zur Unterstützung der Interpretation von Metadaten und Zusatzinformationen geeignet. Da die Komplexität von Inferenzalgorithmen von den in einer Beschreibungslogik enthaltenen Konstruktoren und Operatoren abhängig ist, kann für die Zusammenarbeit auf der Grundlage von Metadaten keine allgemein gültige Empfehlung für die Verwendung einer bestimmten Beschreibungslogik gegeben werden, da eine entsprechende Auswahl sowohl von der gewünschten Ausdrucksmächtigkeit der Konstrukte als auch von den vorgesehenen Inferenzen abhängig ist.

Insgesamt sollten kooperativ genutzte Metadaten sollten eine Struktur aufweisen, die die direkte Anbindung von beschreibungslogischen Ausdrücken ermöglicht.

3.7 Identifikation geeigneter Konzepte

In den vorangehenden Abschnitten wurden Datenmodelle beschrieben, die Konzepte für die Zusammenarbeit auf der Grundlage von Metadaten bereitstellen. Diese Modelle unterstützen dabei zumeist bestimmte Kooperationsszenarien, bestimmte Vorgehensweisen (vgl. Abschnitt 2.3.1) bei der Erstellung und Verwendung von Metadaten oder bestimmte Anwendungen.

Abbildung 3.3 gibt auf der linken Seite einen Überblick über die für die einzelnen Modelle als relevant identifizierten Konzepte, von denen einige nicht oder nicht vollständig in das integrierende Datenmodell KooMet (vgl. Kapitel 4) übernommen werden. Für die Referenzierung über Schlüssel im relationalen Datenmodell, für die Kapselung im objektorientierten Datenmodell sowie für die Festlegung der Attributreihenfolge in Record-Typen in XML wurde bereits in den entsprechenden Abschnitten diskutiert, warum sie sich für das integrierende Datenmodell KooMet nicht eignen. Die Inferenz von Metadaten und Zusatzinformationen, wie sie im Bereich der Beschreibungslogiken zu Verfügung steht, soll hier nochmals als geeignetes Konzept herausgestellt werden, das zwar außerhalb einer Datenmodellierung liegt, wie sie in Kapitel 4 vorgenommen wird, aber auf der Grundlage der vorhandenen Konzepte realisiert werden kann.

Die rechte Seite von Abbildung 3.3 gliedert die in das Metadatenmodell KooMet übernommenen Konzepte nach strukturierenden Konzepten sowie Konzepten zur Typisierung, zur Einordnung in die Anwendungsdomäne, zur Erfassung von Meta-Metadaten und zur Beschränkung des Erfassungsaufwands und verweist dabei nochmals für jedes Konzept auf das Modell, aus dem es stammt. In Ergänzung hierzu werden im Folgenden die Konzepte, die für das Metadatenmodell KooMet als erforderlich identifiziert worden sind, noch einmal aufgeführt:

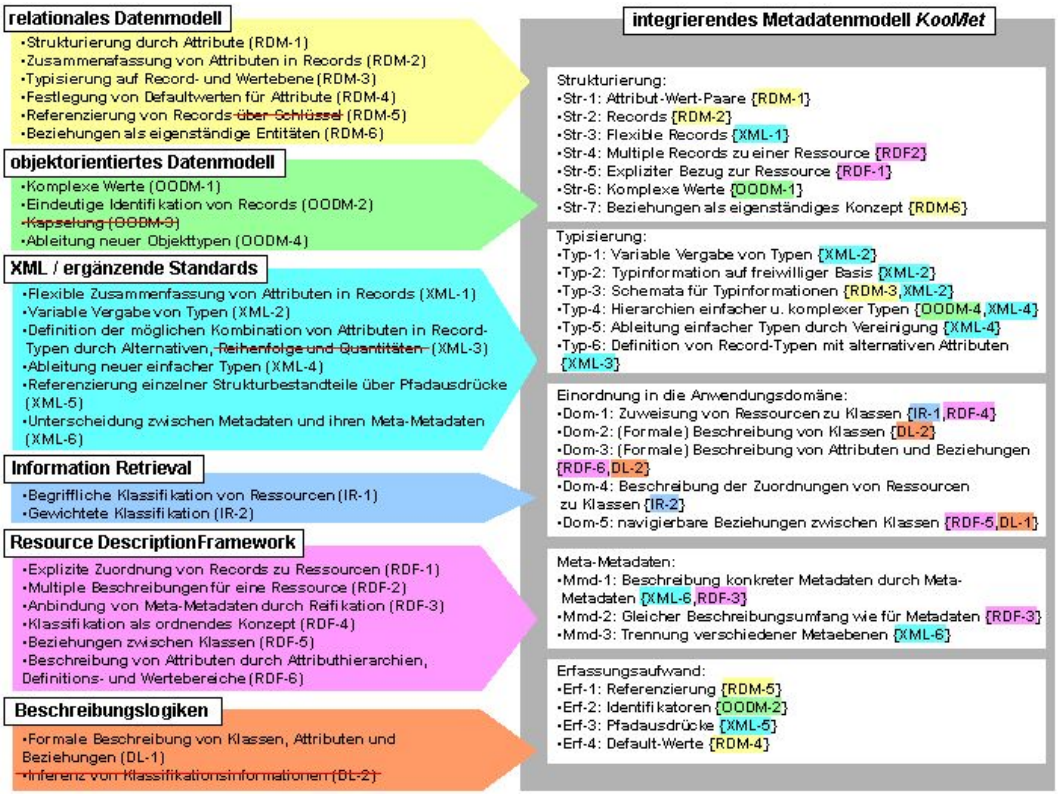

Abbildung 3.3: Gegenüberstellung existierender Datenmodelle und des integrierenden Metadatenmodells KooMet

- **Strukturierung:**

  Str–1: Ressourcen verfügen über verschiedene Eigenschaften, die als Attribut-Wert-Paare voneinander abgegrenzt werden können.

  Str–2: Attribut-Wert-Paare, die Eigenschaften für eine bestimmte Ressource beschreiben, können in einem Record zusammengefasst werden.

  Str–3: Records sind flexibel bezüglich der in ihnen zusammengefassten Attribut-Wert-Paare.

  Str–4: Zu einer Ressource können beliebig viele Records angelegt werden.

  Str–5: Ein expliziter Bezug zwischen Metadaten und der beschriebenen Ressource kann hergestellt werden.

  Str–6: Attributen können komplexe Werte in Form von geschachtelten Records sowie homogener und heterogener Kollektionen zugewiesen werden.

  Str–7: Beziehungen sind ein eigenständiges Strukturierungskonzept, können wiederum beschrieben werden und verbinden Ressourcen in einer Entität.

- **Typisierung:**

  Typ–1: Typinformationen können in variabler Granularität zugeordnet werden. Sie können also sowohl für einzelne Werte als auch für alle Werte eines Attributs und alle Elemente einer Kollektion zur Verfügung gestellt werden.

  Typ–2: Typinformationen können von Erfassern auf freiwilliger Basis zur Verfügung gestellt, aber auch weggelassen werden.

  Typ–3: Typinformationen können in einem Schema für alle Metadaten einer Kooperation bereitgestellt werden.

  Typ–4: Sowohl für komplexe als auch für einfache Typen lassen sich Typhierarchien aufbauen.

  Typ–5: Neue einfache Typen können durch Vereinigung existierender einfacher Typen definiert werden.

  Typ–6: Record-Typen erlauben die Angabe alternativer Attribute.

- **Einordnung in die Anwendungsdomäne:**

  Dom–1: Ressourcen werden durch die Zuweisung zu Klassen semantisch in die Anwendungsdomäne eingeordnet.

  Dom–2: Die Semantik einer Klasse kann formal oder frei beschrieben werden.

  Dom–3: Die Semantik von Attributen und Beziehungen kann formal oder frei beschrieben werden.

  Dom–4: Zuordnungen von Ressourcen zu Klassen können beschrieben werden.

  Dom–5: Klassen werden über navigierbare Beziehungen in einem Klassifikationssystem verbunden. Dabei sollte insbesondere die Spezialisierung von Klassen Berücksichtigung finden, ohne dabei auf die Möglichkeit zur Definition weiterer Klassenbeziehungen zu verzichten.

- **Meta-Metadaten:**

  Mmd–1: Konkrete Metadaten können selbst wiederum durch Meta-Metadaten beschrieben werden.

  Mmd–2: Die Beschreibungsmöglichkeiten für Meta-Metadaten sind ebenso umfangreich wie bei der Beschreibung von Ressourcen selbst.

  Mmd–3: Verschiedene Metaebenen lassen sich voneinander trennen.

- **Reduzierung des Erfassungsaufwands:**

  Erf–1: Durch Referenzierung können verschiedene Bestandteile von Metadaten wiederverwendet werden.

Erf–2: Über Identifikatoren können Metadatenbestandteile benannt und dann referenziert werden.

Erf–3: Die Struktur der Metadaten erlaubt die Verwendung von Pfadausdrücken, über die Metadatenbestandteile ebenfalls referenziert werden können.

Erf–4: Die Verwendung von Default-Werten begrenzt die mehrfache Erfassung sich wiederholender Eigenschaften.

Durch die Gesamtheit der aufgeführten Konzepte zur kooperativen Nutzung von Metadaten in gemeinsamen Informationsräumen werden unterschiedliche Kooperationsszenarien ermöglicht. Diese reichen von einer engen Zusammenarbeit, die neben dem Austausch von Ressourcen und Metadaten in großem Umfang auf zusätzlichen Vereinbarungen der Partner basiert, bis zu einer nur lose gekoppelten Zusammenarbeit zwischen Partnern, die sich zumeist kaum kennen und aus unterschiedlichen Arbeitskontexten stammen.

3.8 Zusammenfassung

Das relationale und das objektorientierte Datenmodell sowie die Datenmodelle in XML, dem Information Retrieval, RDF und den Beschreibungslogiken sind geeignet, um Metadaten für bestimmte, voneinander verschiedene Kooperationsszenarien darzustellen. Sie umfassen hierzu jeweils entsprechende Konzepte zur Strukturierung und Typisierung von Metadaten, zur Einordnung von Ressourcen in Klassifikationssysteme, zur Beschreibung von Metadaten durch Meta-Metadaten sowie zur Verringerung des Erfassungsaufwands für Metadaten.

Um durch ein Metadatenmodell verschiedene Kooperationsszenarien und damit verbundene Vorgehensweisen, wie sie in Kapitel 2 vorgestellt wurden, unterstützen zu können, ist eine Zusammenführung der Konzepte der vorgestellten Modelle notwendig. Diese muss so erfolgen, dass jedes Konzept gleichermaßen für eine bestimmte Kooperationssituation zum Einsatz kommen oder vernachlässigt werden kann. In dem im folgenden Kapitel 4 vorgestellten integrierenden Metadatenmodell KooMet wird eine solche Zusammenführung vollzogen.

Kapitel 4

Das Metadatenmodell KooMet

Im vorhergehenden Kapitel 3 wurden innerhalb konkreter bereits existierender Modelle Konzepte identifiziert, durch die Metadaten in bestimmten Kooperationsszenarien geeignet repräsentiert werden können. Diese Konzepte werden im vorliegenden Kapitel nun so zusammengeführt, dass sie gemeinsam nutzbar sind, um für verschiedene Projekte, in denen innerhalb eines Informationsraums dieselben Ressourcen unterschiedlich genutzt werden, verschiedene Kooperationsszenarien realisieren zu können. Dabei wird an der im vorigen Kapitel getroffenen Einteilung der Konzepte nach Strukturierung, Typisierung, Klassifikation, Erfassung von Meta-Metadaten und Verringerung des Erfassungsaufwandes festgehalten.

Aus der Zusammenführung der Konzepte resultiert das integrierende Metadatenmodell *KooMet* (*Koo*peratives *Met*adatenmodell), bei dem es sich zunächst um ein rein konzeptuelles Modell handelt. Implementationsmodelle, die auf bestimmte Kooperationskontexte zugeschnitten sind, indem sie eine entsprechend optimierte Erfassung, Ablage und Interpretation von Metadaten unterstützen, können daraus abgeleitet werden.

4.1 Strukturierung von Metadaten

Die Kenntnis der Struktur von Metadaten erleichtert Kooperationspartnern die Interpretation. Durch die Struktur kann dabei sowohl zum Ausdruck gebracht werden, welche semantischen Aspekte einer Ressource in welchen Metadatenbestandteilen beschrieben werden, als auch in welcher Form diese Beschreibung erfolgt.

In Abschnitt 3.7 wurden eine Reihe von Konzepten zur Strukturierung von Metadaten benannt, die den Austausch dieser Metadaten in gemeinsamen Informationsräumen erleichtern können. Diese Strukturierungskonzepte werden nun in KooMet umgesetzt. Wie diese Umsetzung erfolgt, veranschaulicht das UML-Klassendiagramm in Abbildung 4.1[1], wobei die in Abschnitt 3.7 verwen-

[1]Ein Klassendiagramm, das alle Klassen des Metadatenmodells KooMet sowie der Anfragesprache KooMet-QL (vgl. Abschnitt 5.2.3) miteinander in Beziehung setzt, findet sich in

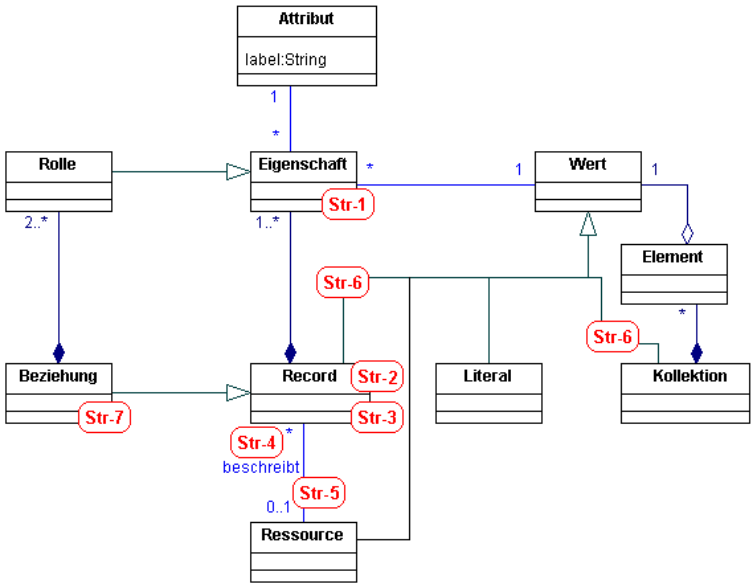

Abbildung 4.1: Strukturierung von Metadaten

deten Kürzel denjenigen Teil des Modells markieren, der das jeweilige Konzept abbildet.

Metadaten werden als eine Reihe voneinander abgrenzbarer *Eigenschaften* erfasst, wobei der jeweils beschriebene Aspekt der Ressource durch genau ein *Attribut* gekennzeichnet wird. Das Attribut ist hierzu mit einem Label ausgestattet, das den beschriebenen Aspekt explizit benennt. Die Eigenschaft ordnet dem Attribut einen *Wert* zu, der die entsprechende Beschreibung enthält (vgl. Abschnitt 3.7, Konzept Str-1). Eigenschaften werden in *Records* zusammengefasst (vgl. Abschnitt 3.7, Konzept Str-2). Ein Record kann dabei beliebig viele Eigenschaften enthalten. Da leere Records keine Information zur Beschreibung von Ressourcen enthalten, werden sie nicht verwendet.

Die Flexibilität von Records (vgl. Abschnitt 3.7, Konzept Str-3) manifestiert sich dadurch, dass weder für Records selbst noch für Attribute Festlegungen getroffen werden müssen, die die Zusammensetzung eines Records vorwegnehmen. Das Modell unterscheidet sich somit von Datenmodellen wie dem relationalen oder dem objektorientierten Modell, in denen über starre Record-Typen die Struktur eines jeden Records vorgegeben ist.

Eine Ressource kann durch beliebig viele Records beschrieben werden (vgl. Abschnitt 3.7, Konzept Str-4). Um dabei den eindeutigen Bezug zwischen einem Record und der durch ihn beschriebenen Ressource herzustellen, wird die Ressource durch den Record explizit benannt (vgl. Abschnitt 3.7, Konzept Str-5).

Bei der Erfassung von Eigenschaften können neben einfachen Werten, die in Abbildung 4.1 als *Literale* bezeichnet sind, und Verweisen auf weitere Ressourcen (vgl. hierzu auch Abschnitt 4.5) auch komplexe Werte in Form von ge-

Anhang A.

schachtelten Records und *Kollektionen* verwendet werden (vgl. Abschnitt 3.7, Konzept Str-6). Für die *Elemente* einer Kollektion wird an dieser Stelle keine weitere Einschränkung getroffen, so dass die Erfassung sowohl homogener als auch heterogener Kollektionen möglich ist.

Beziehungen werden als eigenständige Entitäten (vgl. Abschnitt 3.7, Konzept Str-7) in Form von speziellen Records repräsentiert. Anders als ein allgemeiner Record beschreibt ein Beziehungsrecord keine Ressource, was in Ergänzung zu Abbildung 4.1 im folgenden Ausdruck der Object Constraint Language (OCL) formuliert wird:

```
context Beziehung inv:
  self.ressource -> isEmpty()
```

An dieser Stelle sei zugleich noch einmal betont, dass alle anderen Records des Modells einen Bezug zu der durch sie beschriebenen Ressource besitzen:

```
context Record inv:
  not(self.oclIsTypeOf(Beziehung)) implies
  self.ressource -> notEmpty()
```

Ebenso wie Records, die Ressourcen beschreiben, umfassen auch Beziehungsrecords eine Reihe von Eigenschaften, unter denen sich jeweils mindestens zwei spezielle Eigenschaften befinden müssen, die hier als *Rollen* bezeichnet werden. Rollen sind Eigenschaften, bei deren Wert es sich um eine Ressource handelt, so dass über sie die eigentliche Beziehung zwischen zwei oder mehr Ressourcen aufgebaut wird:

```
context Rolle inv:
  self.wert.oclType = Ressource
```

Alle weiteren Eigenschaften des Beziehungsrecords dienen der Beschreibung der Beziehung selbst.

4.2 Typisierung von Metadaten

Die Definition und Zuordnung von Datentypen erhöht den Erfassungsaufwand für Metadaten und kann sich in einigen Szenarien als Barriere erweisen, durch die die Erfassung von Metadaten unterdrückt wird. Daher ist anders als bei der Verwaltung von Datenbanken oder in der Programmierung (vgl. Abschnitte 3.1 und 3.2) das Ziel der Typisierung von Metadaten nicht eine vollständige Typsicherheit. Vielmehr wird die Angabe von Typen bei der Kooperation auf der Grundlage von Metadaten als optionale Anreicherung dieser Metadaten verstanden, die den Aufbau von verwendeten Werten beschreibt und in unterschiedlichen, den Vorgehensweisen des jeweiligen Kooperationsszenariums angepassten Schritten durchgeführt werden kann. Anwendungen, die Metadaten interpretieren, gehen einerseits davon aus, dass Werte den angegebenen Typen entsprechen, und müssen andererseits robust gegen Typfehler und fehlende Typangaben sein.

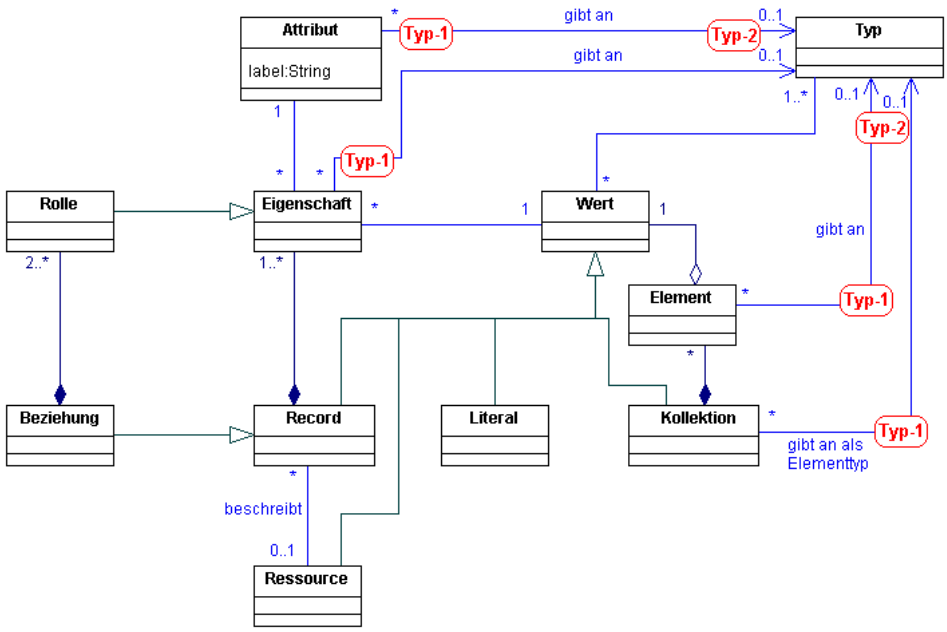

Abbildung 4.2: Typisierung von Metadaten

Abschnitt 3.7 beschreibt eine Reihe von Konzepten, durch die die kooperative Verwendung von Typinformationen zur Anreicherung von Metadaten unterstützt wird. Die Umsetzung dieser Konzepte in KooMet wird im Folgenden erläutert.

Werte erscheinen in der in Abschnitt 4.1 beschriebenen Metadatenstruktur entweder innerhalb einer Eigenschaft oder als Element einer Kollektion und können direkt dort mit einer Typangabe versehen werden (vgl. Abb. 4.2). Darüber hinaus ist es jedoch auch möglich, den *Typ* über ein Attribut für alle seine Eigenschaften und über eine Kollektion für alle ihre Elemente anzugeben. Hierdurch verringert sich zum einen zwar der Erfassungsaufwand, zum anderen lassen sich jedoch typisierte Kollektionen und insbesondere typisierte Attribute nur entsprechend eingeschränkt verwenden. Da sowohl einzelne Werte als auch Mengen von Werten mit Typangaben angereichert werden können, lassen sich unterschiedliche Granularitäten bei der Typisierung erreichen (vgl. Abschnitt 3.7, Konzept Typ-1).

Die Angabe eines Typs für Eigenschaften, Attribute, Kollektionselemente und Kollektionen ist jeweils optional, so dass sie auch entfallen kann, wenn dieses dem jeweiligen Kooperationsszenarium eher entspricht (vgl. Abschnitt 3.7, Konzept Typ-2). Da davon ausgegangen wird, dass ein Metadatenerfasser mit der Typisierung eines Attributs eine bestimmte Absicht verfolgt, die nicht innerhalb der einzelnen Eigenschaften aufgeweicht werden sollte, sollte innerhalb einer Eigenschaft, die das entsprechende Attribut mit einem Wert belegt, kein abweichender Typ für diesen Wert angegeben werden. Eine solche Forderung kann jedoch, anders als in Datenbanksystemen oder Programmierumgebungen, nicht in jedem gemeinsamen Informationsraum bei der Erfassung sichergestellt

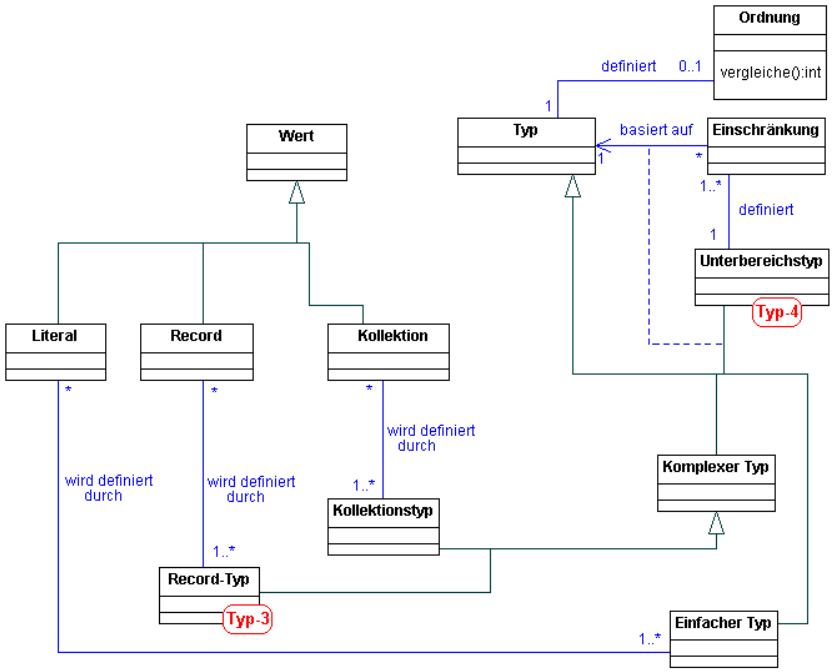

Abbildung 4.3: Typisierung verschiedener Werte

werden. Für den Fall, dass für ein Attribut und eine seiner Eigenschaften widersprüchliche Typen angegeben sind, wird die Angabe in der einzelnen Eigenschaft bei der Interpretation ignoriert und angenommen, dass die Eigenschaft einen Wert beinhaltet, der dem im Attribut deklarierten Typ entspricht. Die Möglichkeit, dass die Eigenschaft hierdurch u.U. nicht korrekt oder gar nicht interpretiert werden kann, wird dabei bewusst in Kauf genommen. Auch in Kollektionen, für die ein einheitlicher Typ angegeben wird, sollte dieser nicht für einzelne Elemente abgeändert werden. Der für die Kollektion angegebene Elementtyp wird bei der Interpretation als für alle Elemente gültig angenommen.

Als Werte zur Beschreibung der Eigenschaften von Ressourcen können sowohl Literale als auch Verweise auf Ressourcen sowie Kollektionen und geschachtelte Records verwendet werden (vgl. Abschnitt 4.1), die jeweils unterschiedlich aufgebaut sind. Dementsprechend umfasst KooMet verschiedene Arten von Typen (vgl. Abb. 4.3). *Einfache Typen* beinhalten und beschreiben den Aufbau von atomaren Werten in Form von Literalen, Kollektionen werden mit Hilfe von *Kollektionstypen* beschrieben, *Record-Typen* reichern Records mit Typinformationen an.

Auch Ressourcen können typisiert werden, indem beispielsweise für digitale Ressourcen ein bestimmtes Dateiformat angegeben wird. Da durch die modellierten Metadaten grundsätzlich jedoch beliebige Ressourcen, wie auch nicht digitale Ressourcen oder Software, beschrieben und typisiert werden können, deren Typinformationen jeweils voneinander abweichen, sei an dieser Stelle auf weitere Ausführungen hierzu verzichtet.

Durch die Festlegung von zulässigen Record-Typen sowie von Typen auf Ba-

sis aller darin geschachtelten Attribute ist es möglich, ein vollständiges Schema als Grundlage für eine Kooperation zu erstellen (vgl. Abschnitt 3.7, Konzept Typ-3), dessen Einhaltung durch entsprechende Erfassungswerkzeuge für Metadaten sichergestellt und das für eine performante Interpretation von Metadaten verwendet werden kann [ABS00].

Für einen Typ lassen sich *Einschränkungen* formulieren, um dadurch einen *Unterbereichstyp* zu definieren (vgl. Abb. 4.3). Dabei können für einen einzelnen Unterbereichstyp mehrere Einschränkungen formuliert werden, diese müssen jedoch alle denselben Basistyp einschränken:

```
context Einschraenkung inv:
  self.unterbereichstyp.einschraenkung ->
  select(e | e.typ <> self.typ) -> isEmpty()
```

Ein Unterbereichstyp umfasst Werte, die an Stellen verwendet werden können, wo auch ein Wert des zugrunde liegenden Basistyps verwendet werden könnte, und spezialisiert so den Basistyp. Einschränkungen lassen sich sowohl für einfache als auch für Kollektions- und Record-Typen formulieren. Für viele Typen lässt sich eine *Ordnung* über ihren Werten definieren. Diese Ordnung kann zur Formulierung von Einschränkungen herangezogen werden. Die Definition von Unterbereichstypen mit Hilfe von Einschränkungen wird in KooMet genutzt, um Hierarchien einfacher und komplexer Werte zu konstruieren (vgl. Abschnitt 3.7, Konzept Typ-4).

Die Informationen, die in einem Typ zum Aufbau seiner Werte bereitgestellt werden können, und die Einschränkungen darauf unterscheiden sich für einfache, Kollektions- und Record-Typen und werden in den folgenden Unterabschnitten für jede Art von Typ einzeln erläutert.

4.2.1 Einfache Typen

Bei den Werten, die ein einfacher Typ umfasst, handelt es sich um Literale, die innerhalb von Metadaten mit Hilfe von Zeichenketten dargestellt werden, die keinerlei interpretierbare Strukturinformation enthalten, wie sie durch z.B. XML-Tags oder andere spezielle Begrenzungszeichen gegeben wäre. Ein einfacher Typ beschreibt den Aufbau dieser Zeichenketten. Hierzu kann er entweder ein Muster angeben, dem seine Werte entsprechen, oder seine Werte einzeln aufzählen, wenn ihre Zahl dieses zulässt. Zur Beschreibung der Werte über Muster können Beschreibungsformen wie die Backus-Naur-Form oder reguläre Ausdrücke eingesetzt werden.

Die Beschreibung aller Werte eines Typs ist umfangreich und kompliziert und erfordert Fachkenntnisse, die bei Erfassern von Typinformationen nicht allgemein vorausgesetzt werden können. Daher wird für den Aufbau einer Hierarchie einfacher Typen häufig auf vordefinierte Typen zurückgegriffen, die anderweitig bereits beschrieben worden sind. Die einfachen Typen in XML Schema [XML01b] oder einer Programmiersprache können hier beispielsweise als Basis verwendet und dann falls nötig über die Formulierung von Einschränkungen an den jeweiligen Kooperationskontext angepasst werden.

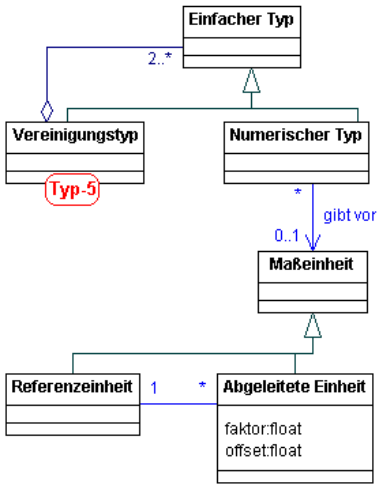

Abbildung 4.4: Vereinigungs- und numerische Typen als spezielle einfache Typen

Zur Bildung einfacher Unterbereichstypen lassen sich eine Reihe von Einschränkungen formulieren, die im Folgenden aufgeführt, kurz erläutert und durch ein Beispiel skizziert werden:

Definition einer Unter- oder Obergrenze: Für Typen, auf denen eine Ordnung definiert ist, enthält der entsprechende Unterbereichstyp nur Werte, die bezüglich dieser Ordnung den gegebenen Grenzwert nicht unter- bzw. nicht überschreiten. Beispiel: Alle ganzen Zahlen, die größer sind als 42

Einschränkung der Länge der Zeichenketten: Der Unterbereichstyp enthält nur diejenigen Literale des Basistyps, die eine bestimmte Anzahl von Zeichen umfassen bzw. diese Anzahl nicht unter- oder überschreiten. Beispiel: Alle vierstelligen natürlichen Zahlen

Angabe eines Musters: Der Unterbereichstyp enthält nur diejenigen Literale des Basistyps, die ein vorgegebenes Muster erfüllen. Beispiel: Alle Gleitkommazahlen, die mit einer 5 beginnen und mindestens zwei aber höchstens drei Nachkommastellen besitzen

Aufzählung: Der Unterbereichstyp enthält nur die in der Einschränkung aufgezählten Literale des Basistyps. Beispiel: Die ganzen Zahlen 1, 2, 3, 4, 5 und 6

Neben der Definition von Unterbereichstypen stellt die Bildung von *Vereinigungstypen* eine weitere Möglichkeit zur Definition einfacher Typen auf der Basis bereits existierender Typen dar (vgl. Abschnitt 3.7, Konzept Typ–5). Ein Vereinigungstyp ist ein spezieller einfacher Typ, der die Literale mehrerer einfacher Typen zusammenfasst (vgl. Abb. 4.4) und sich beispielsweise bei der Zusammenführung von verschiedenen Metadatenbeständen einsetzen lässt, in

denen unterschiedliche Repräsentationen für Werte mit gleicher Semantik existieren.

Eine Besonderheit *numerischer Literale* ist, dass sie häufig in Verbindung mit *Maßeinheiten* auftreten, deren Kenntnis die Interpretation eines numerischen Metadatenwerts erst ermöglicht. Um die Kooperation auf der Grundlage numerischer Werte mit Maßeinheit auch über organisatorische und regionale Grenzen hinaus zu vereinfachen, stellt KooMet hier ein Konzept zur Repräsentation und zur Umrechnung kompatibler Maßeinheiten zur Verfügung.

Maßeinheiten, die dieselbe physikalische Größe beschreiben, lassen sich in einer Gruppe zusammenfassen, wobei es für jede Gruppe eine *Referenzeinheit* gibt (vgl. Abb. 4.4). Alle anderen Maßeinheiten der Gruppe sind *abgeleitete Einheiten*. Werte in einer abgeleiteten Maßeinheit können über einen Faktor und ein Offset in Werte in der dazugehörigen Referenzeinheit umgerechnet werden und umgekehrt, so dass sich transitiv ein Umrechnungssystem zwischen allen Maßeinheiten einer Gruppe ergibt.

Bei der Definition eines numerischen Typs kann dieser mit einer beliebigen Maßeinheit aus einer Maßeinheitengruppe verbunden werden, um zum Ausdruck zu bringen, dass alle Maßeinheiten der Gruppe in Werten dieses Typs auftreten können. Das Konzept mit Maßeinheiten ausgestatteter numerischer Werte ist im Rahmen der Entwicklung des Katalogsystems PIA entstanden (vgl. Abschnitt 6.2 sowie [MS99]).

4.2.2 Kollektionstypen

Der Aufbau von Kollektionen kann mit Hilfe von Kollektionstypen näher beschrieben werden. Zunächst definiert jeder Kollektionstyp eine *Kollektionsart* für die in ihm enthaltenen Kollektionen (vgl. Abb. 4.5). Die Kollektionsart legt fest, ob es sich bei den Kollektionen des Kollektionstyps beispielsweise um Mengen, Multimengen, Alternativen, Sequenzen oder andere Kollektionsarten handelt. Zusätzlich zur Angabe im Kollektionstyp wird direkt innerhalb jeder Kollektion die dazugehörige Kollektionsart benannt. So wird gewährleistet, dass keine untypisierten Kollektionen ohne Kollektionsart erfasst werden, da diese keine klare Semantik besitzen und damit nicht interpretierbar sind. Ist für eine Kollektion in einem Attribut, einer Eigenschaft, einer anderen Kollektion oder einem Element einer anderen Kollektion ein Kollektionstyp angegeben, so muss die in der Kollektion benannte Kollektionsart der im Typ deklarierten Kollektionsart entsprechen:

```
context Kollektion inv:
  self.angegebenerTyp -> notEmpty()
  implies
  self.kollektionsart = self.angegebenerTyp.kollektionsart
```

Die Angabe des Kollektionstyps ist der Übersichtlichkeit halber in Abbildung 4.5 als direkte Assoziation zwischen dem Kollektionswert und dem angegebenen Kollektionstyp dargestellt und wird im obigen OCL-Ausdruck auch so verwendet.

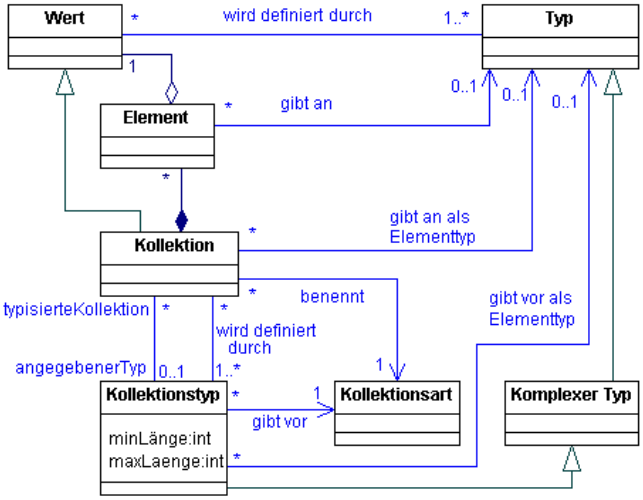

Abbildung 4.5: Angaben in Kollektionstypen

Der Elementtyp in Kollektionen kann entweder direkt innerhalb des jeweiligen Elements oder für alle Elemente einer Kollektion einheitlich festgelegt werden (vgl. Abb. 4.1). Darüber hinaus kann auch in einem Kollektionstyp ein Elementtyp angegeben werden (vgl. Abb. 4.5). Hierdurch wird angegeben, dass alle Elemente in allen Kollektionen des Kollektionstyps vom deklarierten Elementtyp sind. Gibt ein Kollektionstyp einen Elementtyp vor, so ist dieser für alle Elemente in allen Kollektionswerten des Typs gültig. Falls eine Kollektion des Typs oder ein Element in einer Kollektion einen abweichenden Elementtyp angibt, so wird dieser bei der Interpretation ignoriert. Die Möglichkeit eines Typfehlers wird dabei bewusst in Kauf genommen.

Manchmal lässt sich bei der Definition eines Kollektionstyps bereits absehen, dass seine Werte eine bestimmte Elementanzahl nicht unter- oder überschreiten werden. Die Anzahl der Elemente, die in den Kollektionen eines Kollektionstyps vorhanden sind, lässt sich daher über diesen Typ durch die Angabe einer minimalen und/oder einer maximalen Kollektionslänge beschränken (vgl. Abb. 4.5).

Auch für Kollektionswerte lassen sich mit Hilfe von Einschränkungen Unterbereichstypen definieren. Hierzu gibt es zwei verschiedene Möglichkeiten:

Beschränkung der Anzahl der Elemente: Der Unterbereichstyp umfasst diejenigen Kollektionen des Basistyps, die eine bestimmte Anzahl von Elementen beinhalten bzw. diese nicht über- oder nicht unterschreiten. Ist für den Basistyp bereits eine Beschränkung der Elementanzahl definiert, so darf die Beschränkung im Unterbereichstyp dieser Vorgabe nicht zuwiderlaufen. Beispiel: Einschränkung des Typs „Autorenliste mit mindestens drei Namen“ auf den Unterbereichstyp „Autorenliste mit mindestens fünf Namen“

Spezialisierung des Elementtyps Der Unterbereichstyp umfasst diejenigen Kollektionen des Basistyps, deren Elemente einen Elementtyp besitzen,

der eine bestimmte Spezialisierung des im Basistyp festgelegten Elementtyps darstellt. Beispiel: Einschränkung des Typs „Menge ganzer Zahlen“ auf den Unterbereichstyp „Menge positiver ganzer Zahlen“

4.2.3 Record-Typen

Der Aufbau von Records wird mit Hilfe von Record-Typen beschrieben, die die Flexibilität bei der Bildung von Records einschränken zugunsten einer im Kooperationskontext gemeinsam nutzbaren Metadatenstruktur.

Bei der Definition von Record-Typen wird auf eine Reihe von Möglichkeiten verzichtet, die in anderen Datenmodellen, wie für XML in den entsprechenden Schemabeschreibungssprachen (vgl. Abschnitt 3.3), zur Verfügung stehen. So entfällt in KooMet-Record-Typen zum einen die Quantifizierung sich wiederholender Attribute, da durch die Belegung von Attributen durch Kollektionswerte Strukturen gleicher Semantik aufgebaut und durch Kollektionstypen (vgl. Abschnitt 4.2.2) beschrieben werden können, so dass auf eine redundante Beschreibungsmöglichkeit innerhalb von Record-Typen verzichtet werden kann. Zum anderen kann in einem Record-Typ keine Reihenfolge für Attribute vorgesehen werden, da diese einerseits der Formulierung von Attribut-Wert-Paaren als abgrenzbare Eigenschaften widerspricht und andererseits im verteilten asynchronen Kooperationskontext schwierig durchzusetzen ist.

In KooMet kann ein Record-Typ Attribute aufzählen (vgl. Abb. 4.6), die in den Records des Typs auf jeden Fall erscheinen. Der Record-Typ trifft keine Einschränkung hinsichtlich weiterer Attribute, die zusätzlich in den Records erscheinen dürfen.

Daneben können in einem Record-Typ *Varianten* definiert werden (vgl. Abb. 4.6). Diese enthalten jeweils Gruppen von Attributen, die *alternativ* zueinander in einem Record erscheinen können (vgl. Abschnitt. 3.7, Konzept Typ–6). So ließe sich beispielsweise in einem Record-Typ Publikation festlegen, dass die entsprechenden Records einen Zeitschriftentitel, einen Jahrgang und eine Nummer angeben, wenn es sich bei der dazugehörigen Ressource um einen Artikel handelt, oder alternativ dazu eine Konferenz aufführen, wenn die Ressource in Form eines Konferenzbeitrags vorliegt.

Für Record-Typen können mit Hilfe der folgenden Einschränkungen Unterbereichstypen definiert werden:

Hinzufügen von Attributen: Für Basistypen definiert ein Unterbereichstyp weitere Attribute, die zusätzlich zu den im Basistyp benannten Attributen in den Records des Unterbereichstyps enthalten sind. Beispiel: Einschränkung des Record-Typs Person mit den Attributen Name und Vorname auf den Unterbereichstyp Student mit dem zusätzlichen Attribut Matrikelnummer

Weglassen von Varianten: Für Basistypen, die eine Variante mit Gruppen alternativer Attribute enthalten, führt der Unterbereichstyp nur eine Teilmenge der im Basistyp benannten alternativen Attribute auf. Beispiel: Einschränkung des Record-Typs Publikation mit Attributgruppen, die sich für Artikel, Konferenzbeiträge und interne Berichte jeweils unterscheiden,

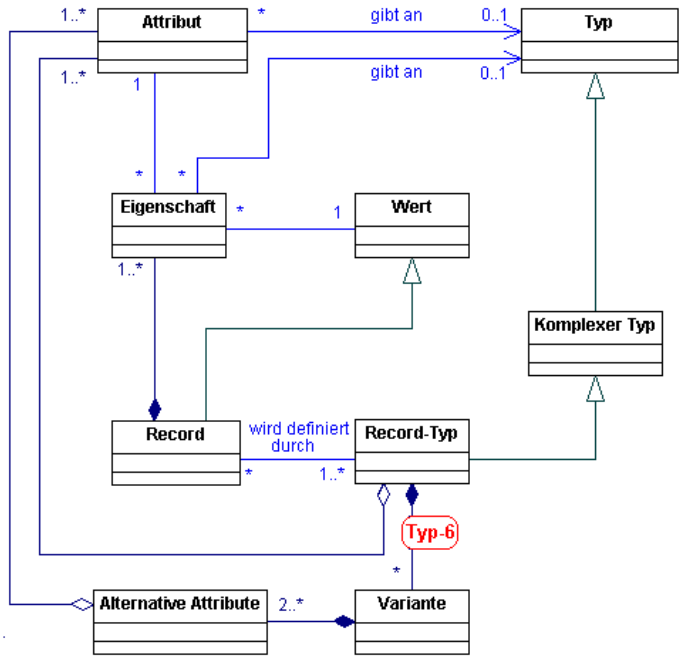

Abbildung 4.6: Angaben in Record-Typen

auf den Unterbereichstyp Begutachtete Publikation durch Weglassen derjenigen Attributgruppe, die interne Berichte beschreibt.

Record-Typen beschreiben den Aufbau von Records und unterstützen damit deren Interpretation. Sie unterscheiden sich dabei von Klassen, die Ressourcen semantisch einordnen und hierzu teilweise auf die Beschaffenheit von Attributen in Records zurückgreifen. Dieser Unterschied wird, unter anderem, im folgenden Abschnitt, der die semantische Einordnung von Ressourcen erläutert, weiter herausgearbeitet werden.

4.3 Einordnung in die Anwendungsdomäne

Über die bisher vorgestellten Konzepte von KooMet können Eigenschaften von Ressourcen beschrieben und zusammengefasst werden sowie Ressourcen zueinander in Beziehung gesetzt werden. Die Zuordnung der Ressourcen in einem gemeinsamen Informationsraum zu *Klassen* kann die Semantik dieser Ressourcen in den verschiedenen, für den Informationsraum relevanten Anwendungsdomänen repräsentieren (vgl. Abschnitt 2.3.2). In Abschnitt 3.7 sind die Konzepte aufgeführt, die die Grundlage für die semantische Beschreibung von Ressourcen mit Hilfe von Klassen bilden. Klassen sind hierbei von Record-Typen (vgl. Abschnitt 4.2) abzugrenzen, die den Aufbau von Records beschreiben, um deren Interpretation zu vereinfachen [Nor94]. Der vorliegende Abschnitt ordnet die Konzepte zur semantischen Beschreibung von Ressourcen durch Klassen in den Zusammenhang des integrierenden Metadatenmodells KooMet ein.

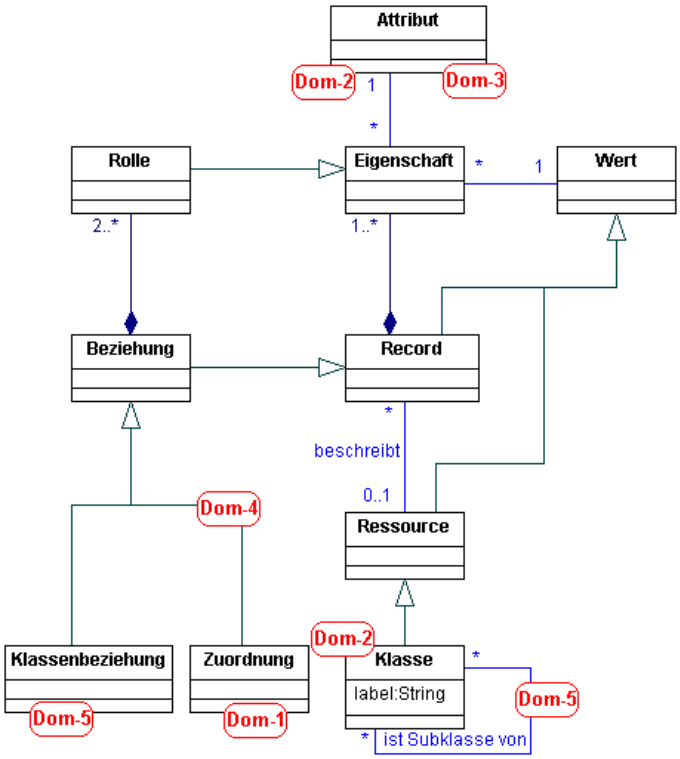

Abbildung 4.7: Einordnung von Ressourcen in durch Klassen beschriebene Anwendungsdomänen

Klassen in gemeinsamen Informationsräumen bilden häufig Begriffe oder Ordnungsstrukturen ab, die außerhalb des Informationsraums geprägt wurden und deren Bedeutung allen Kooperationspartnern bekannt ist, so dass mit den Klassen eine Art gemeinsame Terminologie der Akteure in den Informationsraum eingebracht wird. Klassen tragen daher in KooMet Labels, die den Begriffen in dieser Terminologie entsprechen (vgl. Abb. 4.7).

Klassen werden in KooMet als spezielle Art von Ressourcen modelliert. Hierdurch wird erreicht, dass einerseits für Klassen selbst alle Beschreibungsmechanismen des Datenmodells verwendet werden können (vgl. Abschnitt 3.7, Konzept Dom–2). Andererseits kann auf das Konzept der Beziehung (vgl. Abschnitt 4.1) zurückgegriffen werden, um sowohl die *Zuordnungen* von Ressourcen zu Klassen als auch die *Beziehungen von Klassen* untereinander zu repräsentieren.

Für eine Ressource kann es beliebig viele Zuordnungen geben, die sie jeweils mit einer Klasse verbinden und dadurch in die Semantik einer Anwendungsdomäne einordnen (vgl. Abschnitt 3.7, Konzept Dom–1). Die Zuordnung einer Ressource zu einer Klasse entspricht einer speziellen Art von Beziehung, die genau zwei Rollen enthält, von denen eine durch die Klasse und die andere durch die ihr zugeordnete Ressource besetzt ist:

```
context Beziehung inv:
  self.oclIsTypeOf(Zuordnung) implies
  self.rolle ->
  (size = 2 and
   exists(r | r.wert.oclIsTypeOf(Klasse)) and
   exists(r | r.wert.oclIsTypeOf(Ressource)))
```

Da die Zuordnung einer Ressource zu einer Klasse durch eine Beziehung repräsentiert wird, kann sie durch die Ergänzung weiterer Attribut-Wert-Paare zusätzlich beschrieben werden, beispielsweise durch Gewichtungen, eine Angabe des Erfassers, der die Zuordnung vorgenommen hat, o.ä. (vgl. Abschnitt 3.7, Konzept Dom–4).

Indem Klassen auch zueinander in Beziehung gesetzt werden (vgl. Abschnitt 3.7, Konzept Dom–5), entstehen Klassifikationssysteme, die den Kooperationspartnern im gemeinsamen Informationsraum als semantisches Leitsystem zur Orientierung hinsichtlich der vorhandenen Ressourcen und darüber hinaus hinsichtlich der Zusammenhänge in Anwendungsdomänen zur Verfügung stehen. Der Spezialisierungsbeziehung zwischen Klassen kommt hier eine besondere Bedeutung zu, da sie in vielen Bereichen ihren Platz hat (vgl. hierzu z.B. Abschnitte 3.2, 3.5 und 3.6) und häufig als formale Grundlage in Algorithmen Verwendung findet. Sie wird aus diesem Grund in KooMet entsprechend hervorgehoben (vgl. Abb. 4.7). Generell werden in KooMet Beziehungen zwischen Klassen als spezielle Beziehungen dargestellt, deren Rollen jeweils von einer Klasse gefüllt werden. Da ein Klassifikationssystem navigierbar sein sollte, ist die Anzahl der Rollen in Klassenbeziehungen ebenfalls auf zwei begrenzt:

```
context Beziehung inv:
  self.oclIsTypeOf(Klassenbeziehung) implies
  self.rolle ->
  (size = 2 and
   forAll(r | r.wert.oclIsTypeOf(Klasse))
```

Die Semantik einer Anwendungsdomäne kann formal mit Hilfe von Beschreibungslogiken dargestellt werden (vgl. Abschnitt 3.6). Da verschiedene Beschreibungslogiken existieren, die unterschiedliche Mengen an Konstruktoren und Axiomen umfassen, aus denen sich auch eine unterschiedliche Komplexität für entsprechende Inferenzdienste ableitet [Don03], legt das Metadatenmodell KooMet Akteure in gemeinsamen Informationsräumen nicht auf eine spezielle Beschreibungslogik fest. Vielmehr ist diese in Abhängigkeit der für die entsprechenden Anwendungsdomänen erforderlichen Ausdrucksmächtigkeit sowie der gewünschten Inferenzalgorithmen frei wählbar.

KooMet stellt mit den Konzepten der Klasse, der Klassenbeziehung und des Attributs eine Schnittstelle zur Verfügung, an die Ausdrücke in der TBox einer Beschreibungslogik anknüpfen können (vgl. Abschnitt 3.7, Konzepte Dom–2 und Dom–3). Die entsprechende ABox kann direkt durch KooMet-Strukturen repräsentiert werden. Zuordnungen weisen dabei Instanzen ihren Klassen zu. Beziehungen zwischen den Instanzen können gleichermaßen durch eigenständige Beziehungen und durch Ressource-wertige Eigenschaften dargestellt werden, da

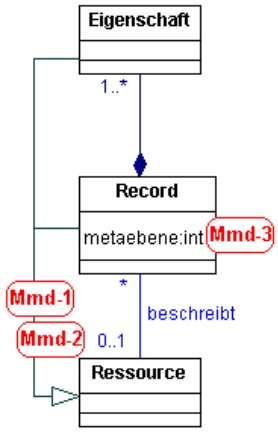

Abbildung 4.8: Beschreibung von Metadaten durch Meta-Metadaten

in KooMet beide Konzepte auf der Grundlage von Attributen definiert werden (vgl. Abschnitt 4.1).

4.4 Beschreibung von Metadaten durch Meta-Metadaten

Meta-Metadaten dienen nicht der Beschreibung beliebiger Ressourcen, sondern beschreiben ausschließlich andere Metadaten und werden daher in Kooperationskontexten verwendet, in denen Metadaten selbst die Arbeitsressourcen darstellen. Abschnitt 3.7 führt hierzu einige Konzepte an, die im Folgenden für KooMet umgesetzt werden, damit es auch Kooperationsszenarien unterstützt, die die Arbeit mit Meta-Metadaten vorsehen.

In einer Metadatenstruktur, die dem Metadatenmodell KooMet entsprechend aufgebaut ist (vgl. Abschnitt 4.1), findet die konkrete Erfassung durch die Angabe von Eigenschaften für Ressourcen statt. Um Metadaten selbst wiederum beschreiben zu können (vgl. Abschnitt 3.7, Konzept Mmd–1), müssen daher Eigenschaften ebenfalls als Ressourcen aufgefasst werden (vgl. Abb. 4.8), wodurch für sie alle Beschreibungsmittel von KooMet zur Verfügung stehen (vgl. Abschnitt 3.7, Konzept Mmd–2).

Eigenschaften werden innerhalb von Records zusammengefasst. Da diese Records häufig in einem einzigen Arbeitsschritt entstehen, gibt es Meta-Metadaten, die nicht nur für einzelne Eigenschaften, sondern für alle Eigenschaften eines Records Gültigkeit haben. Um solche Meta-Metadaten nicht für jede Eigenschaft wiederholt erfassen zu müssen, werden auch Records als spezielle Ressourcen betrachtet, die durch Meta-Metadaten beschrieben werden können, wodurch sich der Erfassungsaufwand (vgl. Abschnitt 4.5) für Meta-Metadaten verringert.

Um zyklische Abhängigkeiten zwischen Metadaten zu vermeiden und dadurch zu verhindern, dass diese sich letztendlich selbst beschreiben, und um allgemein Kooperationskontexte trennen zu können, in denen Metadaten einer-

seits und Meta-Metadaten andererseits die Arbeitsgrundlage bilden, lässt sich der Metadatenbestand eines Informationsraums in mehrere Metaebenen aufteilen (vgl. Abschnitt 3.7, Konzept Mmd-3). KooMet weist hierzu jedem Record eine Zahl zu, die seine Metaebene repräsentiert. Ein Meta-Metadaten-Record darf nur Records, die eine niedrigere Metaebene besetzen, oder Eigenschaften, die sich in Records einer niedrigeren Ebene befinden, beschreiben:

```
context Record inv:
  self.ressource.oclIsTypeOf(Record) implies
  self.metaebene > self.ressource.metaebene

inv:
  self.ressource.oclIsTypeOf(Eigenschaft) implies
  self.metaebene > self.ressource.record.metaebene
```

Durch die Trennung und die Ordnung der Metaebenen von beschriebenen und beschreibenden Records wird unterbunden, dass ein Record direkt oder transitiv über eine oder mehrere Ebenen hinweg sich selbst oder eine seiner Eigenschaften beschreibt.

4.5 Reduzierung des Erfassungsaufwands

Die Erfassung von Metadaten stellt im Rahmen einer Zusammenarbeit neben der Bearbeitung der eigentlichen Kooperationsaufgabe einen Zusatzaufwand dar, dessen Nutzen erst bei erneutem Zugriff auf die Metadaten sichtbar wird. Der Erfassungsaufwand für Metadaten sollte daher so gering wie möglich gehalten werden.

Abschnitt 3.7 stellt mit der Referenzierung bereits erfasster Metadatenbestandteile und der Vorgabe von Default-Werten Konzepte vor, die unnötige Erfassungsschritte vermeiden können. Beide Konzepte kommen an verschiedenen Stellen innerhalb von KooMet zum Tragen, wurden aber in den entsprechenden Erläuterungen und Diagrammen bisher nicht erwähnt, um die übrigen Konzepte des Metadatenmodells jeweils präzise beschreiben zu können. Der vorliegende Abschnitt zeigt nun auf, für welche Teile des Modells die Konzepte zur Begrenzung des Erfassungsaufwands als relevant erachtet werden können.

Der Erfassungsaufwand für Metadaten kann erheblich reduziert werden, wenn bereits erfasste Metadaten und Zusatzinformationen in einem anderen Kontext wiederverwendet werden können (vgl. Abschnitt 3.7, Konzept Erf–1). Verschiedene Metadatenbestandteile können zu verschiedenen Zwecken referenziert werden:

Bezug auf die beschriebenen Ressourcen: Metadaten beziehen sich explizit auf die Ressourcen, die sie beschreiben, oder setzen diese, ebenfalls explizit, zueinander in Beziehung. Für eine Ressource, die außerhalb der Metadaten selbst existiert, kann ein Bezug in den sie beschreibenden Metadaten allein durch die Referenzierung dieser Ressource hergestellt werden. Doch auch innerhalb der Metadaten selbst existieren spezielle Ressourcen,

die durch Metadaten beschrieben werden können. Hierzu zählen zum einen Klassen, die zur Repräsentation von Klassifikationssystemen der Anwendungsdomäne als Ressourcen modelliert werden (vgl. Abschnitt 4.3, Abb. 4.7), und zum anderen Eigenschaften und Records, die bei der Beschreibung durch Meta-Metadaten als Ressourcen betrachtet werden (vgl. Abschnitt 4.4, Abb. 2.10). Weitere Spezialisierungen von Eigenschaften und Records, wie Beziehungen und Rollen, sind ebenfalls spezielle Ressourcen.

Das für Ressourcen außerhalb der Metadaten verwendete Konzept der Referenzierung ist auch für die aufgeführten Ressourcen, die selbst Bestandteile von KooMet sind, angebracht, da diese Ressourcen sonst an jeder Stelle, an der sie beschrieben werden, explizit wiederholt werden müssten. Durch ihre Referenzierung kann dieser Aufwand vermieden werden.

Wiederverwendung umfangreicher Beschreibungen: Werte, die in einer Eigenschaft oder als Element einer Kollektion einen Aspekt einer Ressource beschreiben, können umfangreich sein. So können sie beispielsweise Texte enthalten, die sich über viele Absätze erstrecken, aus mehreren ineinander geschachtelten Records und Kollektionen bestehen oder sogar in einer Multimedia-Datei abgelegt sein.

Werte, die durch eine eigenständige Ressource außerhalb der Metadaten repräsentiert sind, werden wiederum in Form von Referenzen auf diese Ressource dargestellt. Auch hier lässt sich das Konzept der Referenzierung verallgemeinern auf alle Werte des Modells, also auch auf diejenigen, die nur innerhalb der Metadaten selbst existieren. Eine Referenzierung von Literalen, Kollektionen und Records als Werte innerhalb von Eigenschaften und Kollektionselementen ist daher zulässig und vermeidet den Aufwand, den die wiederholte Erfassung umfangreicher Werte an verschiedenen Stellen eines Metadatenbestandes mit sich bringt.

Bezug auf Typinformationen: Um eine Flexibilität bei der Typisierung von Metadaten zu erreichen, können Typen an verschiedenen Stellen in einem Metadatenbestand deklariert werden (vgl. Abschnitt 4.2). Zumeist kommt ein Typ dabei häufiger als einmal zum Einsatz. Darüber hinaus wird beim Aufbau von Typhierarchien auf die eingeschränkten Basistypen Bezug genommen. Daher kann eine Referenzierung von Typen bei ihrer Deklaration oder bei ihrer Einschränkung in einem Unterbereichstyp den Erfassungsaufwand sowohl für Metadaten als auch für Zusatzinformationen verringern.

Grundsätzlich können alle Ressourcen, Werte und Typen in KooMet mit einem Identifikator versehen werden, auf den bei der Referenzierung Bezug genommen werden kann (vgl. Abschnitt 3.7, Konzept Erf–2). Für Attribute und Klassen, die die Semantik einer Anwendungsdomäne repräsentieren, sieht KooMet jeweils ein Label vor, über das diese Konstrukte entsprechend dieser Domäne benannt werden können. Kooperationspartner, die die Anwendungsdomäne kennen, können Attribute und Klassen dann intuitiv über dieses Label referenzieren.

Die Vergabe eindeutiger Identifikatoren und Labels lässt sich durch die Verwendung verschiedener Namensräume, wie sie z.B. für XML realisiert sind (vgl. Abschnitt 3.3), unterstützen. In diesem Fall müssen Identifikatoren nur innerhalb eines Namensraums eindeutig sein. Wird in verschiedenen Namensräumen derselbe Identifikator verwendet, kann die Eindeutigkeit über eine Qualifizierung durch den Namensraum wiederhergestellt werden. Die Verwendung von Namensräumen gewährleistet hierdurch in einem Informationsraum die gegenseitige Unabhängigkeit verschiedener Metadatenerfasser.

Die Erfassung von Identifikatoren ist für Fälle sinnvoll einsetzbar, in denen vorhergesehen werden kann, dass das entsprechende Konstrukt häufig referenziert wird. Dieses gilt für Typen, da sie an vielen Stellen deklariert werden, und auch für Ressourcen, da sie zumeist durch verschiedene Kooperationspartner in verschiedenen Kontexten beschrieben werden. Für Werte ist jedoch nur selten prognostizierbar, ob sie im Metadatenbestand wiederholt Verwendung finden können. Daher läuft die Zuordnung von Identifikatoren zu beliebigen Werten der Begrenzung des Erfassungsaufwands zuwider, da hier im Zweifelsfall überflüssige Informationen erfasst werden.

Eine mögliche Strategie für die Referenzierung von Werten ist es, zunächst nur Werte zu referenzieren, deren Wiederverwendung sicher ist. Werte ohne Identifikator können dann über Pfadausrücke referenziert werden (vgl. Abschnitt 3.7, Konzept Erf–3). Nach dem Vorbild von XPath (vgl. Abschnitt 3.3) können die Labels von Attributen in Kombination mit Platzhaltern und Prädikaten verwendet werden, um schrittweise beliebig tief in eine Metadatenstruktur hinein zu navigieren und Werte auf einer bestimmten Ebene der Struktur zu referenzieren.

Neben der Referenzierung bereits erfasster Metadaten ist die Festlegung von Default-Werten, die als gegeben angenommen werden, sofern sie innerhalb konkreter Metadaten nicht anders erfasst worden sind, eine weitere Möglichkeit, um den Erfassungsaufwand für Metadaten zu reduzieren (vgl. Abschnitt 3.7, Konzept Erf–4). Die Angabe von Default-Werten eignet sich für Metadatenbestandteile, die in direkter Verbindung zu erfassten Werten stehen, häufig verwendet werden und innerhalb der Anwendungsdomäne eine Bedeutung besitzen, aus der sich ein typischer Wert im Zusammenhang mit dem entsprechenden Metadatenbestandteil ableiten lässt. Konstrukte in KooMet, auf die diese Bedingungen zutreffen, sind sowohl Attribute als auch Typen. Daher kann ein Wert, der in einer Eigenschaft ein bestimmtes Attribut beschreibt, über dieses Attribut standardmäßig vorgegeben werden und braucht nicht explizit erfasst zu werden, falls er der Vorgabe entspricht. Gleiches gilt für Werte, die durch einen Typ, der einen Default-Wert festlegt, typisiert sind. Gibt sowohl der Typ eines Werts als auch das Attribut, das durch ihn belegt ist, einen Default-Wert vor, so gilt der Default-Wert des Attributs, da dieses einen engeren Bedeutungsausschnitt der Anwendungsdomäne umfasst, für den eher ein typischer Wert gewählt werden kann.

4.6 Zusammenfassung

Das vorliegende Kapitel führt Konzepte zur Strukturierung und Typisierung von Metadaten, zur Einordnung von Ressourcen in Anwendungsdomänen mit Hilfe von Klassifikationssystemen, zur Beschreibung von Metadaten durch Meta-Metadaten und zur Verringerung des Erfassungsaufwands für Metadaten in dem integrierenden Metadatenmodell KooMet zusammen, das sich für die Repräsentation von Metadaten für unterschiedliche Kooperationsszenarien innerhalb gemeinsamer Informationsräume eignet.

Auch wenn Ressourcen in gemeinsamen Informationsräumen durch umfangreiche und detaillierte Metadaten und Zusatzinformationen ergänzt werden, wie sie z.B. mit Hilfe von KooMet dargestellt werden können, lässt sich nicht garantieren, dass das Verständnis von der Ressource auf Seiten des Metadatenerfassers einem Kooperationspartner mit Hilfe dieser Metadaten vollständig vermittelt werden kann. Abschnitt 2.3.3 stellt hier den Einsatz toleranter Algorithmen bei der Interpretation von Metadaten als einen Lösungsansatz vor. Das folgende Kapitel 5 erläutert tolerante Algorithmen auf der Grundlage von Fuzzy-Prädikaten, die ihrerseits auf den Konzepten des Metadatenmodells KooMet aufbauen und dieses entsprechend ergänzen.

Bei KooMet handelt es sich um ein konzeptuelles Modell, das als Grundlage für konkrete Implementationsmodelle dienen kann, die auf eine Optimierung der Erfassung, Speicherung, Suche und Interpretation von Metadaten abzielen. Zwei konkrete Systeme, die jeweils auf einem anderen Kooperationsszenarium basieren und die daher jeweils nur die für sie relevanten Konzepte des integrierenden Metadatenmodells umsetzen, werden in Kapitel 6 vorgestellt.

Kapitel 5

Tolerante Selektionsalgorithmen

Auch mit Hilfe von Metadaten und Zusatzinformationen, wie sie z.B. durch das Metadatenmodell KooMet repräsentiert werden können (vgl. Kapitel 4), kann ein Akteur seinem Kooperationspartner oft nur teilweise sein individuelles Verständnis einer Ressource vermitteln. Dass dieses nicht vollständig gelingt, liegt hauptsächlich darin begründet, dass beide Partner in einem jeweils unterschiedlichen Kontext arbeiten, aus dem sich ergibt, wie sie Metadaten formulieren und interpretieren (vgl. Abschnitt 2.3.3).

Tolerante Algorithmen werden in gemeinsamen Informationsräumen eingesetzt, um bei der Interpretation Metadaten, die innerhalb eines Kontexts erfasst wurden, auf einen anderen Kontext abzubilden, damit diese dort besser verstanden werden können. Die Verfahren der Fuzzy-Logik (vgl. Abschnitt 5.1) stellen eine Grundlage zur Realisierung toleranter Algorithmen dar, indem sie es ermöglichen, bei der Interpretation eine vage Einschätzung konkreter Metadaten zu treffen und diese an einen interpretierenden Akteur entsprechend weiterzugeben.

Das Metadatenmodell KooMet erlaubt eine Realisierung toleranter Fuzzy-Algorithmen, die über Fuzzy-Prädikate an die modellierten Metadatenstrukturen angebunden werden (vgl. Abschnitt 5.2) und innerhalb verschiedener Dienste in gemeinsamen Informationsräumen für die Interpretation von Metadaten verwendet werden können. Fuzzy-Prädikate stellen somit eine weitere Ergänzung des Metadatenmodells KooMet dar, die ebenfalls auf den Anforderungen der Kooperation in gemeinsamen Informationsräumen beruht. Das vorliegende Kapitel stellt die Verwendung von Fuzzy-Prädikaten anhand der Anfragesprache KooMet-QL vor, die der Selektion von Metadaten-Records dient.

5.1 Fuzzy-Logik

Die Begriffe, die wir verwenden, um Sachverhalte zu beschreiben, sind nicht immer eindeutig und werden von verschiedenen Akteuren unterschiedlich interpretiert. So ist beispielsweise für jemanden, der nicht viel liest, ein Buch mit 200 Seiten bereits ein „dickes Buch“, während jemand, der viel liest, vielleicht erst

ein Buch mit 700 Seiten als „dickes Buch“ bezeichnen würde. Subjektive Interpretationsansätze dieser Art lassen sich durch die Methoden der Fuzzy-Logik repräsentieren und auch formalisieren. Die formalen Grundlagen, auf denen diese Methoden basieren, werden im Folgenden kurz erläutert.

5.1.1 Fuzzy-Mengen

Begriffe, die nicht eindeutig interpretierbar sind, werden in der Fuzzy-Logik als *linguistische Variablen* bezeichnet und durch Fuzzy-Mengen repräsentiert, die das Konzept der klassischen Menge erweitern.

Ein klassische Menge M kann über einer gegebenen Grundmenge X durch ihre *Zugehörigkeitsfunktion* χ_M definiert werden, die denjenigen Werten in X, die in der Menge M enthalten sind, eine 1 und denjenigen Werten, die nicht zu M gehören, eine 0 zuordnet:

$$\chi_M : X \rightarrow \{0,1\} \quad mit \quad \chi_M(x) = \begin{cases} 1 & \text{falls } x \in M \\ 0 & \text{falls } x \notin M \end{cases} \tag{5.1}$$

Die so definierte Funktion χ_M wird im Folgenden auch als *scharfe Zugehörigkeitsfunktion* bezeichnet, da sie für jeden Wert eine eindeutige Entscheidung über dessen Zugehörigkeit zu M beinhaltet.

Das Konzept der klassischen Mengen lässt sich zum Konzept der Fuzzy-Mengen verallgemeinern, die bereits 1965 von Zadeh präsentiert wurden [Zad65]. Zur Definition einer Fuzzy-Menge F wird nicht nur zwischen der Zugehörigkeit und der Nicht-Zugehörigkeit von Werten zu F unterschieden. Stattdessen wird für jeden Wert x der Grundmenge X ein Grad der Zugehörigkeit zu F angegeben. Hierzu wird die Definition 5.1 der scharfen Zugehörigkeitsfunktion χ_M erweitert auf eine unscharfe Zugehörigkeitsfunktion μ_F:

$$\mu_F : X \rightarrow [0,1] \tag{5.2}$$

$\mu_F(x)$ gibt den Grad an, zu dem ein Wert x aus X zu der Fuzzy-Menge F gehört. Dabei gilt, dass x genau dann ein Element von F ist, wenn $\mu_F(x) > 0$ gilt. Auch die Grundmenge X selbst sowie die leere Menge lassen sich als spezielle Fuzzy-Mengen über unscharfe Zugehörigkeitsfunktionen μ_X und $\mu_\emptyset$ definieren:

$$\begin{aligned} \forall x \in X: \quad \mu_X(x) &= 1 \\ \forall x \in X: \quad \mu_\emptyset(x) &= 0 \end{aligned}$$

In praktischen Anwendungen, die auf Fuzzy-Mengen basieren, kommen häufig lineare konvexe Zugehörigkeitsfunktionen zum Einsatz, die durch wenige Stützstellen parametrisiert sind und einfach ausgewertet werden können [Gra95]. Beispiele für lineare konvexe Zugehörigkeitsfunktionen sind die L-Funktion mit den Parametern a_1 und a_2, die Γ-Funktion mit den Parametern a_3 und a_4 sowie die Trapezfunktion mit den Parametern a_1, a_2, a_3 und a_4, wobei a_1, a_2, a_3 und a_4 jeweils Elemente der Grundmenge X sind:

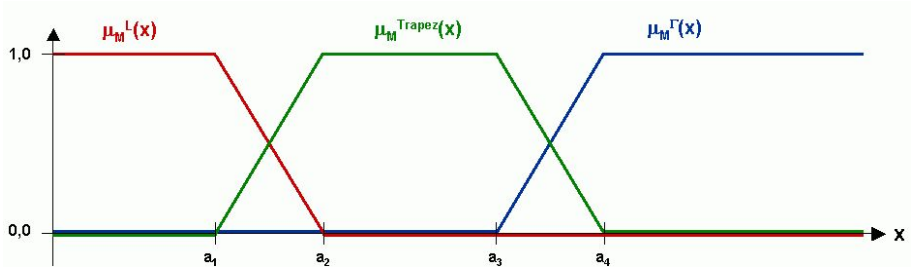

Abbildung 5.1: Die Funktionsgraphen einer L-, einer Γ- und einer Trapezfunktion

$$\chi_M^L(x) = \begin{cases} 1 & \text{falls } x < a_1 \\ \frac{(a2-x)}{(a_2-a_1)} & \text{falls } a_1 \leq x \leq a_2 \\ 0 & \text{falls } x > a_2 \end{cases} \tag{5.3}$$

$$\chi_M^\Gamma(x) = \begin{cases} 0 & \text{falls } x < a_3 \\ \frac{(x-a_3)}{(a_4-a_3)} & \text{falls } a_3 \leq x \leq a_4 \\ 1 & \text{falls } x \geq a_4 \end{cases} \tag{5.4}$$

$$\chi_M^{Trapez}(x) = \begin{cases} 0 & \text{falls } x < a_1 \\ \frac{(x-a_1)}{(a_2-a_1)} & \text{falls } a_1 \leq x \leq a_2 \\ 1 & \text{falls } a_2 \leq x \leq a_3 \\ \frac{(a_4-x)}{(a_4-a_3)} & \text{falls } a_3 \leq x \leq a_4 \\ 0 & \text{falls } x > a_4 \end{cases} \tag{5.5}$$

Abbildung 5.1 stellt die Funktionsgraphen für alle drei Arten der Zugehörigkeitsfunktion dar.

Die Stützstellen von linearen konvexen Zugehörigkeitsfunktionen müssen zunächst für jede linguistische Variable und für jede Grundmenge X individuell definiert werden. Hier bietet sich die Möglichkeit für eine Vereinheitlichung, indem man zunächst für die Grundbereiche X_i der individuellen Zugehörigkeitsfunktionen μ_i eine Maßstabstransformation auf das Intervall $[0, 1]$ vornimmt und über diesem Grundbereich dann einen Satz unscharfer Zugehörigkeitsfunktionen für *linguistische Standardterme*, wie „nicht sehr“, „mittel“ oder „sehr“ (vgl. Abb. 5.2), zur Darstellung einer Gruppe unterschiedlicher linguistischer Variablen verwendet [Bie97]. Wenn man für den anfangs aufgeführten Fall des „dicken Buchs“ beispielsweise einen Grundbereich mit Seitenzahlen zwischen 0 und 800 annimmt, diesen linear auf das Intervall [0,1] abbildet und dann die allgemeinen Zugehörigkeitsfunktionen aus Abbildung 5.2 darauf anwendet, so ergibt sich hier für ein Buch mit einer Seitenzahl von 280, dass es zu einem Grad von $0,5$ „nicht sehr dick“, zu einem Grad von $0,5$ „mittel dick“ und zu einem Grad von 0 „sehr dick“ ist. Die Maßstabstransformation von Grundmengen X_i

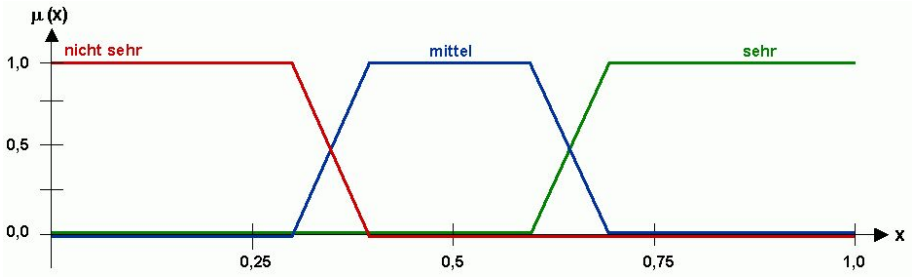

Abbildung 5.2: Zugehörigkeitsfunktionen mit allgemeiner Semantik auf einem vereinheitlichten Definitionsbereich

auf das Intervall $[0, 1]$ kann wie im obigen Beispiel linear erfolgen, wobei der Minimal- und der Maximalwert in X_i als Grenzparameter verwendet können, extreme Werte aber ausgeschlossen werden sollten [Wie02].

5.1.2 Operationen auf Fuzzy-Mengen

Ebenso wie für klassische Mengen lassen sich auch für Fuzzy-Mengen die Mengenoperationen zur Bildung des Komplements sowie von Schnittmengen und von Vereinigungsmengen definieren. Damit klassische Mengen weiterhin als Spezialfall von Fuzzy-Mengen betrachtet werden können, ist die Mindestanforderung für alle Operationen auf Fuzzy-Mengen, dass sie auch auf klassische Mengen mit scharfer Zugehörigkeitsfunktion $\mu : X \rightarrow \{0, 1\}$ anwendbar sind.

5.1.2.1 Das Komplement für Fuzzy-Mengen

Für eine Fuzzy-Menge F gibt die Komplementfunktion c für jeden Zugehörigkeitsgrad $\mu_F(x)$ einen Zugehörigkeitsgrad zur komplementären Fuzzy-Menge von F an, also ein $\mu_{\bar{F}}(x)$ an:

$$\mu_{\bar{F}} \quad = \quad c(\mu_F(x)) \tag{5.6}$$

c ist somit eine Funktion von $[0, 1]$ nach $[0, 1]$. Damit c auf klassische Mengen anwendbar ist, muss es die folgende Bedingung erfüllen:

> **Bedingung 1 (Anwendbarkeit auf klassische Mengen):**
> $c(0) = 1 \ \wedge \ c(1) = 0$.

Des weiteren wird für die Komplementabbildung gefordert, dass sie monoton fallend ist:

> **Bedingung 2:**
> $\forall a, b \in [0, 1] : a < b \Rightarrow c(a) \geq c(b)$.

Alle Funktionen $c_i : [0, 1] \rightarrow [0, 1]$, die Bedingung 1 und 2 erfüllen, sind Komplementfunktionen für Fuzzy-Mengen. Als spezielle Komplementfunktionen lassen sich darüber hinaus diejenigen Funktionen betrachten, die zusätzlich noch stetig und involutiv sind.

Zadeh selbst definiert die folgende Komplementfunktion, die zu den stetigen, involutiven Komplementfunktionen gehört:

$$c_{Zadeh}(x) \quad = \quad 1 - x \tag{5.7}$$

5.1.2.2 Die Schnittmengenbildung für Fuzzy-Mengen

Die Schnittmengenfunktion i für Fuzzy-Mengen ist eine Abbildung von $[0,1] \times [0,1]$ nach [0,1], so dass gilt:

$$\mu_{A \cap B}(x) = i(\mu_A(x), \mu_B(x)) \tag{5.8}$$

Damit i auch auf klassische Mengen anwendbar ist, muss es die folgenden Anforderungen erfüllen:

Bedingung 1 (Anwendbarkeit auf klassische Mengen):
$\forall a \in [0,1] : \; i(0,0) = 0 \; \wedge \; i(a,1) = i(1,a) = a \; \wedge \; i(1,1) = 1.$

Des weiteren müssen Schnittmengenfunktionen i für Fuzzy-Mengen kommutativ, monoton und assoziativ sein:

Bedingung 2:
$\forall a, b \in [0,1] : i(a,b) = i(b,a).$

Bedingung 3:
$\forall a, b, a', b' \in [0,1] : a \leq a' \wedge b \leq b' \; \Rightarrow \; i(a,b) \leq i(a',b').$

Bedingung 4:
$\forall a, b, c \in [0,1] : i(i(a,b),c) = i(a,i(b,c)).$

Schnittmengenfunktionen für Fuzzy-Mengen, die die Bedingungen 1-4 erfüllen, werden auch als *t-Normen* bezeichnet. Es existiert eine Reihe von t-Normen, die teilweise über Parameter an den Kontext konkreter Anwendungen anpassbar sind. Eine spezielle Untergruppe der t-Normen stellen diejenigen Funktionen dar, die über die genannten Bedingungen hinaus zusätzlich stetig und idempotent sind.

Zadeh definiert die folgende Schnittmengenfunktion:

$$i_{Zadeh}(x,y) \quad = \quad \min(x,y) \tag{5.9}$$

5.1.2.3 Die Vereinigungsmenge für Fuzzy-Mengen

Auch die Vereinigungsmengenfunktion u für Fuzzy-Mengen ist eine Funktion von $[0,1] \times [0,1]$ nach $[0,1]$. Für sie gilt:

$$\mu_{A \cup B}(x) = u(\mu_A(x), \mu_B(x)) \tag{5.10}$$

Damit eine Vereinigungsmengenfunktion auf den Spezialfall der klassischen Mengen anwendbar ist, muss sie die folgende Eigenschaft besitzen:

Bedingung 1 (Anwendbarkeit auf klassische Mengen):
$\forall a \in [0,1] : \; u(0,0) = 0 \; \wedge \; u(a,0) = u(0,a) = a \; \wedge \; u(1,1) = 1$

Funktionen, die diese Bedingung erfüllen und darüber hinaus monton, kommutativ und assoziativ sind, werden als *t-Conormen* oder auch als *s-Normen* bezeichnet. Auch t-Conormen besitzen eine Unterklasse, die Funktionen umfasst, die zusätzlich stetig und idempotent sind.

Die von Zadeh definierte t-Conorm verwendet das Maximum der Zugehörigkeitsgrade:

$$u_{Zadeh}(x, y) \quad = \quad \max(x, y) \tag{5.11}$$

5.1.2.4 Kompensatorische Operatoren

Sowohl t-Normen als auch t-Conormen besitzen die Eigenschaft, dass sie für zwei Werte x und y niemals Ergebnisse zwischen deren Minimum und deren Maximum liefern. Es gilt also:

$$\forall x, y \in [0, 1] : i(x, y) \leq \min(x, y) \leq \max(x, y) \leq u(x, y) \tag{5.12}$$

Diese Eigenschaft reflektiert zwar die mathematisch logische Bedeutung der Bildung von Schnittmengen und Vereingungsmengen, Menschen gehen bei der Aggregation unscharfer Eigenschaften jedoch oft anders vor. Hier lässt sich beobachten, dass eine hohe individuelle Bewertung der einen Eigenschaft die niedrigere Bewertung der anderen Eigenschaft kompensiert und umgekehrt. Wenn man beispielsweise einschätzen müsste, zu welchem Grad ein Buch „dick und preiswert“ ist, das zu einem Grad von $0,2$ als „dick“ und zu einem Grad von $0,9$ als „preiswert“ eingeschätzt wird, so kann der gute Preis die Einschätzung so beeinflussen, dass sie für die Eigenschaft „dick und preiswert“ einen höheren Grad ergibt als $0,2$, wie er maximal durch die Anwendung einer t-Norm erreicht werden kann.

Zur Modellierung derartiger Kompensationseffekte werden in der Fuzzy-Logik verschiedene *mittelnde Operatoren* m verwendet, für die jeweils gilt, dass sie monoton und kommutativ sind und darüber hinaus die folgende Forderung erfüllen:

$$\min(x, y) \leq m(x, y) \leq \max(x, y) \tag{5.13}$$

Ein Beispiel für kompensatorische Operatoren sind die geordneten, gewichteten mittelnden Operatoren *(ordered weighted averaging aggregation operators, OWA)* [Yag88]. Sie aggregieren n Zugehörigkeitsgrade, indem sie jedem von ihnen eine Gewichtung zuweisen, über die sein Einfluss auf das Ergebnis der Aggregation gesteuert wird. Formal ist ein OWA definiert als eine Funktion $F : [0, 1]^n \rightarrow [0, 1]$ mit einem dazugehörigen n-dimensionalen Vektor

$$\begin{array}{lll} w = (w_1, w_2, \ldots, w_n) & \text{mit} & \\ w_i \in [0, 1], 1 \leq i \leq n & \text{und} & \sum_{i=1}^{n} w_i = 1 \end{array}$$

F berechnet sich dann wie folgt:

$$F(a_1, \ldots, a_n) = \sum_{j=1}^{n} w_j b_j$$

wobei es sich bei b_j jeweils um das j-größte Element innerhalb der Multimenge $< a_1, \ldots, a_n >$ handelt.

Indem die Zugehörigkeitsgrade a_j vor der Aggregation geordnet werden, lässt sich über die Gewichtungen w_i detailiert steuern, wie stark Zugehörigkeitsgrade mit einer bestimmten, relativ zu den weiteren aggregierten Werten betrachteten Größe in die Aggregation eingebracht werden. Hierdurch lassen sich neben den vorgestellten t-Normen und t-Conormen auch beliebige kompensatorische Operatoren über OWA darstellen.

Die Kompensation kann auch direkt bei der Ausführung einer Aggregation erfolgen, indem, beispielsweise durch einen Benutzer, der eine Fuzzy-Anfrage stellt, den aggregierten Fuzzy-Termen jeweils eine bestimmte Relevanz zugeordnet wird. Auf diese Weise können Benutzer explizit ihre individuelle Einschätzung möglicher Kompensationseffekte zum Ausdruck bringen. Die angegebenen Gewichtungen fließen dann für jeden Einzelfall in die Auswertung von Aggregationsoperationen ein. Bei einer Kombination dieser Vorgehensweise mit OWA können die Benutzergewichtungen mit den internen Gewichtungen des Aggregationsoperators verrechnet werden [Yag97]. Der konkrete Einsatz von Gewichtungen bei der Interpretation von Metadaten wird in den Abschnitten 5.2.3 und 5.2.4 näher beschrieben.

5.1.2.5 Auswahl von Operatoren für Fuzzy-Mengen

Wie oben bereits erwähnt gibt es neben den von Zadeh definierten Operatoren für Fuzzy-Mengen eine Vielzahl weiterer Operatoren. Die Auswahl eines geeigneten Satzes von Operatoren für eine konkrete Anwendung ist dabei von einer Reihe verschiedener Kriterien abhängig.

Zunächst haben verschiedene Operatoren unterschiedliche mathematische Eigenschaften. So lässt sich zeigen, dass kein Satz von Fuzzy-Operatoren alle Gesetze der Booleschen Algebra erfüllen kann [Bie97]. Zadehs Operatoren kommen der Booleschen Algebra sehr nah, erfüllen aber nicht den Satz vom ausgeschlossenen Dritten und den Satz vom Widerspruch. Sie genügen jedoch als einzige dem Distributivgesetz. Eine häufig geforderdte mathematische Eigenschaft von Fuzzy-Operatoren ist ihre Assoziativität. Da nur durch sie eine eindeutige Auswertung von Aggregationen mit einer variablen Anzahl von Argumenten gewährleistet werden kann, spielt sie auch bei der Interpretation von Metadaten mit Hilfe von Fuzzy-Algorithmen eine wichtige Rolle.

Neben den mathematischen Kriterien finden auch pragmatische Gesichtspunkte bei der Auswahl geeigneter Fuzzy-Operatoren Berücksichtigung. Hier fließt beispielsweise die Anpassbarkeit von Operatoren an unterschiedliche Anwendungskontexte über Parameter ein [Zim01]. Weiterhin muss betrachtet werden, ob eine Operator-Menge den Kontext der Anwendung inhaltlich angemessen repräsentiert, so dass beispielsweise eine Suche mit Hilfe von Fuzzy-Operatoren keine Ergebnisse produziert, mit denen der Anfragende nicht rechnet und die daher für ihn nicht verwertbar sind.

5.1.3 Fuzzy-Relationen

Die Elemente der bisher beschriebenen Fuzzy-Mengen sind einzelne Werte. Der Begriff der Fuzzy-Menge lässt sich darüber hinaus jedoch auch ausweiten auf den Begriff der Fuzzy-Relation, die den Grad beschreibt, in dem mehrere Werte in einem bestimmten Zusammenhang stehen. Fuzzy-Relationen können bei der Interpretation von Metadaten sowohl als Grundlage für Vergleiche mehrerer Werte dienen (vgl. Abschnitt 5.2) als auch bei der Auswertung von Relationen zwischen Ressourcen (vgl. Abschnitt 4.1) Verwendung finden.

Da eine klassische n-stellige Relation R eine Teilmenge des kartesischen Produkts $A_1 \times A_2 \times \ldots \times A_n$ ist, lässt sie sich wiederum über eine scharfe Zugehörigkeitsfunktion definieren:

$$\chi_R : A_1 \times A_2 \times \ldots \times A_n \ \rightarrow \{0,1\} \quad mit$$

$$\chi_R(a_1, a_2, \ldots, a_n) = \begin{cases} 1 & \text{falls } (a_1, a_2, \ldots, a_n) \in R \\ 0 & \text{falls } (a_1, a_2, \ldots, a_n) \notin R \end{cases} \tag{5.14}$$

Analog wird eine Fuzzy-Relation R mit Hilfe einer unscharfen Zugehörigkeitsfunktion μ definiert:

$$\mu_R : A_1 \times A_2 \times \ldots \times A_n \ \rightarrow [0,1] \tag{5.15}$$

Obwohl sich in der Literatur auch Ansätze finden, Fuzzy-Mengen über Fuzzy-Relationen zu verbinden [Uma83], sei hier wie allgemein üblich angenommen, dass es sich bei den verbundenen Mengen $A_1, \ldots, A_n$ nicht um Fuzzy-Mengen sondern um klassische Mengen handelt. Dieses entspricht auch der Vorgehensweise bei der toleranten Interpretation von Metadaten, wobei die Metadaten selbst nicht als vage angenommen werden, sondern nur durch eine Interpretation auf der Grundlage von Fuzzy-Methoden in einen fremden Anwendungskontext eingeordnet werden.

Die durch Fuzzy-Relationen verbundenen Mengen können sowohl kontinuierlich als auch diskret sein. Für Fuzzy-Relationen auf kontinuierlichen Definitionsbereichen können entsprechende Zugehörigkeitsfunktionen zumeist direkt über Formeln dargestellt werden. Als Beispiel seien die folgenden unscharfen Vergleichsrelationen für reelle Zahlen angeführt [Bie97]:

$$\mu_{viel\ größer}(x,y) = \begin{cases} 0 & \text{falls } x \leq y \\ (1 + c(x-y)^{-2})^{-1} & \text{falls } x > y \end{cases} \tag{5.16}$$

$$\mu_{viel\ kleiner}(x,y) = \begin{cases} (1 + c(y-x)^{-2})^{-1} & \text{falls } x < y \\ 0 & \text{falls } x \geq y \end{cases} \tag{5.17}$$

$$\mu_{ungefähr\ gleich}(x,y) \ = \ e^{-c|x-y|} \tag{5.18}$$

Die Zugehörigkeitsfunktionen für binäre Fuzzy-Relationen über diskreten Wertebereichen werden häufig in Form von Matrizen dargestellt, die die entsprechenden Zugehörigkeitsgrade enthalten. Für n-stellige Fuzzy-Relationen lassen

sich diese Matrizen auf n-dimensionale Zugehörigkeitsfelder ausweiten. Da diese recht schnell unübersichtlich werden und für Fuzzy-Relationen, die nicht viele Tupel umfassen, viele 0en enthalten, greift man häufig auch auf die aus dem relationalen Datenmodell (vgl. Abschnitt 3.1) geläufige Darstellung von Relationen in Tabellenform zurück, wobei jede Zeile $(a_1, \ldots, a_n)$ der Tabelle durch den entsprechenden Zugehörigkeitswert $\mu_R(a_1, \ldots, a_n)$ für das durch die Zeile repräsentierte Tupel ergänzt wird.

Die für klassische Relationen bekannten Eigenschaften der Reflexivität, der Irreflexivität, der Symmetrie, der Asymmetrie, der Antisymmetrie und der Transitivität lassen sich auch für Fuzzy-Relationen definieren. Hierbei gibt es in einigen Fällen mehrere unterschiedliche Definitionen für eine Eigenschaft, die ihren Ursprung in der Existenz verschiedener t-Normen und t-Conormen (vgl. Abschnitt 5.1.2) haben. Auf Grundlage der Eigenschaften lassen sich bestimmte Arten von Fuzzy-Relationen, wie die Ähnlichkeitsrelation analog zur klassischen Äquivalenzrelation und die unscharfe Ordnungsrelation analog zur klassischen Ordnungsrelation, definieren. [Bie97]

Neben Eigenschaften sind auch verschiedene Operationen auf klassischen Relationen für Fuzzy-Relationen umgesetzt worden. So existieren beispielsweise Fuzzy-Äquivalente für die aus den Anfragesprachen relationaler Datenbanken bekannten Operationen der *Projektion* und der *Selektion* (vgl. hierzu auch Abschnitt 3.1). Eine weitere, häufig verwendete Operation ist die *Verkettung* zweier unscharfer Relationen $R_1 \circ R_2$, die ermittelt, in welchem Grad zwei Werte über zwei Relationen und einen dritten mittleren Wert verbunden sind. Die Verkettung wird allgemein durch die folgende Zugehörigkeitsfunktion definiert:

$$\mu_{R_1 \circ R_2} \quad = \quad \sup_y \; t(\mu_{R_1}(x, y), \mu_{R_2}(y, z)) \tag{5.19}$$

Die Verkettung wird bei der toleranten Interpretation von Metadaten insbesondere innerhalb von Fuzzy-Prädikaten eingesetzt, die die Einordnung von Ressourcen in Klassifikationssysteme auswerten (vgl. Abschnitt 5.2 sowie [Büc99]).

5.2 Fuzzy-Prädikate

Metadaten in gemeinsamen Informationsräumen werden zumeist in unterschiedlichen Arbeitskontexten erfasst und interpretiert. So bleiben Unstimmigkeiten zwischen dem erfassenden und dem interpretierenden Akteur hinsichtlich ihrer Auffassung über eine Ressource bestehen, die je nach Umfang und Qualität der für die Ressource bereitgestellten Metadaten mehr oder weniger gravierend sein können. Um diese Unstimmigkeiten auszugleichen und daneben ebenfalls existierende Gemeinsamkeiten für die Zusammenarbeit auf der Grundlage von Ressourcen und Metadaten zu nutzen, werden tolerante Algorithmen eingesetzt.

Die tolerante Interpretation existierender Metadaten ist eine Ausprägung dieser toleranten Algorithmen. Hier wird u.a. auf Methoden zurückgegriffen, die auf den oben erläuterten theoretischen Grundlagen der Fuzzy-Logik (vgl. Abschnitt 5.1) beruhen. Im Rahmen des integrierenden Metadatenmodells KoMet (vgl. Kapitel 4) werden diese Methoden in Form von Fuzzy-Prädikaten realisiert und in das Modell eingebunden. Sie können dann verwendet werden,

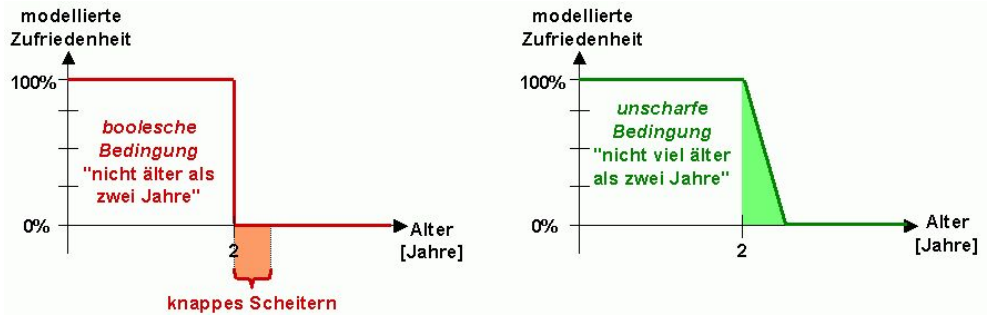

Abbildung 5.3: Vergleich der Modellierung von Anforderungen an Eigenschaften mittels boolescher Prädikate und mittels Fuzzy-Prädikate am Beispiel der Eigenschaft Alter

um unscharfe Bedingungen zu formulieren und auszuwerten, die von interpretierenden Akteuren an Metadaten-Records und damit indirekt an die durch sie beschriebenen Ressourcen gestellt werden.

Unscharfe Bedingungen können für verschiedene Anwendungen innerhalb gemeinsamer Informationsräume genutzt werden. Die Formulierung und Auswertung von Suchanfragen, die über einen Metadatenbestand für eine Aufgabe geeignete Ressourcen und ihre Beschreibungen ermitteln, ist ein Anwendungsbeispiel für die Verwendung unscharfer Bedingungen, das häufig auftritt.

5.2.1 Realisierung unscharfer Bedingungen für Metadaten

Die boolesche Interpretation von Metadaten schlägt nicht selten deshalb fehl, weil sie die unscharfen Anforderungen, die ein Akteur an die Eigenschaften einer Ressource stellt, nicht ausreichend widerspiegelt. So ist beispielsweise ein Akteur, der in einem Informationsraum nach Publikationen sucht, die nicht älter als zwei Jahre sind, sicher auch mit einer Publikation zufrieden, deren Veröffentlichung zwei Jahre und zwei Wochen zurückliegt, die aber dafür genau das für ihn relevante Themengebiet abdeckt. Die Formulierung *unscharfer Bedingungen*, wie „nicht viel älter als zwei Jahre“ verhindern das *knappe Scheitern (near miss)* bei der Auswertung von Metadaten (vgl. Abb. 5.3) und bilden dadurch eine Grundlage für deren tolerante Interpretation.

Durch die Möglichkeit, unscharfe Bedingungen für die Eigenschaften von Ressourcen zu formulieren, wird zum einen die fehlende Übersicht eines interpretierenden Akteurs über die in einem Informationsraum befindlichen Ressourcen und Metadaten kompensiert. So wird im obigen Beispiel dem Anfragenden, der durch eine boolesche Anfrage nicht in einem Schritt ermitteln kann, dass eine passende, zwei Jahre und zwei Wochen alte Publikation existiert, zu einem Suchergebnis verholfen, das diese Veröffentlichung umfasst. Zum anderen wird ein mangelnder Austausch an Zusatzinformationen zwischen erfassenden und interpretierenden Akteuren ausgeglichen. So kann die Unkenntnis des interpretierenden Akteurs über die Art und Weise, in der der Erfasser die Metadaten formuliert hat, eine boolesche Interpretation scheitern lassen. Beispielsweise

können boolesche Anfragen fehlschlagen, wenn nicht bekannt ist, dass in den Metadaten eines Informationsraums keine deutschen Umlaute verwendet werden, Vornamen von Autoren nach dem ersten Buchstaben abgekürzt werden, Briefe in der Klasse Sekretariat eingeordnet sind o.ä.

Für Metadaten, die entsprechend dem Metadatenmodell KooMet (vgl. Kapitel 4) aufgebaut sind, werden unscharfe Bedingungen jeweils für ein bestimmtes Attribut formuliert und ausgewertet (vgl. Abb. 5.4). Für die Auswertung eines Records bezüglich einer unscharfen Bedingung wird diese mit einem Fuzzy-Prädikat verknüpft. Für den oder die Werte x des Attributs innerhalb des Records ermittelt das Fuzzy-Prädikat ihren Zugehörigkeitsgrad $\mu(x)$ zu einer Fuzzy-Menge (vgl. Abschnitt 5.1.1), falls die unscharfe Bedingung sich allein auf die Attributwerte x bezieht wie die Bedingung $istGroß(x)$. Beinhaltet die unscharfe Bedingung hingegen weitere Argumente wie die Bedingung $liegtZwischen(x, a_1, a_2)$ so wird im entsprechenden Fuzzy-Prädikat der Zugehörigkeitsgrad $\mu(x, a_1, ..., a_{n-1})$ zu einer n-stelligen Fuzzy-Relation (vgl. Abschnitt 5.1.3) ermittelt. Das Fuzzy-Prädikat realisiert auf diese Weise die unscharfe Semantik der Bedingung.

Konkret werden bei der Formulierung einer unscharfen Bedingung demnach die folgenden Schritte durchlaufen:

- Festlegung des Attributs, für dessen Eigenschaften die unscharfe Bedingung zutreffen soll
- Festlegung der Semantik der unscharfen Bedingung durch die Auswahl eines entsprechenden Fuzzy-Prädikats
- u.U. Festlegung von weiteren vom Fuzzy-Prädikat benötigten Argumenten
- u.U. Negation der unscharfen Bedingung, so dass sie das Komplement (vgl. Abschnitt 5.1.2) des durch das Fuzzy-Prädikat ermittelten Zugehörigkeitsgrades zum Ergebnis hat.

Ist eine unscharfe Bedingung einmal auf diese Weise formuliert, so kann sie auf einen oder beliebig viele Metadaten-Records angewendet werden, indem ihre Auswertungsmethode resultat für jeden auszuwertenden Record rec erneut aufgerufen wird. Die Auswertungsmethode gibt sowohl den Record als auch das Attribut att, für das die unscharfe Bedingung formuliert wurde, und etwaige weitere Argumente args an die Auswertungsmethode resultat des Fuzzy-Prädikats p weiter. Der durch das Fuzzy-Prädikat p ermittelte Erfüllungsgrad der unscharfen Bedingung wird in deren Auswertungsmethode ggf. mit Hilfe des Operators op negiert und als Ergebnis der Auswertung der Bedingung für den Record rec zurückgegeben:

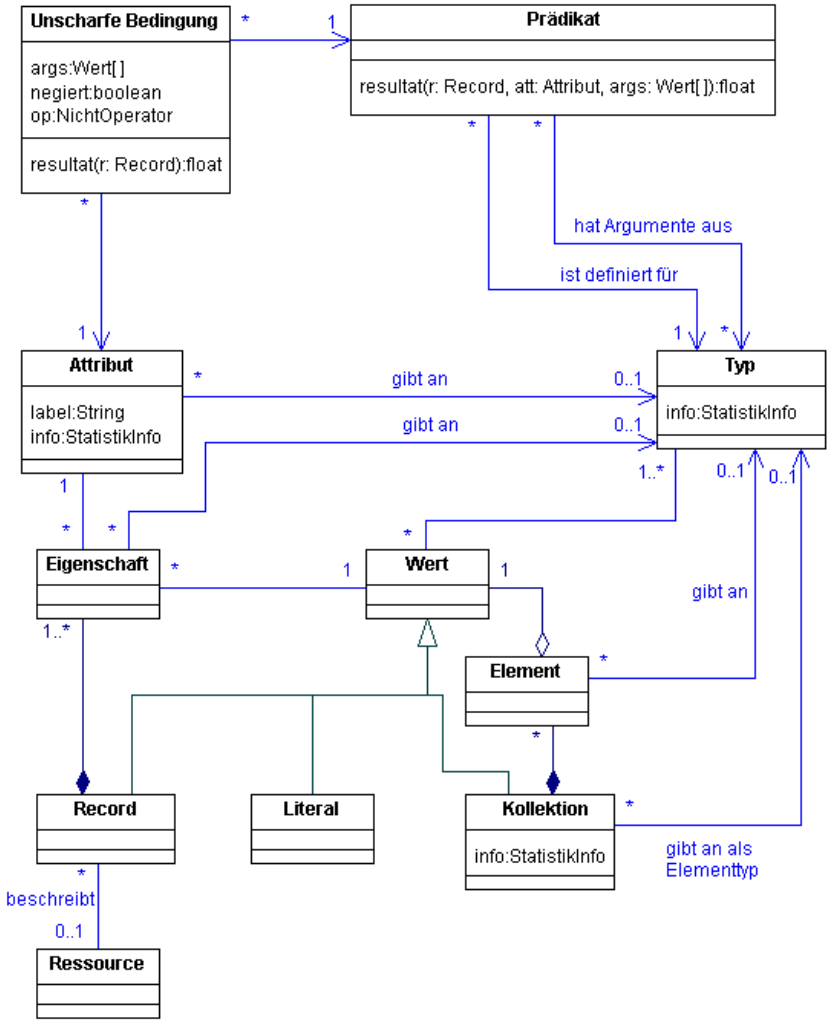

Abbildung 5.4: Fuzzy-Prädikate zur Formulierung von unscharfen Bedingungen für Metadaten

```
class Unscharfe_Bedingung{
  Attribut att; Praedikat p; Wert[ ] args;
  boolean negiert; NichtOperator op;

  float resultat(Record rec){
    return negiert ?
    op.nicht(p.resultat(rec,att,args)) :
    p.resultat(rec,att,args);
  }
}
```

Da das Metadatenmodell KooMet die Erfassung mehrerer Records für eine einzige Ressource unterstützt, entspricht die Auswertung einer unscharfen Be-

dingung für einen Record nicht automatisch der Auswertung dieser Bedingung für die Ressource selbst. Um eine unscharfe Bedingung für eine Ressource auszuwerten, müssen die Auswertungsergebnisse aller sie beschreibenden Records aggregiert werden. Dabei wird hier davon ausgegangen, dass die Ressource innerhalb des Metadatenbestandes eindeutig benannt ist und daher alle relevanten Records dort aufgefunden werden können. Die für die unscharfe Bedingung vorgesehene Semantik bestimmt die Wahl des Aggregationsoperators, mit dem die Auswertungsergebnisse der Records verknüpft werden. Hier sind t-Normen, t-Conormen und kompensatorische Operatoren (vgl. Abschnitt 5.1.2) gleichermaßen einsetzbar. Der folgende Programmauszug zeigt nun eine zweite, überladene Auswertungsmethode resultat, die den Erfüllungsgrad einer unscharfen Bedingung für eine Ressource res ermittelt. Zur Vereinfachung wird hier davon ausgegangen, dass der verwendete Aggregationsoperator das Assoziativgesetz erfüllt:

```
float resultat(Ressource res){
  Iterator records = res.holeRecords();
  float ergebnis = -1.0f;
  while(records.hasNext()){
    Record rec = (Record)records.next();
    ergebnis = (ergebnis < 0.f) ?
      resultat(rec) :
      aggregiere(resultat(rec),ergebnis);
  }
  return (ergebnis < 0.f) ? 0.f : ergebnis;
}
```

Jedes Fuzzy-Prädikat ist für Werte eines bestimmten Typs definiert und gibt auch für die weiteren Argumente jeweils einen Typ vor (vgl. Abb. 5.4). Da die Auswertung des Fuzzy-Prädikats auf der Annahme basiert, dass die übergebenen Werte eine ihrem Typ entsprechende Repräsentation und die damit verbundene Semantik besitzen, kann das Prädikat nur dann korrekt arbeiten, wenn sichergestellt ist, dass der eigentlichen Auswertung ausschließlich Werte der vorgegebenen Typen zugrundeliegen. Dementsprechend darf eine unscharfe Bedingung ein Fuzzy-Prädikat nur mit einem Attribut verbinden, das entweder für seine Werte denselben Typ deklariert, für den auch das Fuzzy-Prädikat definiert ist, oder mit einem Attribut, das beliebige Werte zulässt, die potentiell zum Fuzzy-Prädikat passen könnten:

```
context Kriterium inv:
  self.attribut.typ.notEmpty() implies
  self.attribut.typ = self.praedikat.istDefiniertFuer
```

Während für Attribute, die einen einheitlichen Typ deklarieren, sowie für die zusätzlichen Argumente bereits bei der Formulierung und somit bei der Initialisierung der unscharfen Bedingung sichergestellt werden kann, dass ihr Typ zum Fuzzy-Prädikat der Bedingung passt, können Attribute, die keinen Typ deklarieren, in Records mit Werten beliebigen Typs verbunden sein (vgl. Abschnitt

4.2). In diesem Fall muss in der Auswertungsmethode des Prädikats für jede das untypisierte Attribut betreffende Eigenschaft des übergebenen Records im Einzelnen ein Typ ermittelt und mit dem durch das Prädikat vorgegebenen Typ verglichen werden. Treten hier Typfehler auf, so schlagen sich diese im Ergebnis der Auswertungsmethode nieder, z.B. indem der Fehler als Nichterfüllung der unscharfen Bedingung gewertet und der Wert 0.0 zurückgegeben wird.

KooMet nimmt jedoch nicht nur über die wahlweise Typisierung von Attributen und Eigenschaften Einfluss auf die Auswertung von Fuzzy-Prädikaten. Auch die Tatsache, dass in einem einzigen Metadaten-Record ein und dasselbe Attribut mehrfach in Eigenschaften mit Werten belegt werden kann (vgl. Abschnitt 4.1), muss in der Auswertungsmethode von Fuzzy-Prädikaten berücksichtigt werden. Zu diesem Zweck werden bei der Auswertung im Fuzzy-Prädikat alle Eigenschaften des übergebenen Records durchlaufen. Wenn in einer dieser Eigenschaften das betreffende Attribut mit einem Wert belegt ist, wird für diesen Wert jeweils ein Zugehörigkeitsgrad ermittelt, bevor alle ermittelten Zugehörigkeitsgrade zu einem Gesamtergebnis für den übergebenen Record aggregiert werden. Die Wahl des Aggregationsoperators hängt auch hier von der Semantik des Fuzzy-Prädikats ab.

Zur Ermittlung eines Zugehörigkeitsgrads benötigen die Auswertungsmethoden von Fuzzy-Prädikaten oft *statistische Informationen* bezüglich des ihnen übergebenen Attributs, wie den größten, den kleinsten oder den durchschnittlichen Wert über alle Eigenschaften dieses Attributs, um daraus beispielsweise die entsprechenden Stützstellen für lineare konvexe Zugehörigkeitsfunktionen (vgl. Abschnitt 5.1) zu ermitteln. Ein Attribut verwaltet daher die benötigten statistischen Informationen (vgl. Abb. 5.4) und aktualisiert sie, wenn neue das Attribut betreffende Eigenschaften erfasst werden, so dass das Fuzzy-Prädikat bei der Auswertung eines Records auf diese Informationen zugreifen kann.

Unter Berücksichtigung individuell typisierter Eigenschaften, mehrfach auftretender Attribute sowie der Verwendung von statistischen Informationen zur Berechnung von Zugehörigkeitsgraden vollzieht sich die Auswertung innerhalb von Fuzzy-Prädikaten nach einem einheitlichen Muster, das in der folgenden Auswertungsmethode resultat dargestellt ist:

```
class Praedikat {

  private float z(Wert x, Wert[] args, StatistikInfo si){
    /* Berechne Zugehoerigkeitsfunktion einer
       Fuzzy-Menge oder Fuzzy-Relation */
    ...
  }

  public float resultat(Record rec, Attribut att, Wert[] args){
    Iterator eigenschaften = rec.holeEigenschaften();
    float ergebnis = -1.f;
    while(eigenschaften.hasNext()){
      Eigenschaft e = (Eigenschaft)eigenschaften.next();
```

```
      if(e.holeAttribut() == att){
        if(e.deklariertTyp()){
          if(e.holeTyp() == att.holeTyp())
            ergebnis = (ergebnis < 0.f) ?
              z(e.holeWert, args, att.holeStatistik()) :
              aggregiere(z(e.holeWert, args, att.holeStatistik()),
                         ergebnis);
        }
        else //kein Typ oder falscher Typ
          ergebnis = (ergebnis < 0.f) ?
            0.f :
            aggregiere(0.f, ergebnis);
      } // else: keine Eigenschaft fuer att -> keine Auswertung
    }
    return (ergebnis < 0.f) ? 0.f : ergebnis;
  }
}
```

Ein Attributwert in einem Record, für den über ein Fuzzy-Prädikat ein Erfüllungsgrad ermittelt wird, kann zu einem beliebigen Typ gehören, gemäß dem Metadatenmodell KooMet also auch ein komplexer Wert in Form einer Kollektion oder eines geschachtelten Records sein. Für Kollektionen und Records ermittelt ein Fuzzy-Prädikat `p` häufig, ob ein anderes Fuzzy-Prädikat `q` von den Elementen einer Kollektion bzw. von bestimmten Werten in einem Record erfüllt wird. Um dieses zu ermöglichen, muss weiter zwischen einfachen Fuzzy-Prädikaten und Fuzzy-Prädikaten, die zur Auswertung ein weiteres Prädikat, eben das Prädikat q, als Parameter benötigen, unterschieden werden (vgl. Abb. 5.5). Prädikate der zweiten Art seien hier als *Fuzzy-Prädikate höherer Ordnung* bezeichnet. Bei dem Prädikat `q`, das der Auswertungsmethode eines Fuzzy-Prädikats höherer Ordnung übergeben wird, kann es sich sowohl um ein einfaches als auch um ein weiteres Fuzzy-Prädikat höherer Ordnung handeln.

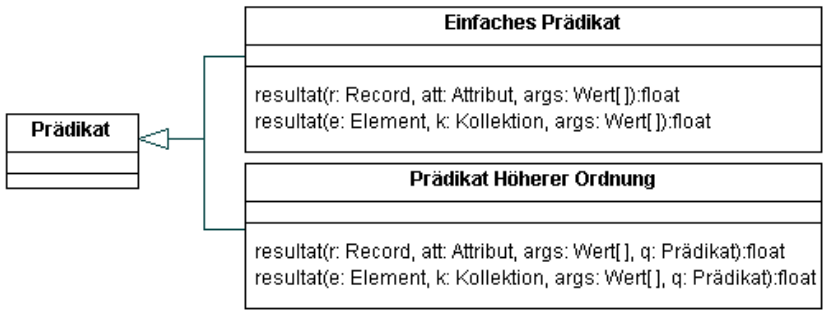

Abbildung 5.5: Spezialisierung von Fuzzy-Prädikaten in einfache Prädikate und Prädikate höherer Ordnung

Der folgende Programmauszug zeigt die interne Zugehörigkeitsfunktion `z` eines Prädikats höherer Ordnung, das für einen Kollektionswert `k` bestimmt, zu welchem Grad das einfache Prädikat `q` für alle Elemente dieser Kollektion gilt. Auch hier wird analog zur Auswertung von Attributwerten angenommen, dass eine Typüberprüfung direkt bei der Auswertung nur dann stattfindet, wenn der Typ für die Elemente der Kollektion einzeln festgelegt ist, während Typfehler bei Typen, die für die gesamte Kollektion deklariert sind, bereits im voraus abgefangen werden:

```
private float z(Kollektion k,Wert[] args,Einfaches_Praedikat q){

  Iterator elemente = k.holeElemente();
  float ergebnis = -1.f;
  while(elemente.hasNext()){
    Element e = (Element)elemente.next();
    if(e.deklariertTyp()){
      if(e.holeTyp() == q.holeTyp())
        ergebnis = (ergebnis < 0.f) ?
          q(e, k, args) :
          schnitt(ergebnis, q(e, k, args));
    }
    else
      ergebnis = (ergebnis < 0.f) ?
        0.f :
        schnitt(ergebnis, 0.f);
  }
  return (ergebnis < 0.f) ? 0.f : ergebnis;
}
```

Wie das obige Beispiel bereits zeigt, muss ein Fuzzy-Prädikat `q`, das in der Auswertungsmethode eines Prädikats höherer Ordnung aufgerufen wird, seinerseits in der Lage sein, Werte nicht nur im Kontext von Records sondern auch im Kontext von Kollektionen auszuwerten. Daher wird die Auswertungmethode resultat sowohl für einfache als auch für Prädikate höherer Ordnung durch eine zweite Auswertungsmethode überladen, der anstelle eines Records und eines Attributs ein Kollektionselement und die Kollektion, in deren Kontext dieses Element erscheint, als Parameter übergeben werden können (vgl. Abb. 5.5).

Auch für die Auswertung von Fuzzy-Prädikaten für Kollektionen können u.U. statistische Informationen benötigt werden. Als Beispiel sei hier ein Prädikat angeführt, das ermittelt, ob in einer Kollektion ein Element sehr viel größer ist als der Rest der Elemente. Da bisher statistische Informationen nur für die Werte von Attributen bereit stehen, wird das Modell an dieser Stelle erweitert, indem statistische Informationen je nach der intendierten Semantik auch für alle Elemente einer Kollektion oder für alle Werte eines Typs verwaltet werden können (vgl. hierzu auch Abb. 5.4).

Prädikat	**Typ der Argumente**
x ist so groß wie möglich; x ist [sehr|nicht sehr] groß; x ist so klein wie möglich; x ist [sehr|nicht sehr] klein; x ist durchschnittlich	
x ist genau a_1; x ist ungefähr a_1; x ist [strikt|sehr viel|nicht viel] größer als a_1; x ist [strikt|sehr viel|nicht viel] kleiner als a_1	numerischer Typ
x ist [genau|ungefähr] zwischen a_1 und a_2	numerischer Typ, numerischer Typ

Tabelle 5.1: Beispiele für numerische Fuzzy-Prädikate

5.2.2 Fuzzy-Prädikate im Metadatenmodell KooMet

Das Metadatenmodell KooMet sieht mit Literalen, Kollektionen, geschachtelten Records und Referenzen auf andere Ressourcen verschiedene Arten von Werten vor. Im Folgenden werden verschiedene Fuzzy-Prädikate aufgeführt und erläutert, die zur Auswertung jeweils einer dieser Arten von Werten geeignet sind. Die Darstellung jedes Prädikats erfolgt dabei für einen Wert x, bei dem es sich entsprechend den beiden möglichen Aufrufkontexten eines Fuzzy-Prädikats sowohl um einen Attributwert in einem Record als auch um den Wert eines Kollektionselements handeln kann (vgl. Abschnitt 5.2.1).

Der Satz der vorgestellten Prädikate besitzt dabei weder einen Anspruch auf Vollständigkeit noch muss er von jeder kooperativen Metadatenanwendung komplett implementiert werden. Stattdessen sollte er in Umfang und Ausprägung dem jeweiligen Kooperationskontext angepasst sein.

5.2.2.1 Fuzzy-Prädikate für Literale

Einfache Typen (vgl. Abschnitt 4.2.1) fassen Literale zusammen, die innerhalb von Metadaten in Form von nicht weiter unterteilten Zeichenketten repräsentiert werden. Diese Zeichenketten können beliebig sein, sie können aber auch durch syntaktische Vorgaben eingeschränkt werden. Numerische Typen enthalten Zeichenketten, die Zahlen und unter Umständen dazugehörige Maßeinheiten repräsentieren. Darüber hinaus lassen sich weitere spezielle Typen wie für Personennamen definieren. Der folgende Überblick beschränkt sich jedoch auf Fuzzy-Prädikate für numerische Werte und allgemeine Zeichenketten.

Numerische Prädikate machen zumeist von der Ordnung innerhalb numerischer Typen Gebrauch. Eine Reihe von ihnen ist in Tabelle 5.1 zusammengestellt, gemeinsam mit den Typen, aus denen die weiteren, zur Auswertung des Prädikats benötigten Argumente $a_1, \ldots, a_{n-1}$ stammen müssen. Für numerische Prädikate handelt es sich dabei durchweg um Vergleichswerte, so dass alle Argumente ebenso wie der tatsächlich untersuchte Wert x stets einem numerischen Typ angehören müssen.

Prädikat	**Typ der Argumente**
x ist lang|kurz	
x ist nicht [viel] kürzer|länger als a_1 Wörter|Zeichen	numerischer Typ
x enthält [häufig] (das Wort a_1)| (die Passage a_1)|(ein verwandtes Wort zu a_1)| (eine Passage, die mit a_1 beginnt); x ist ähnlich zu a_1; x klingt wie a_1	Zeichenkette
x enthält a_1 (in der Nähe)|(gefolgt) von a_2	Zeichenkette, Zeichenkette
x enthält a_1 und a_2 getrennt durch [mindestens|höchstens] a_3 Wörter	Zeichenkette, Zeichenkette, numerischer Typ
x enthält eine Passage der Länge a_1, die die Wörter $a_2, \ldots, a_{n-1}$ beinhaltet	numerischer Typ, Zeichenkette[ ]

Tabelle 5.2: Beispiele für Fuzzy-Prädikate auf Zeichenketten

Numerische Werte treten häufig in Verbindung mit Maßeinheiten auf. Durch die Typisierung der Argumente für Fuzzy-Prädikate kann zwar gewährleistet werden, dass die Maßeinheiten verglichener Werte untereinander kompatibel sind, nicht jedoch, dass sie gleich sind. Daher wird bei der Implementation von Fuzzy-Prädikaten für numerische Werte mit Maßeinheiten auf das in Abschnitt 4.2.1 definierte Umrechnungssystem für diese Maßeinheiten zurückgegriffen.

Tabelle 5.2 zeigt eine Reihe von Beispielen für Fuzzy-Prädikate auf Zeichenketten. Dabei sind hier zunächst Prädikate aufgeführt, die sich auf beliebige Zeichenketten anwenden lassen. Für speziellere Typen, die auf Zeichenketten basieren, wie „Personenname“, lassen sich entsprechende Prädikate implementieren (vgl. hierzu auch [NBYA01, CRF03]), die sich auch auf den besonderen Aufbau der Zeichenketten beziehen können, wie „enthält Nachname ähnlich zu x“ oder „ist Doppelname“.

Prädikate für beliebige Zeichenketten basieren zum einen auf dem Umfang der ausgewerteten Zeichenkette und zum anderen auf dem Enthaltensein von einer oder mehreren Unterzeichenketten, wobei für mehrere Unterzeichenketten zusätzlich ihr räumlicher Zusammenhang innerhalb der untersuchten Zeichenkette eine Rolle spielen kann. Für die Untersuchung der Struktur von Textpassagen können dabei insbesondere auch Methoden des Information Retrieval verwendet werden (vgl. z.B. [Kor97, Por80, Cav94]), für die es verschiedene Implementationen mit offenen Schnittstellen gibt [Ste96], auf denen bei der Realisierung von Fuzzy-Prädikaten aufgesetzt werden kann.

Bereits die kurze Liste der Beispiele in Tabelle 5.2 macht deutlich, dass Literale bei ihrer Auswertung bis auf einzelne Zeichen hinunter untersucht werden können. Hier ist bei der Auswahl der zu implementierenden Fuzzy-Prädikate abzuwägen, in welchem Interpretationskontext diese Prädikate zum Einsatz kommen. Für Kooperationspartner beispielsweise, die mit den interpretierten Metadaten nicht bis ins Detail vertraut sind, ist oft eine einfache Möglichkeit zur Textsuche ausreichend, die sich auf Prädikate wie „ist lang“ oder „enthält das

Prädikat	**Typ der Argumente**
x ist groß|klein	
x hat [mindestens|höchstens|ungefähr] a_1 Elemente	numerischer Typ
für alle Elemente in x gilt q; für (viele|wenige) Elemente in x gilt q	Prädikat
für [mindestens|höchstens|ungefähr] a_1 Elemente in x gilt q	numerischer Typ, Prädikat

Tabelle 5.3: Beispiele für Fuzzy-Prädikate auf Kollektionen

Wort“ beschränkt, da es keine bekannten Anhaltspunkte zur Nutzung detaillierterer Prädikate gibt. Kooperationspartner dagegen, die häufig mit bestimmten Metadaten-Records arbeiten, greifen auch auf feingranulare Prädikate zurück, die sie genau zu einer bereits bekannten Beschreibung im Bestand führen.

5.2.2.2 Fuzzy-Prädikate für Kollektionen

In Tabelle 5.3 sind mögliche Fuzzy-Prädikate für die Auswertung von Kollektionswerten aufgeführt. Diese Prädikate beziehen sich zum einen auf die Größe der Kollektion selbst oder machen zum anderen die Elemente der Kollektion einzeln zum Gegenstand der Auswertung. Da die Elemente einer Kollektion wiederum Werte repräsentieren, kann einem Kollektionsprädikat p ein weiteres Prädikat q für diese Werte übergeben werden, mit dessen Hilfe die einzelnen Elemente der Kollektion ausgewertet werden können. Bei dem Kollektionsprädikat p handelt es sich in diesem Fall um ein Fuzzy-Prädikat höherer Ordnung (vgl. Abschnitt 5.2.1).

5.2.2.3 Fuzzy-Prädikate für Records

Fuzzy-Prädikate für Records werten zum einen deren Struktur aus, indem sie überprüfen, ob ein Record, direkt oder in weiteren Records geschachtelt, ein bestimmtes Attribut enthält (vgl. Tab. 5.4). Neben der Auswertung der reinen Record-Struktur bewerten Fuzzy-Prädikate auch die Werte, die den Attributen innerhalb des Records, wiederum direkt oder geschachtelt, zugewiesen sind. Hierzu werden, ebenso wie für Kollektionen, Fuzzy-Prädikate höherer Ordnung eingesetzt, denen Prädikate zur Auswertung der Werte in den einzelnen Eigenschaften übergeben werden (vgl. Abschnitt 5.2.1). Das übergebene Prädikat nimmt dabei auf ein bestimmtes Attribut Bezug oder führt die Auswertung für alle im Record beschriebenen Attribute durch. Die mehrfache Rekursion über Fuzzy-Prädikate für Metadaten-Records lässt sich konzeptuell mit der Verwendung bestimmter Pfadausdrücke für semistrukturierte Daten (vgl. Abschnitt 3.3) vergleichen, wobei die Auswahl der Schritte und die Auswertung der Werte der Struktur hier jedoch der Unschärfe unterliegen.

Der Metadatenerfasser muss die Struktur von Records und weiteren darin geschachtelten Records nicht global festlegen, sondern kann einzelne Records fle-

Prädikat	**Typ der Argumente**
x hat [direktes] Attribut a_1	Attribut
x hat Attribut a_1 (auf|bis zu) Ebene a_2	Attribut, numerischer Typ
x hat [direktes] beliebiges Attribut, für das q gilt	Prädikat
x hat beliebiges Attribut (auf|bis zu) Ebene a_1, für das q gilt	numerischer Typ, Prädikat
x hat Attribut a_1, für das q gilt	Attribut, Prädikat
x hat Attribut a_1 (auf|bis zu) Ebene a_2, für das q gilt	Attribut, numerischer Typ Prädikat

Tabelle 5.4: Beispiele für Fuzzy-Prädikate auf Records

xibel strukturieren. Die Schachtelungstiefe für eine Eigenschaft, die die gleiche Information übermitteln soll, kann sich daher von Record zu Record unterscheiden, wie das folgende Beispiel zeigt:

```
<description about="http://www.somedomain.de/somedoc.html">
  <title>Institutsreport Nr. 5</title>
  <institution>
    <name>Institut zur Untersuchung von Beispielen</name>
    <address>
      <zip>12345</zip>
      <city>Testhausen</city>
    </address>
  </institution>
</description>

<description about="http://www.somedomain.de/somedoc.html">
  <title>Institutsreport Nr. 5</title>
  <institution>
    <name>Institut zur Untersuchung von Beispielen</name>
    <contact>
      <email>examples@somedomain.de</email>
      <address>
        <zip>12345</zip>
        <city>Testhausen</city>
      </address>
    </contact>
  </institution>
</description>
```

Während beispielsweise die Angabe der Stadt, in der das herausgebende Institut ansässig ist, im ersten Record auf Schachtelungsebene drei zu finden ist, ist sie im zweiten Record auf Ebene vier erfasst. Ein interpretierender Akteur kann im Regelfall nicht überblicken, in welcher Schachtelungstiefe sich ein von

ihm gesuchtes Attribut in den Metadaten seiner Kooperationspartner befindet. Fuzzy-Prädikate auf Records, wie das Prädikat „x hat Attribut a_1, für das q gilt“, sind daher nicht auf die erste Ebene eines Records begrenzt, sondern werten Records in beliebiger Tiefe aus, falls sie nicht diesbezüglich eingeschränkt sind, wie das Prädikat „x hat Attribut a_1 bis zu Ebene a_2, für das q gilt“. Fuzzy-Prädikate für Records mit beliebiger Schachtelungstiefe entsprechen konzeptuell Wildcards in Pfadausdrücken (vgl. Abschnitt 3.3).

Ein weiteres Phänomen bei der Formulierung von Metadaten durch mehrere Kooperationspartner ist die unterschiedliche Granularität bei der Strukturierung der Records, die durch das folgende Beispiel verdeutlicht werden soll, das einen dritten Record für die oben beschriebene Ressource „somedoc.html“ enthält:

```
<description about="http://www.somedomain.de/somedoc.html">
  <title>Institutsreport Nr. 5</title>
  <institution>
    <name>Institut zur Untersuchung von Beispielen</name>
    <address>12345 Testhausen</adress>
  </institution>
</description>
```

In diesem Beispiel ist der Wert des Attributs „address“, der zuvor ein geschachtelter Record war, in ein Literal aufgelöst worden und hat somit seine Struktur auf dieser Ebene verloren. Für Fälle, in denen ein interpretierender Akteur keine Kenntnis von der Granularität der Record-Struktur besitzt, sollten ihm Prädikate zur Verfügung stehen, die Eigenschaften für beliebige Attribute auf die Gültigkeit einer Bedingung hin überprüfen, wie das Prädikat „x hat beliebiges Attribut, für das q gilt“. Für das obige Beispiel kann dieses Prädikat verwendet werden, um Einträge zu suchen, in denen eine Eigenschaft für ein beliebiges Attribut, also sowohl für „address“ als auch für „city“, die Zeichenkette „Testhausen“ enthält.

Bei Beziehungen zwischen Ressourcen sowie bei Zuordnungen von Ressourcen zu Klassen handelt es sich jeweils um spezielle Records (vgl. Abschnitte 4.1 und 4.3). Für sie lassen sich Fuzzy-Prädikate formulieren, wie „x verknüpft die Ressource a_1 in der Rolle a_2 mit einer anderen Ressource“ oder „x ordnet der Klasse a_1 eine Ressource zu“. Diese Prädikate können auf die in Tabelle 5.4 aufgezählten Prädikate für Records abgebildet werden und werden hier nicht extra aufgeführt. Ihre Existenz und ihre Bezeichnung erleichtert als syntaktischer Zucker dem interpretierenden Akteur die Formulierung von Anfragen an den Metadatenbestand.

5.2.2.4 Fuzzy-Prädikate für Ressourcen

Attribute können in Eigenschaften auch mit Referenzen auf andere Ressourcen belegt werden, die sich dabei sowohl innerhalb als auch außerhalb des gemeinsamen Informationsraums befinden können. Auch Ressourcen und die Referenzen darauf können durch Fuzzy-Prädikate ausgewertet werden, von denen einige in Tabelle 5.5 aufgeführt sind.

Prädikat	**Typ der Argumente**
Referenz auf x ist [URL|DOI]; x hat Record; x hat (viele|wenige) Records	
Referenz auf x ist genau a_1; Referenz auf x ist ähnlich zu a_1; Referenz auf x enthält a_1	Zeichenkette
x hat [mindestens|höchstens] a_1 Records	numerischer Typ
x hat Record, für den p_ν gilt x hat (viele|wenige) Records, für die p_ν gilt	Prädikat für Records
x hat [mindestens|höchstens] a_1 Records, für die p_ν gilt	numerischer Typ, Prädikat für Records

Tabelle 5.5: Beispiele für Fuzzy-Prädikate auf referenzierten Ressourcen

Fuzzy-Prädikate auf Ressourcereferenzen werten zum einen die Referenz selbst aus, indem sie untersuchen, ob diese einem bestimmten Muster entspricht. Hier wird beispielsweise ermittelt, ob die Referenz eine bestimmte Zeichenkette enthält oder nach bestimmten Regeln, wie denen für URLs oder DOIs [Pas00] aufgebaut, ist. Zum anderen ermitteln Fuzzy-Prädikate für referenzierte Ressourcen, ob und wie häufig sie durch Records im vorliegenden Metadatenbestand beschrieben sind, und überprüfen darüber hinaus, ob die entsprechenden Records bestimmte Bedingungen erfüllen.

Fuzzy-Prädikate, die überprüfen, ob die Metadaten-Records für eine referenzierte Ressource eine bestimmte Bedingung erfüllen, benötigen als zusätzlichen Parameter ein weiteres Fuzzy-Prädikat q, das diese Bedingung implementiert. Das übergebene Prädikat q wertet stets Records aus und muss hierfür geeignet sein (vgl. Tab. 5.5). Die Implementation der aufrufenden Fuzzy-Prädikate für durch weitere Records beschriebene Ressourcen stellt eine Verwendungsmöglichkeit für die Verknüpfungsoperation für Fuzzy-Relationen dar (vgl. Abschnitt 5.1.3).

Die in Tabelle 5.5 aufgeführten Fuzzy-Prädikate gehen von der Minimalannahme aus, dass eine Ressource in einem gemeinsamen Informationsraum in verschiedenen Beschreibungen einheitlich über einen Identifikator referenziert wird. Sie benötigen keinen direkten Zugriff auf die Ressource selbst. Ist eine Ressource digital im Informationsraum abgelegt und somit verfügbar, so ergeben sich Implementationsmöglichkeiten für weitere Fuzzy-Prädikate, die die Ressource selbst entsprechend ihrem Format auf bestimmte Eigenschaften hin untersuchen, und hierfür beispielsweise Verfahren der Volltextsuche [BYRN99], des Image- oder des Video-Retrievals [GJ97] einsetzen.

5.2.3 KooMet-QL – eine Anfragesprache für Metadaten

Einzelne unscharfe Bedingungen hinsichtlich der Eigenschaften einer Ressource lassen sich formulieren, indem ein Fuzzy-Prädikat auf Werte eines bestimmten Attributs angewendet wird (vgl. Abschnitt 5.2.1). Auf dieser Grundlage lässt sich nun eine einfache Anfragesprache realisieren, mit deren Hilfe in ei-

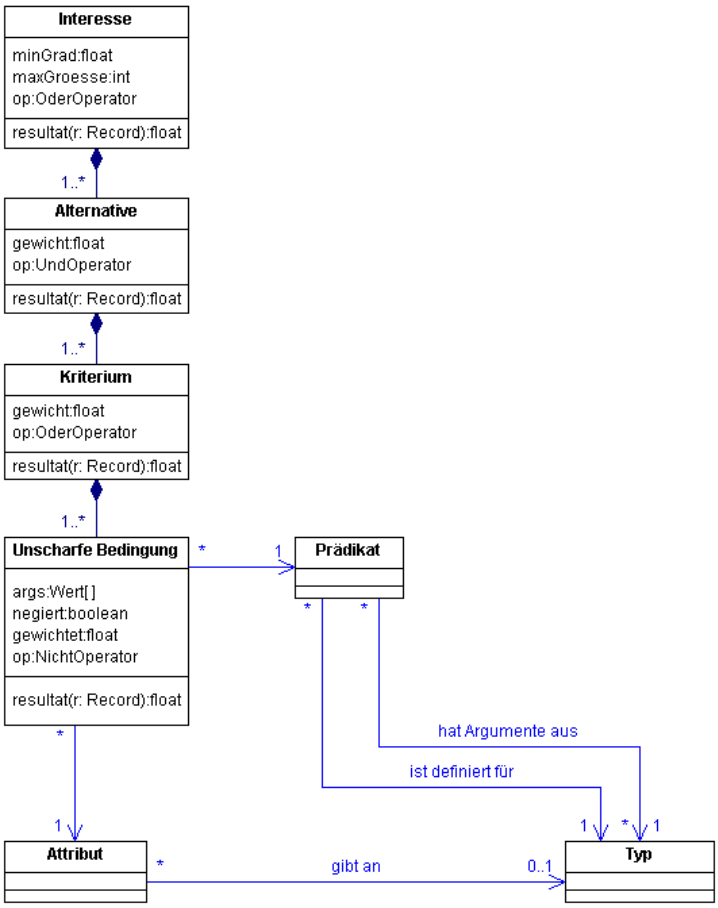

Abbildung 5.6: Anfragesprache zur Suche in Metadatenbeständen mit Hilfe von Fuzzy-Prädikaten

nem gemeinsamen Informationsraum Metadaten und Ressourcen zur Lösung kooperativer Arbeitsaufgaben aufgefunden werden können. Die besonderen Gegebenheiten des integrierenden Metadatenmodells KooMet werden dabei von den Konstrukten der Anfragesprache reflektiert. Vom Metadatenmodell leitet sich auch der Name *KooMet-QL* (KooMet *Q*uery *L*anguage) dieser Anfragesprache ab. Durch den Einsatz toleranter Algorithmen liegen die Resultate von KooMet-QL in Form von Trefferpaaren vor, die Records ein Auswertungsergebnis zuordnen, und nicht als eigenständige Metadatenstrukturen, gegen die wiederum Anfragen gestellt werden könnten. KooMet-QL ist somit nicht als vollständige Sprache, sondern lediglich als Selektionssprache aufzufassen.

Da es für Metadaten in KooMet kein einheitliches Schema geben muss (vgl. Abschnitt 4.2), sind die Aspekte einer Ressource, die für die Aufgabe eines anfragenden Akteurs von Bedeutung sind, nicht unbedingt in Eigenschaften für ein und dasselbe Attribut erfasst. In Abschnitt 5.2.2.3 wurde hierzu bereits ein Beispiel aufgeführt, in dem der Name einer Stadt in einem Record als Wert für das Attribut city auftrat, während er in einem zweiten Record im Wert des

Attributs address enthalten war. Um auf Metadatenbeständen mit heterogenen Record-Strukturen dennoch zu einem möglichst vollständigen Suchergebnis zu kommen, erlaubt es KooMet-QL, verschiedene unscharfe Bedingungen für verschiedene Record-Strukturen zu formulieren, die semantisch jeweils eine ähnliche Anforderung an eine Ressource repräsentieren. Diese werden innerhalb eines *Kriteriums* zusammengefasst (vgl. Abb. 5.6), das semantisch der Anwendung eines unscharfen Oder-Operators entspricht. Dieser Oder-Operator kann für die Auswertung des Kriteriums durch einen Vereinigungsoperator für Fuzzy-Mengen realisiert werden. Ein Kriterium, das unscharfe Bedingungen für die beiden Records des obigen Beispiels umfasst, kann man z.B. wie folgt formulieren:

```
Kriterium(ist_aehnlich_zu(city,"Testhausen"),
          enthaelt_das_wort(address,"Testhausen"))
```

Da in unterschiedlichen Attributen ein Aspekt mit unterschiedlicher Deutlichkeit zum Ausdruck kommen kann, ist es zusätzlich möglich, jede unscharfe Bedingung innerhalb eines Kriteriums mit einem Gewicht zu versehen, um diese Unterschiede in eine Anfrage mit einzubringen. Zur Anwendung von Gewichtungen auf die Ergebnisse von Fuzzy-Prädikaten und Fuzzy-Operatoren wird angenommen, dass es sich bei der internen Darstellung von Gewichtungen hier und auch im Folgenden um Gleitkommazahlen handelt. Um die Formulierung von Anfragen für menschliche Akteure zu vereinfachen, umfassen Anfrageschnittstellen darüber hinaus jedoch Zuordnungen von einzelnen Gewichtungen zu Ausdrücken in natürlicher Sprache, die die Semantik dieser Gewichtungen repräsentieren. So könnte beispielsweise eine Gewichtung von 1.0 dem Ausdruck „besonders relevant“ zugeordnet sein, während der Ausdruck „nicht sehr relevant“ eine Gewichtung von 0.2 beschreibt. Der Anfragende verwendet diese Ausdrücke in seiner Anfrage dann anstelle der Gleitkommazahlen.

Ein anfragender Akteur besitzt nicht immer die Übersicht über die Struktur aller Records in einem gemeinsamen Informationsraum. Daher muss die Formulierung von Kriterien nicht notwendigerweise direkt durch diesen Akteur erfolgen. Eine weitere Option ist hier, dass eine Anfrageschnittstelle automatisch anhand einer Reihe von zuvor festgelegten Regeln eine einzelne vom Akteur formulierte unscharfe Bedingung in mehrere semantisch ähnliche unscharfe Bedingungen expandiert und daraus intern ein entsprechendes Kriterium erstellt.

Um Ressourcen oder sie beschreibende Records aufzufinden, die zur Lösung einer bestimmten Arbeitsaufgabe beitragen können, muss ein Akteur diese genau beschreiben und somit in einer Suchanfrage beliebig viele Aspekte dieser Ressourcen innerhalb einer einzigen Anfrage in Form von Kriterien formulieren können. KooMet-QL sieht daher vor, dass sich Kriterien wiederum innerhalb von *Alternativen* zusammenfassen lassen (vgl. Abb. 5.6), was semantisch der Anwendung eines unscharfen Und-Operators auf die Kriterien entspricht. Der Und-Operator kann für die Auswertung der Alternative durch einen Schnittmengenoperator für Fuzzy-Mengen realisiert werden (vgl. Abschnitt 5.1.2). Das folgende Beispiel fasst das oben erstellte Kriterium mit einem weiteren Kriterium in einer Alternative zusammen:

```
Alternative(
  Kriterium(ist_aehnlich_zu(city,"Testhausen"),
            enthaelt_das_wort(address,"Testhausen")),
  Kriterium(ist_aehnlich_zu(titel,"Institutsreport 5")))
```

Auch Kriterien können innerhalb einer Alternative gewichtet werden, so dass ein Akteur zum Ausdruck bringen kann, welches Kriterium für die von ihm benötigten Ressourcen oder Records die größte Bedeutung besitzt.

Oft verwenden Akteure bei der Bearbeitung einer Aufgabe mehr als eine Ressource. Da in einer Alternative jeweils nur die Anforderungen an eine einzelne Ressource angegeben werden können, stellt KooMet-QL ein weiteres Konstrukt zur Verfügung, um Anfragen zu formulieren, die mehrere Alternativen umfassen und die die Vereinigungsmenge der durch die einzelnen Alternativen spezifizierten Records bzw. Ressourcen zum Ergebnis haben. Eine einzelne Anfrage, die nach mehreren alternativen Ressourcen sucht, wird als *Interesse* bezeichnet (vgl. Abb. 5.6), da sie das Gesamtinteresse des Anfragenden für im Informationsraum befindliche Records und Ressourcen reflektiert. Das Interesse stellt somit zugleich die eigentliche Anfrage des Akteurs an den Metadatenbestand dar.

Auch innerhalb eines Interesses kann der Anfragende bestimmten Alternativen mit Hilfe von Gewichtungen Priorität einräumen. Darüber hinaus kann er hier den Umfang eines Anfrageergebnisses einschränken, indem er bei der Formulierung seines Interesses eine maximale Anzahl oder einen minimalen Erfüllungsgrad für die zurückgelieferten Records angibt. Diese Beschränkungen für Ergebnismengen können zur Verringerung von Berechnungsschritten bei der Anfrageauswertung beitragen (vgl. Abschnitt 5.2.4).

5.2.4 Auswertung von Anfragen auf einem Metadatenbestand

Anfragen, die in der oben beschriebenen Anfragefragesprache formuliert sind und das Interesse eines Akteurs am gemeinsamen Informationsraum widerspiegeln (vgl. Abschnitt 5.2.3), werden ausgewertet, um in einem Metadatenbestand relevante Records oder indirekt auch durch sie beschriebene Ressourcen aufzufinden. Hierzu werden die Auswertungsmechanismen, die Fuzzy-Prädikate für einzelne Records zu Verfügung stellen (vgl. Abschnitt 5.2.1), wiederholt auf die Records des Bestandes angewendet. Für jeden Record wird dabei in Abhängigkeit vom in der Anfrage geäußerten Interesse ein Erfüllungsgrad berechnet und ihm innerhalb eines *Treffers* zugeordnet (vgl. Abb. 5.7). Die Gesamtheit dieser Treffer stellt das Ergebnis der Anfrage und damit das *Angebot* des Informationsraums an den suchenden Akteur dar.

Um die Schritte, die für die Auswertung von Anfragen in KooMet-QL in einem Metadatenbestand notwendig sind, zu veranschaulichen, wird zunächst ein einfacher Auswertungsalgorithmus vorgestellt. Im Anschluss daran werden mögliche Anpassungen des Algorithmus erläutert, durch die die Performanz der Auswertung erhöht werden kann.

Der Auswertungsalgorithmus für ein Interesse berechnet als Ergebnismenge ein Angebot, dem die Auswertung aller in einem Metadatenbestand vorhandenen Records zugrunde liegt. Die Zusammenstellung des Angebots verläuft

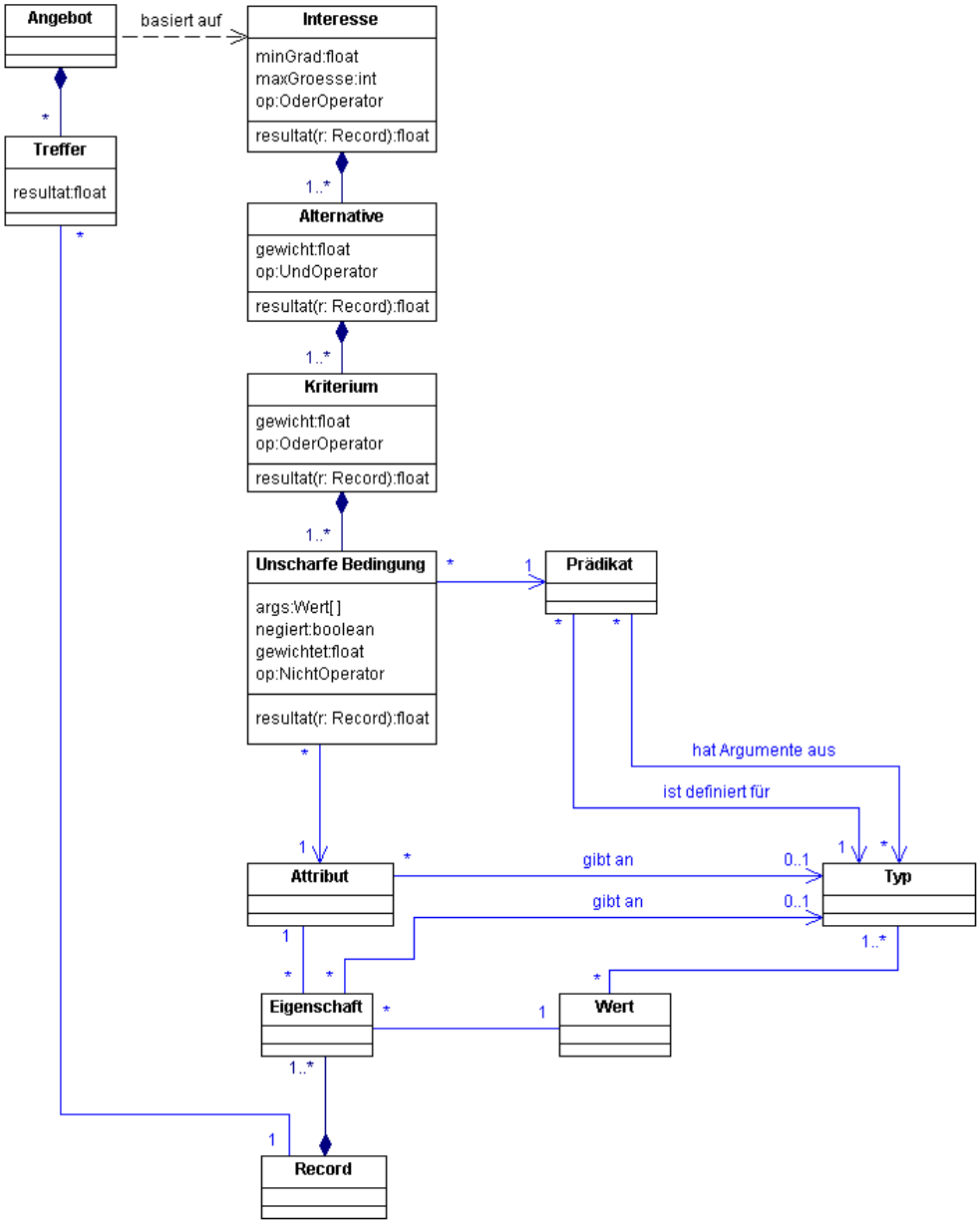

Abbildung 5.7: Zusammenhang zwischen Metadaten, Anfrage und Anfrageergebnis

dabei unterschiedlich, je nachdem ob das Interesse einen Mindesterfüllungsgrad oder eine Höchstanzahl für die enthaltenen Treffer festlegt oder die Größe des Angebots unbegrenzt bleibt:

```
class Metadatenbestand {

    Angebot suche(Interesse interesse){
      Angebot angebot = new Angebot();
      Iterator records = holeRecords();
      if(interesse.hatMinGrad()){
        //Ergebnis qualitativ begrenzt
        float minGrad = interesse.holeMinGrad();
        while(records.hasNext()){
          Record rec = (Record)records.next();
          float grad = interesse.resultat(rec);
          if(erg >= minGrad)
            angebot.fuegeHinzu(new Treffer(rec,grad));
        }
      }
      else if(interesse.hatMaxGroesse()){
        //Ergebnis quantitativ begrenzt
        int maxGroesse = interesse.holeMaxGroesse();
        while(records.hasNext()){
          Record rec = (Record)records.next();
          float grad = interesse.resultat(rec);
          if(angebot.holeGroesse() < maxGroesse)
            angebot.fuegeHinzu(new Treffer(rec,grad));
          else{
            Treffer letzter = Angebot.holeLetzten();
            if(letzter.holeGrad() < grad)
              Angebot.tausche(letzter,new Treffer(rec,grad));
           }
        }
      }
      else
        //Ergebnis nicht begrenzt
        while(records.hasNext()){
          Record rec = (Record)records.next();
          angebot.fuegeHinzu(
            new Treffer(rec,interesse.resultat(rec)));
        }
      return Angebot;
    }
}
```

Legt das Interesse für das Angebot einen Erfüllungsgrad fest, den enthaltene Treffer mindestens besitzen müssen, so wird, nachdem das Interesse für einen

Record ausgewertet wurde, überprüft, ob das jeweilige Ergebnis entsprechend groß ist. Nur dann werden der Record und sein Ergebnis in einem Treffer zusammengefasst und dem Angebot hinzugefügt. Ist dagegen eine Höchstanzahl für die im Angebot enthaltenen Treffer festgelegt, so werden dem Angebot Treffer zunächst hinzugefügt, bis diese Höchstanzahl erreicht ist. Für jeden weiteren ausgewerteten Record wird dann überprüft, ob sein Erfüllungsgrad höher ist, als der des schlechtesten Treffers im Angebot. Ist dieses der Fall, so wird für den neu ausgewerteten Record ein Treffer erstellt, durch den der schlechteste Treffer des Angebots ersetzt wird. Gibt das Interesse keinerlei Beschränkungen für das zu ermittelnde Angebot vor, so wird einfach für jeden Record ein Treffer erzeugt und dem Angebot hinzugefügt.

Die im Algorithmus aufgerufene Auswertungsmethode des Interesses ruft für den übergebenen Record ihrerseits die Auswertungsmethoden aller ihrer Alternativen auf, die den Record wiederum an die Auswertungsmethoden ihrer Kriterien weiterreichen. Diese schließlich rufen dann die Auswertungsmethoden der einzelnen unscharfen Bedingungen auf, deren Arbeitsweise in Abschnitt 5.2.1 im Detail erläutert wurde. Die Ergebnisse der unscharfen Bedingungen werden entlang der Hierarchie der Anfragekonstrukte wieder nach oben durchgereicht, wobei die Zwischenergebnisse in den entsprechenden Auswertungsmethoden jeweils mit etwaigen Gewichtungen verrechnet und aggregiert werden. Die Auswertungsmethoden eines Interesses, einer Alternative und eines Kriteriums seien zur Veranschaulichung im Folgenden aufgeführt:

```
class Interesse {
  Alternative[] alts;
  float minGrad; int maxGroesse; OderOperator op;

  float resultat(Record rec){
    float erg = -1.f
    for(int i = 0; i < alts.length; i++)
      erg = op.oder(erg,
        alts[i].resultat(rec)*alts[i].holeGewicht());
    return (erg < 0.f) ? 0.f : erg;
  }
}

class Alternative {
  Kriterium[] krits;
  float gewicht; UndOperator op;

  float resultat(Record rec){
    float erg = -1.f
    for(int i = 0; i < krits.length; i++)
      erg = op.und(erg,
        krits[i].resultat(rec)*krits[i].holeGewicht());
    return (erg < 0.f) ? 0.f : erg;
  }
}
```

```
class Kriterium {
  Unscharfe_Bedingung[] beds;
  float gewicht; OderOperator op;


  float resultat(Record rec){
    float erg = -1.f
    for(int i = 0; i < beds.length; i++)
      erg = op.oder(erg,
        beds[i].resultat(rec)*beds[i].holeGewicht());
    return (erg < 0.f) ? 0.f : erg;
  }

}
```

Auch wenn ein Interesse den Umfang des Angebots auf eine bestimmte Anzahl von Treffern beschränkt, werden im einfachen Auswertungsalgorithmus für jeden Record in der Auswertungsmethode des Interesses die Ergebnisse aus allen Alternativen aggregiert. Zur Kombination von Einzelergebnissen, wie sie durch die Auswertung der Alternativen entstehen, stellt Fagin in [Fag96] für monotone Aggregatoren wie t-Normen und t-Conormen (vgl. Abschnitt 5.1.2) einen Algorithmus vor, der mit weniger Schritten auskommt. Der Algorithmus basiert auf der Annahme, dass für alle Objekte eines Datenbestandes, hier also für alle Records, die zu kombinierenden Ergebnisse der Unteranfragen, hier also die Ergebnisse der einzelnen Alternativen, in absteigend geordneten Listen zur Verfügung stehen, auf die sowohl sequentiell als auch gezielt zugegriffen werden kann. Die Ordnung dieser Listen und die Tatsache, dass die Ergebnismenge zahlenmäßig begrenzt ist, macht sich Fagins Algorithmus zu Nutze, um die Aggregation der Einzelergebnisse nur für wenige Objekte durchzuführen.

Obwohl Fagins Algorithmus die Performanz gegenüber einem naiven Aggregationsalgorithmus verbessern kann, ist er dennoch insbesondere für Aggregatoren, die nicht streng monoton sind, nicht optimal [Fag02]. Ein weiterer Algorithmus, der mit denselben Voraussetzungen arbeitet wie Fagins Algorithmus, der aber dessen Schwächen ausräumt, wurde u.a. in [GBK02] und [FLN03] vorgestellt.

Um die beschriebenen optimierenden Aggregationsalgorithmen bei der Auswertung eines Interesses für einen Metadatenbestand einzusetzen, muss die Vorgehensweise, die Auswertungsmethode resultat des Interesses wiederholt für jeden Record aufzurufen, aufgegeben werden. Stattdessen werden alle Records gemeinsam an eine Auswertungsmethode, die hier resultate genannt sei, übergeben, die die für die optimierte Aggregation benötigten Ergebnislisten für jede Alternative erstellt, um dann die Aggregation durchzuführen. Das Ergebnis der Auswertung ist entsprechend kein einzelner Erfüllungsgrad, sondern ein Angebot, das die geforderte Anzahl von Treffern enthält:

```
class Metadatenbestand {

  Angebot suche(Interesse interesse){
    Angebot angebot = new Angebot();
    Iterator records = holeRecords();

    ...
    else if(interesse.hatMaxGroesse()){
      angebot = interesse.resultate(records);
    }
    ...
  }
}

class Interesse {
  Alternative[] alts; int MaxGroesse;
  ...

  Angebot resultate(Iterator records){
    Ergebnisliste[] listen = new Ergebnisliste[alts.length];
    while(redords.hasNext()){
        Record rec = (Record)records.next();
        for(int i = 0; i < alts.length; i++)
          listen[i].fuegeEin(rec,
            alts[i].resultat(rec)*alts[i].holeGewicht());
    }
    return Fagin.aggregiere(listen,maxGroesse);
  }
}
```

Die optimierte Aggregation lässt sich für die Anfragesprache KooMet-QL durchgehend nur auf der Ebene des Interesses durchführen. Da hier die Ergebnisse aller Alternativen benötigt werden, liefert die Auswertung in der Alternative selbst kein Optimierungspotential, da sie nicht mit beschränkten Ergebnismengen arbeiten kann. Eine Ausnahme stellen Anfragen dar, in denen ein Interesse nur eine einzige Alternative umfasst. In diesem Fall kann die Beschränkung der Ergebnismenge in der Alternative übernommen und für die Optimierung genutzt werden. Gleiches gilt, wenn eine einzelne Alternative nur ein Kriterium umfasst, und wiederum auch, wenn dieses einzelne Kriterium aus nur einer unscharfen Bedingung besteht.

Weitere Optimierungen bei der Auswertung von Anfragen werden ermöglicht, wenn für die Speicherung von Metadaten persistente Datenspeicher verwendet werden, die bereits effiziente Suchverfahren realisieren. Mit der Kenntnis der Charakteristika der verwendeten Fuzzy-Operatoren (vgl. Abschnitt 5.1.2) können beispielsweise boolesche Anfragen an ein relationales Datenbanksystem einer unscharfen Auswertung vorgeschaltet werden, um so Records, für die durch die Belegung bestimmter Attribute mit bestimmten Werten bereits ein

eindeutiges Ergebnis vorweggenommen ist, von vornherein aus einzelnen unscharfen Auswertungsschritten herauszulassen.

5.2.5 Verwendung von Klassifikationsinformationen

Die Zuordnung von Ressourcen zu Klassen stellt diese Ressourcen in einen Bedeutungszusammenhang (vgl. Abschnitt 4.3). Dieser Zusammenhang besteht zum einen zwischen der klassifizierten Ressource und ihrer Klasse, die ein Konzept der Anwendungsdomäne repräsentiert. Zum anderen stehen aber auch die Ressourcen derselben Klasse untereinander inhaltlich in Beziehung.

In KooMet kann ein und dasselbe Attribut Ressourcen aus unterschiedlichen Klassen beschreiben. Aufgrund der inhaltlichen Zusammenhänge zwischen den Ressourcen derselben Klasse und der inhaltlichen Unterschiede zwischen Ressourcen unterschiedlicher Klassen können die Werte für dieses Attribut über Klassengrenzen hinweg stark voneinander abweichen. Das Attribut „Stückpreis“ für Ressourcen in einem Online-Katalog mit breiter Produktpalette kann hier als Beispiel dienen: Während Kleinartikel wie Kugelschreiber, Streichhölzer, Büroklammern o.ä. einen Stückpreis von wenigen Cent erzielen, werden Bekleidungsartikel zu zwei- oder dreistelligen Eurobeträgen gehandelt, während Artikel aus dem Bereich Unterhaltungselektronik noch teurer sind.

Gibt in einem Informationsraum mit klassifizierten Ressourcen der interpretierende Akteur bei einer Anfrage nicht explizit eine Klasse von Ressourcen vor, in der er sucht, kann die Suche anhand eines klassenübergreifenden Attributs unvorhergesehene Ergebnisse liefern. So könnte beispielsweise die Suche nach Preis-Schnäppchen in unserem Online-Katalog wie folgt formuliert werden:

```
Interesse(
  Alternative(
    Kriterium(ist_niedrig(Preis)))
```

Um das gewünschte Ergebnis zu erzielen, darf die Auswertung dieser Anfrage nicht einheitlich über den gesamten Bestand erfolgen, da ansonsten Kleinartikel den gesamten oberen Bereich der Ergebnisliste für sich einnehmen würden. Um die jeweils preisgünstigsten Artikel bezüglich einer Produktkategorie zu finden, muss die Auswertung daher zwischen den einzelnen Klassen differenzieren.

Um für einzelne Klassen entsprechende Auswertungsergebnisse berechnen zu können, bedarf es detaillierterer statistischer Informationen, die nicht nur pro Attribut, sondern pro Attribut und Klasse verwaltet werden (vgl. Abb. 5.8). Die Auswertung erfolgt dann klassenweise, indem für jede Klasse alle Records der darin befindlichen Ressourcen betrachtet werden. Der Auswertungsmethode des Interesses muss nun neben dem Record selbst auch die entsprechende Klasse mit übergeben werden. Diese wird gemeinsam mit dem Record über alle Ebenen der Anfrage bis in die Fuzzy-Prädikate hinunter gereicht, die mit den ihnen zur Verfügung stehenden Parametern auf die von ihnen benötigten statistischen Informationen für Attribut und Klasse zugreifen können:

```
class Metadatenbestand {

    Angebot suche(Interesse interesse){
      Angebot angebot = new Angebot();
      Iterator klassen = holeKlassen();
      while(klassen.hasNext()){
        Klasse k = (Klasse)klassen.next();
        Iterator ressourcen = k.holeRessourcen();
        while(ressourcen.hasNext()){
          Ressource res = (Ressource)ressourcen.next();
          Iterator recs = res.holeRecords();
          while(recs.hasNext()){
            Record rec = (Record)recs.next();
            angebot.fuegeHinzu(
              new Treffer(rec,k,interesse.resultat(rec,k));
          }
        }
      }
      return Angebot;
    }
}
```

Ein Treffer enthält neben dem Record und dem Erfüllungsgrad nun auch die Klasse, für die dieser berechnet wurde. Hierdurch ist beispielsweise eine klassenweise Visualisierung des Anfrageergebnisses möglich. Diese erscheint besonders vor dem Hintergrund der Tatsache sinnvoll, dass eine Ressource verschiedenen Klassen gleichzeitig zugeordnet sein kann, so dass bei der klassenweisen Auswertung für ein und denselben Record in jeder Klasse ein anderer Erfüllungsgrad berechnet wird.

Ob zur Verfügung stehende Klassifikationsinformationen bei der Auswertung einer Anfrage verwendet werden sollten, hängt einzig vom Informationsbedarf des interpretierenden Akteurs ab. Bezogen auf das obige Beispiel ist es sowohl möglich, dass ein Kunde des Online-Katalogs nach den Schnäppchen der Produktkategorien sucht, als auch dass er an den günstigsten Artikeln, die überhaupt im Angebot sind, interessiert ist. Um bestandsweise und klassenweise Anfragen gleichermaßen zu ermöglichen, muss daher direkt über die Anfrageschnittstelle eine entsprechende Auswahl ermöglicht werden, die dann den dazugehörigen Auswertungsalgorithmus in Gang setzt.

5.3 Zusammenfassung

Tolerante Algorithmen überbrücken die Kluft, die sich zwischen einem Metadatenerfasser und einem interpretierenden Akteur aufgrund ihrer verschiedenen Arbeitskontexte ergibt, indem sie die Interpretation dem jeweiligen Kontext anpassen, in dem sie stattfindet. Die Fuzzy-Logik hält eine Reihe theoretischer Grundlagen für die Realisierung toleranter Algorithmen bereit.

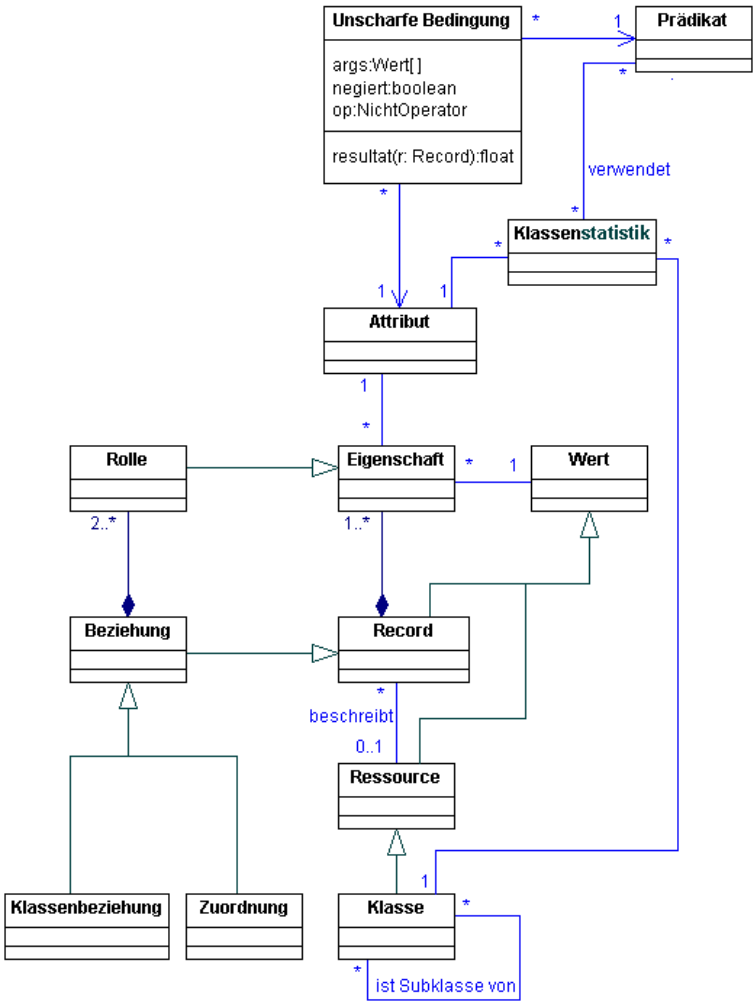

Abbildung 5.8: Fuzzy-Prädikate zur klassenweisen Auswertung von unscharfen Bedingungen

Mit Hilfe von Fuzzy-Prädikaten lassen sich tolerante Fuzzy-Algorithmen auf der Basis des integrierenden Metadatenmodells KooMet (vgl. Kapitel 4) realisieren, wobei Besonderheiten des Modells wie flexible Record-Strukturen oder multiple Records für einzelne Ressourcen berücksichtigt werden.

Ein Beispiel für den Einsatz von Fuzzy-Prädikaten bei der asynchronen Nutzung gemeinsamer Informationsräume ist die Formulierung von Suchanfragen an einen Metadatenbestand. Die Anfragesprache KooMet-QL kombiniert unscharfe Bedingungen zu Suchanfragen für die Selektion von Metadaten-Records und verwendet Fuzzy-Prädikate dabei als einen wichtigen Grundbaustein. Die in KooMet vorgesehenen Klassifikationsinformationen können dazu verwendet werden, KooMet-QL-Anfragen entsprechend den Klassen einer Anwendungsdomäne auszuwerten.

Kapitel 6

Praktische Umsetzung

Mit dem integrierenden Metadatenmodell KooMet (vgl. Kapitel 4) liegt ein konzeptuelles Datenmodell vor, auf dessen Grundlage konkrete Systeme zur Verwaltung von Metadaten in gemeinsamen Informationsräumen entworfen werden können. Während das Ziel von KooMet die Unterstützung einer möglichst breiten Palette an Kooperationsszenarien und den damit verbundenen Vorgehensweisen (vgl. Kapitel 2) ist, begünstigen die Datenmodelle konkreter Systeme zumeist ein einzelnes Kooperationszenarium, so dass sie entsprechende Spezialisierungen des allgemeineren Modells KooMet darstellen.

Im Folgenden werden mit dem infoAssetBroker (vgl. Abschnitt 6.1) und dem Katalogsystem PIA (vgl. Abschnitt 6.2) zwei konkrete Systeme zur Verwaltung kooperativ genutzter Metadaten vorgestellt und die Auswirkungen ihres jeweiligen Kooperationsszenariums auf die Umsetzung der durch KooMet vorgegebenen Konzepte untersucht.

Zur Überbrückung der semantischen Kluft zwischen den individuellen Kontexten von Metadatenerfassern und interpretierenden Akteuren lassen sich tolerante Algorithmen an die Konzepte des Metadatenmodells KooMet anbinden (vgl. Kapitel 5) und ebenfalls in konkreten Systemen umsetzen. Die Verwendung toleranter Algorithmen im infoAssetBroker und in PIA wird daher im Verlauf dieses Kapitels ebenfalls betrachtet.

6.1 Enge Kooperation im infoAssetBroker

Der infoAssetBroker wurde in einer Kooperation zwischen dem Arbeitsbereich Softwaresysteme der Technischen Universität Hamburg-Harburg und der Münchner infoAsset AG entwickelt und dient als anpassbare Standard-Software dem Aufbau von organisationsspezifischen Wissensportalen. Die Einbindung in den Kontext der Organisation prägt dabei die Verwendung und Gestaltung der im Wissensportal befindlichen Metadaten.

6.1.1 Kooperationsszenarium

Mit Hilfe der infoAssetBroker-Standard-Software werden Wissensportale innerhalb des WorldWideWeb realisiert, die sowohl der externen Darstellung von Or-

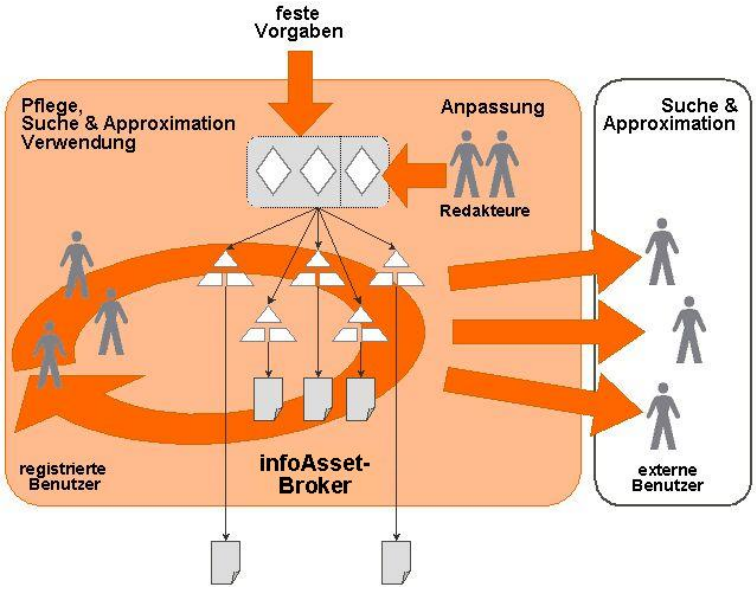

Abbildung 6.1: Zusammenarbeit im infoAssetBroker

ganisationen als auch der Verwaltung von Ressourcen und Metadaten innerhalb dieser Organisationen dienen. Bei den Akteuren, die über ein infoAssetBroker-Portal kooperieren, wird entsprechend zwischen externen Benutzern und Benutzern, die in Verbindung zur betreffenden Organisation stehen und daher im System registriert sind, unterschieden (vgl. Abb. 6.1).

Interne Benutzer greifen innerhalb ihrer Arbeitsprozesse auf Ressourcen und Metadaten zu, verwenden sie und verändern sie nach Bedarf. Durch ein Berechtigungskonzept, das Gruppen registrierter Benutzer die Rechte zum lesenden und schreibenden Zugriff jeweils auf eine bestimmte Teilmenge der im Portal abgelegten Ressourcen und Metadaten gewährt, wird ermöglicht, dass verschiedene Arbeitsgruppen innerhalb der Organisation ihre Materialien unabhängig voneinander über das Portal verwalten können, so dass die Pflege des Portals insgesamt an die jeweilige Organisationsstruktur angepasst werden kann.

Zur Bewältigung ihrer Aufgaben stehen internen Benutzern eine Reihe von Diensten zur Verfügung. Hierzu zählen u.a. die Verwaltung von Versionen, die Nutzung persönlicher Sammelmappen, ein Empfehlungsdienst [Die01], ein Awareness-Dienst, ein Dienst zur automatischen begrifflichen Klassifikation von neuen Volltext-Ressourcen [Büc02] sowie ein Dienst zur Zusammenführung verschiedener Portalbestände [Rie03].

Da die Kooperation der internen Benutzer im Kontext und nach den Regeln der Organisation stattfindet, kann ein großer Teil der Zusatzinformationen zu den im Portal erfassten Metadaten bereits bei der Initialisierung des Portals fest vorgegeben werden. Ein Teil der Klassifikationssysteme ist jedoch dynamisch anpassbar. Diese werden durch eine spezielle Gruppe interner Benutzer gepflegt, denen hierdurch die Funktion einer *Redaktion* für das Portal zukommt. Die dynamische Anpassung von Klassifikationssystemen unterstützt die exak-

te Einordnung von Ressourcen in die aktuell in der Organisation existierenden Anwendungsdomänen.

Ein Teil der im Portal gespeicherten Ressourcen und Metadaten wird für die Außendarstellung der Organisation verwendet, indem sie externen Benutzern zur Verfügung gestellt werden, die darin navigieren und suchen können. Welche Informationen für externe Benutzer sichtbar sind, wird ebenfalls über das Berechtigungskonzept des infoAssetBrokers gesteuert.

6.1.2 Struktur von Metadaten

Die Struktur der Metadaten in einem infoAssetBroker-Portal lässt sich mit Hilfe des Metadatenmodells KooMet beschreiben (vgl. Abschnitt 4.1). Da es sich bei KooMet um ein konzeptuelles Modell handelt, das verschiedene Kooperationsszenarien gleichermaßen unterstützt, fließen jedoch nur einige der in ihm enthaltenen Konzepte zur Strukturierung von Metadaten in das tatsächliche Datenmodell des infoAssetBrokers ein, während andere eingeschränkt werden oder nicht in Erscheinung treten. Diejenigen Konzepte zur Strukturierung von Metadaten, die in infoAssetBroker-Portalen eine besondere Rolle spielen, werden im Folgenden vor dem Hintergrund des für den infoAssetBroker beschriebenen Kooperationsszenariums (vgl. Abschnitt 6.1.1) erläutert.

Beschreibung einer Ressource durch genau einen Record

In einem infoAssetBroker-Portal wird jede Ressource durch genau ein Asset repräsentiert, das einem Metadaten-Record entspricht (vgl. Abb. 6.2). Die Bezeichnung „Asset“ trägt dabei der Tatsache Rechnung, dass es sich sowohl bei den Metadaten-Records als teilweise auch bei den repräsentierten Ressourcen um informationelle Vermögenswerte der betreibenden Organisation handelt. Die folgende Korrespondenztabelle fasst den Zusammenhang zwischen Assets, Records und Ressourcen zusammen:

infoAssetBroker	**Zusammenhang**	**KooMet**
Asset	entspricht	Record
Asset	repräsentiert	Ressource

Da im infoAssetBroker keine Trennung zwischen Ressourcen und den sie repräsentierenden Assets besteht, ist die in KooMet vorgesehene Möglichkeit der Ressourcebeschreibung durch multiple Metadaten (vgl. Abschnitt 2.3.1, Vorgehensweise–2, sowie Abschnitt 4.1) hier nicht gegeben. Die dadurch erreichte eindeutige Beschreibung der Ressourcen durch genau ein Asset besitzt im Kooperationsszenarium des infoAssetBrokers den Vorteil, dass innerhalb der Organisation ein einheitlicher Blick auf die verwendeten Ressourcen besteht, der für die Zusammenarbeit eine erste organisationsweite Diskussionsgrundlage bietet. Als ein weiterer Vorteil lässt sich die Vereinfachung der Qualitätskontrolle für die Metadaten innerhalb des Portals aufführen, die sich dadurch ergibt, dass ein und dasselbe Asset von mehreren Akteuren gepflegt und verbessert wird.

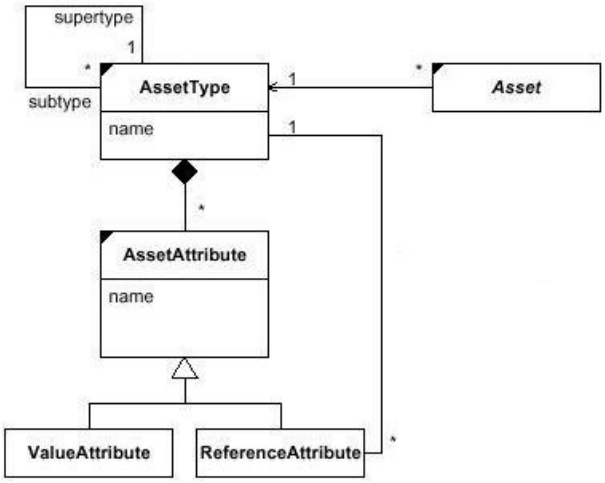

Abbildung 6.2: Aufbau von Asset-Typen im infoAssetBroker [Weg02a]

Belegung von Attributen mit einfachen Werte und Ressourcen

Zur weiteren Vereinheitlichung der Metadaten innerhalb der Organisation ist die Struktur eines jeden Assets durch einen AssetType fest vorgegeben (vgl. Abb. 6.2), der die Attribute des Assets (AssetAttributes) umfasst (vgl. hierzu auch Abschnitt 6.1.3). Hierbei kann es sich zum einen um ValueAttributes handeln, denen einfache Werte zugewiesen werden, und zum anderen um ReferenceAttributes, denen wiederum Ressourcen wie Grafikdateien als Werte zugeordnet werden. Attribute, die mit komplexen Werten belegt werden (vgl. Abschnitt 4.1), sind im infoAssetBroker nicht vorgesehen. Die folgende Korrespondenztabelle fasst die geschilderten Zusammenhänge zwischen den Attributen und Werten des infoAssetBrokers einerseits und des Metadatenmodells KooMet andererseits zusammen:

infoAssetBroker	**Zusammenhang**	**KooMet**
AssetAttribute	entspricht	Attribut
ValueAttribute	entspricht	Attribut, das in Eigenschaften nur mit Literalen belegt ist
ReferenceAttribute	entspricht	Attribut, das in Eigenschaften nur mit Ressourcen belegt ist
–	–	Attribut, das in Eigenschaften nur mit Kollektionen oder Records belegt ist

Durch die Beschränkung auf einfache Werte wird erreicht, dass viele Mitglieder einer Organisation das Portal aktiv zur Metadatenerfassung nutzen können,

ohne dass dafür zuvor ein tiefes Spezialwissen über den Aufbau komplexer Metadatenstrukturen erlernt werden muss. Die Möglichkeit, Ressourcen als Attributwerte zu verwenden, wird hingegen den Anforderungen an ein Portal als Repräsentationsplattform gerecht, wo häufig auch auf multimediale Darstellungsformen zurückgegriffen wird. Abbildung 6.3 zeigt in diesem Zusammenhang die Beschreibung eines Projekts innerhalb eines infoAssetBroker-Portals, die sowohl auf Grafik- als auch auf Video-Ressourcen zurückgreift.

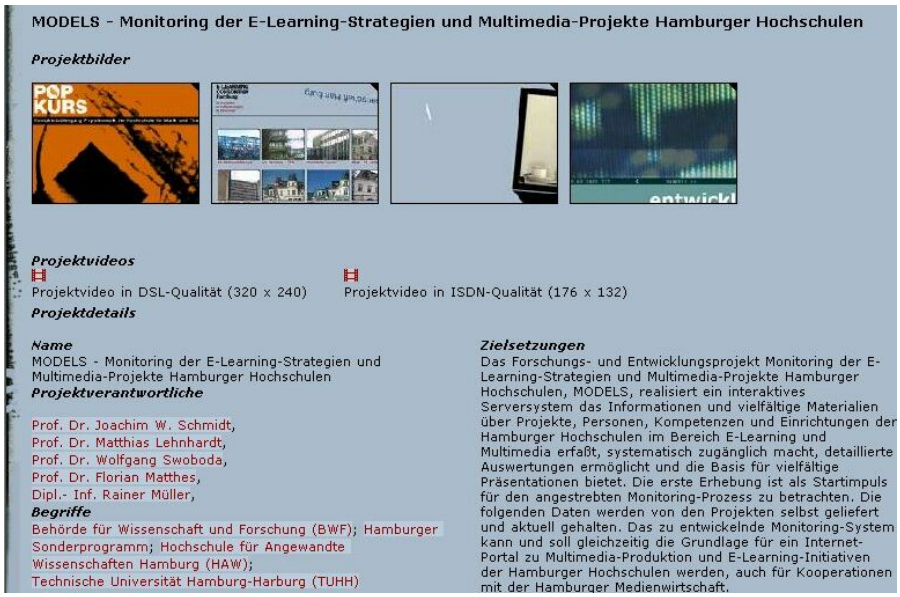

Abbildung 6.3: Verwendung von Multimedia-Dateien im infoAssetBroker

Erfassung von Beziehungen zwischen Ressourcen

Ein weiteres Kennzeichen für den Portalcharakter des infoAssetBrokers ist mit der Möglichkeit gegeben, Beziehungen (Relationships) zwischen verschiedenen Assets aufzubauen (vgl. Abb. 6.4). Relationships werden in infoAssetBroker-Portalen insbesondere eingesetzt, um bidirektional zwischen in Verbindung stehenden Ressourcen zu navigieren, wodurch sich individuelle Pfade für die Erkundung der im Portal vorhandenen Inhalte ergeben. Aus diesem Grund verbindet eine Relationship, anders als im Metadatenmodell KooMet, stets genau zwei Assets miteinander.

Ebenso wie die Beziehungen in KooMet lassen sich auch Relationships weiter spezialisieren. Hierbei kann die Art der durch die spezielle Relationship verbundenen Assets ebenfalls verfeinert werden (vgl. z.B. Abb. 6.4, Klasse Annotation). Die folgende Korrespondenztabelle stellt den Zusammenhang zwischen Beziehungen im infoAssetBroker und im Metadatenmodell KooMet her:

infoAssetBroker	**Zusammenhang**	**KooMet**
Relationship	entspricht	*(binärer)* Beziehung

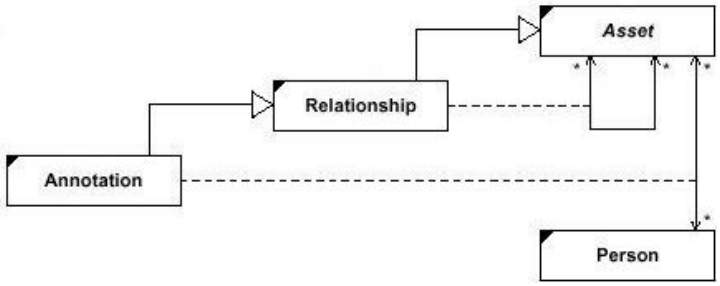

Abbildung 6.4: Aufbau von Beziehungen im infoAssetBroker [Weg02a]

Da alle Informationen zu einer Relationship innerhalb eines zentralen Objekts verwaltet werden, sind für die Erfassung und Änderung von Beziehungen nur wenige Arbeitsschritte notwendig, während die referentielle Integrität zwischen den Assets erhalten bleibt.

6.1.3 Typisierung

Das Metadatenmodell KooMet gestattet sowohl die Erfassung untypisierter Metadaten als auch eine Typisierung von Metadaten in variabler Granularität (vgl. Abschnitt 4.2). Das konkrete Datenmodell des infoAssetBrokers sieht die Typisierung für alle erfassten Assets und die darin enthaltenen Werte vor und unterstützt so die enge Kooperation innerhalb von Organisationen.

Vorgegebene Record-Typen

Im infoAssetBroker ist für jede Art von Asset ein fester AssetType vorgesehen, der vorgibt, durch welche AssetAttributes Assets beschrieben werden (vgl. Abb. 6.2). AssetTypes entsprechen somit Record-Typen im Metadatenmodell KooMet (vgl. Abschnitt 4.2.3), mit der zusätzlichen Bedingung, dass ihre Vergabe zwingend erforderlich ist und nicht nach eigenem Ermessen des Erfassers erfolgen kann:

infoAssetBroker	**Zusammenhang**	**KooMet**
AssetType	entspricht	*(obligatorischer)* Record-Typ

Feste AssetTypes sind ein weiterer Schritt in Richtung einer organisationsweit einheitlichen Sicht auf die in einem infoAssetBroker-Portal erfassten Metadaten und entsprechen daher dem für den infoAssetBroker vorgesehen Kooperationsszenarium (vgl. Abschnitt 6.1.1).

Zugleich können feste Record-Typen direkt auf die Datenmodelle von Datenbanksystemen, wie auf das relationale Datenmodell (vgl. Abschnitt 3.1), abgebildet werden, so dass die Metadaten des infoAssetBrokers ohne Aufwand in kommerziellen persistenten Datenspeichern abgelegt werden können (vgl. Abschnitt 6.1.5). Hierdurch stehen zum einen ausgereifte Methoden zur effizienten Suche und Sortierung bereit. Zum anderen können aber auch weitere Funktiona-

litäten eines Datenbanksystems genutzt werden, wie das Anlegen von Backups oder die Durchführung von Recovery-Maßnahmen, die insbesondere im organisatorischen Kontext von hoher Relevanz sind.

Feste Record-Typen besitzen jedoch den Nachteil, dass die Flexibilität bei der Metadatenerfassung eingeschränkt wird. Insbesondere bei der Erfassung von Metadaten für neue Arbeitsaufgaben kann es vorkommen, dass der erfassende Akteur in einer festen Record-Struktur kein Attribut vorfindet, durch das eine bestimmte, für die Aufgabe relevante Eigenschaft einer Ressource beschrieben werden kann. Um diesem Problem entgegenzuwirken, sieht das Datenmodell des infoAssetBrokers einerseits von vornherein für alle Assets breite Record-Strukturen mit einer Vielzahl von möglichen Attributen vor. Andererseits bietet die Software-Architektur des Systems die Möglichkeit, Record-Strukturen außerhalb des laufenden Betriebs in wenigen Schritten um zusätzlich benötigte Attribute erweitern zu können [Weg02a].

Vorgegebene Typen für Attributwerte

Ebenso wie die Struktur der Records sind auch die Typen für alle Attributwerte im infoAssetBroker fest vorgegeben, so dass auch auf der Wertebene zum einen ein einheitlicher Blick auf die Metadaten im Portal entsteht und zum anderen der Einsatz effizienter Speichertechnologie gewährleistet wird.

Daneben werden die Typinformationen zu den Attributwerten in der Benutzerschnittstelle des Portals verwendet, um syntaktische Fehler bereits bei der Metadatenerfassung zu erkennen und abzufangen, so dass sich Erfassungsfehler in geringerem Maße auf die Qualität des Metadatenbestandes auswirken.

6.1.4 Verwendung von Klassifikationssystemen

Die durch das Metadatenmodell KooMet vorgegeben Konzepte zur Klassifikation von Ressourcen (vgl. Abschnitt 4.3) werden einerseits als Möglichkeit zum Aufbau beliebiger Klassifikationssysteme im konzeptuellen Modell der Dienstschicht des infoAssetBrokers übernommen (vgl. Abb. 6.5). Klassen werden hier als Kategorien und die Zuordnung zu einer Klasse als Klassifikation bezeichnet. Anders als im Metadatenmodell KooMet sind zunächst keine Beziehungen zwischen den Kategorien vorgesehen:

infoAssetBroker	**Zusammenhang**	**KooMet**
Kategorie	entspricht	Klasse
Klassifikation	entspricht	Zuordnung

Darüber hinaus umfasst das Datenmodell des infoAssetBrokers jedoch bereits verschiedene Ausprägungen von Klassifikationssystemen, die den Benutzern eines infoAssetBroker-Portals standardmäßig zur Verfügung stehen. So treten Klassen bei der Festlegung von Record-Strukturen zu Tage, werden zur begrifflichen Einordnung und Navigation verwendet und haben Anteil am Berechtigungskonzept des infoAssetBrokers.

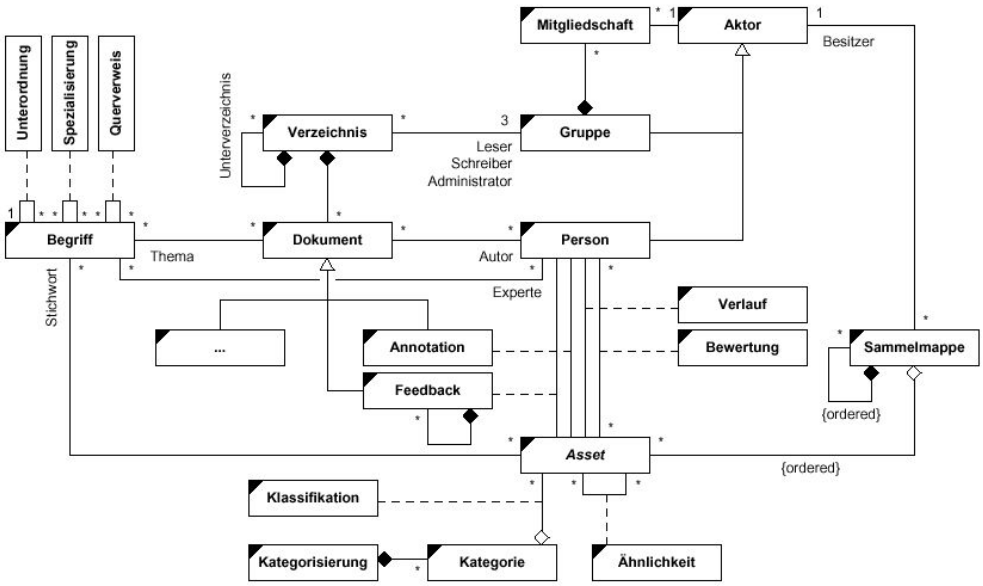

Abbildung 6.5: Konzeptuelles Modell der Dienstschicht des infoAssetBrokers [Weg02a]

Klassifikation nach Art der beschriebenen Ressource

Innerhalb eines infoAssetBroker-Portals können je nach den in der betreibenden Organisation anfallenden Aufgabenstellungen verschiedene Arten von Assets verwaltet werden. Dabei zählen Personen und Personengruppen ebenso zu den Ressourcen der Organisation wie Dokumente oder Gegenstände. So können beispielsweise in einem E-Learning-Portal Lerneinheiten und Tests sowie Informationen über Lernende, Lehrende, Übungsgruppen usw. als relevante Ressourcen identifiziert werden [Koc02], während im Portal eines Handelsunternehmens Produktbeschreibungen, Bestellungen sowie Informationen über Kunden oder Konkurrenzunternehmen zur Verfügung gestellt werden. Im konzeptuellen Klassendiagramm in Abbildung 6.5, das die grundlegenden Klassen der in infoAssetBroker-Portalen angebotenen Dienste in Zusammenhang bringt, sowie in den Abbildungen 6.2 und 6.4 sind die Klassen, die eine Spezialisierung der Asset-Klasse darstellen, mit einem schwarzen Dreieck in der oberen linken Ecke gekennzeichnet.

Die Einteilung der Assets in verschiedene Arten stellt eine semantische Gruppierung dar, so dass jede konkrete Asset-Subklasse zugleich einer Klasse im Sinne des Metadatenmodells KooMet (vgl. Abschnitt 4.3) entspricht:

infoAssetBroker	**Zusammenhang**	**KooMet**
Subklasse von Asset	instanziiert	Klasse

Die in einem infoAssetBroker-Portal vorhandenen Arten von Assets werden bereits beim Aufsetzen des Systems festgelegt, so dass es sich bei dem auf ihnen

beruhenden Klassifikationssystem um eine fest vorgegebene Zusatzinformation handelt. Die verschiedenen Arten von Assets werden durch verschiedene Eigenschaften beschrieben, so dass jeder Asset-Art durch das Datenmodell des infoAssetBrokers ein eigener AssetType zugewiesen ist (vgl. Abschnitt 6.1.3).

Die Zuordnung einer Ressource zu einer Asset-Art wird im infoAssetBroker nicht weiter beschrieben. Dass es sich bei einer solchen Zuordnung semantisch um eine Instanzbeziehung handelt, wird bereits aus der Bezeichnung der Asset-Arten deutlich. Untereinander stehen die Asset-Arten nicht in Beziehung, sondern bilden einfach eine Menge gleichwertiger Teilmengen des Portalbestands.

Da zur Lösung von kooperativen Arbeitsaufgaben zumeist bestimmte Arten von Assets für bestimmte Arbeitsschritte herangezogen werden, stellt die Unterteilung des Portalbestands nach Art der beschriebenen Ressourcen ein erstes grobes Leitsystem dar, an dem sich die Akteure innerhalb eines infoAssetBroker-Portals orientieren können, wenn sie Assets für einen solchen Arbeitsschritt benötigen. Asset-Arten erscheinen daher zumeist in der grafischen Benutzerschnittstelle eines infoAssetBroker-Portals, wo sie zum einen als Menupunkte eine Navigation in die entsprechende Teilmenge des Portalbestandes hinein erlauben und wo für sie zum anderen unterschiedliche Suchformulare zur Verfügung gestellt werden, die entsprechend dem AssetType der jeweiligen Asset-Art aufgebaut sind (vgl. Abb. 6.6, links).

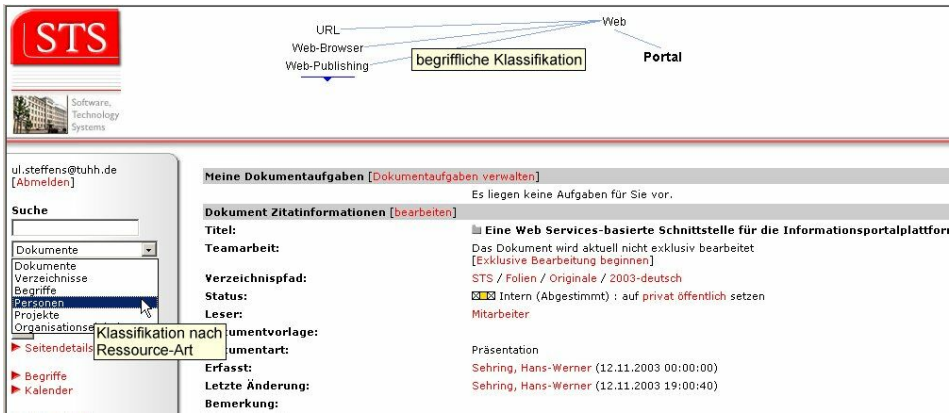

Abbildung 6.6: Klassifikation nach Art der beschriebenen Ressource und begriffliche Klassifikation im infoAssetBroker

Begriffliche Klassifikation

Bei der Lösung der verschiedenen kooperativen Aufgaben innerhalb einer Organisation treten immer wieder Begriffe in Erscheinung, die in ihrer Gesamtheit ein für die Organisation spezifisches Fachvokabular bilden, das von allen Organisationsmitgliedern verstanden wird.

Die Klassifikation von Assets anhand dieses Fachvokabulars ist ebenfalls im infoAssetBroker vorgesehen. Da die verwendeten Begriffe im Vergleich zu den verschiedenen Asset-Arten eine detailliertere Einordnung von Ressourcen gestatten und zugleich einem stärkeren und rascheren Wandel unterzogen sind,

kann das Vokabular eines infoAssetBroker-Portals von den Redakteuren direkt über die Benutzerschnittstelle des laufenden Systems an die aktuellen Gegebenheiten innerhalb der Organisation angepasst werden (vgl. hierzu auch Abb. 6.1), so dass sich im Ergebnis alle verwendeten Assets mit großer Genauigkeit begrifflich klassifizieren lassen. Da das Vokabular als Kommunikationsgrundlage zur Lösung kooperativer Aufgaben innerhalb der Organisation gilt, stellt es eine Zusatzinformation dar, die einheitlich von allen Akteuren verwendet wird.

Die Zuordnung eines Assets zu einem Begriff des Fachvokabulars repräsentiert die Tatsache, dass dieses Asset weiterführende Informationen zu diesem Begriff umfasst. So kann der Begriff beispielsweise eines der Themen in einem Dokument oder das Spezialgebiet einer Person darstellen (vgl. Abb. 6.5). Die Zuordnung von Assets zu Begriffen ist optional. Insbesondere wenn Assets neue Begriffe behandeln, die von der Redaktion des Portals noch nicht in das Vokabular übernommen worden sind, können diese auch ohne begriffliche Klassifikation abgelegt werden.

Untereinander stehen die Begriffe eines infoAssetBroker-Portals ebenfalls in Beziehung (vgl. Abb. 6.5). Einerseits bilden sie über *Spezialisierungsbeziehungen* eine Begriffshierarchie, andererseits verbinden *Querverweise* inhaltlich zusammengehörige Begriffe, so dass insgesamt ein *Begriffsnetz* entsteht. Weitere Beziehungsklassen zwischen Begriffen sind im infoAssetBroker, anders als im allgemeinen Metadatenmodell KooMet, nicht vorgesehen. Die Beschränkung auf wenige Beziehungsklassen begünstigt die Übersichtlichkeit und Verständlichkeit des Begriffsnetzes und damit seine einfache Verwendung durch eine Vielzahl von Akteuren.

Insgesamt bestehen zwischen der Verwendung von Begriffsnetzen im infoAssetBroker und der allgemeineren Klassifikation von Ressourcen im Metadatenmodell KooMet die folgenden Zusammenhänge:

infoAssetBroker	**Zusammenhang**	**KooMet**
Begriff	spezialisiert	Klasse
Assoziation Asset–Begriff	spezialisiert	Zuordnung
Assoziation Person–Begriff	spezialisiert	Zuordnung
Assoziation Dokument–Begriff	spezialisiert	Zuordnung
Unterordnung	spezialisiert	Klassenbeziehung
Spezialisierung	spezialisiert	Klassenbeziehung
Querverweis	spezialisiert	Klassenbeziehung

Die Aufzählung der Assoziationen zwischen den verschiedenen Subklassen der Klasse Asset und der Klasse Begriff besitzt dabei keinen Anspruch auf Vollständigkeit, sondern nennt lediglich Beispiele der begrifflichen Klassifikation als einer Spezialisierung der Zuordnung von Ressourcen zu Klassen, die gemäß des Metadatenmodells KooMet eine Anwendungsdomäne repräsentieren.

Die Benutzerschnittstelle eines infoAssetBroker-Portals verwendet das Begriffsnetz zur Navigation durch das Vokabular der Organisation, so dass ein Akteur von einem von ihm besuchten Begriff direkt zu benachbarten und damit inhaltlich verwandten Begriffen und den durch sie klassifizierten Assets gelan-

gen kann. Für die Navigation werden sowohl Hyperlinks eingesetzt als auch ein grafischer Begriffsnavigator (vgl. Abb. 6.6, oben). Neben der Navigation führt auch die direkte Suche über einen Begriff zu den durch ihn klassifizierten Assets.

Eine denkbare Erweiterung zu einem organisationsweit einheitlichen Begriffsnetz, wie es in einem infoAssetBroker-Portal besteht, stellt die in [MNS01] beschriebene Verwaltung mehrerer Vokabulare dar, die beispielsweise auf persönlicher oder auf Gruppenebene gepflegt und in regelmäßigen Abständen mit dem allgemeinen Organisationsvokabular zusammengeführt werden. Zur Zusammenführung wird dabei, ähnlich wie bei der Zusammenführung von Ressourcebeständen [Rie03], die Identifikation ähnlicher Begriffe durchgeführt [Cha01] und im Anschluss eine Entscheidung über die Vereinheitlichung von Begriffspaaren getroffen [Tre01]. Für beide Schritte werden die Beziehungen der Begriffe innerhalb des Begriffsnetzes und zu den ihnen zugeordneten Ressourcen berücksichtigt.

Klassifikation nach Ablageort

Eine klassische Methode bei der Verwaltung kooperativ genutzter digitaler Ressourcen ist die Nutzung eines gemeinsamen Dateisystems. Auch der infoAssetBroker stellt einen Verzeichnisbaum zur Verfügung, in den sich Dokumente als spezielle Art von Assets einordnen und damit auch klassifizieren lassen [RMS+01]. Verzeichnisse können dabei als Klassen angesehen werden, denen Dokumente als Ressourcen zugeordnet werden und die untereinander über Unterverzeichnisbeziehungen verbunden sind:

infoAssetBroker	**Zusammenhang**	**KooMet**
Verzeichnis	spezialisiert	Klasse
Assoziation Dokument–Verzeichnis	spezialisiert	Zuordnung
Assoziation Verzeichnis–Verzeichnis	spezialisiert	Klassenbeziehung

Die Klassifikation von Dokumenten innerhalb von Verzeichnissen bietet zum einen den Vorteil, dass Akteure insbesondere in neu installierten infoAssetBroker-Portalen ein ihnen bereits vertrautes Leitsystem vorfinden. Zum anderen verwendet der infoAssetBroker Verzeichnisse zur Realisierung seines Berechtigungskonzepts. Jedem Verzeichnis können dabei Benutzergruppen zugewiesen werden, die auf die Dokumente in diesem Verzeichnis lesend oder schreibend zugreifen oder die Eigenschaften des Verzeichnisses selbst verändern dürfen (vgl. Abb. 6.5). Durch die dedizierte Rechtevergabe können Verzeichnisse als virtuelle Gruppenarbeits- und leseräume für kooperative Projekte [RMSS03] genutzt werden, die für Organisationsmitglieder außerhalb der Gruppe unzugänglich sind.

6.1.5 Software-Architektur

Der infoAssetBroker ist in einer mehrschichtigen Client-/Server-Architektur realisiert, die sich modular erweitern lässt. Die Module implementieren zunächst grundlegende Dienste wie die Suche im Metadatenbestand, eine Linkverwaltung und den Schutz von Dokumenten. Darüber hinaus können jedoch auch Module

angebunden werden, die Funktionalitäten für einen spezifischen Anwendungskontext bieten, wie eine Zahlungsabwicklung [Brö00] für kommerziell genutzte Assets, Kompetenztests im Rahmen einer Personalverwaltung [Weg02a] oder die Organisation von Lehrveranstaltungen im E-Learning-Kontext [Koc02]. Beim Aufbau eines neuen Portals werden die benötigten Module in Abhängigkeit vom entsprechenden Anwendungskontext kombiniert.

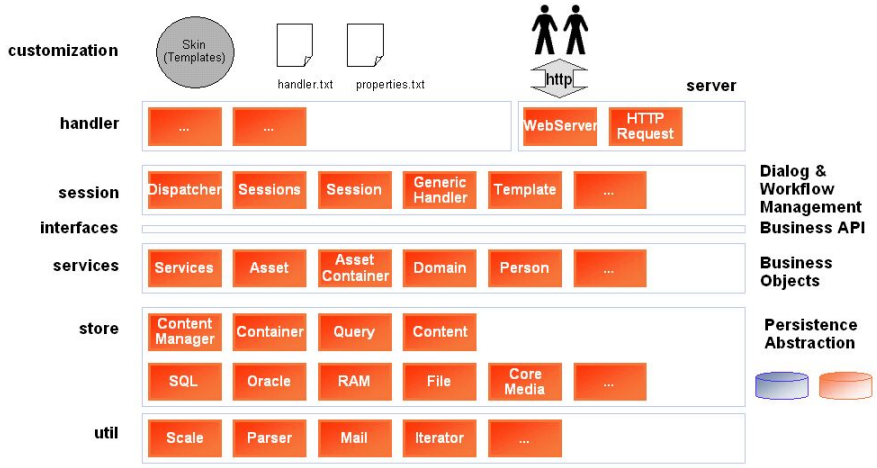

Abbildung 6.7: Schichten-Architektur des infoAssetBrokers

Die Schichtenarchitektur des infoAssetBrokers (vgl. Abb. 6.7) ist in Java implementiert und durch Java-Packages strukturiert. Die Packages server, handler und session bilden die Grundlage für eine Web-Benutzerschnittstelle, indem sie einen Web-Server, Klassen zur Generierung von statischen und dynamischen HTML-Seiten auf der Basis von Templates sowie Klassen für das Dialog- und Workflow-Management zur Verfügung stellen. Durch den Einsatz von Template-Technologie kann die Benutzerschnittstelle sowohl an die grafischen als auch an die inhaltlichen Anforderungen der betreibenden Organisation angepasst werden, so dass für jedes infoAssetBroker-Portal Layout, Darstellung und Aufteilung der Inhalte sowie Workflows für die Arbeit im Portal individuell gestaltet werden können [RMS+01]. Abbildung 6.8 zeigt in diesem Zusammenhang zwei verschiedene infoAssetBroker-Portale, von denen eines ein reich bebildertes Layout besitzt, während das zweite eher informationsbetont aufgemacht ist.

Das Package services stellt allen höheren Schichten die Kerndienste des infoAssetBrokers über entsprechende Schnittstellen zur Verfügung. In dieser Schicht sind insbesondere die Klasse Asset und ihre Subklassen realisiert.

Die Persistenzabstraktion für die Dienste und somit auch die Abbildung der Record-Typen auf den für ein infoAssetBroker-Portal gewählten Datenspeicher, der z.B. in Form eines relationalen Datenbanksystems (vgl. Abschnitt 3.1), eines Content-Management-Systems [SWBH00] oder des Dateisystems gegeben sein kann, wird im package store umgesetzt. Hier sind auch die Klassen angesiedelt, die benötigt werden, um in der Schemadefinition eines infoAssetBroker-Portals Record-Typen um weitere Attribute zu ergänzen (vgl. Abschnitt 6.1.3).

Abbildung 6.8: infoAssetBroker-Portale in unterschiedlichem Layout

6.1.6 Einsatz toleranter Algorithmen

Neben der Verwendung von Fuzzy-Prädikaten (vgl. Abschnitt 5.2) existiert eine Reihe weiterer Verfahren für die Implementation toleranter Algorithmen für gemeinsame Informationsräume. Innerhalb des infoAssetBrokers kommen vektorbasierte Verfahren wie das *k-Nearest-Neighbor-Verfahren* (*kNN-Verfahren*) [BEK+98] oder Verfahren auf der Grundlage von *Support Vector Machines* (*SVM*) [Joa98] zum Einsatz, um drei verschiedene Arten toleranter Algorithmen zu realisieren [Büc02]:

Automatische Klassifikation: Ressourcen werden anhand ihrer Eigenschaften in einem bestehenden Begriffsnetz (vgl. Abschnitt 6.1.4) klassifiziert, ohne dass die Zuordnung zu den Begriffen explizit von einem Metadatenerfasser angegeben wird (vgl. Abb. 6.9, oben).

Clustering: Über einer Menge von Ressourcen werden Cluster gebildet, die zueinander ähnliche Ressourcen zusammenfassen. Diese Cluster können dann die Grundlage für ein neues, noch nicht erfasstes Klassifikationssystem, wie beispielsweise für ein neues Begriffsnetz, darstellen (vgl. Abb. 6.9, Mitte).

Assoziative Suche: Für eine gegebene Ressource werden ähnliche Ressourcen gefunden, ohne dass diese Ähnlichkeiten explizit innerhalb der Metadaten erfasst werden (vgl. Abb. 6.9, unten).

Alle drei Verfahren basieren auf der Darstellung von Ressource-Eigenschaften in mehrdimensionalen Vektorräumen, wobei jede Eigenschaft auf eine der Dimensionen abgebildet wird. Eine Ressource entspricht dabei einem bestimmten Vektor innerhalb des Vektorraums, und der Abstand zwischen zwei Vektoren repräsentiert die Ähnlichkeit der entsprechenden Ressourcen. Je geringer der Abstand der Vektoren ist, desto ähnlicher sind sich die Ressourcen in den durch den Vektorraum dargestellten Eigenschaften.

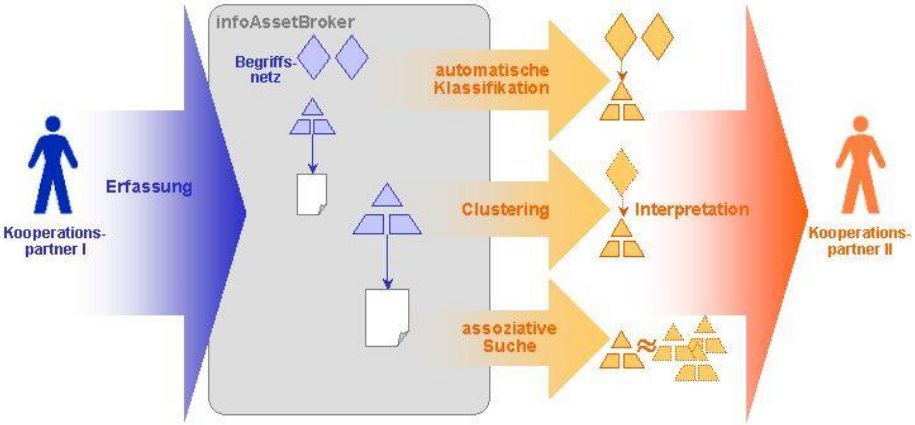

Abbildung 6.9: Verwendung toleranter Algorithmen im infoAssetBroker

6.2 Lose gekoppelte Kooperation in PIA

Das Katalogsystem *PIA (Personal Information Assistant)* wurde im Verlauf des Projekts Kolibri [MWS00] im Rahmen des DFG-Forschungsschwerpunkts „Verteilte Verarbeitung und Vermittlung digitaler Dokumente“ entwickelt. Es vermittelt zwischen mehreren, voneinander unabhängigen Anbietern und ihren potentiellen Kunden, indem es den Anbietern ermöglicht, Metadaten in Form von Produktbeschreibungen in einem online-Katalog zu veröffentlichen. Die Erfassung und Nutzung dieser Produktbeschreibungen unterliegt dabei dem Einfluss des durch PIA realisierten Szenariums einer nur lose gekoppelten Kooperation, bei der die Hauptaufgabe des Systems in der Vermittlung zwischen Partnern liegt, die aus verschiedenen Kontexten stammen und einander nicht kennen.

6.2.1 Kooperationsszenarium

PIA vermittelt zwischen *Anbietern*, die die von ihnen angebotenen Produkte durch Metadaten in PIA-Katalogen beschreiben, und *Kunden*, die nach Produktbeschreibungen suchen, um anhand dieser Beschreibungen eine Kaufentscheidung zu treffen (vgl. Abb. 6.10). Bei den Ressourcen, die in PIA durch Metadaten beschrieben sind, handelt es sich demnach hauptsächlich um Produkte wie Waren oder Dienstleistungen. Darüber hinaus können aber auch Metadaten für andere Arten von Ressourcen, wie für Dokumente, Projekte oder Personen, in einem PIA-Katalog verwaltet werden. PIA arbeitet dabei ausschließlich auf Metadaten. Digitale Ressourcen selbst werden hier nicht gespeichert. PIA-Kataloge können sowohl homogene als auch eher heterogene Produktmengen beschreiben, wie einerseits die Produktpalette eines einzelnen Herstellers oder andererseits das komplette Angebot eines Versandhauses. Insbesondere kann ein PIA-Katalog Produktbeschreibungen verschiedener, voneinander unabhängiger Anbieter umfassen.

Anbieter haben in PIA die Freiheit, ihre Produkte ihren eigenen Wünschen

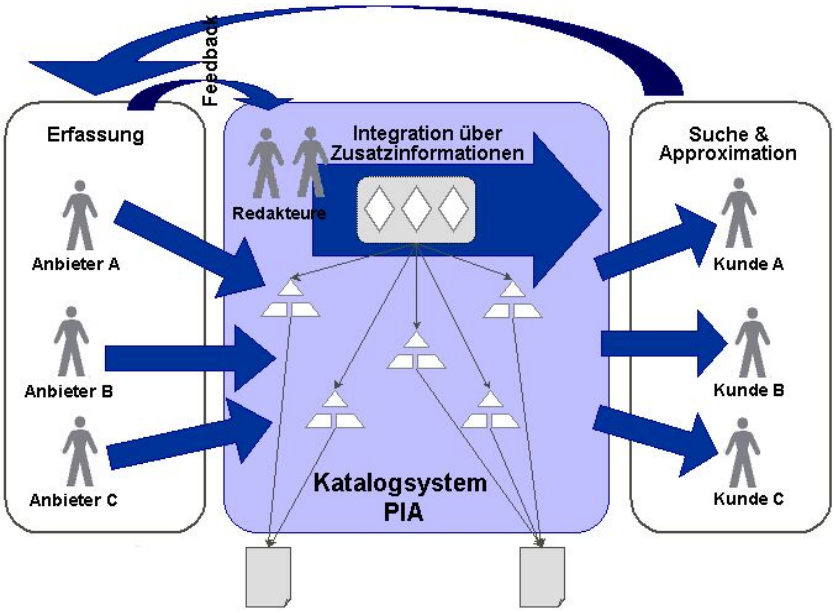

Abbildung 6.10: Zusammenarbeit in PIA

entsprechend darzustellen, ohne dabei an ein rigides Schema gebunden zu sein. Insbesondere können so die vorteilhaften Eigenschaften eines Produkts in den Vordergrund gestellt und andere Eigenschaften, die nur durchschnittlich oder sogar nachteilig sind, bei der Produktbeschreibung unerwähnt gelassen werden.

Um die Vergleichbarkeit der verschiedenen Produktbeschreibungen in einem PIA-Katalog zu gewährleisten, werden in PIA eine Reihe schematisierter Zusatzinformationen bereitgestellt, an denen sich Anbieter bei der Erfassung von Produktbeschreibungen orientieren müssen. Um den Anbietern dennoch die notwendige Flexibilität einzuräumen, sind die Zusatzinformationen in PIA weitestgehend im laufenden Betrieb anpassbar. Notwendige Anpassungen werden von einer besonderen Gruppe von Akteuren, den *Redakteuren* des PIA-Systems, auf der Grundlage eines Anbieter-Feedbacks vorgenommen, das über einen Kommunikationskanal außerhalb des eigentlichen PIA-Systems übermittelt wird.

Bei Kunden setzt PIA weder die Kenntnis der durch den Katalog beschriebenen Produktpalette noch die feste Vorstellung über das gesuchte Produkt voraus. Ein Kunde kann durch mit Hilfe von Fuzzy-Prädikaten (vgl. Abschnitte 5.2 sowie 6.2.6) vage formulierte, schrittweise Suchanfragen, für die ihm Zwischenergebnisse angezeigt werden, einen Überblick über das präsentierte Angebot gewinnen und dabei zugleich sein eigenes Kaufinteresse bezüglich dieses Angebots ausmachen.

Die an PIA gestellten Suchanfragen werden persistent verwaltet [MS99]. Hierdurch können sich Kunden von PIA über etwaige neue Angebote hinsichtlich ihrer Anfragen informieren lassen, die gespeicherten Anfragen zu einem späteren Zeitpunkt erneut ausführen oder sie als Grundlage für neue, leicht

abgeänderte Anfragen nutzen, während zugleich die Anbieter Anfragen einsehen können, um Rückschlüsse auf Kundenwünsche zu ziehen und das eigene Angebot entsprechend anzupassen (vgl. Abb. 6.10, oben).

6.2.2 Struktur von Metadaten

Die Struktur der Metadaten in PIA basiert auf dem allgemeinen Metadatenmodell KooMet (vgl. Abschnitt 4.1). Ebenso wie das Datenmodell des infoAssetBrokers (vgl. Abschnitt 6.1) wurde auch das Datenmodell von PIA [MS99] an das konkrete Kooperationsszenarium angepasst, das durch dieses Datenmodell unterstützt wird, so dass auch hier nicht alle Konzepte von KooMet vollständig zur Anwendung kommen. Diejenigen Konzepte zur Strukturierung von Metadaten, die eine besondere Relevanz für die lose gekoppelte Kooperation in PIA besitzen, werden im Folgenden aufgeführt und erläutert. Anders als beim infoAssetBroker ist das Datenmodell von PIA in weiten Teilen im Aufbau sowie bei der Benennung von Klassen und Assoziationen direkt an das Metadatenmodell KooMet angelehnt, so dass in diesem und den folgenden Abschnitten auf die Darstellung durch Klassendiagramme verzichtet wird.

Flexible Kombination von Attributen in Records

Die durch PIA gegebene Flexibilität der Anbieter bei der Erfassung von Produktbeschreibungen basiert insbesondere darauf, dass Attribute in einem PIA-Record beliebig, auch mehrfach, zusammengefasst werden können, so dass die Qualitäten eines angebotenen Produkts angemessen dargestellt werden können.

Die Anbieter sind bei der Auswahl der in einem Record zu kombinierenden Attribute allerdings auf die durch die Redakteure eines PIA-Systems zur Verfügung gestellte Attributmenge eingeschränkt. Hierdurch wird für PIA-Kunden eine Vergleichbarkeit von Records gewährleistet und ausgeschlossen, dass mit einer Vielzahl verwendeter Attribute gleicher oder ähnlicher Bedeutung eine mehrdeutige Terminologie für die Beschreibung von Produkten entsteht.

Beschreibung einer Ressource durch beliebig viele Records

In PIA kann eine Ressource durch beliebig viele Records beschrieben werden. Hiermit wird der Tatsache Rechnung getragen, dass ein PIA-System von einer Menge voneinander unabhängiger Anbieter zur Beschreibung ihrer Produkte verwendet wird. Befindet sich das gleiche Produkt im Angebot mehrerer Anbieter, so muss es diesen möglich sein, dieses Produkt je nach ihrer persönlichen Verkaufsstrategie unterschiedlich und damit auch in unterschiedlichen Records zu beschreiben.

Belegung von Attributen mit einfachen und Kollektionswerten

Zur Belegung von Attributen wurden in PIA zunächst ausschließlich einfache Werte verwendet [MS99]. Diese Einschränkung wurde getroffen, damit das System für eine möglichst große Gruppe von Anbietern und Kunden verständlich und damit nutzbar bleibt.

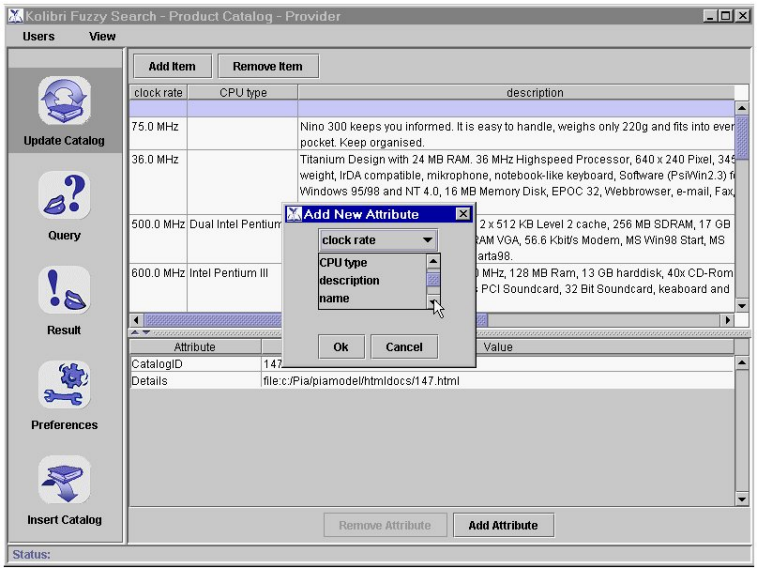

Abbildung 6.11: Auswahl eines Attributs zur Beschreibung einer Ressource in PIA

Bei ersten Anwendungstests in der Metadatenerfassung mit PIA wurde jedoch schnell erkennbar, dass das Fehlen von Kollektionswerten für ein Katalogsystem eine nicht hinnehmbare Einschränkung bedeutet, da in einer Produktbeschreibung häufig mehrere Ausführungen eines Produkts, beispielsweise in unterschiedlichen Farben oder Größen, zusammengefasst werden. Bei der aufgrund dieser Anforderung vorgenommenen Erweiterung des PIA-Datenmodells konnte die Tatsache ausgenutzt werden, dass dieses Datenmodell auf dem allgemeineren Metadatenmodell KooMet beruht, so dass das PIA-Datenmodell ohne großen Aufwand um Kollektionswerte ergänzt werden konnte.

6.2.3 Typisierung

Das PIA-Datenmodell schränkt das Konzept der freiwilligen Typisierung von Metadaten im Metadatenmodell KooMet (vgl. Abschnitt 4.2) entsprechend dem für PIA-Kataloge beschriebenen Kooperationsszenarium ein, indem es einerseits die Erfassung untypisierter Records vorsieht, die andererseits typisierte Attributwerte enthalten, wobei für numerische Werte insbesondere das Umrechnungssystem für Maßeinheiten aus KooMet übernommen wird. Im Folgenden wird beschrieben, inwiefern die Typisierungskonzepte in PIA die Zusammenarbeit über PIA-Kataloge begünstigen.

Untypisierte Records

Produkte in einem PIA-Katalog werden mit Hilfe von untypisierten Records beschrieben, so dass Anbieter die Möglichkeit haben, flexibel beliebige Eigenschaften für ein Produkt zu erfassen oder wegzulassen, um es den PIA-Kunden gegenüber vorteilhaft darzustellen (vgl. Abschnitt 6.2.2).

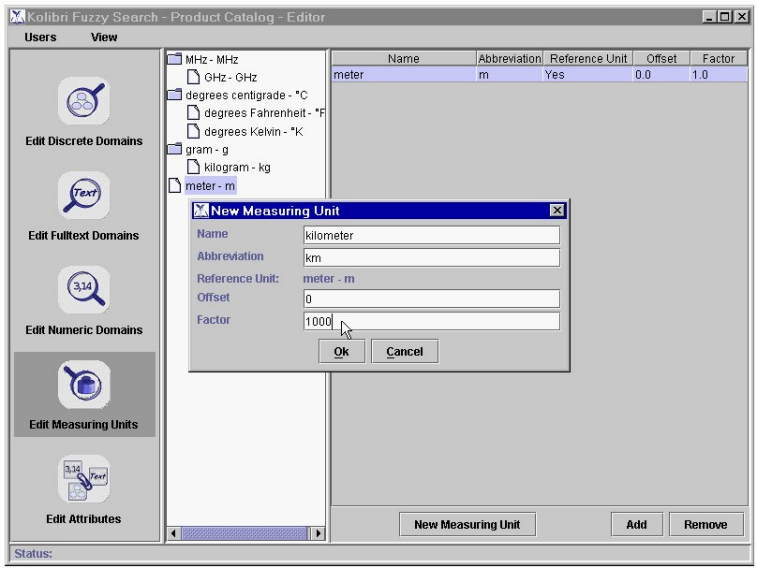

Abbildung 6.12: Verwaltung umrechenbarer Maßeinheiten in PIA

Um dennoch einen einheitlichen Zugang zu allen in PIA erfassten Metadaten zu erreichen, sind PIA-Anbieter bei der Erfassung von Eigenschaften auf eine durch die Redakteure des Katalogs festgelegte Menge von Attributen eingeschränkt (vgl. Abb. 6.11). Diese Attribute können für jedes Produkt unabhängig von seiner Produktkategorie (vgl. Abschnitt 6.2.4) verwendet werden und bilden somit das Vokabular eines PIA-Kataloges, das die semantische Grundlage sowohl für die Erfassung als auch für die Suche bildet. Für die Suche ermöglicht es insbesondere die Formulierung von unscharfen Bedingungen (vgl. Abschnitt 6.2.6) über alle erfassten Beschreibungen ohne die Kenntnis von entsprechenden Produktkategorien.

Vorgegebene Typen für Attributwerte

Ein zweiter Schritt für die effiziente und effektive Suche in PIA-Katalogen ist Vorgabe eines einheitlichen Typs für alle Werte, die einem bestimmten Attribut zugeordnet werden. Hierdurch wird erreicht, dass dem Attribut eindeutig Fuzzy-Prädikate zur Formulierung unscharfer Bedingungen zugeordnet werden können, die ohne weitere Typüberprüfungen auf den entsprechenden Attributwerten arbeiten können (vgl. Abschnitt 5.2).

Die Typisierung der Attributwerte erfolgt, ebenso wie die Festlegung der Attribute selbst, während des laufenden Betriebs eines PIA-Systems und wird durch die PIA-Redakteure vorgenommen. Diese ordnen dabei nicht nur jedem Attribut den entsprechenden Typ zu, sondern definieren auch die Typen selbst. Hierzu gibt ihnen das PIA-Datenmodell die einfachen Basistypen numerischer Typ, Volltexttyp und hierarchischer Typ (vgl. Abschnitt 6.2.4) vor, die sie entsprechend der im Katalog angebotenen Produktpalette in eigenen Typen weiter einschränken können (vgl. Abschnitt 4.2.1).

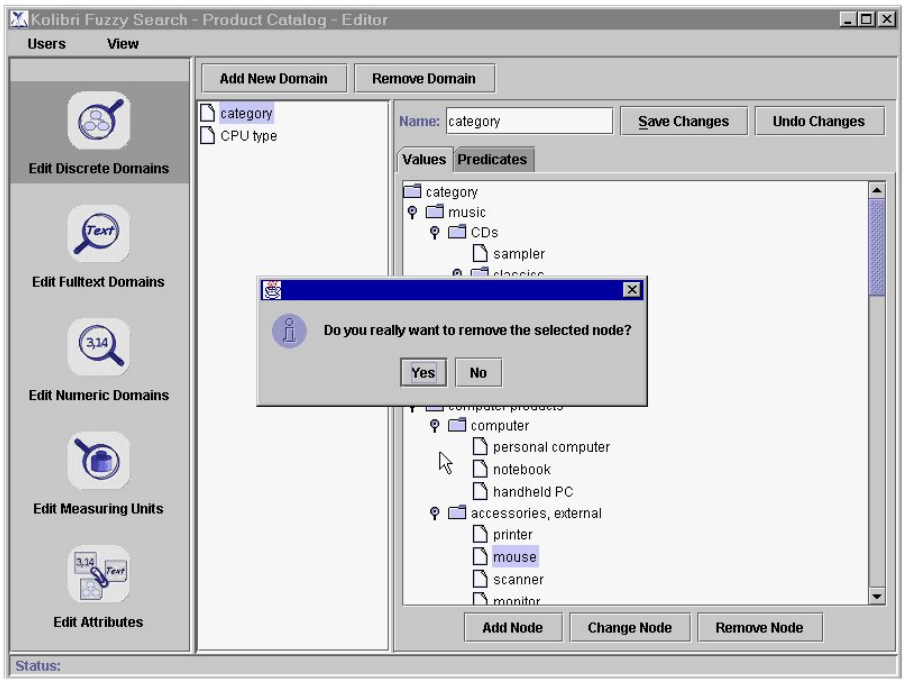

Abbildung 6.13: Verwaltung von Klassifikationssystemen über hierarchische Typen in PIA

Umrechnungssystem für Maßeinheiten numerischer Typen

Gerade bei der Beschreibung und dem Vergleich von Produkten kommen häufig numerische Werte in Verbindung mit Maßeinheiten zum Einsatz. Um zum einen PIA-Anbietern die Flexibilität bieten zu können, Eigenschaften mit von ihnen bevorzugten Maßeinheiten zu erfassen, und andererseits die Vergleichbarkeit von numerischen Werten mit unterschiedlichen, kompatiblen Maßeinheiten zu erhalten, wird im PIA-Datenmodell das in KooMet vorgestellte Konzept zur Umrechnung kompatibler Maßeinheiten (vgl. Abschnitt 4.2.1) übernommen und den Redakteuren für die Pflege numerischer Typen zugänglich gemacht (vgl. Abb. 6.12).

6.2.4 Verwendung von Klassifikationssystemen

Anders als in KooMet werden in PIA Klassifikationssysteme nicht als eigenständiges Konzept (vgl. Abschnitt 4.3), sondern als spezielle *hierarchische Typen* aufgefasst. Dabei können die Redakteure eines PIA-Katalogs beliebig viele hierarchische Typen und damit auch beliebig viele Klassifikationssysteme anlegen (vgl. Abb. 6.13).

Innerhalb eines hierarchischen Typs werden Klassen allein durch ihren Namen beschrieben, so dass sich hierarchische Typen mit Aufzählungstypen in Programmiersprachen [Wat90] vergleichen lassen. Zusätzlich sind die Klassen eines hierarchischen Typs jedoch über Spezialisierungsbeziehungen miteinander

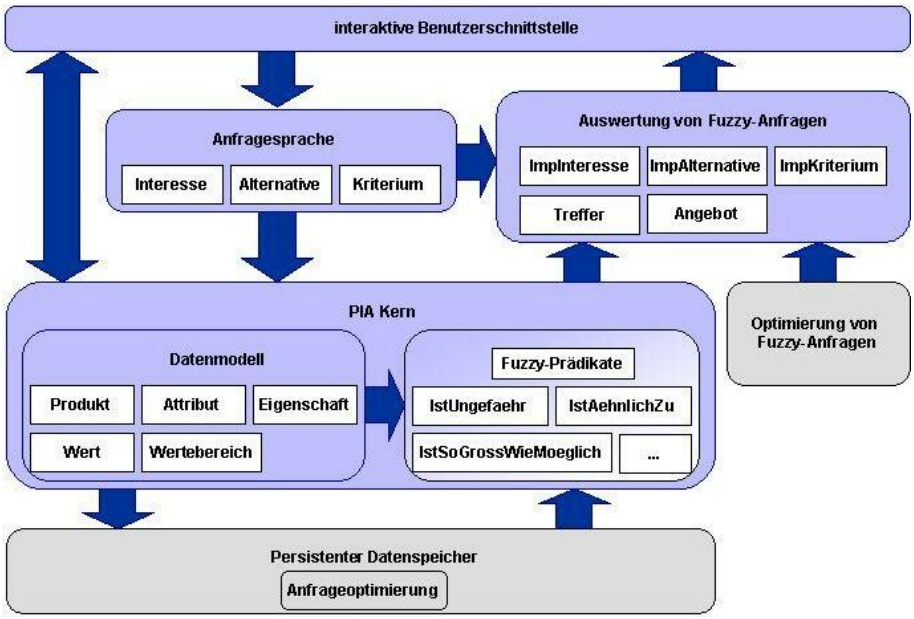

Abbildung 6.14: Architektur von PIA

verbunden [Büc99], so dass der Typ insgesamt eine Klassenhierarchie repräsentiert.

Die Modellierung von Klassifikationssystemen als hierarchische Typen hat sich bei der Verwendung von PIA als ausreichend erwiesen. Sie entspricht der Verwendung hierarchischer Klassifikationssysteme in Internet-Katalogen wie Yahoo! [LF99] und der Struktur des Angebots vieler Online-Versandanbieter. Die durch die Modellierung von Klassen als eigenständiges Konzept erlangten Vorteile, wie die Beschreibbarkeit von Klassen und ihrer Beziehungen sowohl untereinander als auch zu ihren Ressourcen, haben für die Anwendungsdomäne nur geringe Relevanz. Zudem lässt sich das Konzept der hierarchischen Typen in PIA auf das Konzept der Klassen als eigenständige Ressourcen in KooMet abbilden, so dass es in späteren Entwicklungsschritten entsprechende Erweiterungen erfahren kann.

6.2.5 Architektur

Das PIA-System ist in einer Schichten-Architektur aufgebaut (vgl. Abb. 6.14). Die obere Schicht umfasst dabei die interaktive Benutzerschnittstelle des Systems. Diese ist eine wichtige Komponente von PIA, da ihr die Aufgabe zukommt, dem Benutzer zu jeder Zeit einen möglichst vollständigen und dennoch übersichtlichen Eindruck der Möglichkeiten zu vermitteln, die ihm der entsprechende Katalog bietet. Für jede zu treffende Entscheidung, sei es für einen Wertebereich, ein Attribut oder ein Fuzzy-Prädikat, stellt die Benutzerschnittstelle eine entsprechende Liste aller möglichen Kandidaten zur Verfügung. Diese Liste vermittelt dem Benutzer einen Überblick, sie begrenzt jedoch auch die Auswahl auf Elemente, die den durch die Redakteure aufgestellten Konsistenzregeln des

Katalogs entsprechen. So werden in einer Liste der Fuzzy-Prädikate, die ein Kunde bei der Formulierung einer Anfrage für ein Attribut wählen kann, nur diejenigen Prädikate angezeigt, die tatsächlich für den Bildbereich des Attributs definiert sind. Ähnliches gilt für die Eingabe von Werten, für die unmittelbar überprüft wird, ob sie dem jeweiligen Wertebereich entsprechen, also z.B. innerhalb der Wertegrenzen eines numerischen Wertebereichs liegen.

Die Benutzerschnittstelle interagiert mit dem darunter liegenden PIA-Kern, der die Implementation des Metadatenmodells von PIA und der Fuzzy-Prädikate umfasst. Zur Anzeige und Erfassung von Metadaten werden von der Benutzerschnittstelle get- und set-Methoden der entsprechenden Klassen des Datenmodells direkt aufgerufen. Für die Formulierung von Anfragen durch PIA-Kunden stellt das System die in Abschnitt 6.2.6 näher beschriebene Anfragesprache auf der Basis der Anfragesprache KooMet-QL (vgl. Abschnitt 5.2.3) zur Verfügung. Die Benutzerschnittstelle setzt die Eingaben des Benutzers bei der Anfrage in Instanzen der Implementationen der Interfaces Kriterium, Alternative und Interesse um und übergibt Attribute, Fuzzy-Prädikate und Werte aus dem PIA-Kern als Parameter für die Auswertung der Anfrage.

Die Auswertung der Anfrage erfolgt im Grunde nach dem in Abschnitt 5.2.4 beschriebenen Auswertungsalgorithmus. Zusätzlich werden jedoch verschiedene Optimierungsmöglichkeiten genutzt, um die Auswertung effizienter zu gestalten. So wird für Anfragen, die boolesche Prädikate enthalten, über das PIA-Datenmodell anhand dieser Prädikate zunächst eine Anfrage an den persistenten Datenspeicher, der die untere Schicht der PIA-Architektur markiert, abgesetzt. Hierbei können die im Datenspeicher implementierten Algorithmen zur Anfrageoptimierung genutzt werden, so dass schnell eine eingeschränkte Ergebnismenge möglicher Produktbeschreibungen ermittelt werden kann, die dem booleschen Teil der Anfrage entsprechen. Nur diese Produktbeschreibungen werden über das Datenmodell in Form von Iteratoren der Komponente zur Anfrageauswertung zur Verfügung gestellt, die auf ihnen den durch Fuzzy-Prädikate repräsentierten Teil der Anfrage auswertet. Hierbei kann sie weitere Optimierungsalgorithmen (vgl. Abschnitt 5.2.4) zum Einsatz bringen, um die Auswertung weiter zu beschleunigen.

Es bleibt zu erwähnen, dass insbesondere bei der Optimierung auf ausgereifte Verfahren, wie sie z.B. in relationalen Datenbanken (vgl. Abschnitt 3.1) bereits existieren, zurückgegriffen wird, anstatt ähnliche Algorithmen von Grund auf neu zu implementieren. Komponenten, die außerhalb von PIA implementiert wurden, sind in Abbildung 6.14 grau eingefärbt.

6.2.6 Einsatz toleranter Algorithmen

Die in PIA eingesetzten toleranten Algorithmen zur Interpretation von Metadaten sind auf der Grundlage von Fuzzy-Prädikaten realisiert (vgl. Abschnitt 5.2). Sie dienen einerseits der Auswertung von Suchanfragen an das Katalogsystem [MS99] und bilden andererseits die Basis für einen Dienst zur Zusammenführung mehrerer PIA-Kataloge [Tra01].

Die Sprache, in der mit Hilfe von Fuzzy-Prädikaten Anfragen an einen PIA-Katalog gestellt werden, ist im Vergleich zur Anfragesprache KooMet-QL (vgl.

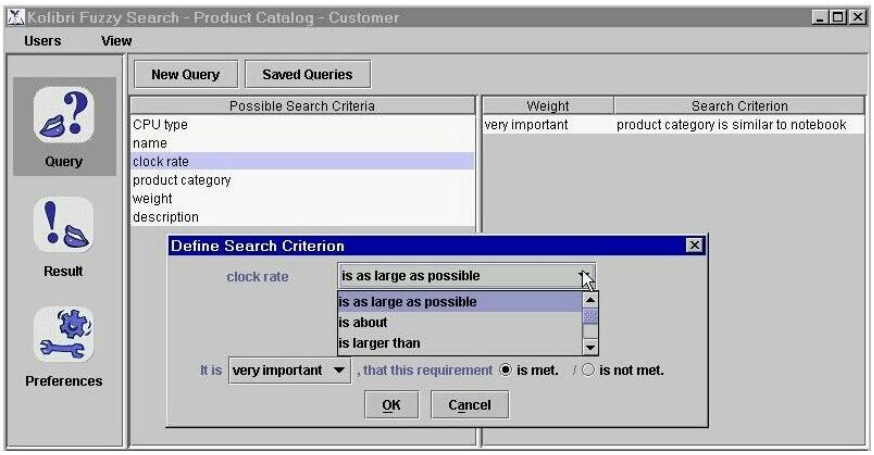

Abbildung 6.15: Formulierung eines Suchkriteriums in PIA

Abschnitt 5.2.3) leicht vereinfacht. Da es für Metadaten, die dem allgemeinen Metadatenmodell KooMet entsprechen, kein einheitliches Schema geben muss, können hier semantisch ähnliche Aspekte durch verschiedene Attribute ausgedrückt sein. Dieser Tatsache wird in KooMet-QL Rechnung getragen, indem ein Suchkriterium mehrere unscharfe Bedingungen mit gleicher oder ähnlicher Bedeutung durch einen unscharfen Oder-Operator zusammenfasst (vgl. Abschnitt 5.2.3). In PIA hingegen werden die Attribute für die Beschreibung der semantischen Aspekte der Ressourcen durch die Redakteure eindeutig festgelegt, so dass ein Suchkriterium durch eine einzige unscharfe Bedingung für ein einziges Attribut ausgedrückt werden kann (vgl. Abb. 6.15). Die Formulierung mehrerer, semantisch ähnlicher unscharfer Bedingungen entfällt daher. Die Übergabe von Vergleichswerten für das verwendete Fuzzy-Prädikat sowie die etwaige Negation des Prädikats werden direkt innerhalb des Kriterums abgewickelt (vgl. Abb. 6.16).

Bei der Zusammenführung von verschiedenen PIA-Katalogen werden zunächst auf beiden Seiten Typen und Attribute identifiziert, die einander in ihrer Bedeutung entsprechen. Im Anschluss daran wird eine Reihe von Attributen ausgewählt, die für den Vergleich der in den Katalogen beschriebenen Ressourcen geeignet sind, wie die Attribute „Produktbezeichnung“ oder „Artikelnummer“ für Produktkataloge. Für die ausgewählten Attribute werden intern der oben erläuterten Anfragesprache entsprechende Kriterien erstellt. Die Fuzzy-Prädikate in diesen Kriterien ermitteln für einen Attributwert einer Ressource des ersten Katalogs jeweils eine Ähnlichkeit mit einem entsprechenden Attributwert einer Ressource des zweiten Katalogs. Eine Alternative fasst die Kriterien für alle als relevant festgelegten Attribute zusammen und wird mehrfach für alle Ressourcen des ersten Katalogs ausgewertet, wodurch sich schließlich Ähnlichkeitswerte für Ressourcenpaare der beiden Kataloge ergeben, anhand derer die Bestände zusammengeführt werden können.

Die in PIA verwendeten Fuzzy-Prädikate umfassen einerseits eine Teilmenge der für Literale und Kollektionswerte aufgeführten Beispielprädikate (vgl.

Prädikat	**Typ der Argumente**
x ist genau a_1; x ist ähnlich zu a_1; x spezialisiert a_1	hierarchischer Typ
x spezialisiert eine der Klassen $a_1, \ldots, a_{n-1}$	hierarchischer Typ[ ]

Tabelle 6.1: Beispiele für Fuzzy-Prädikate auf hierarchischen Werten in PIA

Abschnitt 5.2.2). Da in PIA auch Klassifikationssysteme durch Typen repräsentiert sind (vgl. Abschnitt 6.2.3), werden sie andererseits ergänzt durch spezielle Fuzzy-Prädikate für hierarchische Typen [Büc99], die in Tabelle 6.1 aufgeführt sind.

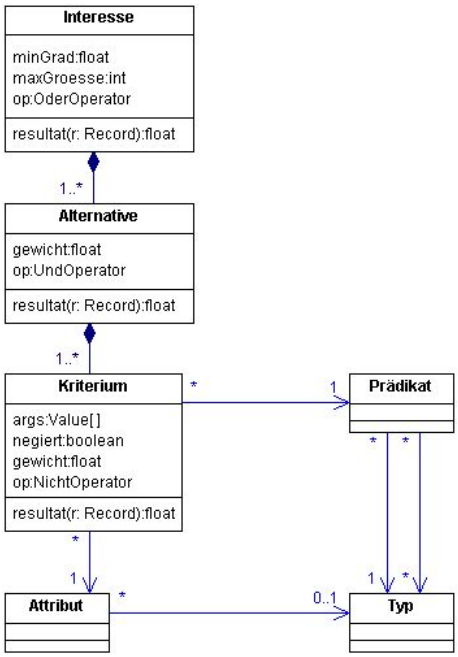

Abbildung 6.16: Anfragesprache zur Suche in PIA-Katalogen

6.3 Zusammenfassung

Das integrierende konzeptuelle Metadatenmodell KooMet (vgl. Kapitel 4) bildet die Grundlage für die konkrete Datenmodellierung in verschiedenen Systemen zur Verwaltung kooperativ genutzter Metadaten. Diese Systeme, von denen in diesem Kapitel mit dem infoAssetBroker und dem Katalogsystem PIA zwei Ausprägungen vorgestellt wurden, unterstützen jeweils ein bestimmtes Kooperationsszenarium, für das nur eine Teilmenge der in KooMet enthaltenen Konzepte zur Strukturierung, Typisierung und Klassifikation von Metadaten geeignet ist. Die konkreten Datenmodelle der Systeme stellen daher jeweils Einschränkungen von KooMet dar, die eine optimierte Erfassung und Verwendung von Metadaten

innerhalb einer entsprechenden Software-Architektur erlauben.

Der Aufbau von Metadaten und Zusatzinformationen wird dabei sowohl durch die Erfahrung der beteiligten Akteure im Umgang mit Metadaten als auch durch die Art der beschriebenen Ressourcen selbst beeinflusst. So führt eine heterogene Benutzermenge sowohl beim infoAssetBroker als auch in PIA zu einer relativ einfachen und flachen Metadatenstruktur. Der Einsatz von PIA zur Verwaltung von Katalogdaten nimmt darüber hinaus Einfluss auf den Aufbau der dort verwendeten Klassifikationssysteme.

Die Beschreibung des infoAssetBrokers macht deutlich, dass die Schematisierung von Metadaten für gemeinsame Informationsräume durch die Einbettung in eine Organisation erleichtert wird, so dass sie im Idealfall weit vor der Inbetriebnahme eines entsprechenden Informationssystems erfolgen und in die Datenmodellierung und den Entwurf der Software-Architektur mit einfließen kann.

Sowohl die Verwendung von Klassifikationssystemen im infoAssetBroker als auch die Verwaltung von Attributen und Typen in PIA zeigen jedoch darüber hinaus auf, dass Zusatzinformationen veränderbar sind und sich im laufenden Betrieb eines Informationssystems an die Gegebenheiten in einem gemeinsamen Informationsraum anpassen lassen müssen. Um die Verwendbarkeit dynamisch angepasster Zusatzinformationen als gemeinsame kooperative Arbeitsgrundlage gewährleisten zu können, kann die Berechtigung zur Änderung von Zusatzinformationen auf eine kleine Gruppe von Akteuren eingeschränkt werden, die als Redaktion eine gute Kenntnis der individuellen Arbeitskontexte und Anforderungen aller durch einen Informationsraum verbundener Kooperationspartner mitbringen sollten.

Je weniger Gemeinsamkeiten die Akteure in einem Informationsraum aufweisen, desto größer ist ihr Bedarf, flexibel mit Metadaten umgehen zu können. Dieses wird noch einmal deutlich durch die Erfassung untypisierter Produktbeschreibungen in PIA reflektiert, die in der Unabhängigkeit der verschiedenen Anbieter voneinander und vom Katalogsystem begründet ist. Dennoch können auch uneinheitliche Metadaten durch den Rückgriff auf entsprechend flexible Algorithmen wie bei der Anfrageauswertung in PIA einer kooperativen Nutzung zugute kommen.

Da bei der Verwendung toleranter Algorithmen (vgl. Kapitel 5) eine zusätzliche Aufbereitung existierender Metadaten und damit eine Übertragung in weitere mögliche Arbeitskontexte erfolgt, bildet diese sowohl in Szenarien der engen als auch der lose gekoppelten Kooperation eine zusätzliche Verbindung zwischen Metadatenerfasser und interpretierendem Akteur.

Kapitel 7

Ergebnisse und Ausblick

Zum Abschluss werden die Ergebnisse der vorliegenden Arbeit zusammengefasst und ein Ausblick auf verbleibende offene Fragestellungen im Zusammenhang mit der kooperativen Nutzung von Metadaten gegeben, die sich als Gegenstand weiterer Forschungsaktivitäten in diesem Bereich anbieten.

7.1 Ergebnisse

Für den Umgang mit Metadaten sind unterschiedliche *Vorgehensweisen* identifiziert worden, die ihrerseits wiederum die Grundlage für eine Reihe von Kooperationsszenarien bilden, die von einer sehr engen bis hin zu einer offen, flexiblen Form der Zusammenarbeit reichen. Die Unterschiede in den Vorgehensweisen beziehen sich dabei einerseits darauf, ob und zu welchem Grad Metadaten durch Zusatzinformationen beschrieben und schematisiert werden, und andererseits auf die Rolle, die die beschriebenen Ressourcen beim Aufbau der Metadaten spielen. Die Unterstützung aller identifizierten Vorgehensweisen ist eine grundlegende Anforderung an ein allgemeines Metadatenmodell, das eine Vielzahl von Kooperationsszenarien unterstützt.

Für Metadaten selbst ist eine weitere Unterteilung in verschiedene *Beschreibungsebenen* getroffen worden, die einerseits die tatsächlichen Metadaten umfassen, die der direkten Beschreibung von Ressourcen dienen, und andererseits Zusatzinformationen, die in Form von Typsystemen und Klassifikationssystemen den intensionalen Rahmen für die Kooperation auf der Grundlage von Metadaten abstecken. Die Unterteilung gibt zusätzlichen Aufschluss über Inhalte und Verwendung von Metadaten und bietet zugleich Anhaltspunkte bezüglich der möglichen Bestandteile und deren Strukturierung innerhalb eines allgemeinen Metadatenmodells.

Die *Untersuchung existierender Datenmodelle*, die teils allgemeinen Charakter besitzen und teils ausschließlich für die Repräsentation von Metadaten genutzt werden, hat ergeben, dass eine Verwendung dieser Modelle als allgemeine Metadatenmodelle im Sinne einer Unterstützung unterschiedlicher Kooperationsszenarien verschiedene Nachteile mit sich bringt. So zielen die meisten Datenmodelle auf ein spezielles Kooperationsszenarium ab. Das relationale und das objektorientierte Datenmodell begünstigen eine enge Zusammenarbeit, bei

der ein Großteil der Metadaten einem festen Schema unterliegt. Am anderen Ende der Skala hingegen findet sich RDF, das eine offene Kooperation unterstützt, in der Metadaten völlig frei gestaltbar bleiben. Wiederum andere Modelle beschränken sich auf die Modellierung eines bestimmten Anteils von Metadaten, wie im Falle von Datenmodellen des Information Retrieval und der Beschreibungslogiken, mit deren Hilfe Ressourcen klassifiziert und Klassifikationssysteme aufgebaut werden können, die aber keine oder nur einige wenige Konzepte zur Strukturierung oder zur Typisierung von Metadaten aufweisen. In allgemeinen Modellen wie XML schließlich finden sich kaum Konzepte, mit denen sich speziell die Arbeit mit Metadaten unterstützen lässt. Für die Modellierung von Metadaten müssten sie entsprechend stark adaptiert werden und können daher eher als Beschreibungsmittel, nicht aber als Metadatenmodelle zum Einsatz kommen.

Die untersuchten Datenmodelle weisen jedoch durchgehend auch Konzepte auf, die für die Kooperation auf der Grundlage von Metadaten in hohem Maße relevant sind und die in ihrer Gesamtheit der Unterstützung der erarbeiteten Vorgehensweisen beim Umgang mit Metadaten und den damit verbundenen Kooperationsszenarien dienen.

Die Beschreibung des im Rahmen dieser Arbeit entwickelten *integrierenden Metadatenmodells KooMet* hat gezeigt, dass die für die Nutzung von Metadaten in gemeinsamen Informationsräumen relevanten Konzepte innerhalb eines allgemeinen Gesamtmodells zusammengeführt werden können. Das Hauptaugenmerk liegt dabei darauf, dass die Konzepte von KooMet nebeneinander ohne Widersprüche nutzbar sind, so dass eine hohe Bandbreite möglicher Kooperationsszenarien durch das Metadatenmodell abgedeckt werden kann.

Innerhalb von KooMet lassen sich Konzepte zur Strukturierung und Typisierung von Metadaten sowie Konzepte zur Klassifikation von Ressourcen voneinander abgrenzen, so dass sich bei einer Verwendung des Modells die innerhalb von Metadaten gegebenen Beschreibungsebenen klar voneinander trennen und gegebenenfalls auch ausblenden lassen.

Die *Strukturierung von Metadaten* in KooMet beruht hauptsächlich auf dem Konzept von Attribut-Wert-Paaren mit einfachen und komplexen Werten sowie auf der Zusammenfassung dieser Paare in Records und der Bildung von Beziehungen zwischen Ressourcen.

Die Konzepte zur *Typisierung* in KooMet erlauben sowohl den Aufbau von Schemata in unterschiedlicher Granularität als auch die freiwillige Bereitstellung von Typen und den vollständigen Verzicht auf eine Typisierung.

Die Konzepte zur *Klassifikation* in KooMet gestatten die formale Definition von Klassifikationssystemen ebenso wie die spontane Festlegung beliebiger Klassen und Beziehungen zwischen ihnen. Da die Klassen innerhalb eines Klassifikationssystems wiederum als Ressourcen betrachtet werden, ergibt sich eine Vielzahl von Beschreibungsmöglichkeiten für Klassen, für Klassenbeziehungen und für Zuordnungen von Ressourcen zu Klassen.

Durch die durch KooMet gegebene Metadatenstruktur erleichtert sich die Anbindung *toleranter Algorithmen* an das Modell. Im Rahmen dieser Arbeit sind Interpretationsalgorithmen auf der Grundlage von Fuzzy-Prädikaten in das Metadatenmodell KooMet integriert worden, die die Strukturierung von Meta-

daten anhand von Attributen nutzen, um das Zutreffen unscharfer Bedingungen hinsichtlich bestimmter Eigenschaften von Ressourcen zu überprüfen. Fuzzy-Prädikate können innerhalb gemeinsamer Informationsräume für verschiedene Aufgaben eingesetzt werden. Hierzu zählt insbesondere die Selektion einzelner Records aus einem Metadatenbestand, für die die Anfragesprache KooMet-QL beschrieben worden ist.

Das Metadatenmodell KooMet dient einerseits als konzeptuelle Grundlage für die Integration unterschiedlicher Metadaten. Andererseits kann es in konkreten Metadatenmodellen vor dem Hintergrund konkreter Kooperationsszenarien eines zu realisierenden Systems eingeschränkt und vereinfacht werden. Mit dem infoAssetBroker und dem Katalogsystem PIA sind zwei *konkrete Systeme* mit unterschiedlichen Kooperationsszenarien vorgestellt worden, die beide auf unterschiedlichen Teilmengen der Konzepte von KooMet beruhen.

7.2 Ausblick

Im Rahmen dieser Arbeit wurde das integrierende Metadatenmodell KooMet bisher als konzeptuelle Grundlage für konkrete Systeme mit bestimmten Kooperationsszenarien verwendet. Die Realisierung eines auf KooMet basierenden gemeinsamen Informationsraums, der verschiedene gegenläufige Kooperationsszenarien unterstützt, steht hingegen noch aus. Ein solches System würde einen weiteren Beleg für das Zusammenwirken der Konzepte in KooMet erbringen und könnte beispielsweise als eine Erweiterung des infoAssetBrokers implementiert werden.

Formal kann der Nachweis dafür, dass KooMet keine widersprüchlichen Konzepte enthält, beispielsweise durch die Umsetzung des Modells in einer Beschreibungslogik geführt werden, für die dann ein Kohärenztest durchgeführt wird. Auch dieser Nachweis steht noch aus.

Da KooMet eine Vielzahl von Konzepten zur Arbeit mit Metadaten zusammenführt, kann es auch als eine Art lingua franca dienen, anhand derer Metadaten aus unterschiedlichen spezialisierten Metadatenmodellen ineinander überführt werden können. Die Entwicklung eines Frameworks, das KooMet für die Umwandlung von Metadatenmodellen nutzt, sowie die Implementation entsprechender Dienste, stellt einen weiteren Schritt zur Zusammenführung verschiedener Ansätze bei der Unterstützung einer kooperativen Nutzung von Metadaten dar.

Das WorldWideWeb als globaler Informationsraum befindet sich in einem Wandlungsprozess, der sich auch in den Bestrebungen hinsichtlich dem Aufbau eines Semantic Webs widerspiegelt. So wird das WorldWideWeb heute nicht mehr nur als Materialspeicher menschlicher Kooperationspartner betrachtet, sondern auch als eine Datenbasis für autonome, ineinander verwobene Software-Dienste, wie sich bereits durch die Entstehung spezieller Technologien abzeichnet [VN02]. Die Untersuchung, welche spezifischen Anforderungen insbesondere mobile Software-Agenten [SLK98] an Metadaten in einem gemeinsamen Informationsraum stellen und inwieweit der in dieser Arbeit vorgestellte Ansatz einer flexiblen Anreicherung von Metadaten mit Zusatzinformationen vor diesem

speziellen Hintergrund Bestand haben kann, ist somit eine weitere wichtige Fragestellung im Bereich kooperativ genutzter Metadaten.

Die Forschung im Bereich des Semantic Web orientiert sich gegenwärtig in hohem Maße an Ansätzen der Beschreibungslogik. Hier bleibt zu untersuchen, in welchen Szenarien die Formalismen von Beschreibungslogiken zur inhaltlichen Beschreibung von Ressourcen eingesetzt werden können [Sta02], und ob darüber hinaus weitere Verfahrensweisen für die Anreicherung von Web-Inhalten notwendig werden. In diesem Zusammenhang stellt sich insbesondere die Frage, ob Kooperationsprozesse im Semantic Web, an denen menschliche Akteure, die keine Spezialisten auf dem Gebiet der Beschreibungslogiken sind, teilhaben, auf einer eindeutigen formalen Beschreibung von Ressourcen beruhen sollten und können. Neben der Entwicklung geeigneter Benutzerschnittstellen ist auch hier der Einsatz toleranter Algorithmen, die in der Lage sind formal fehlerhafte semantische Beschreibungen zu interpretieren oder korrekt umzuwandeln, ein möglicher Lösungsansatz.

Hinsichtlich der Kooperation über die Grenzen verschiedener Arbeitskontexte, die jeweils auf eigenen spezifischen Klassifikationssystemen beruhen, hinweg, ist außerdem die Verbindung sowie die Verschmelzung unterschiedlicher Klassifikationssysteme der Gegenstand aktueller Forschung. Hierzu existieren erste Ansätze sowohl für formale [FNM00] als auch für freie Klassifikationssysteme [MNS01].

Zusätzlich ist das Konzept der toleranten Algorithmen ein potentieller Gegenstand weiterer Untersuchungen. Während die Eignung des Metadatenmodells KooMet für die Anbindung von speziellen toleranten Algorithmen in Form von Fuzzy-Prädikaten nachgewiesen werden konnte, steht dieses für tolerante Algorithmen allgemein noch aus. Hier ist zu untersuchen, über welche Art von Schnittstellen tolerante Algorithmen mit Informationssystemen interagieren. [Büc02] liefert hier für vektorbasierte Verfahren einen ersten exemplarischen Ansatz.

Literaturverzeichnis

[ABK01] Wolfgang Appelt, Uwe Busbach, und Thomas Koch. Kollaborationsorientierte asynchrone Werkzeuge. In: Gerhard Schwabe, Norbert Streitz, und Rainer Unland, Hrsg., *CSCW-Kompendium: Lehr- und Handbuch zum computerunterstützen kooperativen Arbeiten*, Seite 194–203. Springer-Verlag, 2001.

[ABS00] Serge Abiteboul, Peter Bunemann, und Dan Suciu. *Data on the Web - From Relations to Semistructured Data and XML.* Morgan Kaufmann, San Francisco, 2000.

[ABW+90] Malcom P. Atkinson, François Bancilhon, David De Witt, Klaus Dittrich, David Maier, und Stanley Zdonik. The Object-Oriented Database System Manifesto. In: Won Kim, Jean-Marie Nicolas, und Shojiro Nishio, Hrsg., *Proceedings of the First International Conference on Deductive and Object-Oriented Databases, DOOD'89, Kyoto.* Elsevier Science Publishers, Amsterdam, Dezember 1990.

[AMN94] Helena Ahonen, Heikki Mannila, und Erja Nikunen. Forming grammars for structured documents: an application of grammatical inference. In: Rafael C. Carrasco und José Oncina, Hrsg., *Proceedings of Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante*, Band LNCS 862, *Lecture Notes in Computer Science*, Seite 153–167. Springer-Verlag, September 1994.

[Arm00] William Y. Arms. *Digital libraries - Digital Libraries and Electronic Publishing.* The MIT Press, Cambridge, 2000.

[BB97] Liam Bannon und Susanne Bødker. Constructing Common Information Spaces. In: *Proceedings of the 5th European Conference on Computer-Supported Cooperative Work, ECSCW'97, Lancaster*, Shared Information Spaces, Seite 81–96. Kluwer Academic Publishers, September 1997.

[BD99] Vincent Bouthors und Olivier Dedieu. Pharos, a Collaborative Infrastructure for Web Knowledge Sharing. In: Serge Abiteboul und Anne-Marie Vercoustre, Hrsg., *Research and Advanced Technology for Digital Libraries, Proceedings of the Third European*

Conference, ECDL 1999, Paris, Band LNCS 1696, *Lecture Notes in Computer Science*, Seite 215–233. Springer-Verlag, September 1999.

[BEK+98] Stefan Berchtold, Bernhard Ertl, Daniel A. Keim, Hans-Peter Kriegel, und Thomas Seidl. Fast Nearest Neighbor Search in High-Dimensional Spaces. In: *Proceedings of the 14th International Conference on Data Engineering, ICDE'98, Orlando*, Seite 209–218. IEEE Computer Society, Februar 1998.

[BHST95] Richard Bentley, Thilo Horstmann, Klaas Sikkel, und Jonathan Trevor. Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System. In: *Proceedings of the Fourth International World Wide Web Conference, Boston*, Seite 63–74, Dezember 1995.

[Bie97] Benno Biewer. *Fuzzy-Methoden - Praxisrelevante Rechenmodelle und Fuzzy-Programmiersprachen.* Springer-Verlag, Berlin, 1997.

[BKT01] Peter Buneman, Sanjeev Khanna, und Wang Chiew Tan. Why and Where: A Characterization of Data Provenance. In: Jan van den Bussche und Victor Vianu, Hrsg., *Proceedings of Database Theory - ICDT 2001, 8th International Conference, London*, Band LNCS 1973, *Lecture Notes in Computer Science*, Seite 316–330. Springer-Verlag, Januar 2001.

[BL84] Ronald J. Brachman und Hector J. Levesque. The Tractability of Subsumption in Frame-Based Description Languages. In: Ronald J. Brachman, Hrsg., *Proceedings of the 4th National Conference on Artificial Intelligence AAAI'84, Austin*, Seite 34–37. AAAI Press, August 1984.

[BLHL01] Tim Berners-Lee, James Hendler, und Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, Mai 2001.

[Bli35] Henry Evelyn Bliss. *The Organization of knowledge, part 3: A system of bibliographic classification.* Wilson, New York, 1935.

[BM72] Rudolf Bayer und Edward M. McCreight. Organization and Maintenance of Large Ordered Indices. *Acta Informatica*, 1:173–189, 1972.

[BN03] Franz Baader und Werner Nutt. Basic Decsription Logics. In: Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, und Peter F. Patel-Schneider, Hrsg., *The Description Logic Handbook: Theory, Implementation and Applications*, Kapitel 2, Seite 47–100. Cambridge University Press, März 2003.

[Boo94] G. Booch. *Object-Oriented Design with Applications.* Addison-Wesley Publishing Company, Reading, 2. Auflage, 1994.

[Bos97] Jon Bosak. XML, Java and the future of the Web. Technischer Bericht, Sun Microsystems, 1997.

[BP98] Sergey Brin und Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[Brö00] Jan Brötzmann. Abrechnungen und Zahlungskonzepte für Dienstleistungen digitaler Bibliotheken. Studienarbeit, Fachbereich Informatik, Universität Hamburg, Juli 2000.

[BS85] Ronald J. Brachman und James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[BS91] Liam J. Bannon und Kjeld Schmidt. CSCW: Four Characters in Search of a Context. In: John M. Bowers und Steven D. Benford, Hrsg., *Studies in Computer Supported Cooperative Work*, Seite 3–16. North-Holland, 1991.

[Büc99] Thomas Büchner. Konzeption und Implementation von Fuzzy-Prädikaten in Digitalen Bibliotheken. Studienarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, November 1999.

[Büc02] Thomas Büchner. Entwurf und Realisierung eines Java-Frameworks zur inhaltlichen Erschließung von Dokumenten. Diplomarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Januar 2002.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, und Lynn Andrea Stein. OWL Web Ontology Language Reference. W3C Recommendation, Februar 2004.

[BYRN99] Ricardo Baeza-Yates und Berthier Ribiero-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing Company, Harlow, 1999.

[Cav94] William B. Cavnar. Using an N-Gram-Based Document Representation with a Vector Processing Retrieval Model. In: Donna K. Harman, Hrsg., *Proceedings of the Third Text Retrieval Conference (TREC-3), Gaithersburg*, Seite 269–278, 1994.

[CB97] Roderic G. G. Cattell und Douglas K. Barry, Hrsg. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, San Francisco, 1997.

[Cha01] Hendry Chandra. Semi-automatic Merging of Content Networks : Matching. Studienarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, März 2001.

[Cod70] E. F. Codd. A Relational Model for Large Shared Databanks. *Communications of the ACM*, 13(6):377–387, Juni 1970.

[CRF03] William Cohen, Pradeep Ravikumar, und Stephen E. Fienberg. A Comparison of String Metrics for Matching Names and Records. In: *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation, Washington D.C.*, Seite 13–18, 2003.

[DB92] Paul Dourish und Victoria Bellotti. Awareness and Coordination in Shared Workspaces. In: *Proceedings of CSCW '92 - Sharing Perspectives*, Seite 107–114, Toronto, Canada, November 1992. ACM Press.

[DH98] Lorcan Dempsey und Rachel Heery. Metadata: A current review of practice and issues. *Journal of Documentation*, 54(2):145–172, 1998.

[Die01] Lars Diestelhorst. Recommendation Engines. Studienarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Juli 2001.

[DN66] Ole-Johan Dahl und Kristen Nygaard. SIMULA, an ALGOL-based simulation language. *Communications of the ACM*, 9(9):671–678, September 1966.

[DNR97] Francesco M. Donini, Daniele Nardi, und Riccardo Rosati. Autoepistemic Description Logics. In: *Proceedings of the 15th International Joint Conference on Artificial Intelligence, IJCAI'97, Nagoya*, Seite 136–141. Morgan Kaufmann, August 1997.

[Don03] Francesco M. Donini. Complexity of Reasoning. In: Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, und Peter F. Patel-Schneider, Hrsg., *The Description Logic Handbook: Theory, Implementation and Applications*, Kapitel 3, Seite 101–141. Cambridge University Press, März 2003.

[Eck98] Bruce Eckel. *Thinking in Java.* Prentice Hall, Englewood Cliffs, Upper Saddle River, 1998.

[Fag96] Ronald Fagin. Combining Fuzzy Information from Mutliple Systems. In: *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Montreal*, Seite 216–226. ACM Press, Juni 1996.

[Fag02] Ronald Fagin. Combining Fuzzy Information: an Overview. *ACM SIGMOD Record*, 31(2):109–118, Juni 2002.

[Fen00] Dieter Fensel. The semantic Web and its languages. *IEEE Intelligent Systems*, 15(6):67–73, November/Dezember 2000.

[FLN03] Ronald Fagin, Amnon Lotem, und Moni Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656, Juni 2003.

[FNM00] Natalya Fridman Noy und Mark Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: *Proceedings of the 7th Conference on Artificial Intelligence AAAI-00 and of the 12th Conference on Innovative Applications of Artificial Intelligence IAAI–00*, Seite 450–455, Menlo Park, Juli 2000. AAAI Press.

[Fur00] Betty Furrie. *Understanding MARC Bibliographic: Machine-Readable Cataloging.* Cataloging Distribution Service, Customer Service Section, Library of Congress, Washington, D.C., 6. Auflage, 2000. In Zusammenarbeit mit dem Data Base Development Department der Follett Software Company.

[GBK02] Ulrich Güntzer, Wolf-Tilo Balke, und Werner Kießling. Optimizing Multi-Feature Queries for Image Databases. In: Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, und Kyu-Young Whang, Hrsg., *Proceedings of the 26th International Conference on Very Large Databases VLDB 2000, Cairo*, Seite 419–428. Morgan Kaufmann, 2002.

[GJ97] Amarnath Gupta und Ramesh Jain. Visual Information Retrieval. *Communications of the ACM*, 40(5):70–79, Mai 1997.

[GR83] Adele Goldberg und David Robson. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley Publishing Company, Reading, 1983.

[Gra95] Adolf Grauel. *Fuzzy-Logik: Einführung in die Grundlagen mit Anwendungen.* BI-Wissenschaftsverlag, Mannheim, 1995.

[GS98] Anne J. Gilliland-Swetland. Defining Metadata. In: Murtha Baca, Hrsg., *Introduction to Metadata: Pathways to Digital Information.* Getty Research Institute, Los Angeles, 1998.

[GW97] Roy Goldman und Jennifer Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In: Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, und Manfred A. Jeusfeld, Hrsg., *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB97, Athen*, Seite 436–445. Morgan Kaufmann, August 1997.

[HG02] Farshad Hakimpour und Andreas Geppert. Global Schema Generation Using Formal Ontologies. In: Stefano Spaccapietra, Salvatore T. March, und Yahiko Kambayashi, Hrsg., *Proceedings of*

the 21st International Conference on Conceptual Modeling, Tampere, Band LNCS 2503, *Lecture Notes in Computer Science*, Seite 307–321. Springer-Verlag, Oktober 2002.

[HM93] Joachim Hammer und Dennis McLeod. An Approach to Resolvin Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. *Journal of Intelligent and Cooperative Information Systems*, 2(1):51–83, 1993.

[HM01] Volker Haarslev und Ralf Möller. Description of the RACER System and its Applications. In: Carole A. Goble, Deborah L. McGuinness, Ralf Möller, und Peter F. Patel-Schneider, Hrsg., *Working Notes of the 2001 International Description Logics Workshop DL 2001, Stanford*, Band 49, *CEUR Workshop Proceedings*, August 2001.

[Hor02] Ian Horrocks. DAML+OIL: A Reason-able Web Ontology Language. In: Christian S. Jensen, Keith G. Jeffery, Jaroslav Pokorný, Simonas Saltenis, Elisa Bertino, Klemens Böhm, und Matthias Jarke, Hrsg., *Proceedings of the 8th International Conference on Extending Database Technology, EDBT'02, Prague*, Band LNCS 2287, *Lecture Notes in Computer Science*, Seite 2–13. Springer-Verlag, März 2002.

[HT99] David Hicks und Klaus Tochtermann. Metadata Support for Customization in Environmental Information Management Systems. In: *Proceedings of the 2nd GI Workshop Hypermedia im Umweltschutz, Nürnberg*, Seite 136–145, März 1999.

[HTM99] HTML 4.01 Specification. W3C Recommendation, Dezember 1999.

[ISO86] ISO 8879: Information Processing – Standard Generalized Markup Language, 1986. International Organization for Standardization.

[ISO87] ISO 9075: Information processing systems – Database language SQL, 1987. International Organization for Standardization.

[Joa98] Thorsten Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: Claire Nedellec und Céline Rouveirol, Hrsg., *Proceedings of the 10th European Conference on Machine Learning, ECML-98, Chemnitz*, Band LNCS 1398, *Lecture Notes in Computer Science*, Seite 137–142. Springer-Verlag, April 1998.

[Joh88] Robert Johansen. *Groupware: Computer Support for Business Teams*. The Free Press, Macmillan Inc., New York, 1988.

[JvBA+03] Henk Jonkers, René van Buuren, Farhad Arbab, Frank de Boer, Marcello Bonsangue, Hans Bosma, Hugo ter Doest, Luuk Groe-

newegen, Juan Guillen Scholten, Stijn Hoppenbrouwers, Maria-Eugenia Iaco, Wil Janssen, Marc Lankhorst, Diederik van Leeuwen, Erik Andries Stam, Leon van der Torre, und Gerd Veldhuijzen van Zanten. Towards a Language for Coherent Enterprise Architecture Decsriptions. In: *Proceedings of Enterprise Distributed Object Computing EDOC 2003, Brisbane*, 2003.

[KK02] Dimitris Karagiannis und Harald Kühn. Metamodelling Platforms. In: K. Bauknecht, A. Min Tjoa, und G. Quirchmayer, Hrsg., *Proceedings of the Third International Conference EC-Web 2002 - Dexa 2002, Aix-en-Provence*, Band LNCS 2455, *Lecture Notes in Computer Science*. Springer-Verlag, September 2002.

[KM03] Meike Klettke und Holger Meyer. *XML und Datenbanken: Konzepte, Sprachen und Systeme*. xml.bibliothek. dpunkt.verlag, Heidelberg, 2003.

[Koc02] Leif Koch. Wissensportalsoftware als Learning Management System – Fachliche Analyse, Entwurf, prototypische Realisierung. Diplomarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Oktober 2002.

[Kor97] Robert R. Korfhage. *Information Storage and Retrieval*. Wiley Computer Publishing, New York, 1997.

[Lag01] Carl Lagoze. Keeping Dublin Core Simple - Cross-Domain Discovery or Resource Description. *D-Lib Magazine*, 7(1), Januar 2001.

[Lea74] Harold J. Leavitt. *Grundlagen der Führungspsychologie: Individuum, Gruppe, Organisation*. Verlag Moderne Industrie, München, 1974.

[LF99] Yannis Labrou und Tim Finin. Yahoo! as an Ontology: Using Yahoo! Categories to Describe Documents. In: Susan Gauch und Il-Yeol Soong, Hrsg., *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM'99), Kansas City*, Seite 180–187. ACM, November 1999.

[LMSP95] Clifford Lynch, Avra Michelson, Craig Summerhill, und Cecilia Preston. The Nature of the NIDR Challenge. Technischer Bericht, Coalition for Networked Information, 1995.

[LS87] Peter C. Lockemann und Joachim W. Schmidt, Hrsg. *Datenbank-Handbuch*. Springer-Verlag, Berlin, 1987.

[Mar00] Catherine C. Marshall. The Future of Annotation in a Digital (Paper) World. In: Susan Harum und Michael Twidale, Hrsg., *Successes and Failures of Digital Libraries*, Seite 97–117. Urbana-Champaign, University of Illinois, 2000.

[Mey88] Bertrand Meyer. *Object-oriented Software Construction*. International Series in Computer Science. Prentice Hall, Englewood Cliffs, 1988.

[Min81] Marvin Minsky. A framework for representing knowledge. In: J. Haugeland, Hrsg., *Mind Design*. The MIT Press, Cambridge, 1981.

[MNS01] Florian Matthes, Claudia Niederée, und Ulrike Steffens. C-Merge: A Tool for Policy-Based Merging of Resource Classifications. In: Panos Constantopoulos und Ingeborg Sølvberg, Hrsg., *Research and Advanced Technology for Digital Libraries, Proceedings of the 5th European Conference, ECDL2001, Darmstadt, Germany*, Band LNCS 2163, *Lecture Notes in Computer Science*, Seite 352–365. Springer-Verlag, September 2001.

[MNSS99] Florian Matthes, Claudia Niederée, Joachim W. Schmidt, und Ulrike Steffens. Das Internet als Wissensmarkt - Möglichkeiten und Grenzen. In: Christel Kumbruck und Wolfgang Kersten, Hrsg., *Wissensmarkt Internet - zwischen betrieblichem Wissensmanagement und virtueller Universität*, Harburger Beiträge zur Psychologie und Soziologie der Arbeit. Technische Universität Hamburg-Harburg, 1999.

[MS99] Florian Matthes und Ulrike Steffens. PIA - A Generic Model and System for Interactive Product and Service Catalogs. In: Serge Abiteboul und Anne-Marie Vercoustre, Hrsg., *Research and Advanced Technology for Digital Libraries, Proceedings of the 3rd European Conference, ECDL'99, Paris*, Band LNCS 1696, *Lecture Notes in Computer Science*, Seite 403–422. Springer-Verlag, September 1999.

[MW99] Jason McHugh und Jennifer Widom. Query Optimization for XML. In: Malcom P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, und Michael L. Brodie, Hrsg., *Proceedings of the 25th International Conference on Very Large Databases, VLDB'99, Edinburgh*, Seite 315–326. Morgan Kaufmann, September 1999.

[MWS00] Florian Matthes, Joachim W.Schmidt, und Ulrike Steffens. Kolibri - Entwicklung von Modellen und Werkzeugen für kooperative digitale Handbibliotheken. *Informationstechnik und Technische Informatik*, 42(6), Dezember 2000.

[NBYA01] Gonzalo Navarro, Ricardo Baeza-Yates, und João Marcelo Azevedo Arcoverde. Matchsimile: A Flexible Approximate Matching Tool for Personal Names Searching. In: Marta Mattoso und Geraldo Xexéo, Hrsg., *Proceedings of the XVI Brazilian Symposium on Databases SBBD 2001*, Seite 228–242, 2001.

[Nor94] Moira C. Norrie. Distinguishing Typing and Classification in Object Data Models. In: *European-Japanese conferences on Information Modelling and Knowledge Bases: Conference 1994, Kista, Stockholm*, Seite 399–412. IOS Press, Mai/Juni 1994.

[NSH02] Claudia Niederée, Ulrike Steffens, und Matthias Hemmje. Towards Digital Library Mediation for Web Services. In: *EurAsian Workshop on Knowledge Foraging for Dynamic Networking of Communities and Economies 2002, Shiraz*, Oktober 2002.

[NSS+98] Claudia Niederée, Ulrike Steffens, Hans-Werner Sehring, Florian Matthes, und Joachim W. Schmidt. On Indexing in Digital Libraries: Cooperation, Personalization, and Evolution. Technischer Bericht, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Juni 1998.

[Pas00] Norman Paskin. Digital Object Identifier: implementing a standard digital identifier as the key to effective digital rights management. Technischer Bericht, International DOI Foundation, April 2000.

[Pep00] Steve Pepper. The TAO of Topic Maps: Finding the Way in the Age of Infoglut. In: *XML Europe*, 2000.

[PH01] Jeff Z. Pan und Ian Horrocks. Metamodeling architecture of web ontology languages. In: Isabel F. Cruz, Stefan Decker, Jérôme Euzenat, und Deborah L. McGuinness, Hrsg., *The Emerging Semantic Web, Selected papers from the first Semantic Web Working Symposium, Stanford*, Band 75, *Frontiers in Artificial Intelligence and Applications*. IOS press, Juli 2001.

[Por80] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[Qui67] M. Ross Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, (12):410–430, 1967.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, und W. Lorensen, Hrsg. *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, 1991.

[RCR93] John F. Roddick, Noel G. Craske, und Thomas J. Richards. A Taxonomy for Schema Versioning Based on the Relational and Entity Relationship Models. In: Ramez Elmasri, Vram Kouramajian, und Bernhard Thalheim, Hrsg., *Proceedings of the 12th International Conference on the Entity-Relationship Approach, ER'93, Arlington*, Band LNCS 823, *Lecture Notes in Computer Science*, Seite 137–148. Springer-Verlag, Dezember 1993.

[RDF04a] Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, Februar 2004.

[RDF04b] RDF Primer. W3C Recommendation, Februar 2004.

[RDF04c] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, Februar 2004.

[RDF04d] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, Februar 2004.

[RDF04e] RDF/XML Syntax Specification (Revised). W3C Recommendation, Februar 2004.

[RG96] Mark Roseman und Saul Greenberg. TeamRooms: Network Places for Collaboration. In: *Proceedings of the ACM 1996 Conference on Computer-Supported Cooperative Work, CSCW'96, Boston*, Places for Collaboration, Seite 325–333. ACM Press, November 1996.

[Rie03] Sebastian Riedel. Entwicklung eines Modells, Verfahrens und softwaretechnischen Rahmenwerks für Record Linkage in semantischen Netzen. Diplomarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, August 2003.

[RMS+01] Martin Raulf, Rainer Müller, Ulrike Steffens, Florian Matthes, Klaus J. Scheunert, und Joachim W. Schmidt. Begriffsorientierte Dokumentenverwaltung für das internetgestützte Projektmanagement - Der FHH InfoBroker für das Projekt "sap für hamburg". In: *Tagungsband 4. GI-Fachgruppentagung "Management und Controlling von IT-Projekten"*, Glasshütten (Taunus), März 2001.

[RMSS03] Monika Renz, Florian Matthes, Ulrike Steffens, und Joachim W. Schmidt. Erschließung heterogener Wissensquellen durch eine digitale Bibliothek für die öffentliche Verwaltung. In: Ralph Schmidt, Hrsg., *Tagungsband 25. DGI Online Tagung Competence in Content*, Frankfurt, Main, Juni 2003. Deutsche Gesellschaft für Informationswissenschaft und Informationspraxis.

[SL90] Amit P. Sheth und James A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SLK98] Katia Sycara, Jianguo Lu, und Matthias Klusch. Interoperability among Heterogeneous Software Agents on the Internet. Technischer Bericht CMU-RI-TR-98-22, Robotics Institute, Carnegie Mellon University, Pittsburgh, Oktober 1998.

[SM84] Gerard Salton und Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, Auckland, 1984.

[Smi96] Terence R. Smith. The Meta-Information Environment of Digital Libraries. *D-Lib Magazine*, Juli 1996.

[Sow00] John F. Sowa. *Knowledge Representation - Logical, Philosophical, and Computational Foundations*. Brooks / Cole, Pacific Grove, 2000.

[SSS91] Manfred Schmidt-Schauß und Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[SSW01] Joachim W. Schmidt, Hans-Werner Sehring, und Martin Warnke. Der Bildindex zur Politischen Ikonographie in der Warburg Electronic Library - Einsichten eines interdisziplinären Projektes. In: *Archivprozesse*, 2001.

[Sta02] Steffen Staab. Ontologies' KISSES in Standardization. *IEEE Intelligent Systems*, Seite 70–79, März 2002.

[Ste96] Ulrike Steffens. Integration von Information Retrieval Funktionalität in eine offene, persistente Programmierumgebung. Informatik Mitteilung FBI-HH-M-257/96, Fachbereich Informatik, Universität Hamburg, April 1996. (Studienarbeit).

[SW49] Claude E. Shannon und Warren Weaver. *A Mathematical Theory of Communication*. University of Illinois Press, Urbana, 1949.

[SWBH00] Erwin Schuster, Stephan Wilhelm, und Hans-Jörg Bullinger Hrsg. Content Management Systeme: Auswahlstrategien, Architekturen und Produkte. Forschungsbericht, Fraunhofer Institut Arbeitswirtschaft und Organisation (IAO), Stuttgart, 2000.

[SZ92] J. F. Sowa und J. A. Zachman. Extending and formalizing the framework for information systems architecture. *IB; Systems Journal*, 31(3):590–616, 1992.

[TC92] H. R. Turtle und W. Bruce Croft. A comparison of text retrieval models. *Computer Journal*, 35(3):279–290, 1992.

[Tea01] ADL Technical Team. Sharable Content Object Reference Model (SCORM) Version 1.2, Oktober 2001.

[Tra01] Alexander Trapp. Fuzzy Vergleich in digitalen Bibliotheken am Beispiel des Katalogsystems PIA. Studienarbeit, Fachbereich Informatik, Universität Hamburg, Juli 2001.

[Tre01] Siripong Treetasanatavorn. Semi-automatic Merging of Content Networks: Policy-based Customization. Studienarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Juni 2001.

[TSMB95] Stephanie Teufel, Christian Sauter, Thomas Mühlherr, und Kurt Bauknecht. *Computerunterstützung für die Gruppenarbeit*. Addison-Wesley Publishing Company, Zürich, 1995.

[Uma83] M. Umano. Retrieval from Fuzzy Database by Fuzzy Relational Algebra. In: E. Sanchez, Hrsg., *Fuzzy Information, Knowledge Representation and Decision Analysis. Proceedings of the IFAC (International Federation of Automatic Control) Symposium, Marseille*, Seite 1–6. Pergamon, Juli 1983.

[VN02] Stephen J. Vaughan-Nichols. Web Services: Beyond the Hype. *Computer*, 35(2):18–21, Februar 2002.

[vR75] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1975.

[Wal02] Priscilla Walmsley. *Definitive XML Schema*. The Charles F. Goldfarb Definitve XML Series. Prentice Hall, Englewood Cliffs, Upper Saddle River, 2002.

[Wat90] David A. Watt. *Programming language concepts and paradigms*. Prentice Hall international series in computer science. Prentice Hall, Englewood Cliffs, 1990.

[Weg02a] Holm Wegner. *Analyse und objektorientierter Entwurf eines integrierten Portalsystems für das Wissensmanagement*. Dissertation, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, 2002.

[WEG+02b] Axel Wienberg, Matthias Ernst, Andreas Gawecki, Olaf Kummer, Frank Wienberg, und Joachim W. Schmidt. Content Schema Evolution in the CoreMedia Content Application Platform CAP. In: Christian S. Jensen, Keith G. Jeffery, Jaroslav Pokorný, Simonas Saltenis, Elisa Bertino, Klemens Böhm, und Matthias Jarke, Hrsg., *Proceedings of the 8th International Conference on Extending Database Technology, EDBT'02, Prague*, Band LNCS 2287, *Lecture Notes in Computer Science*, Seite 712–721. Springer-Verlag, März 2002.

[WIC97] Stuart Weibel, Renato Iannella, und Warwick Cathro. The 4th Dublin Core Metadata Workshop Report. *D-Lib Magazine*, Juni 1997.

[Wie02] Ernst August Wieden. Unscharfe Suche in Datenbanken - Konzeptuelle Grundlagen, Systementwurf und prototypische Implementierung. Diplomarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, August 2002.

[Wir71] Niklaus Wirth. The Programming Language Pascal. *Acta Informatica*, 1(1):35–63, 1971.

[XLi01] XML Linking Language (XLink) Version 1.0. W3C Recommendation, Juni 2001.

[XML99] Namespaces in XML. W3C Recommendation, Januar 1999.

[XML00] Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, Oktober 2000.

[XML01a] XML Schema Part 0: Primer. W3C Recommendation, Mai 2001.

[XML01b] XML Schema Part 2: Datatypes. W3C Recommendation, Mai 2001.

[XPa99] XML Path Language (XPath) Version 1.0. W3C Recommendation, November 1999.

[XPT02] XML Pointer Language (XPointer). W3C Working Draft, August 2002.

[XSL01] Extensible Stylesheet Language (XSL), Version 1.0. W3C Recommendation, Oktober 2001.

[Yag88] Ronald R. Yager. On ordered weighting averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, And Cybernetics*, 18(1):183–190, 1988.

[Yag97] Ronald R. Yager. On the inclusion of importances in OWA aggregation. In: Ronald R. Yager und Janusz Kacprzyk, Hrsg., *The ordered weighted averaging operators: theory and applications*. Kluwer Academic Publishers, 1997.

[Zad65] Lotfi A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[Zim01] Hans-Jürgen Zimmermann. *Fuzzy-Set Theory and its Application*. Kluwer Academic, Boston, 4. Auflage, 2001.

Anhang A

Klassendiagramm zur Übersicht über KooMet und KooMet-QL

Das Klassendiagramm in Abbildung A.1 setzt die Klassen des Metadatenmodells KooMet (vgl. Kapitel 4) und der daran angebundenen Anfragesprache KooMet-QL (vgl. Abschnitt 5.2.3) zueinander in Beziehung. In Ergänzung zum Klassendiagramm gelten einige Constraints, die im folgenden anhand ihrer Rolle im Metadatenmodell noch einmal gemeinsam aufgeführt werden:

- Beziehungen als spezielle Records

```
context Beziehung inv:
  self.ressource -> isEmpty()

context Record inv:
  not(self.oclIsTypeOf(Beziehung)) implies
  self.ressource -> notEmpty()

context Rolle inv:
  self.wert.oclType = Ressource
```

- Bildung von Unterbereichstypen durch Einschränkungen

```
context Einschraenkung inv:
  self.unterbereichstyp.einschraenkung ->
  select(e | e.typ <> self.typ) -> isEmpty()
```

- Abgleich von definierter und tatsächlicher Kollektionsart

```
context Kollektion inv:
  self.angegebenerTyp -> notEmpty()
  implies
  self.kollektionsart = self.angegebenerTyp.kollektionsart
```

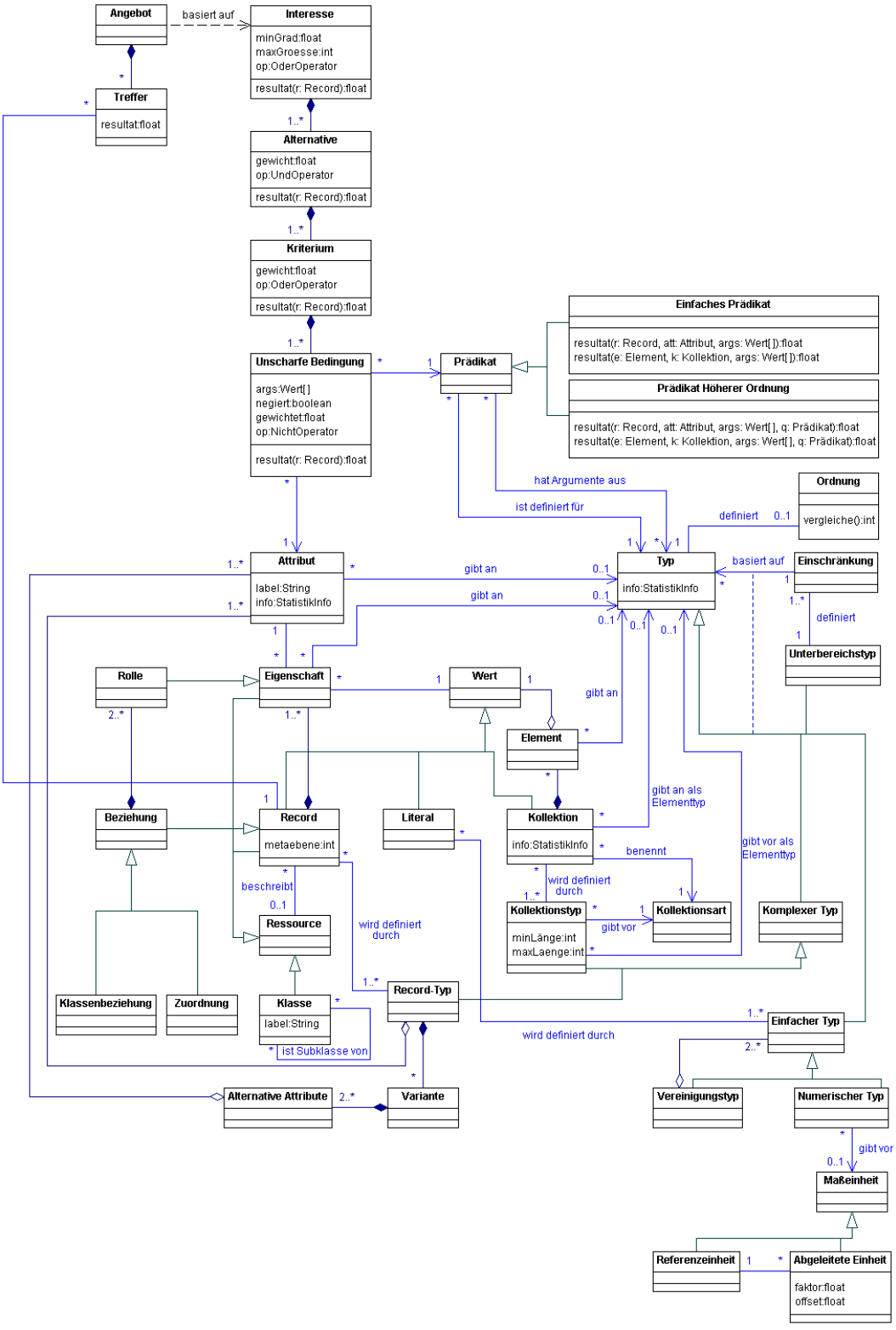

Abbildung A.1: Klassen des Metadatenmodells KooMet und der Anfragesprache KooMet-QL im Überblick

- Zuordnungen und Klassenbeziehungen als spezielle Beziehungen

```
context Beziehung inv:
  self.oclIsTypeOf(Zuordnung) implies
  self.rolle ->
  (size = 2 and
   exists(r | r.wert.oclIsTypeOf(Klasse)) and
   exists(r | r.wert.oclIsTypeOf(Ressource)))

context Beziehung inv:
  self.oclIsTypeOf(Klassenbeziehung) implies
  self.rolle ->
  (size = 2 and
   forAll(r | r.wert.oclIsTypeOf(Klasse))
```

- Trennung von verschiedenen Metaebenen für Meta-Metadaten

```
context Record inv:
  self.ressource.oclIsTypeOf(Record) implies
  self.metaebene > self.ressource.metaebene

inv:
  self.ressource.oclIsTypeOf(Eigenschaft) implies
  self.metaebene > self.ressource.record.metaebene
```

- Sicherstellung des Typs für Prädikate

```
context Kriterium inv:
  self.attribut.typ.notEmpty() implies
  self.attribut.typ = self.praedikat.istDefiniertFuer
```