Entwicklung und Implementierung eines modellgestützten Datenrepositories: Das KBS Hyperbook System

Vom Fachbereich Elektrotechnik und Informationstechnik

der Universität Hannover

zur Erlangung des akademischen Grades

Doktor-Ingenieur/Doktor-Ingenieurin

genehmigte Dissertation

von

Dipl.-Ing. MARTIN WOLPERS

geboren am 14. Februar 1969 in Langenhagen

2001

1. Referent: Prof. Dr. techn. Dipl.-Ing. Wolfgang Nejdl
2. Koreferent: Prof. Dr.-Ing. Bernado Wagner
30. November 2001

Zusammenfassung

Im Rahmen dieser Arbeit wurde das KBS Hyperbook System zur Verwaltung, Speicherung und Bereitstellung von Hypermedia-Dokumenten entwickelt und implementiert. Dieses WWW-basierte Hypermedia-System bildet als Forschungssystem die Basis für weitere Arbeiten, unter anderem in den Themenbereichen Modellierung und Adaption der Inhalte des Systems. Darüber hinaus wird das System beispielsweise in der Lehre als Vorlesungsskript, Nachschlagewerk und zur Prüfungsvorbereitung von Lehrenden und Lernenden eingesetzt.

Zu Beginn der Arbeit wurden zunächst grundlegende Anforderungen auf der Basis der Aufgabenstellungen der Systeme an ein solches Hypermedia-System hergeleitet. Die Aufgabenstellungen umfassen die Erweiterbarkeit des Systems um neue Inhalte und neue Funktionalitäten, seine Offenheit gegenüber Hypermedia-Dokumenten, die Flexibilität des Systems gegenüber unterschiedlichen Verwendungsszenarien und Modifikationen des Inhalts sowie die Wiederverwendbarkeit der Hypermedia-Dokumente und ihrer Beschreibungen. Diese Anforderungen werden vorteilhaft durch den Einsatz von Datenmodellen, die in der Modellierungssprache O-Telos geschrieben sind, im KBS Hyperbook System erfüllt. Datenmodelle beschreiben unter Verwendung von Meta-Informationen und Meta-Modellen die Informationen und die Struktur der durch die Hypermedia-Dokumente dargestellten Wissensgebiete (Domänen). Darüber hinaus nutzt das KBS Hyperbook System die Datenmodelle auch zur Realisierung zusätzlicher Funktionen wie beispielsweise der Benutzeradaption.

Ein in dieser Arbeit angefertigter Vergleich existierender modellbasierte Hypermedia-Systeme zeigt, daß deren Ansätze zumeist entweder die Domänen- oder die Benutzermodellierung verwenden. Das KBS Hyperbook System kombiniert diese beiden Ansätze, indem sowohl die explizite Domänen- als auch die explizite Benutzermodellierung unter Einsatz von Metamodellierung verwendet werden. Hierdurch ist es möglich, im KBS Hyperbook System die notwendige Flexibilität, Erweiterbarkeit und Offenheit zu realisieren. An verschiedenen Beispielen aus den Bereichen der Terminologien, der Lehre und der Integration von Resource Description Frameworks (RDF) Schemata werden die Modelle und die zugehörigen Hyperbooks ausführlich beschrieben. Insbesondere für die Integration von RDF-Modellen wurde das O-Telos-RDF Schema als Alternative zum RDF-Schema entwickelt und beschrieben.

Das KBS Hyperbook System ist unter Verwendung eines modularen Ansatzes in der Programmiersprache Java implementiert. Es setzt sich aus den Modulen für den Anschluß der Datenbank (Modul Datenbank), für die Verwaltung der Daten-Objekte (Modul Storage), für die Erstellung der WWW-Seiten (Modul Visualisierung) und für die mittels eines Servlets realisierte Integration in einen Web-Server (Modul Hyperbook) zusammen. Als Datenspeicher wird eine objekt-orientierte Datenbank (z.B. ConceptBase oder ObjectStore) verwendet. Als Webserver wird ein handelsüblicher Servletfähiger Webserver eingesetzt, z.B. der JavaWebServer der Firma Sun.

Der modulare Aufbau des KBS Hyperbook Systems ermöglicht den Austausch der Module, wobei die Kompatibilität durch Schnittstellenspezifikationen sichergestellt

wird. Auf diese Weise wird auch die Integration weiterer Module in das System ermöglicht. Beispielsweise verwendet das Modul Adaption die Schnittstellen der Module Storage und Visualisierung zur Integration adaptiver Funktionalitäten in das KBS Hyperbook System.

Das KBS Hyperbook System ist mit einigen der verschiedenen Anwendungsfälle unter der URL `http://www.kbs.uni-hannover.de/hyperbook` veröffentlicht.

Abstract

Within this work the development and implementation of the KBS Hyperbook System is described. It is a WWW-based hypermedia system for administration, storage and provision of hypermedia documents which is also a platform for further research activities, e.g. in the area of data modeling and user adaptation. Furthermore, it is used by teachers and students as a lecture script, as a reference book and for exam preparations.

To start with, the basic requirements for this kind of hypermedia systems are derived from the main tasks of the systems. The requirements comprise extensibility for new contents and new functionalities, openness regarding the hypermedia documents, reusability of the hypermedia documents including their models, flexibility regarding the systems different use cases and flexibility regarding the modifications of the content. These requirements are fulfilled by the usage of meta-, metadata- and datamodels stated in the modeling language O-Telos. By utilizing meta-information and metamodels the datamodels describe the information and the structure of the knowledge domain which is represented by hypermedia documents. Furthermore, the system uses the datamodels for implementing additional functionalities like e.g. user adaptation.

A comparison of existing model-based hypermedia systems shows that they either use the domain modeling or the user modeling approach. The KBS Hyperbook System combines both modeling approaches by describing the knowledge domains and the users in datamodels at the same time. The utilized meta-modeling enables the necessary flexibility, extensibility and openness. Several existing hyperbooks from the various areas of teaching, terminologies and Resource Description Framework (RDF) model integration serve as examples for the description of the systms modeling approach and its capabilities. Besides the development of datamodels an alternative RDF-Schema called O-Telos-RDF is defined for the integration of RDF-models.

The KBS Hyperbook System is implemented in the programming language Java using a modular approach. It consists of modules for the database connection, for the management of the data objects, for the generation of the webpages and for the integration in a webserver which is realized as a servlet. As a data repository an object-oriented database (ConceptBase or ObjectStore) is used. As a webserver a standard server capable of servlets like Suns JavaWebServer is employed.

The modular architecture of the system enables the replacement of the modules or the implementation of new modules while the compatibility is guaranteed by their respective interface specifications. The module adaptation uses for example the interfaces of the modules for storage and visualization in order to integrate additional adaptive functionality into the KBS Hyperbook System.

The KBS Hyperbook System and some of its applications are published at the URI: `http://www.kbs.uni-hannover.de/hyperbook`.

Schlagworte/Keywords

Deutsch: Metadaten, Konzeptuelle Modellierung, Hypermedia-System
English: metadata, conceptual modelling, hypermedia-system

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Beispiele und Quelltexte

Kapitel 1

Einleitung

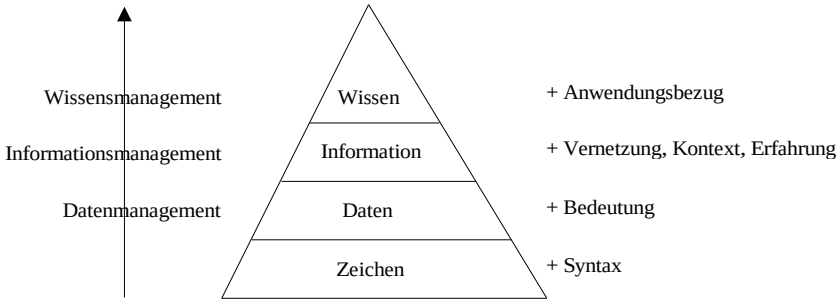

Abbildung 1.1: Die Wissenspyramide, aus [38]

Informationen sind ein wichtiger Baustein der menschlichen Gesellschaft. Beispielsweise stellt das auf Informationen basierende Wissen einen wichtigen Bestandteil bei der Findung von Entscheidungen dar. Nach [38] sind Informationen die Basis zur Bildung von Wissen (vergleiche auch Abbildung 1.1). In [94] wird festgestellt, daß keine Repräsentationsform von Wissen bekannt ist, daher kann Wissen nicht in Dokumenten oder Datenbanken abgelegt werden. Gespeichert werden nur Informationen, die aber individuell immer wieder zu Wissen zusammengesetzt werden.

Die Vermittlung von Informationen wird sowohl durch mündliche als auch durch bildliche/schriftliche Kommunikationswege und -medien vorgenommen. Die schriftlichen Kommunikationsmedien setzen Dokumente ein, um die Informationen zu speichern und auszutauschen. Oftmals werden die Dokumente durch Verweise miteinander vernetzt. Eine Literaturangabe in einem Vorlesungsskript ist ein solcher Verweis. Beispiele für existierende Kommunikationsmedien sind Druckerzeugnisse wie Bücher und Zeitschriften, die eine Reihe von Dokumenten zu einem Wissensgebiet als strukturierte Sammlung enthalten.

Der Austausch von Dokumenten wird in Erweiterung der existierenden Kommunikationswege in immer stärkerem Maße auch mittels des World Wide Webs (WWW) [5]

und der darin verwendeten WWW-Seiten betrieben. Das WWW beherbergt eine große Anzahl von Dokumenten, die als WWW-Seiten oder Hypermedia-Dokumente bezeichnet werden und aus beliebigen Informationen bestehen. Die WWW-Seiten verweisen aufeinander, so daß sie durch diese Verbindungen zu einem Netz verknüpft werden. Das Netz stellt den Kontext dar, in dem die Informationen einer WWW-Seite betrachtet werden. Durch schnell wechselnde Inhalte und neu hinzukommende WWW-Seiten werden die Informationen oftmals in einem für den Leser verwirrenden oder falschen Kontext dargestellt, so daß die Intentionen der Seiten verloren gehen können. Dieses Problem ist insbesondere dann gravierend, wenn durch die WWW-Seiten ein Wissensgebiet beschrieben wird, dessen Zusammenhänge durch die Beziehungen falsch dargestellt werden.

In dieser Arbeit soll ein Hypermedia-System zur Verwaltung und Repräsentation von WWW-Seiten entwickelt werden, das die Konsistenz der enthaltenen Informationen sicherstellt. Das System soll gleichzeitig auch ein offenes System sein, dessen Erweiterbarkeit sowohl inhaltlich um weitere WWW-Seiten als auch funktional um zusätzliche Funktionen möglich sein soll. Offenheit und Erweiterbarkeit sind notwendig, da die in den WWW-Seiten enthaltenen, schnell veränderlichen Informationen oftmals für spezifische Belange eingesetzt werden. Beispiele für solche spezifischen Anwendungen der Informationen sind ein im Internet abrufbarer Zugfahrplan, dessen Informationen an den Benutzer und dessen Wünsche angepaßt werden können (Wahl von Abfahrts- und Ankunftsort, -zeit, etc.), die Präsentation von Börsennachrichten, das Management von Bankkonten und das Erstellen und Lesen von personalisierten Büchern und Zeitschriften.

Ein weiteres Beispiel für eine spezielle Anwendung von Hypermedia-Systemen sind Lehrveranstaltungen. Diese Anwendungen bilden einen Schwerpunkt dieser Arbeit, denn die in den Lehrveranstaltungen verwendeten Lehrmaterialien können und sollen über das WWW den Lernenden zur Verfügung gestellt werden. Die entsprechenden WWW-Seiten beschreiben dann ein Wissensgebiet, das konsistent sein muß, also keine falschen Informationen und Verbindungsstrukturen aufweisen darf. Die Pflege und Wartung des Materials soll im Hypermedia-System einfach gestaltet sein, so daß sich Inhalte schnell ändern lassen, ohne daß die Verbindungsstruktur inkonsistent wird. Außerdem sollen neue WWW-Seiten auf einfache Weise in das Hypermedia-System integrierbar sein.

Zur Repräsentation von Inhalt und Struktur eines durch WWW-Seiten dargestellten Wissensgebiets verwendet das Hypermedia-System dieser Arbeit Datenmodelle. Die Datenmodelle schematisieren und abstrahieren den Inhalt und die Struktur eines Wissensgebiets. Die Modelle beschreiben beispielsweise den Aufbau und den Inhalt einer oder mehrerer Lehrveranstaltungen und der darin verwendeten WWW-Seiten. Sie vereinfachen die Wartung und Pflege des Materials, da sie die Informationen der WWW-Seiten in deren Kontexten darstellen. Die Datenmodelle werden in einer formalisierten Sprache beschrieben, so daß sie in automatisierten Hypermedia-Systemen eingesetzt werden können.

Neben der Modellierung der Wissensgebiete setzt das in dieser Arbeit zu ent-

wickelnde Hypermedia-System Datenmodelle auch für die Benutzeradaption ein. Das Hypermedia-System soll z.B. in der Lage sein, einen Lernenden aufgrund seines aktuellen Wissenstandes und einer gewählten Zielsetzung durch das Lehr-/Lernmaterial zu führen. Die Adaption umfaßt die Anpassung der Navigationsstruktur der WWW-Seiten und/oder die inhaltliche Modifikation der WWW-Seiten entsprechend der Bedürfnisse des Benutzers.

Das in dieser Arbeit entwickelte Hypermedia-System wird KBS Hyperbook System genannt, wobei KBS für die englischsprachige Bezeichnung des Fachgebiets des Instituts steht: **K**nowledge **B**ased **S**ystems. Es ist ein Hypermedia-System zur Pflege und Verwaltung der in WWW-Seiten enthaltenen Informationen. Datenmodelle der Wissensgebiete beschreiben die Inhalte der Dokumente, die im KBS Hyperbook System strukturiert, auf Konsistenz überwacht und zur Präsentation aufbereitet werden. Das KBS Hyperbook System ermöglicht darüber hinaus die Erprobung weiterer, in wissenschaftlichen Arbeiten entwickelter Komponenten, wie etwa die Adaptivitätskomponente als Teil der Dissertation von N. Henze [53].

Die Arbeit weist die folgende Struktur auf: Im Kapitel 2 werden zunächst die Grundlagen von Hypertext- und Hypermedia-Systemen erläutert. Auf der Basis der Grundlagen werden dann Anforderungen an Hypermedia-Systeme entwickelt und beschrieben. Ein Beispiel, in dem das KBS Hyperbook System in einer universitären Lehrveranstaltung eingesetzt wird, verdeutlicht die identifizierten Anforderungen. Diese lassen sich vorteilhaft durch den Einsatz von Datenmodellen der Wissensgebiete implementieren. Die Modelle werden neben der reinen Verwaltung der Hypermedia-Dokumente auch für die Repräsentation der Hypermedia-Dokumente und deren Vernetzung eingesetzt.

Ein Vergleich einer Reihe modellbasierter Hypermedia-Systeme in Kapitel 3 zeigt, daß die Datenmodelle neben der Modellierung des Wissensgebiets auch zur Benutzeradaption verwendet werden können. Die beschriebenen Systeme verwenden im allgemeinen entweder den Ansatz der Modellierung des Wissensgebiets oder den der Modellierung des Benutzers. Das KBS Hyperbook System kombiniert beide Ansätze, um eine explizite Darstellung des Wissensgebiets und eine Benutzeradaption desselben zu erreichen.

Ein Vergleich zwischen den Anforderungen an die Modellierung der Datenmodelle und den Anforderungen an Hypermedia-Systeme verdeutlicht, daß die Anforderungen an Hypermedia-Systeme sich größtenteils auf die ihnen zugeordneten Datenmodelle übertragen lassen. Verschiedene dieser Datenmodelle, die im KBS Hyperbook System eingesetzten werden, werden im Kapitel 4 erläutert. Diese Beispiele existierender Hyperbooks dokumentieren die Modellierung und die Vielfältigkeit der Einsatzgebiete des KBS Hyperbook Systems. Darüber hinaus werden in diesem Kapitel die Integration von Resource Description Framework (RDF) Modellen in das KBS Hyperbook System und ein alternatives RDF-Schema namens O-Telos-RDF für die Modellierung mittels RDF vorgestellt. Das Schema O-Telos-RDF resultiert aus einem Vergleich von RDF mit der Modellierungssprache O-Telos und kombiniert die Vorteile beider Modellierungssprachen.

Im Kapitel 5 werden die Eigenschaften des KBS Hyperbook Systems ausführlich beschrieben, und es wird dargestellt, daß es die im Kapitel 2 aufgestellten Anforderungen erfüllt. Die Implementierung des Systems wird in Kapitel 6 vorgestellt. Ausgehend von der Beschreibung einer vom System generierten WWW-Seite und der Abbildung der Modelle der Wissensgebiete im System werden die einzelnen Module und Klassen des Systems erörtert. Zusammenfassung und Ausblick beschließen diese Arbeit.

Kapitel 2

Hypertext-/ Hypermediasysteme: Das Hyperbook

In diesem Kapitel werden zunächst die in dieser Arbeit verwendeten Begriffe erklärt. Es wird insbesondere auf die Definitionen von Hypertext und Hypermedia und die darauf aufsetzenden Hypertext- und Hypermedia-Systeme eingegangen.

Auf der Basis der Definitionen und unter Berücksichtigung der Einsatzgebiete der Systeme, wie z.B. in der Lehre, werden dann eine Reihe von Anforderungen an die Systeme abgeleitet. Ein Beispiel aus der universitären Lehre erläutert im Anschluß die aufgestellten Anforderungen.

Zum Ende des Kapitels wird dargestellt, daß die entwickelten Anforderungen der Systeme durch den Einsatz von Datenmodellen größtenteils erfüllt werden.

2.1 Hypertext / Hypermedia

Der Begriff Hypertext wurde von Ted Nelson [89] während seiner Arbeit im Xanadu Projekt (1965) geprägt. Hypertext bezeichnet ein oder mehrere Dokumente, die durch assoziative bidirektionale Verbindungen miteinander in Beziehung gesetzt sind. Im Idealfall bestimmt der Inhalt der Dokumente, welche Dokumente miteinander in Beziehung stehen. Die Beziehungen sind Bestandteil der Dokumente, so daß ein Dokument immer aus seinem Inhalt und mindestens einer Beziehung zu einem weiteren Dokument besteht.

Der Inhalt der Hypertext-Dokumente besteht aus Texten. Kommen in einem solchen Dokument auch Bilder und sonstige Medien wie Audio oder Video vor, wird üblicherweise von Hypermedia-Dokumenten gesprochen. Die Begriffe Hypertext und Hypermedia werden oft synonym verwendet [78]. In dieser Arbeit wird jedoch explizit der Begriff Hypermedia-Dokument bevorzugt, um der obigen Definition entsprechend die

multimedialen Inhalte der Dokumente herauszustellen.

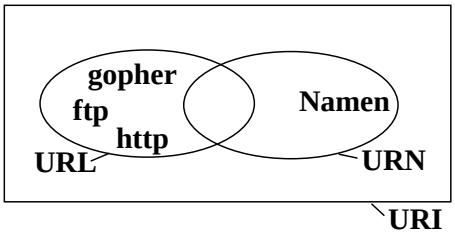

Abbildung 2.1: Die Gültigkeitsbereiche von URL, URN, URI

Ein Hypermedia-Dokument wird im WWW durch einen sogenannten Uniform Resource Identificator (URI) identifiziert [38]. Eine URI existiert im Internet genau einmal. Sie identifiziert eine Ressource im Internet eindeutig. Als Ressource wird alles bezeichnet, was im Internet eine Identität annehmen bzw. dem ein Identifikator zugeordnet werden kann. Neben der URI wird im WWW auch die Bezeichnung Uniform Resource Locator (URL) verwendet, die den Speicherort eines Hypermedia-Dokuments eindeutig beschreibt [7].

Eine URL spezifiziert eine Ressource nach dessen primärer Zugangs- und Speicherart. Damit ist gemeint, daß als Teil einer URL immer auch das Protokoll mitangegeben werden muß, das den Zugang zur jeweiligen Ressource bzw. zum jeweiligen Dokument ermöglicht (z.B. das Hypertext Transmission Protokoll (HTTP)). Die URL ist somit Teil der Menge aller möglichen URIs, da sie nur einen eingeschränkten Geltungsbereich hat: den Netzwerkspeicherort. Abbildung 2.1 verdeutlicht diesen Zusammenhang, indem die URL als Teilmenge der URI dargestellt wird. Um also beliebige Ressourcen im Internet identifizieren zu können, wird heutzutage die URI verwendet. Sie kann eine URL oder eine Uniform Resource Name (URN) sein. Eine URN stellt die Untermenge von URIs dar, die auch nach dem Löschen der entsprechenden Ressource noch gültig bleiben muß (s. wiederum Abbildung 2.1). Entsprechend lautet die folgende Definition für URIs aus dem Request for Comments (RFC) 2396 [6]:

> "A Uniform Resource Identifier is a compact string of characters for identifying an abstract or physical resource." (Aus [6])

URLs und URNs reichen also zur eindeutigen Identifikation eines Hypermedia-Dokuments nicht aus. In dieser Arbeit wird daher die URI zur Identifikation der Hypermedia-Dokumente verwendet.

Eine URI setzt sich aus drei, wahlweise vier Komponenten zusammen. Zuerst wird das Schema angegeben, das den Namensraum der URI definiert. Als Namensraum werden unter anderem die Protokolle angesehen, mit denen z.B. eine WWW-Seite über das Internet übertragen wird (HTTP, FTP, Gopher, etc.). Die Protokolle werden daher in Abbildung 2.1 als zur URL gehörig dargestellt. An das Schema schließt sich der Pfad zur beschriebenen Ressource an. Der Pfad wiederum besteht beispielsweise aus dem

Server, auf dem die Ressource zu finden ist und aus dem absoluten Pfad zur Ressource auf dem Server. An das Schema und den Pfad wird dann wahlweise ein Textstring angefügt, der von der Ressource selbst interpretiert wird und ohne Bedeutung für den Webserver ist.

Beispiel 1:

```
http://www.kbs.uni-hannover.de/hyperbook/index.html
```

Beispiel 2:

```
http://www.kbs.uni-hannover.de/
                          KBSbozen?do=Informatik1&m=gesamt
```

Beispiel 2.1.1: Zwei Beispiele für gültige URIs

In Beispiel 2.1.1 sind zwei Beispiele für gültige URIs angegeben. Das erste Beispiel `http://www.kbs.uni-hannover.de/hyperbook/index.html` ist die URI einer Ressource `index.html`. Diese Ressource wird über das im Internet gebräuchliche Protokoll HTTP vom Webserver `www.kbs.uni-hannover.de` geladen. Der Webserver findet die Ressource durch Verfolgen und Auflösen des angegebenen Pfads `/hyperbook/`. Das zweite Beispiel `http://www.kbs.uni-hannover.de/KBSbozen?do=Informatik1&m=gesamt` beschreibt eine Ressource `KBSbozen`, die mittels des HTTP Protokolls vom Server `www.kbs.uni-hannover.de` geladen wird. Dieser Ressource wird zur weiteren Verarbeitung der Textstring `do=Informatik1&m=gesamt` übergeben. In diesem Beispiel enthält der Textstring zwei Variablen-Paare `do=Informatik1` und `m=gesamt`. Programme zum Anzeigen von WWW-Seiten, Browser genannt, erzeugen diese URIs, wenn sie eine Anfrage an einen Webserver mittels der HTTP-Get-Methode senden. Der Browser übernimmt die Variablenpaare aus einer Eingabe des Benutzers, z.B. einem WWW-Formular oder einem entsprechend aufgebauten Link.

Im Kontext des Internets bzw. des WWW werden die Beziehungen der Hypermedia-Dokumente untereinander auch “Links” (engl. Verweis) genannt. Ein Link ist somit das unidirektionale WWW-Pendant der Beziehung von zwei Hypermedia-Dokumenten. Für den Hintergrund dieser Arbeit genügt die folgende Definition von Links:

> Ein Link ist eine unidirektionale Beziehung zwischen zwei Ressourcen. Er ist in einer der beiden beteiligten Ressourcen enthalten. Er besteht mindestens aus der URI der zweiten Ressource und einem beliebigen Textstring, der als sein Name angesehen wird. (Aus [93])

Die Programme zum Betrachten der Hypermedia-Dokumente im WWW (Browser genannt) ermöglichen die Aktivierung eines Links mittels Tastatureingabe, Mausklick,

etc. Die Folge des Aktivierens ist, daß der Browser das im Link mittels seiner URI spezifizierte Dokument aus dem WWW lädt und anzeigt [93].

Die Hypermedia-Dokumente werden im WWW durch sogenannte Webserver den Betrachtern zur Verfügung gestellt. Die Webserver wiederum laden die Dokumente beispielsweise aus dem lokalen Dateisystem oder einer Datenbank. In diesem Kontext wird ein System zur Verwaltung, Bereitstellung und Speicherung der Hypermedia-Dokumente und deren Beziehungen (Links) als **Hypermedia-System** bezeichnet. Ein Hypermedia-System beinhaltet einen Speicher, in dem die Hypermedia-Dokumente abgelegt sind. Wenn in diesem Speicher neben den Dokumenten auch deren Beziehungen sowie weitere die Dokumente und Beziehungen beschreibende Daten enthalten sind, wird dieser **Repository** (engl. Lager, Magazin) genannt. Der Speicher bewahrt dabei die Integrität der Dokumente. Ein Repository kann nur die Daten beinhalten, die die Hypermedia-Dokumente und deren Beziehungen beschreiben. Zusätzlich charakterisiert sich ein Repository durch die Eigenschaft, daß die enthaltenen oder beschriebenen Dokumente eine inhaltliche Nähe zueinander aufweisen. Entsprechend wird ein Repository oftmals einem Thema, Themengebiet oder Wissensgebiet zugeordnet.

Hypermedia-Dokumente, die inhaltlich eine Nähe zueinander aufweisen, beschreiben im Allgemeinen ein Themen- oder Wissensgebiet. Wenn das Wissensgebiet durch die Hypermedia-Dokumente fest umrissen und gegen andere Wissensgebiete eindeutig abgegrenzt ist, beschreiben die Hypermedia-Dokumente eine Domäne, die den Inhalt des Wissensgebiets umfaßt. Umgekehrt gilt, daß eine Domäne im Gegensatz zum Wissensgebiets einen eindeutig definierten Inhalt hat. Eine ausführlichere Definition des Begriffs Domäne wird im Kapitel 4.1 vorgestellt.

2.2 Hyperbooks

Ein Hyperbook ist ein Hypermedia-System, das das WWW mit den damit verbundenen Technologien wie der Sprache HTML, den Webservern und Browsern, etc. als Hauptpräsentationsmedium nutzt. Die Entwicklung von Hyperbooks hat mit einer Sammlung von elektronischen Texten [73], die eine Einheit gebildet haben, begonnen. In den meisten Fällen behielten diese Sammlungen eine konventionelle Buchstruktur mit Einteilungen in Kapitel, Unterkapitel, Anhänge, usw. [81] bei. In diesem Zusammenhang wird auch von elektronischen Büchern gesprochen, die nach [61] gedruckte Bücher bezeichnen, die am Bildschirm lesbar sind.

Mittlerweile ist die Übertragung gedruckter Bücher in die elektronische Darstellung zu einem großen Forschungsgebiet geworden [95], auf dem sich auch vermehrt Firmen positionieren [114, 26]. Neben der Entwicklung einfacher WWW-Schnittstellen für solche Bücher (auch Web-Portale genannt), wie z.B. die Darstellung eines Inhaltsverzeichnisses als WWW-Seite [8], werden zusätzliche Hypertext- und multimediale Elemente den elektronischen Büchern hinzugefügt [91]. Eine weitere Entwicklung läßt sich durch das Schlagwort “Disaggregated Content” beschreiben. Es wird versucht, Bücher kapitelweise elektronisch darzustellen und aus diesen Kapiteln neue Bü-

cher zusammenzusetzen [29, 117, 114]. Zur Orientierung der Leser werden Systeme zur Benutzerführung eingesetzt, die das Erstellen eines auf den einzelnen Leser zugeschnittenen Buches unterstützen, z.B. die Slicing Information Technology [26, 11].

Weitergehende Entwicklungen umfassen die Erstellung von Netzwerken aus Dokumenten, die in gedruckter Version nicht realisierbar wären, z.B. Dickens Web [76] basierend auf dem INTERMEDIA-System [75] und die mit dem ATHENA-System [59] realisierten Anwendungen. Diese Hyperbook-Entwicklungen versuchen, die Inhalte der Themengebiete durch die Vernetzung der Dokumente zusammenhängend darzustellen.

Der Begriff **Hyperbook** bezeichnet in dieser Arbeit ein Hypermedia-System, das eine Sammlung von Hypermedia-Dokumenten präsentiert und verwaltet (vergleiche [73]). Ein Hyperbook basiert auf einem Repository, in dem die Inhalte und Strukturen durch Modelle und zugehörige semantische Einheiten dargestellt werden.

In dieser Arbeit wird ein System zur Entwicklung von Hyperbook-Systemen vorgestellt, das am Institut für Technische Informatik, Abteilung Rechnergestützte Wissensverarbeitung entwickelt wird. Es wird “KBS Hyperbook System” genannt, wobei “KBS” synonym für die Anfangsbuchstaben der englischsprachigen Bezeichnung des Fachgebiets des Instituts steht: **K**nowledge **B**ased **S**ystems. Das KBS Hyperbook System ermöglicht die Entwicklung und den Betrieb von Hyperbooks zu beliebigen Themengebieten, unter anderem aus dem Bereich der Lehre. Die Hyperbooks basieren auf Modellen der jeweiligen Themengebiete und den damit assoziierten Dokumenten. Zur Verwaltung der Modelle setzen die Hyperbooks Repositories ein. Aufgrund der Modellbasierung der Hyperbooks liegt ein Fokus ihrer Entwicklung auf der Erstellung und Verarbeitung der Modelle und der jeweiligen Besonderheiten der Modelle im System.

In der Forschung wird verstärkt an der Entwicklung von Hyperbooks für Ausbildungszwecke, insbesondere im Bereich der Entwicklung von Lernumgebungen, siehe [18, 28, 31, 56, 57, 58, 72, 80, 105], gearbeitet. Daher wird im Rahmen dieser Arbeit am Beispiel der Lehrveranstaltung “Grundzüge der Informatik 1 - Einführung in die Programmierung” beschrieben, wie ein mit diesem System entwickeltes Hyperbook in der Lehre eingesetzt wird. Es dient dem Vortragenden z.B. als Präsentationsmedium der Folien, während es dem Studierenden den Lehrstoff um zusätzliche Informationen wie Beispiele, weitere Erläuterungen und Dokumentationen ergänzt präsentiert.

2.3 Anforderungen an Hypermedia-Systeme

An ein Hypermedia-System werden eine Reihe von Anforderungen gestellt, die im Folgenden aufgestellt und erläutert werden. Zur Verdeutlichung wird als Beispiel aus der Lehre das Hyperbook der Vorlesung “Grundzüge der Informatik 1 (GdI 1)” von Professor Nejdl, Institut für Technische Informatik, Fachgebiet Rechnergestützte Wissensverarbeitung, Universität Hannover [85] verwendet. Die Vorlesung vermittelt an-

hand der Programmiersprache Java eine Einführung in die Informatik mit allen dazugehörigen Konzepten. In dieser Vorlesung wird ein Hyperbook eingesetzt, um zum einen den Vorlesungsinhalt zu vermitteln (als Skriptum) und um zum anderen den Studierenden beim Lernen zusätzlich weitere Informationsquellen in einer personalisierten Umgebung zur Verfügung zu stellen. Der Inhalt des Hyperbooks beschreibt das für eine Einführung in die Informatik notwendige Wissensgebiet anhand der Programmiersprache Java und der damit zusammenhängenden Teilgebiete wie Algorithmen, Daten, nebenläufige Programme und Projekte. Die Modellierung der Projekte resultiert aus dem konstruktivistischen Ansatz der Vorlesung, der stark auf der Durchführung von studentischen Programmierprojekten in Gruppenarbeiten aufbaut [54]. Das Hyperbook wird aufgrund der Veranstaltung, in der es eingesetzt wird, "GdI 1 Hyperbook" genannt. Es basiert auf dem KBS Hyperbook System. Das KBS Hyperbook System wird in den Kapiteln 5ff. beschrieben, während hier zunächst die Anforderungen erläutert werden.

Die folgenden Anforderungen leiten sich hauptsächlich aus den Aufgaben der Hyperbooks ab. So soll ein Hyperbook-System der Anforderung der **Erweiterbarkeit** des Systems um neue Inhalte (Dokumente und Beziehungen) gerecht werden. Die Dokumente und die Beziehungen sollen dabei jedoch nicht in ihrer Integrität verändert werden. In der Vorlesung GdI 1 werden die Inhalte der jeweiligen Vorlesung von Veranstaltung zu Veranstaltung erweitert, um neue Erkenntnisse und Weiterentwicklungen darstellen zu können. Gleichzeitig soll es sowohl dem Lehrenden als auch den Lernenden möglich sein, neue Hinweise, Kommentare, Fragen oder Antworten in das System zu integrieren.

Die Erweiterbarkeit bezeichnet auch die **Integration neuer Funktionalitäten** in das Hyperbook-System. Neue Funktionalitäten eines Hyperbooks bezeichnen in diesem Kontext zusätzliche Module zur erweiterten Verarbeitung der Daten. So integriert das in der Vorlesung GdI 1 eingesetzte Lehrsystem beispielsweise eine Adaptionskomponente, die die personalisierte Anpassung der Navigationsstrukturen ermöglicht [53].

Aus der Erweiterbarkeit eines System folgt dessen **Offenheit** gegenüber dem Hinzufügen von neuen Inhalten (Dokumente und Beziehungen der Dokumente). Auf einfache Weise können (beliebige) Nutzer des Systems diesem neue Inhalte vermitteln. Offenheit in diesem Kontext beschreibt aber auch die Fähigkeit des Hypermedia-Systems, Dokumente außerhalb seiner Grenzen aufzunehmen und zu verarbeiten. Entsprechend sollte ein Hypermedia-System für den Einsatz im WWW die Möglichkeit bieten, die im Internet verteilt liegenden Dokumente zu integrieren und zu verwalten, ohne sie von ihrem jeweiligen Ort zu entfernen. Das in der Vorlesung GdI 1 eingesetzte System umfaßt unter anderem WWW-Dokumente aus sehr unterschiedlichen Quellen, wie das Vorlesungsskript GdI 1 [85] und das SUN Java Tutorial [19] in den jeweils aktuellen Versionen. Die Offenheit der Systeme gegenüber der Lokalität der Dokumente unterstützt auch die Wiederverwendbarkeit der Inhalte.

Weitere Anforderungen sind die **Flexibilität** und die **Konsistenzwahrung** des Systems gegenüber der Erweiterbarkeit bzw. der Modifikation von Hypermedia-Dokumenten im System und gegenüber dem Einsatzgebiet des Systems. Neben

dem Einfügen neuer Informationen in ein System müssen die Inhalte ggf. modifiziert oder gelöscht werden können. Das System muß die während dieser Vorgänge möglicherweise entstehenden Inkonsistenzen, also beispielsweise in Hinsicht auf die Hypermedia-Dokumente fehlerhafte und falsche Beziehungen, vermeiden. Flexibilität wird auch vom GdI 1 Hyperbook im Rahmen der Vorlesung GdI 1 gefordert. Dessen Inhalte werden von Semester zu Semester an die entsprechenden Entwicklungen von Lehrenden angepaßt, um der notwendigen Aktualität der Lehrveranstaltung Rechnung zu tragen.

Ein Hypermedia-System muß sich auch gegenüber dem beabsichtigten Einsatzgebiet bzw. Verwendungszweck flexibel zeigen. Wird z.B. ein System in der Lehre eingesetzt, soll es dem Vortragenden während einer Veranstaltung die Präsentation des Inhalts (die Folien) ermöglichen, zur Nachbereitung der Veranstaltung dann dem Lernenden neben den Folien auch zusätzliche Informationen (weiterführende Literatur, Beispiele, Projekte, Aufgabenstellungen, usw.) zum Lehrstoff bieten. Der Inhalt der GdI 1 Vorlesung wird im GdI 1 Hyperbook unter anderem durch das Vorlesungsskriptum und das Sun Java Tutorial dargestellt. Dasselbe GdI 1 Hyperbook wird in der GdI 1 Vorlesung an der Universität Hannover und in einer via Videokonferenz übertragenen Kooperations-Lehrveranstaltung an der Freien Universität Bozen eingesetzt. Aufgrund der technischen Bedingungen einer Videokonferenz wie geringerer Bild- und Tonqualität sind die in der Bozener Veranstaltung verwendeten Vorlesungseinheiten aus weniger Einheiten aufgebaut als in der in Hannover gehaltenen Vorlesung.

Eine weitere Anforderung an Hypermedia-Systeme ist die **Annotation** der Dokumente und Beziehungen mit zusätzlichen Daten. Die zusätzlichen Daten können z.B. eine Zusammenfassung eines Dokuments sein, oder aber Informationen über eine Beziehung wie Namen oder Sichtbarkeit der Beziehung beschreiben. Diese Arten der Annotation werden Metadaten genannt, weil sie aus Daten bestehen, die wiederum Daten im System beschreiben. Beispielsweise sind das in Hinsicht auf die Hypermedia-Dokumente Daten über den Inhalt und die Art der Hypermedia-Dokumente, oder es sind Daten, die die Dokumente strukturieren, wie etwa eine Beschreibung eines Unterthemas, das eine Gruppe von Hypermedia-Dokumenten im System behandelt. So sind im GdI 1 System etwa die Dokumente um zusätzliche Informationen wie eine kurze Inhaltsangabe erweitert, um in den Links bereits eine Art Vorschau auf die Dokumente zu integrieren.

In einem Hyperbook stellen die Dokumente und ihre Beziehungen auf Modellierungsebene gesehen **strukturierte Daten** dar. Beispiele von Datenstrukturen können z.B. eine hierarchische Ordnung oder, allgemeiner, eine Netzstruktur sein. Hypermedia-Dokumente etwa bilden eine Netzstruktur, indem sie durch ihre Beziehungen miteinander vernetzt sind. In der Netzstruktur stellen die Dokumente die Knoten und die Beziehungen die Kanten (Verbindungen) dar. Diese Art der Struktur wird auch semantische Struktur genannt, da die Knoten jeweils einen Inhalt repräsentieren, während die Kanten deren Semantik verdeutlichen.

Die Strukturierung der Daten unterstützt durch die Gruppierung von Hypermedia-Dokumenten in Themen und Unterhemen den Aufbau einer thematischen Hierarchie. Zur Repräsentation solcher Strukturen eignen sich Modelle, die neben den einzelnen

Dokumenten auch deren strukturelle Informationen darstellen (z.B. generell semantische Modelle, spezieller aber auch objekt-orientierte Klassendiagramme). Die Modelle werden üblicherweise in einer anerkannten Modellierungssprache wie O-Telos, Object Modeling Technique (OMT) oder Unified Modeling Language (UML) dargestellt (siehe beispielsweise die in [66, 39, 10, 99, 96, 101, 67, 83] beschriebenen Modellierungsmethoden, Entwurfsverfahren und Modellierungssprachen).

In Bezug auf Hypermedia-Dokumente und deren Strukturierung werden im WWW Modelle verwendet, um den Dokumenten und deren Beziehungen im Rahmen der so gebildeten Netze Semantik zu verleihen [77]. Vor diesem Hintergrund wird das Resource Description Framework (RDF) [13] vom World Wide Web Consortium (W3C) [123] als Modellierungs- und Beschreibungssprache von WWW-Ressourcen entwickelt. Durch den Einsatz von RDF soll das WWW von einer Ansammlung von Hypermedia-Dokumenten mit wenig aussagekräftigen Strukturen in ein strukturiertes Netz aus Hypermedia-Dokumenten transformiert werden. Auch der Ansatz des KBS Hyperbook Systems umfaßt die Entwicklung von semantischen Modellen zur Strukturierung der Hypermedia-Dokumente. So wird z.B. aus dem allgemeineren KBS Hyperbook System durch den Einsatz von Modellen der Veranstaltung, der Inhalte und der Benutzer das GdI 1 Hyperbook generiert.

Der Einsatz von Modellen unterstützt auch die **Wiederverwendbarkeit** von Inhalten und Metadaten. Durch die Generalisierung und Abstraktion im Modellierungsprozess können Modelle geschaffen werden, die übergreifend für mehrere Wissens- oder Einsatzgebiete Gültigkeit haben. Es wurde z.B. im Rahmen des GdI 1 Hyperbooks ein Vorlesungsmodell entwickelt, das auch in anderen Vorlesungen angewendet wird.

2.4 Repräsentation der Daten durch Modelle

Die semantischen Netze, die die Hypermedia-Dokumente im Rahmen der Hyperbooks bilden, werden durch Modelle beschrieben. Die Modelle werden im KBS Hyperbook System eingesetzt, um die jeweiligen Hyperbooks zu generieren. Die verwendete Modellierung erschafft Modelle bestimmter Wissensgebiete, um die Inhalte der Wissensgebiete durch Abstraktionen erfassen, darstellen und dem Rechner in geeigneter Form zur Verarbeitung zuführen zu können. Die Modelle werden also im System durch Daten in unterschiedlichen, aufeinander aufbauenden Abstraktionsebenen dargestellt. Die jeweiligen Abstraktionen der Daten ergeben sich aus den Aufgaben, die das System mit den Darstellungsformen erfüllen kann [88].

In Abbildung 2.2 sind die grundsätzlichen Abstraktionsebenen der die Modelle repräsentierenden Daten dargestellt. Auf der obersten Ebene sind in der Abbildung die Dokumente, Beziehungen und die Metadaten als solche erkennbar, d.h. hier existiert das Modell in der Modellierungssprache im System. Die Modelle werden durch die Sprachelemente der Modellierungssprache ausgedrückt. Diese Ebene dient in erster Linie dem Betrachter bzw. Anwender des Systems, z.B. zur Formulierung des Modells und zum Abrufen der Dokumente. Sie wird daher auch Modellebene genannt.

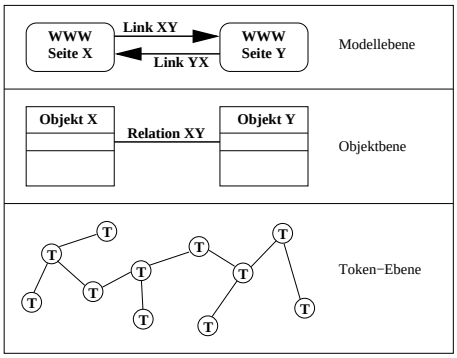

Abbildung 2.2: Die Abstraktionsebenen der Modelle

Die Modelle und deren Modellierung werden in Kapitel 4 eingehend beschrieben.

Die Modellebene basiert, wie aus Abbildung 2.2 ersichtlich wird, auf der Objektebene. In der Objektebene werden die Modelle durch Objekte dargestellt. Die Objekte resultieren aus der objekt-orientierten Darstellung der Sprachkonstrukte der verwendeten Modellierungssprache. Um aus den Modellen die Objekte generieren zu können, werden die Grammatik und Semantik der Modellierungssprache auf die Objekte in der Objektebene angewendet. Die Modelle werden dem System also in maschinenlesbarer Form zugeführt, wobei deren formulierte Abhängigkeiten und Mechanismen (z.B. Vererbung und Instanzierung) aufgelöst werden.

Die Verarbeitung der Modelle, insbesondere die Speicherung und Verwaltung der Objekte im Repository basiert auf der Darstellung der Objekte in der Objektebene durch sogenannte Token. Token sind die atomaren Bestandteile, aus denen die Objekte bestehen. Als atomar wird ein Token bezeichnet, wenn es bei weiterer Teilung seinen Sinn verlieren würde. Besteht z.B. ein Objekt aus einem Bezeichner und einem Attribut, so läßt sich das Attribut als atomares Token darstellen, das über eine Referenz/Relation mit dem das Objekt darstellende Token verbunden ist. Das das Objekt darstellende Token hingegen wird durch den zugehörigen Bezeichner eindeutig identifiziert, weshalb der Bezeichner nicht als eigenständiges atomares Token dargestellt wird.

Die Ebene, in der die Token als solche existieren, wird Token-Ebene genannt (s. Abb. 2.2). Sie ermöglicht die Betrachtung der Modelle losgelöst von ihren bisherigen Darstellungsformen als einfache Daten. Somit können die Modelle unabhängig von der verwendeten Modellierungssprache, der verwendeten Modellierung, der Dokumenten- und Relationstypen verarbeitet und in einer Datenbank beliebigen Typs gespeichert werden. In Bezug auf Hyperbook-Systeme existieren die Dokumente und deren Relationen auf dieser Ebene nicht mehr, womit ihre Betrachtung und Verarbeitung als reine Daten möglich wird.

Kapitel 3

Vorstellung und Vergleich modellbasierter Hypermedia-Systeme

In den vergangenen Jahren wurden eine Reihe von Methoden zur Modellierung von Hypermedia-Domänen entwickelt. Insbesondere für die Hochschullehre wurden diverse Systeme entwickelt, die sowohl Lehrenden als auch Lernenden in der Ausübung der jeweiligen Aufgaben (Lehre und Lernen) unterstützen sollen. Diese Systeme beschreiben zum einen eine Domäne, die durch eine Sammlung von Hypermedia-Dokumenten, welche inhaltlich einander nahe sind, repräsentiert wird. Zum anderen ermöglichen sie oftmals eine Anpassung des Inhalts und der Navigationsstruktur an den spezifischen Benutzer.

Bestehende Systeme verwenden im Allgemeinen einen von zwei Ansätzen: Der eine Ansatz geht von einem Domänenmodell aus, dessen Inhalt eine Sammlung von Hypermedia-Dokumenten beschreibt. Dieser eher datenbank-orientierte Ansatz verwendet Methoden zur Erstellung der Domänenmodelle, die aus der Softwaretechnik (z.B. Object Modeling Technique (OMT) [96]) oder aus dem Datenbankentwurf (z.B. Entity Relationship (ER) [22]) bekannt sind. Der Fokus der Systeme, die diesen Ansatz verwenden, liegt auf der Entwicklung des Datenmodells. Die Datenmodelle werden zur Verwaltung bzw. Indexierung der Dokumente und der Darstellung der Beziehungen zwischen den Dokumenten eingesetzt. Auf der Basis dieser Modelle werden z.T. unter Zuhilfenahme weiterer Navigations- und Visualisierungsmodelle die Hypermedia-Dokumente und ihre Beziehungen dargestellt. Vorhandene Benutzermodelle und/oder statische Auswertungsmechanismen ermöglichen in Abhängigkeit vom jeweiligen System die Adaption der Navigationsstrukturen und der Hypermedia-Dokumente selbst an die Bedürfnisse der Benutzer.

Der andere Ansatz geht von der Entwicklung eines Benutzermodells aus, durch dessen Einsatz eine Sammlung von Hypermedia-Dokumenten an die Bedürfnisse des Benutzers angepasst wird. Dieser wird zur Entwicklung adaptiver Hypermedia-Textbücher

(siehe z.B. [72, 17, 18, 15]) verwendet. Das in den Benutzermodellen reflektierte Wissen über die Benutzer wird eingesetzt, um zum einen die Navigationsstrukturen zwischen Hypermedia-Dokumenten zu modifizieren und/oder zu annotieren oder zum anderen, um den Inhalt des aktuellen Hypermedia-Dokuments anzupassen. Eine Modifikation der Navigationsstrukturen kann das Hinzufügen oder das Ausblenden von Strukturen sein, je nachdem, welchem Lehr-/Lernmodell das System folgt. Eine Annotation der Navigationsstrukturen kann eine Ergänzung der Strukturen um zusätzliche Informationen sein. Beispielsweise können Farben von Links andeuten, ob es für einen Benutzer sinnvoll ist, diesem Link zu folgen. Auch eine aus dem Dokument gewonnene kurze Zusammenfassung kann einen Link annotieren. Es soll erreicht werden, auf diese Weise dem Benutzer bei der Entscheidung zu helfen, ob er diesem Link folgt oder nicht. In diese Systeme sind häufig auch interaktive Komponenten wie Tests mit Auswertungsmechanismen integriert, die dem Benutzer die Einschätzung des eigenen Wissens ermöglichen. Die Modelle werden in semantischen Netzen dargestellt, wobei die Knoten der Netze die Hypermedia-Dokumente indizieren. Die Beziehungen zwischen den Knoten werden durch die Benutzermodelle modifiziert bzw. annotiert. Diesen Methoden fehlen aussagekräftige Domänenmodelle mit der Folge, daß existierende Seitensammlungen nur durch die Benutzermodelle wie Bücher indiziert werden.

In existierenden Systemen wird oftmals nur ein Ansatz verfolgt, wie folgende Ausführungen zeigen.

3.1 Beispiele des Domänenmodell-orientierten Ansatz

Eine Methode, die den Domänenmodell-orientierten Ansatz verwendet, ist die *Hypertext Design Method (HDM)* [43, 44]. Sie stellt ein Objektmodell für hierarchisch strukturierte Einheiten, d.h. Teile von Hypertext-Dokumenten, zur Verfügung. Die Modellierungstechnik ähnelt der ER-Technik, setzt zur Modellierung aber auch strukturierende und hierarchische Einheiten ein.

Eine weitere Methode ist die *Relationship Management Methodology (RMM)* [63], deren Modelle auf HDM und ebenfalls auf ER-Diagrammen basieren. ER-Diagramme werden um die Konzepte der Navigation wie Link, Group und Index erweitert. Die Relationen zwischen den ER-Einheiten werden als mögliche Relationen zur Navigation des Modells betrachtet, aus denen die im Hinblick auf die Intention des Systems angemessenen Relationen zur Navigation ausgewählt werden können. Sie werden daher vom semantischen Modell getrennt in einem eigenen Modell dargestellt. Dieser Ansatz verwendet zur Speicherung der Hypermedia-Inhalte eine Datenbank. Der Zugriff auf die Inhalte der Datenbank wird durch die Indexierung des semantischen Modells ermöglicht.

HDM wird in der *Object-Oriented Hypertext Design Method (OOHDM)* [103] um objekt-orientierte Konzepte ergänzt, deren Modelle in der Sprache OMT [96] mit zusätzlichen Konstrukten ergänzt deklariert werden. OMT wird um Konzepte für Sichtweisen (Views) auf Domänenobjekte und Relationen zur Generierung von Links er-

weitert. OOHDM setzt neben dem konzeptuellen Modell unterschiedliche weitere Modelle zur Navigation der Inhalte ein. Diese Navigationsmodelle beschreiben explizit die möglichen Navigationsarten und das Layout jeden Typs der Konzepte des Domänenmodells. Die Navigationsmodelle werden in Abhängigkeit der Intentionen der Benutzer entworfen. Jedes Navigationsmodell beschreibt eine Sicht auf die aktuelle Domäne. Beispielsweise gibt es ein Navigationsmodell zur Beschreibung der Domäne als Skript, ein weiteres für den Einsatz als Nachschlagewerk, usw.

Das *Hypertext Based On Statecharts (HMBS)* System [111] setzt zur Wiedergabe der Hypermedia-Dokumente ein ausführliches Visualisierungsmodell ein. Die Navigationsstruktur des HMBS Systems basiert auf der Textstruktur der Dokumente der Domäne (Kapitel, Unterkapitel, etc.), setzt also kein semantisches Modell zur Beschreibung der Domäne ein. Das verwendete Modell entspricht vielmehr einer Art Finite State Machine Diagramm oder Zustandsübergangsdiagramm, das um strukturierende Elemente zur Hierarchiebildung, Aggregation, etc. erweitert wird. HMBS stellt einen systematischen Ansatz zur Navigation in formal strukturierten Dokumenten dar. Es ermöglicht also die Navigation in hierarchisch strukturierten Domänen.

Ein weiteres System dieser Art ist das *World Wide Web Design Technique (W3DT)* System [9]. Im W3DT System wird ein Navigationsmodell der Domäne verwendet, das semantische Einheiten beinhaltet. Die Beziehungen der Einheiten werden als HTML-Links modelliert. Die Einheiten selbst werden semantisch als Seite, Menü, Formular, Index, etc. deklariert. Die aktuelle Domäne wird mit Hilfe dieser Konstrukte modelliert, wobei keine Konstrukte zur Modellierung ihres Inhalts zur Verfügung stehen. Im Gegensatz zu RMM und OOHDM beschreibt die Modellierung hier die Navigation in einer Domäne, jedoch nicht den Inhalt der Domäne selbst.

Das *Multimedia Supported Systems Engineering Hypermedia-Database and Client System (MUSE-HM DB and Client System)* [3] setzt ein Domänenmodell zur Strukturierung und Speicherung der multimedialen Dokumente ein, die bei dem Entwurf und der Validierung von integrierten Systemen anfallen. Das verwendete Domänenmodell beschreibt die Metaklassen zur Darstellung hypermedialer Inhalte, zu denen sowohl Konzepte als auch Relationen zählen. Konzepte können zusammengesetzte oder unteilbare Einheiten sein, wobei die Inhalte der Konzepte durch eigenständige Klassen dargestellt werden. Mittels dieser Metaklassen wird die Domäne beschrieben, wobei zur Relation der Konzepte nur binäre Linktypen eingesetzt werden. Die Navigation des Domänenmodells beruht auf den darin verwendeten Beziehungen, die in einem Browser direkt abgebildet werden. Das System verwendet eine Datenbank, die zur Speicherung des Domänenmodells und zum Zugriff auf das Modell eingesetzt wird.

Die bisher vorgestellten Methoden eigenen sich nach [63] am besten für das Design von Zugriffsmöglichkeiten auf strukturierte Daten. Die in diesen Methoden verwendeten Navigationskonzepte werden für die Indizierung großer Mengen relativ einfach zusammenhängender Informationen konzipiert. Sie strukturieren eine Domäne mittels Klassen, Typen, Objekten und deren Beziehungen untereinander. Die eingesetzten semantischen Modelle bestehen im Allgemeinen aus wenigen Konzepten. Die Hypermedia-Dokumente werden entsprechend dieser Modelle klassifiziert, so daß Na-

vigationselemente automatisch auf der Basis der Beziehungen generiert werden können.

Diese Methoden unterstützen entsprechend den jeweiligen implementierten Modellierungen eine begrenzte Menge an Navigationskonzepten. Die Hypermedia-Dokumente werden aus einzelnen Informationsteilen aufgebaut, deren Zusammenhang auf den Modellen beruht. Die Modelle speichern die Informationsteile direkt (RMM) oder indirekt (HDM, OOHDM, z.B. im Dateisystem, etc.) und verbinden sie untereinander, z.B. mittels Relationen, Links oder ähnlichen Konstrukten.

3.2 Beispiele des Benutzermodell-orientierten Ansatzes

Ein Beispiel des Benutzermodell-orientierten Ansatzes ist der *ELM-ART Tutor* [17, 15, 120], ein interaktiver Lisp-Tutor. Dieser stellt neben den Hypermedia-Dokumenten zur Programmiersprache auch interaktive Übungen zur Verfügung. Die interaktiven Übungen werden mittels eines in das System integrierten Lisp Interpreters realisiert. Jede Einheit dieses Systems enthält den Text der Seite, Informationen über die Beziehungen der Seite zu anderen Seiten, und eine Beschreibung der Abhängigkeiten von bzw. zu anderen Konzepten. Die Abhängigkeiten der Konzepte untereinander bilden das konzeptuelle Modell, in dem das Wissen über die Domäne abgebildet wird. Das Benutzermodell wird anhand der jeweiligen Aufgabenstellung und des Domänenmodells sowie des vom Benutzer erstellten Quelltexts generiert. Das Benutzermodell basiert auf einem episodischen Lerner-Modell [119, 118] und dient unter anderem der Adaption der Navigationsstruktur.

Die Strukturen des Modells werden im Nachfolgesystem *ELM-ART II* [120] durch die Implementierung von Lerneinheit, Kapitel, Unterkapitel, Aufgabenstellungen und Tests an übliche Buchformen angepasst, so daß in diesem System Textbücher in elektronische Versionen konvertiert werden können.

Das auf dem ELM-ART Tutor basierende *InterBook* Projekt [18, 16] stellt auf der Basis eines Benutzermodells adaptive Navigationsmöglichkeiten zur Verfügung. Icons, Fonts und Farben spezifizieren die unterschiedlichen Lehr-/Lernzustände eines Hypermedia-Dokuments [32]. Ein semantisches Netz beschreibt das Domänenmodell zur Indizierung des Hyperbooks. Das semantische Netz weist eine dem ELM-ART Tutor II ähnliche hierarchische Struktur mit Kapiteln, Unterkaptiteln, etc. auf. Sowohl ein Glossar als auch das aktuelle Textbuch basieren auf dem Domänenmodell. Das Glossar ist eine visuelle Darstellung des Domänenmodells, das die didaktische Struktur des Hyperbooks darstellt. Die Einheiten des Textbuchs indizieren mehrere Domänenmodell-Einheiten. Diese Indizierung beschreibt, welches Wissen ein Benutzer vor dem Lesen der Seite haben sollte, und welches er nach dem Lesen der Seite hat.

Das System *Slicing Book Technology* [26] dient ebenfalls der Wandlung von existierenden Büchern in elektronische Hyperbücher. Die Texte werden in einer hierarchischen

Struktur u.a. nach Buch, Kapitel, Unterkapitel und durch Unterscheidung in Basis- und Fortgeschrittenenwissen klassifiziert. Diese semantischen Einheiten werden mit Metadaten wie Typ, Titel, Relationen zu anderen Einheiten, Schlagworten, etc. annotiert. Die Adaption basiert auf dem vom Benutzer gewählten Lernziel unter Berücksichtigung des Basis- und Fortgeschrittenenwissens der Einheiten und der bereits gelesenen Einheiten. Die Einheiten werden dann vom System zu einem Dokument zusammengestellt und an den Benutzer gesendet.

3.3 Kombination der Ansätze im KBS Hyperbook System

Die in den beiden vorstehenden Kapiteln 3.1 und 3.2 beschriebenen Ansätze verwenden entweder ein Modell zur Beschreibung der Domäne oder des Benutzers. Die im Kapitel 3.1 vorgestellten Ansätze setzen objekt-orientierte Methoden ein, um ein Domänenmodell zu erstellen. Dieses verwenden sie als Basis zur Generation der Links aus den Relationen zwischen den Objekten. Auf diese Weise ist keine Adaption des Inhalts und der Navigationsstrukturen zwischen den Objekten möglich. Dafür unterstützt diese Art der Modellierung aber die Konsistenz und Kohärenz der Modelle bei Erweiterungen und Modifikationen der Modelle und Domänen.

Im Gegensatz dazu verwenden die im Kapitel 3.2 vorgestellten Beispiele ein Benutzermodell, um Hypermedia-Dokumente zu indizieren und eine Benutzeradaption der Inhalte zu ermöglichen. Ein auf diesem Ansatz basierendes System ist jedoch nicht in der Lage, das verwendete Benutzermodell bei Erweiterungen und Modifikationen der Domäne und des Modells konsistent und kohärent zu halten.

Im KBS Hyperbook System werden die beiden Ansätze kombiniert, um deren Vorteile zu nutzen. Auf der Basis der verwendeten Domänenmodelle sorgt das System für die Konsistenz der zugehörigen Domänen und Modelle. Das Benutzermodell wird für die Adaption der Navigationsstrukturen und der WWW-Seiten eingesetzt, welche die Hypermedia-Dokumente beinhalten. Im Folgenden werden diejenigen Ansätze der zuvor vorgestellten Beispiele näher erläutert, die vom KBS Hyperbook System übernommen werden.

Einige Beispiele, insbesondere RMM und OOHDM, führen die semantische Modellierung für Hypermedia-Dokumentsammlungen ein. Basierend auf Modellierungstechniken aus dem Datenbankentwurf werden Domänenmodelle zur Beschreibung des Wissens der Domäne geschaffen. Gleichzeitig werden von Systemen wie ELM-ART und InterBook eigenständige Benutzermodelle eingeführt, die den Navigationsaufbau eines Hyperbooks verändern können oder vollständig bestimmen.

Das KBS Hyperbook System generalisiert die in den Beispielen verwendeten Methoden RMM und OOHDM, indem Metadaten und konzeptuelle Daten von den inhaltlichen Daten und Hypermedia-Dokumenten getrennt werden. Die Metadaten modellieren explizit alle relevanten Einheiten eines Hyperbooks, während die inhaltlichen Daten die tatsächlichen Hypermedia-Dokumente referenzieren (ähnlich der in [90] be-

schriebenen Idee der Indizierung). Die Hypermedia-Dokumente werden nicht wie bei RMM durch Attribute, sondern durch eigenständige Konzepte dargestellt. Wie auch im OOHDM verwendet das KBS Hyperbook System konzeptuelle Modelle, die das Wissen einer Domäne beschreiben.

Die im KBS Hyperbook System verwendete Modellierungssprache O-Telos erlaubt im Vergleich mit dem im OOHDM eingesetzten OMT die Formulierungen von zusätzlichen deduktiven Regeln und Einschränkungen zur Bildung neuer Konzepte und Beziehungen sowie zur Konsistenzwahrung. Beispiele der im KBS Hyperbook System verwendeten Regeln und Einschränkungen finden sich im Anhang A.5. Die Modellierungstechnik ähnelt der im RMM eingesetzten, erweitert diese jedoch um das Vererbungskonstrukt. Im Gegensatz zu den Systemen RMM, HDM und OOHDM, aber ähnlich dem MUSE HM-DB erlaubt das KBS Hyperbook System die Verwendung mehrerer Abstraktionsebenen in einem Domänenmodell, um die Komplexität großer Domänen explizit beschreiben zu können. Auf diese Weise werden auch Sichtweisen auf die Hypermedia-Dokumente und deren Inhalte möglich, die in anderen Metamodellierungstechniken wie IRDS [64, 65] noch nicht enthalten sind. Die Sichtweisen werden in dieser Arbeit zur Beschreibung unterschiedlicher Aspekte der Domäne verwendet.

Die Visualisierung im KBS Hyperbook System verwendet die Beziehungen zwischen Konzepten zur Generierung von Navigationsstrukturen. Die Beziehungen deklarieren auf der schematischen Ebene bereits, auf welche Weise sie die Navigation im Datenmodell unterstützen. Mögliche Navigationselemente können einfache Links, Indizes, etc. sein. Diese Art der Navigationsdeklarationen wird auch im RMM und im OOHDM verwendet. Das KBS Hyperbook System erweitert diese Navigationsdeklarationen um diverse Indexe. Außerdem implementiert es die Möglichkeit, Sequenzen von Hypermedia-Dokumenten explizit zu modellieren. Im Gegensatz zu RMM werden im Domänenmodell enthaltene Relationen als Beziehungen der Dokumente dargestellt.

Neben dem Domänenmodellen werden im KBS Hyperbook Benutzermodelle verwendet, die die Anpassung der Navigationsstrukturen der Hyperbooks an den jeweiligen Benutzer ermöglichen. Die Adaption im KBS Hyperbook System wird in [53] eingehend beschrieben. Die Einführung der Benutzermodelle, wie sie z.B im ELM-ART Tutor verwendet wird, ermöglicht die Beibehaltung der Buchstruktur, so daß auf diese Weise die meisten Bücher in elektronische Hyperbooks gewandelt werden können. In diesen Systemen wird jedoch auf die Verwendung eines Domänenmodells verzichtet, so daß die Struktur der Domäne nicht explizit ausgedrückt und dem Benutzer vermittelt werden kann.

Kapitel 4

Diskussion der Modelle

Wie bereits in den Kapiteln 2.3 und 2.4 dargestellt, werden Datenmodelle als Basis von Hyperbook-Systemen eingesetzt [63, 103]. Sie modellieren die Inhalte der Systeme, stellen also deklarativ die Dokumente und deren Beziehungen untereinander dar. Werden in der Modellierung auch Metadaten eingesetzt, so beschreiben die Modelle neben thematischen Inhalten des Systems auch die im System enthaltenen Daten. Auf diese Weise wird die Trennung von Inhalten nach Daten und Metadaten und die Navigation dieser Inhalte ermöglicht.

Die im Kapitel 2.3 hergeleiteten Anforderungen an ein Hyperbook-System lassen sich zum Teil auf die verwendeten Modelle übertragen. Im Folgenden werden daher neben den zugehörigen Definitionen der Modelle die entsprechenden Anforderungen an diese analog der Anforderungen an die Systeme erläutert.

4.1 Domänen- und Repräsentationsmodelle

Der Begriff Domäne beschreibt in diesem Kontext das Wissensgebiet, das in dem Modell beschrieben werden soll. Die in dieser Arbeit verwendete Definition der Domänenmodelle lautet somit wie folgt:

> "The **domain model** is a declarative description of the objects and activities possible in the application domain as viewed by a typical user." SIMS [33]

Die obige Definition besagt, daß die Inhalte einer Anwendung bzw. eines Wissensgebiets beliebiger Art auf eine ausdrucksstarke und deklarative Weise beschrieben werden. Das Modell soll dabei nur die Aspekte beinhalten, die für den Nutzer relevant sind. Domänenmodelle sind Abstraktionen der Wirklichkeit, die z.B. die Kommunikation über einen Sachverhalt erleichtern, einen Sachverhalt überhaupt erst darstellen

und die Komplexität des Sachverhalts auf das für die jeweilige Domäne Wesentliche reduzieren [96].

Ein Domänenmodell enthält nur die Inhalte, die für die jeweilige Anwendung des Modells relevant sind. Der Inhalt wird üblicherweise auf ein entsprechendes Themengebiet oder Thema reduziert. Die Forderung nach **Vollständigkeit** wird also bereits von einem Domänenmodell erfüllt, wenn das Wissen soweit vollständig enthalten ist, wie es für die Intention des Modells notwendig ist (siehe hierzu auch [4]).

Ein Domänenmodell erfüllt außerdem die Anforderung nach **Flexibilität** und **Erweiterbarkeit**, wenn neue Konstrukte hinzugefügt und bestehende Konstrukte gelöscht bzw. modifiziert werden können. Zu berücksichtigen ist dabei, daß das Modell weiterhin konsistent (s.u.) bleiben muß.

Die **Offenheit** der Modelle wird in modernen Modellierungen, z.B. in [13] gefordert. Sie ermöglicht die Konstruktion von Modellen, die aufeinander aufbauen. Es können diverse Modelle einer Domäne aufgrund der Verwendung gleicher Konstrukte und eines generelleren Modells miteinander verglichen oder ausgetauscht werden, wie dies z.B. beim Ontolingua Server [51] der Fall ist.

In einem Domänenmodell muß die **Eindeutigkeit/Kohärenz** gegeben sein. Damit ist gemeint, daß jedes Konstrukt des Modells nur genau einmal existiert und von jedem anderem Konstrukt des Modells eindeutig unterschieden werden kann. In den Hyperbooks existiert also genau ein Konstrukt für jede Meta-Klasse, jede Klasse und jede Instanz. Somit wird in Hyperbooks auch jedes Hypermedia-Dokument und jede Beziehung durch je ein eigenes Konstrukt beschrieben. Auf diese Weise werden im KBS Hyperbook System Mehrdeutigkeiten vermieden und Konsistenzkonflikte aufgrund von Bezeichnergleichheit unterbunden.

Oben erwähnte Anforderungen eines Domänenmodells lassen sich mittels der Verwendung von Metadaten in Domänenmodellen z.B. in Form von Meta-Klassen (mit Einschränkungen) erfüllen. Da die Metadaten die Daten des Modells beschreiben [13], geben sie die Regeln vor, nach denen Konstrukte des Modells hinzugefügt, geändert und gelöscht werden können. Neben Erweiterbarkeit und Flexibilität wird so auch die Offenheit der Modelle erzwungen. Metadaten beschreiben mehrere Modelle, so daß die Modelle untereinander austauschbar bzw. vergleichbar werden und aufeinander aufbauen können.

Einschränkend auf die Verwendung von Metadaten wirkt, daß durch sie das Kriterium der Vollständigkeit eines Modells nicht erfüllt wird. Ein durch Metadaten beschriebenes Modell kann für die jeweilige Intention vollständig sein. Die Metadaten erzwingen jedoch nicht, daß das Modell für eine andere Intention trotz Verwendung derselben Metadaten vollständig ist.

Ebenso wirkt für den Einsatz von Metadaten einschränkend, daß sie die Kriterien der Eindeutigkeit und Kohärenz nicht unterstützen. Metadaten beschreiben diese Sachverhalte nicht. Sie werden jedoch durch die eingesetzten Systeme oder Modellierungssprachen und deren Semantiken und Grammatiken gewährleistet.

Im Rahmen dieser Arbeit wird zwischen einem Domänenmodell und einem **Repräsentationsmodell** unterschieden. Ein Repräsentationsmodell ist ein erweitertes Domänenmodell, das neben dem Themengebiet zusätzlich auch Metadaten und weitere Konstrukte, die für die Anwendung sinnvoll oder notwendig sind (z.B. Inhaltsverzeichnisse oder Konstrukte zur Visualisierung), modelliert. Diese Unterscheidung ist für die weitere Betrachtung der Modelle sinnvoll, denn es werden nicht nur Domänenmodelle für spezielle Themen, sondern auch generellere Modelle zur Verwendung in mehreren Domänenmodellen vorgestellt und eingesetzt.

4.2 Ansätze und Methoden der Modellierung

Die Domänen-Modellierung in dieser Arbeit basiert auf der Verwendung von objekt-orientierten Methoden zur Erstellung und Repräsentation der Modelle. Die Modellierung entspricht dabei der Abstraktion der jeweiligen Domäne unter Berücksichtigung der beabsichtigten Intentionen und Anwendungen der Modelle.

Die Erstellung der Modelle zur Verwendung in Hyperbooks läßt sich in zwei Hauptarten unterscheiden:

1. Systematische objekt-orientierte Modellierungstechniken, die für generische Hypermedia-Systeme wie Webserver und Datenbanken Front-Ends entwickelt wurden. Darunter fallen z.B. die Hypertext Design Methode (HDM) [43], die Relation Management Methodology (RMM) [63] und das Object-Oriented Hypermedia Design Model (OOHDM) [103] als bekannteste Beispiele. Diese Methoden strukturieren eine Domäne durch Einsatz von objekt-orientierten Konstrukten wie Klassen, Typen und deren Relationen. Durch Klassifizierung der WWW-Seiten in Typen und Klassen sowie durch Erzeugung der zugehörigen Relationen lassen sich durch Inferenz die Links berechnen.
2. Spezialisierte Techniken für Lehr-/Lernsysteme werden in der Community des adaptiven Hypertexts entwickelt, siehe z.B. auch [18, 17, 72, 80, 100]. Sie basieren auf der Verwendung von Metadaten in und für Vorlesungsseiten, die vom jeweiligen Benutzermodell verwendet werden. Die vom Benutzermodell generierten Daten werden dann z.B. für die Anpassung der existierenden Navigationsstruktur der Vorlesungsseiten an den Benutzer oder für deren Annotation verwendet. Eine Übersicht dieser Systeme findet sich in [14].

Eine Methode zur Modellierung ist der top-down Ansatz, in dem zunächst die allgemeinen Konzepte des Modells gebildet werden. Diese werden mittels Vererbungshierarchien und Bestandteil- (part-whole) Beziehungen immer mehr spezialisiert. Bestandteil-Beziehungen beschreiben den Sachverhalt, daß ein Konzept aus einem oder mehreren anderen Konzepten besteht [96].

Im Gegensatz zu den top-down Methoden ist auch ein bottom-up Ansatz möglich. Die zugrunde liegende Aussage ist, daß “all perceptible things of our world consist

of things undividable" (aus [113]). Somit sind also nur die nicht-teilbaren, atomaren Einheiten zu modellieren, während die komplexeren Zusammenhänge auf der Basis von Konstruktionsregeln zur Laufzeit generiert werden.

Unabhängig davon, ob der bottom-up oder der top-down Ansatz gewählt wird, gibt es zwei grundsätzliche Arten der Modellierung [92]. Eine Art beschreibt die implizite Modellierung, in der komplexe Konstrukte interne, unsichtbare Konstrukte beinhalten. Die internen Konstrukte werden durch objekt-orientierte Modellierungstechniken nicht erfaßt und verschwinden aus der Modellierung. Sie werden nicht durch Bestandteil-Beziehungen dargestellt. Im Gegensatz dazu werden in der expliziten Modellierung alle Konstrukte und Zusammenhänge explizit ausgedrückt. Komplexe Konstrukte werden explizit aus anderen Konstrukten zusammengesetzt, wodurch alle Konstrukte durch objekt-orientierte Modellierungstechniken erfaßt werden können. Die Bestandteil-Beziehungen werden zur Explizierung verwendet.

In dieser Arbeit wird die explizite Modellierung mit dem top-down Ansatz verwendet (siehe hierzu auch [92] und [96]), um die vereinfachte Wiederverwendbarkeit, die Handhabung und die Verwaltung der Modelle zu ermöglichen. Außerdem erscheinen Modelle mit expliziter Modellierung für den Betrachter ausdrucksstärker, wodurch der Vorgang der Modellierung vereinfacht wird. Die jeweiligen Bestandteil- und Vererbungshierarchien werden soweit detailliert, wie es für die Intentionen der Modelle sinnvoll erscheint.

Der Entwurf der Repräsentationsmodelle setzt sich neben der Entwicklung der Domänenmodelle aus der Modellierung von Meta-Modellen zusammen. Im Allgemeinen vereinfachen die Meta-Modelle den Modellierungsprozess, denn aufgrund ihrer Abstraktion können sie für eine Gruppe von Domänen bereits alle notwendigen Konstrukte beschreiben. Diese werden dann vom Domänenmodell auf die jeweilige Domäne bezogen. Die Meta-Modelle beschreiben also auf einer allgemeinen Ebene, wie die Konstrukte der Domänenmodelle zu bilden und einzusetzen sind, bzw. wie deren Zusammenhänge sind. Auch die Meta-Modelle werden nach top-down Methoden in expliziter Modellierung dargestellt.

Die Entwicklung der Meta-Modelle entspricht der Konzeptualisierung von Abstraktionen [87, 96]. Semantische Modellierungsansätze verwenden oftmals mindestens eine Abstraktionsebene, in der das Modell alle notwendigen Konzepte auf einem abstrakten Niveau beschreibt, während die Daten dann Ausprägungen (auch Objekte oder Instanzen genannt) der Konzepte sind. Beispiele für solche Konzepte sind die Klassen in objekt-orientierten Modellierungen und Sprachen wie Java [69] und die Schemata in Datenbanken. Weitere Abstraktionsebenen können eingeführt werden, so daß eine Modell-Architektur entsteht, in der die Konstrukte einer Abstraktionsebene die Instanzen der höheren sind. Solch ein Modellierungsansatz liegt z.B. dem Information Resource Dictionary System (IRDS) Standard zugrunde.

Verglichen mit Entity-Relationship (ER-) Modellen [101] fällt bei der Betrachtung der Hyperbook-Modelle und der damit verbundenen Abstraktionsebenen auf, daß die Klassen einer Ebene durchaus Instanzen einer darüber liegenden Ebene oder Super-

klassen mehrerer Klassen darunter liegender Ebenen sein können [87]. Um diese Verwirrungen zu vermeiden, werden die Konstrukte aller Ebenen in dieser Arbeit auch Konzepte genannt. Konstrukte in der Abstraktionsebene, in der nur Ausprägungen von Konstrukten enthalten sind, werden auch als Instanzen bezeichnet.

Die Modellierung wird in dieser Arbeit in Anlehnung an die Unified Modelling Language (UML) [39, 66] durchgeführt, wobei jedoch nicht ganz so strikte Kriterien zugrunde gelegt werden. Von UML wird insbesondere das Objektmodell übernommen, indem sowohl Klassendiagramm als auch Instanzendiagramm entwickelt werden. Das Klassendiagramm ist eine Erweiterung des ER-Modells und beschreibt die statischen Konzepte und deren Beziehungen zueinander. Das Instanzendiagramm beschreibt die sich eher dynamisch verhaltenden Instanzen und deren Beziehungen in Hinsicht auf das zugrunde liegende Klassendiagramm.

In dieser Arbeit wird der Modellentwurf vereinfacht, in dem die Instanzen als ebenso statische Einheiten wie die Konzepte in einem gemeinsamen Diagramm, dem Klassendiagramm, dargestellt werden. Auch werden die zum UML Entwurf gehörenden dynamischen und funktionellen Modelle vernachlässigt, da durch die Domänenmodelle keine dynamischen Inhalte des Hyperbooks ausgedrückt werden sollen. Hypermedia-Dokumente weisen keine eigenständigen dynamischen oder funktionellen Eigenschaften auf, weshalb die entsprechenden Modelle hier nicht relevant sind.

Die beschriebenen Entwurfsmethoden werden um axiomatisierte Definitionen erweitert, wie das z.B. in den Modellierungssprachen KIF [74, 49], Prolog [113], General Predicate Calculus [21] oder O-Telos [83, 67, 70] der Fall ist. Diese Definitionen werden benötigt, um bei der Erstellung der Modelle die Kriterien der Eindeutigkeit, Integrität und Konsistenz/Kohärenz der Daten zu prüfen und zu erhalten. Die in den Hyperbooks verwendeten Modelle werden also in einer formalen objekt-orientierten Modellierungssprache dargestellt. Diese Modellierungssprache weist eine formal deklarierte und beschriebene Semantik auf, in der die Formulierung von Axiomen zur Einschränkung von Modellen möglich ist [43]. In dieser Arbeit wird die logische, deduktive und objekt-orientierte Modellierungssprache O-Telos und insbesondere deren Axiomatisierung [70] als Basis aller Modelle verwendet. Die grafische Notation basiert auf UML [39, 66] und deren Verwendung im objekt-orientierten Software Entwurfstool TogetherJ [110].

4.3 Modelle und ihre Verwendung im System

Das in dieser Arbeit entwickelte KBS Hyperbook System verwendet ein Meta- oder ein Metadaten-Modell, das die Sprachkonstrukte des Systems definiert. Dieses Modell wird auch allgemeines Modell genannt. Wird das Domänenmodell durch Instanzierung des allgemeinen Modells erzeugt, ist das allgemeine Modell ein Meta-Modell. Wird hingegen die Spezialisierung zur Bildung des Domänenmodells verwendet, wird das resultierende Datenmodell Metadaten-Modell genannt. Das Repräsentationsmodell umfaßt aber immer sowohl das allgemeine Modell als auch das Domänenmodell.

Neben dem allgemeinen Modell werden im Folgenden kurz einige Repräsentationsmodelle vorgestellt, die den Entwicklungsprozess des allgemeinen Modells beschreiben. Zunächst wird auf das Metadaten-Modell der Vorlesung “Grundzüge der Informatik 1 (GdI 1)”, gehalten an der Universität Hannover eingegangen. Daran anschließend wird das Metadaten-Modell einer Terminologie-Datenbank für juristische und administrative Begriffe [41] vorgestellt, die mit Hilfe der European Academy Bozen entwickelt und als Hyperbook im Rahmen des KBS Hyperbook Systems realisiert wurde. Anschließend wird ein Metadaten-Modell kurz erläutert, das in Zusammenarbeit mit der Fachhochschule Flensburg in den Projekten Vineta [102] und Virtual Campus [115] für eine Lehrveranstaltung und deren Verteilung an mehrere Lehranstalten im baltischen Raum entwickelt wurde. Abschließend wird ein Meta-Modell beschrieben, das die Eigenheiten der RDF Schema Spezifikation auflöst und die Verwendung von RDF-Schemata und -Dokumenten im Rahmen des KBS Hyperbook Systems ermöglicht.

Die verschiedenen Modelle demonstrieren auch die flexiblen Einsatzgebiete des KBS Hyperbook Systems. Die Reihenfolge ihrer Darstellung in dieser Arbeit gibt den Entwicklungsprozess wieder, dessen Ergebnis das allgemeine Modell ist. Im Rahmen dieses Entwicklungsprozesses werden die Besonderheiten der einzelnen Modellierungen kurz beschrieben und im Vergleich zum allgemeinen Modell bewertet.

4.3.1 Allgemeines Modell

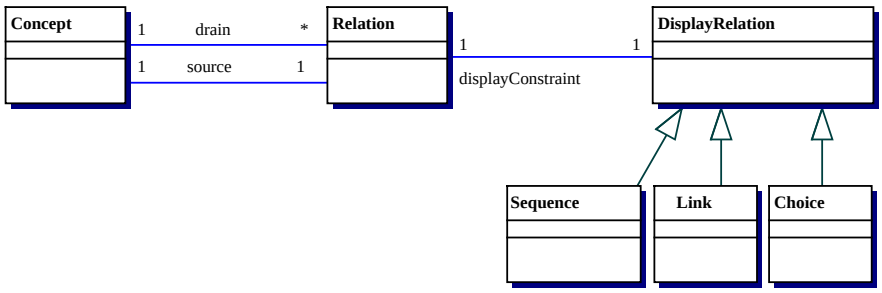

Abbildung 4.1: Allgemeines Modell des KBS Hyperbook Systems

Zunächst soll das in Abbildung 4.1 gezeigte allgemeine Modell des KBS-Hyperbooks erläutert werden. Es besteht in Anlehnung an die semantische Modellierung im Wesentlichen aus den zwei Konzepten *Concept* und *Relation*, die über zwei Beziehungen miteinander verbunden sind. Die Modelle werden in UML-Notation abgebildet, wobei Klassen die Konzepte darstellen.

Das Konzept `Concept` modelliert ein Modellierungskonstrukt, das als wichtigste Funktion im Sinne von Hyperbooks ein Hypermedia-Dokument repräsentieren kann. Gleichzeitig kann es verwendet werden, um als Konzept in Instanz- und Vererbungshierarchien eingesetzt zu werden. Ein `Concept` kann also gleichzeitig modellierende Funktionen wahrnehmen und ein inhaltliches Konzept des zu modellierenden Hyper-

books darstellen. In der aktuellen Version des KBS Hyperbook Systems besitzt das `Concept` die folgenden Attribute: Ein Attribut *name* vom Typ String zur Aufnahme des anzuzeigenden Namens. Ein Attribut *abstract* vom Typ String, das eine kurze Zusammenfassung des Konzeptinhalts aufnimmt und ein Attribut *id* vom Typ String zur Darstellung der URI des Hypermedia-Dokuments. Diese Attribute sind aus Gründen der Vereinfachung nicht in der Abbildung 4.1 enthalten.

Das Konzept `Relation` stellt die Beziehungen in einem Hyperbook zwischen Konzepten dar. Aus diesem Grund steht es mittels der zwei Attribute `source` und `drain` mit dem `Concept` in Beziehung. Die Bezeichnungen `source` und `drain` sollen den Unterschied der Attribute in ihren Wertigkeiten darstellen. Das `source` Attribut ist eine (1:1)-Beziehung, um ein `Concept` (im Folgenden source-Konzept genannt) an die Relation zu binden. Hingegen ist das Attribut `drain` eine (1:n)-Beziehung, um weitere Konzepte (im Folgenden drain-Konzepte genannt) mittels der `Relation` mit dem source-Konzept zu verbinden. Auf diese Weise lassen sich sowohl (1:1)- als auch (1:n)-Relationen darstellen. Die Wertigkeiten der Attribute `source` und `drain` werden durch ein Constraint sichergestellt (siehe auch die Erläuterungen zu Constraints im Anhang A.5).

Auch (m:n)-Relationen lassen sich durch diese Darstellung modellieren. Da sie nicht explizit darstellbar sind, werden sie daher im Sinne der expliziten Modellierung beispielsweise zu (1:n)-Relationen aufgelöst. In den folgenden Modellierungen sind verschiedene Ansätze enthalten, wie (m:n)-Relationen explizit dargestellt werden können. Es hat sich jedoch im Rahmen der Hyperbook-Entwicklungen gezeigt, daß (m:n)-Relationen ohne Nachteile für das jeweilige Hyperbook vernachlässigt bzw. zu (1:n)-Relationen aufgelöst werden können.

In einigen der folgenden Beispiel-Modelle werden die (1:1)-und (1:n)-Relationen explizit modelliert. Sie stellen also explizit die möglichen Beziehungstypen zwischen Konzepten dar. Die Modelle werden dadurch sehr umfangreich und unübersichtlich. Es zeigt sich bei der Anwendung dieser Modelle in der Domänenmodellierung, daß diese zusätzlichen Konstrukte keine sinnvollen Auswirkungen auf die Domänenmodelle oder die darauf basierenden Hyperbooks haben. Aus diesem Grund gibt es im allgemeinen Modell des KBS Hyperbook Systems keine expliziten Konstrukte zur Darstellung von (1:1)-, (1:n)- oder (m:n)-Relationen. Die explizite Modellierung der Instanz- und Vererbungsbeziehungen wird durch die entsprechenden Konstrukte der Modellierungssprache bereits vorgenommen (In und IsA-Beziehungen in O-Telos). Sollen diese Beziehungen angezeigt werden, sind sie mit Instanzen der `DisplayRelation` zu assoziieren (z.B. durch Constraints).

Die in der Abbildung 4.1 enthaltene Vererbungshierarchie `DisplayRelation` mitsamt den Subklassen `Sequence`, `Link` und `Choice` dient dem KBS Hyperbook System zur Anzeige der Relationen, z.B. als Links. Diesen Konzepten sind die folgenden Attribute zugeordnet, die für die Anzeige benötigt werden: `nameSourceDrain` vom Typ `String` stellt den Namen der Beziehung vom source-Konzept zu den drain-Konzepten dar. `nameDrainSource` vom Typ `String` stellt den Namen der Beziehung von einem der drain-Konzepte zum source-Konzept dar. `visualisation` vom

Typ `boolean` bestimmt, ob eine Relation angezeigt wird und `priority` vom Typ `Integer` bestimmt die Position der Relation in der Liste der angezeigten Relationen.

Die Beziehungen der Konzepte zu anderen Konzepten sind Instanzen der Klasse `Relation` und ihrer Subklassen. Gleichzeitig sind diese Objekte Instanzen der Klasse `DisplayRelation` und ihrer Subklassen. Sie prägen die oben genannten Attribute dieser Klasse aus, deren Werte dann in der Konstruktion der Anzeige der Beziehungen in der WWW-Seite verwendet werden. Prägt ein `DisplayRelation` Objekt eines der Attribute der `DisplayRelation` Klasse nicht aus, verfolgt das KBS Hyperbook System die Vererbungshierarchie in Richtung der Klasse und deren Superklassen, bis es ein `DisplayRelation` Objekt findet, das einen Wert für dieses Attribut enthält. Dieser Wert wird dann für den nicht vorhandenen Wert des Ausgangsobjekts verwendet.

Im Folgenden werden nun eine Reihe von Metadaten- und Meta-Modellen vorgestellt, die alle im KBS-Hyperbook-System zur Modellierung der jeweiligen Domäne eingesetzt werden. Das allgemeine Modell aus Abbildung 4.1 stellt die grundsätzlichen Gemeinsamkeiten aller angeführten Metadaten- und Meta-Modelle dar.

4.3.2 Das Metadaten-Modell des GdI 1 Hyperbooks

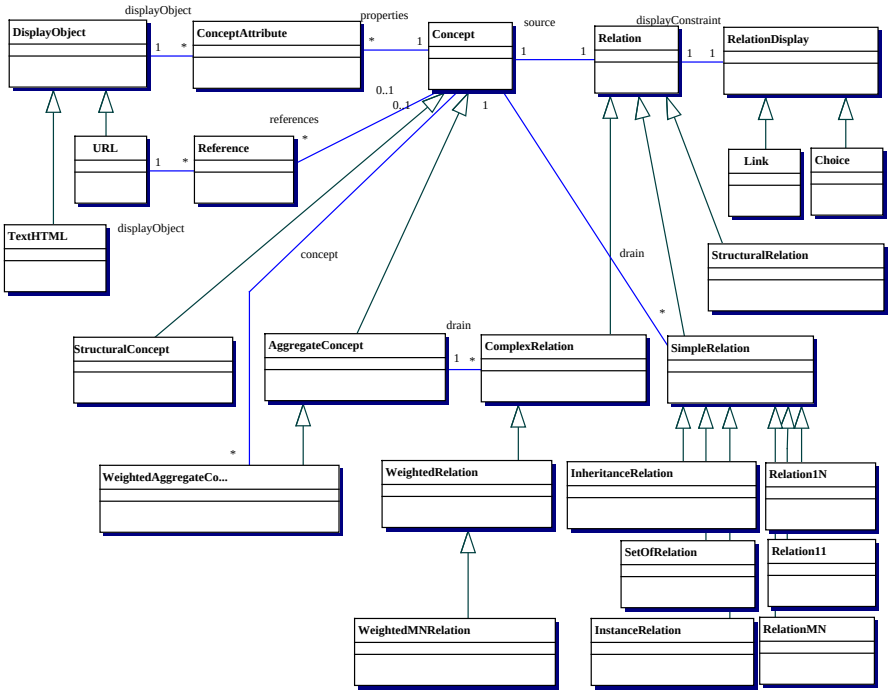

Abbildung 4.2: Metadaten-Modell des GdI1-Hyperbooks

In Abbildung 4.2 wird eines der möglichen Metadaten-Modelle des Hyperbooks zur

Vorlesung “Grundzüge der Informatik 1” (GdI 1) dargestellt. Im Vergleich zum allgemeinen Modell spezialisiert dieses Modell die möglichen Relationstypen, um in der Domänenmodellierung unterschiedliche Relationstypen explizit verwenden zu können. Die Relationstypen modellieren die Relationen mit den unterschiedlichen Wertigkeiten (`Relation11`, `Relation1N`, `RelationMN`) und die zugrundeliegenden Modellierungskonstrukte Vererbungsrelation (`InheritanceRelation`) und Instanzrelation (`InstanceRelation`). Auf diese Weise sollen Vererbungs- und Instanzrelationen sowie deren Hierarchien explizit modelliert werden, unabhängig von der jeweiligen Modellierungssprache und deren Konstrukten.

Die beiden Konzepte `InheritanceRelation` und `InstanceRelation` müssen mit den entsprechenden Konstrukten der Modellierungssprache kombiniert werden, um die Konsistenzprüfung des Modells durch geeignete, für die Sprache existierende Tools, zu ermöglichen. In der Praxis hat sich dieser Ansatz nicht bewährt, denn er bedeutet die gleichzeitige Darstellung desselben Zusammenhangs auf zwei unterschiedliche Weisen (durch ein Modellierungskonstrukt und durch ein Sprachelement). Beispielsweise existiert in der Modellierungssprache O-Telos der Vererbungsmechanismus in der Form der Relation *IsA* und dem damit assoziierten Axiom bereits. Es macht keinen Sinn, ein zweites Konstrukt mit derselben Funktion zu entwickeln. Ebenso existiert in O-Telos auch der Instanzierungsmechanismus in der Form der Relation *In* mit dem zugehörigen Axiom, womit auch die Instanzrelation als explizites Konstrukt im Modell unnötig wird. Diese Konstrukte müssen im Metadaten-Modell vorhanden sein, wenn sie nicht durch die Modellierungssprache oder deren Konstrukte direkt ausgedrückt werden können. Zur Anzeige müssen die Instanz- und Vererbungsbeziehungen mit den zugehörigen `DisplayRelation`-Instanzen assoziiert werden.

Um in der Modellierung mittels dieses Metadaten-Modells zwischen strukturellen und inhaltlichen Konzepten und Relationen unterscheiden zu können, sind die entsprechenden Konstrukte (`StructuralConcept` und `StructuralRelation`) explizit im Modell dargestellt. In der Praxis zeigt sich jedoch auch hier, daß die Domänenmodelle nur aus Konzepten bestehen, mit denen auch Hypermedia-Dokumente assoziiert sind, so daß kein Bedarf für die strukturellen Konzepte besteht.

Des weiteren sind in diesem Metadaten-Modell eine Reihe von Konstrukten enthalten, die für zusätzliche Funktionalitäten im Rahmen des KBS-Hyperbook-Systems notwendig sind. Zu diesen Konstrukten gehören `WeightedAggregateConcept`, `WeightedRelation` und `WeightedMNRelation`, die mittels Integer-Attributierung eine Gewichtung der jeweiligen Relation darstellen können. Diese Konstrukte werden im Rahmen der Benutzeradaption, eines besonderen Moduls des KBS-Hyperbook-Systems (siehe [53]) verwendet.

Ein weiterer Unterschied zum allgemeinen Modell ist in Abbildung 4.2 die explizite Modellierung der mit dem Konzept assoziierten Hypermedia-Dokumente, dargestellt durch `DisplayObject` und dessen Subtypen (`TextHTML` und `URL`). Sie werden durch das Konzept `ConceptAttribute` mit dem `Concept` assoziiert. Auf diese Weise wird explizit dargestellt, daß ein `Concept` Attribute hat, deren Inhalt die Hypermedia-Dokumente sein können. Ein `Concept` kann in diesem Metadaten-

Modell dementsprechend mehrere Hypermedia-Dokumente repräsentieren. Gleichzeitig wird durch das Konstrukt `Reference` die direkte Assoziation eines `Concept` mit einem Hypermedia-Dokument, das durch seine URL repräsentiert wird, dargestellt. Dieser Art der Modellierung von `Concept` und damit verbundener Hypermedia-Dokumente ist die Idee, daß ein Hypermedia-Dokument aus mehreren Teildokumenten besteht.

In der Praxis hat sich dieser Modellierungsansatz nicht durchgesetzt, weil die Hypermedia-Dokumente nicht in Teildokumente zerlegt werden können. Durch die Offenheit des KBS Hyperbook Systems wird das Einbinden von komplexen fremden Hypermedia-Dokumenten unterstützt, die sich nur als unteilbares Ganzes im Zugriff des Systems befinden (vergleiche auch das `WWW-Resource` Konzept in [13]). Entsprechend wird auch das Konstrukt `AggregateConcept` zur expliziten Modellierung zusammengesetzter Konzepte in der Praxis nicht benötigt.

4.3.3 Metadaten-Modell des Terminologie-Hyperbooks

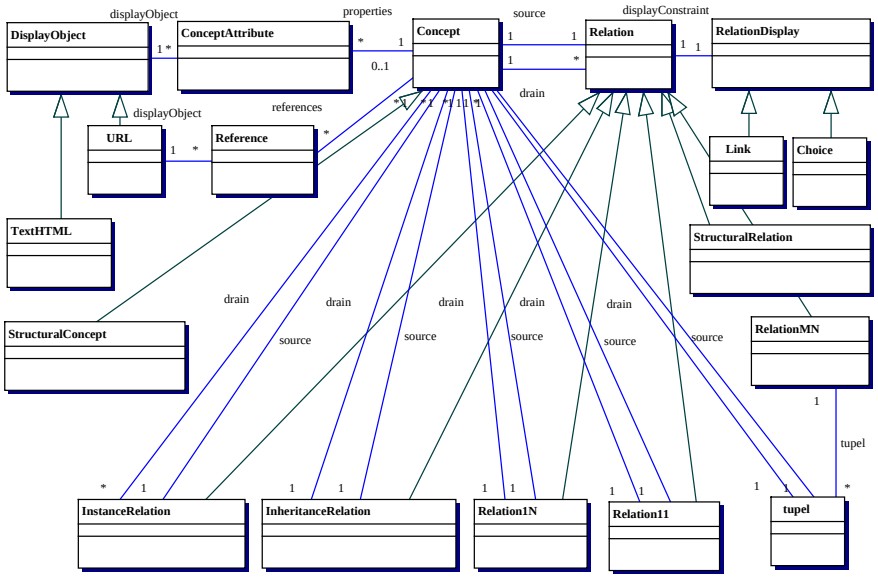

Abbildung 4.3: Metadaten-Modell des Terminologie-Hyperbooks

In Abbildung 4.3 wird das Metadaten-Modell des Terminologie-Hyperbooks für juristische und administrative Begriffe dargestellt. Entwickelt in Zusammenarbeit mit der European Academy Bozen ist dieses Modell die Basis für die Modellierung terminologischen Wissens als Ontologie. Die Erläuterungen der Grundlagen zu Ontologie und Terminologie finden sich ausführlich in der Literatur, z.B. in [41, 98] zur Terminologie und in [52] zum Thema Ontologie. Ein genereller Überblick über unterschiedliche Arten und Definitionen von Ontologien findet sich insbesondere in [45].

Zum Verständnis des Metadaten-Modells soll an dieser Stelle die verwendete Definiti-

on für Ontologien nach [50] kurz erläutert werden. Sie lautet:

> "An **ontology** is an explicit specification of a conceptualization (aus [50])."

In diesem Sinne besteht eine Konzeptualisierung aus den Konzepten einer Domäne und konzeptuellen Beziehungen derselben, die als existierend und relevant angenommen werden. Eine Ontologie fügt der Konzeptualisierung ein Vokabular hinzu, das den Bezug auf die Konzepte der Konzeptualisierung ermöglicht. Außerdem spezifiziert eine Ontologie die Konzeptualisierung explizit in einer formalen Sprache, z.B. First Order Logik. Somit ist dann die intentionale Idee der Konzeptualisierung explizit formalisiert und definiert.

Eine mögliche und in dieser Arbeit verwendete Definition von Terminologie nach [98] lautet:

> "**Terminology** is a theory concerned with those aspects of the nature and the functions of language which permit the efficient representation and transmission of items of knowledge in all their complexity of concepts and conceptual relationships (aus [98])."

Somit können sowohl Ontologien als auch Terminologien Ausprägungen einer gemeinsamen Konzeptualisierung sein. Ontologien erreichen dies durch die formale Definition der Semantik des Vokabulars, wohingegen Terminologien natürlichsprachliche Definitionen verwenden.

Das zugehörige, in Abbildung 4.3 dargestellte Metadaten-Modell ermöglicht aufgrund des Einsatzes der Konstrukte `Concept` und `Relation` ein Domänenmodell, dessen Inhalt eine Terminologie spezifiziert, welche wiederum durch eine Ontologie ausgedrückt wird. Dieses Metadaten-Modell ist im Vergleich zum vorhergehenden Metadaten-Modell (Abbildung 4.2) des GdI-Hyperbooks eine reduzierte Variante, in der die für zusätzliche Funktionalität verwendeten Konstrukte fehlen.

In diesem Modell wird zu Forschungszwecken außerdem die (m:n)-Relation explizit modelliert, indem sie aus beliebig vielen, aber mindestens einem `tupel` besteht. Ein Tupel ist hier eine direkte Beziehung zwischen zwei Konzepten. Somit wird hier eine (m:n)-Beziehung aus ihren einzelnen Tupeln zusammengesetzt. In der Praxis bedeutet diese Modellierung zusätzliche Modellierungsarbeit mit der Folge der höheren Fehleranfälligkeit und Unübersichtlichkeit. Es hat sich herausgestellt, daß die durch die Modellierung der (m:n)-Relationen im Domänenmodell enthaltenen Informationen für den Inhalt des Hyperbooks nicht von Bedeutung sind. Daher wird diese Darstellungsweise der (m:n)-Relationen im allgemeinen Modell vernachlässigt.

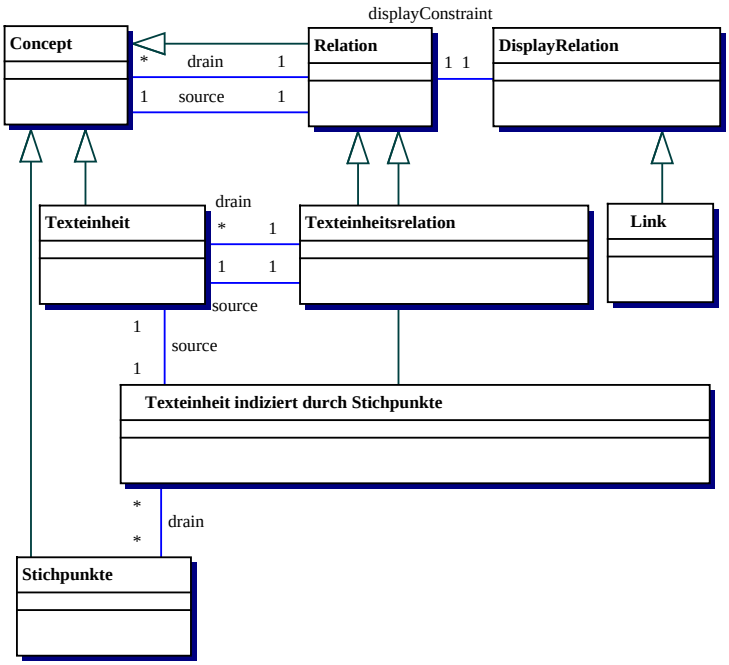

Abbildung 4.4: Meta-Modell des Vineta-Hyperbooks

4.3.4 Metadaten-Modell des Vineta-Hyperbooks

Das Metadaten-Modell des Vineta-Hyperbooks, entwickelt in Zusammenarbeit mit der Fachhochschule Flensburg, besteht aus sehr wenigen Konstrukten. Wie aus Abbildung 4.4 hervorgeht, werden auf der Basis von `Concept` und `Relation` im Metadaten-Modell die für die Modellierung des Domänenmodells notwendigen Konzepte als Subklassen von `Concept` und `Relation` dargestellt. Das Konzept `Texteinheit` stellt die möglichen Hypermedia-Dokumente dar, die über das Konzept `Texteinheitsrelation` miteinander in Beziehung treten können. Die Unterscheidung von `Concept` und `Texteinheit` wird aufgrund des Konzepts `Stichpunkt` notwendig, das die Indizierung von Texteinheiten durch Stichpunkte mittels des Konzepts `Texteinheit_indiziert_durch_Stichpunkte` ermöglicht. Die Indizierung wird für die zusätzliche Funktionalität der Benutzeradaption, die in diesem Hyperbook eingesetzt werden soll, notwendig.

Außerdem ist in diesem Metadaten-Modell das Konzept `Relation` als Subklasse des Konzepts `Concept` dargestellt. Diese Beziehung bedeutet, daß Relationen auch Konzepte sind und deren Attribute ausprägen können. Als direkte Folge ist es möglich, sowohl `Concept` als auch `Relation` zur Modellierung zu verwenden und ihnen Hypermedia-Dokumente zuzuordnen. Es sind noch nicht alle Implikationen dieser Modellierung ausgewertet, jedoch wird sie bereits auch im RDF-Standard verwendet. Im Rahmen des KBS Hyperbook Systems wird sie verwendet, um neben den Konzepten auch die Relationen des Metadaten-Modells auf den zugeordneten WWW-Seiten

beschreiben zu können.

4.3.5 Meta-Modell zur Verwendung von RDF-Schemata

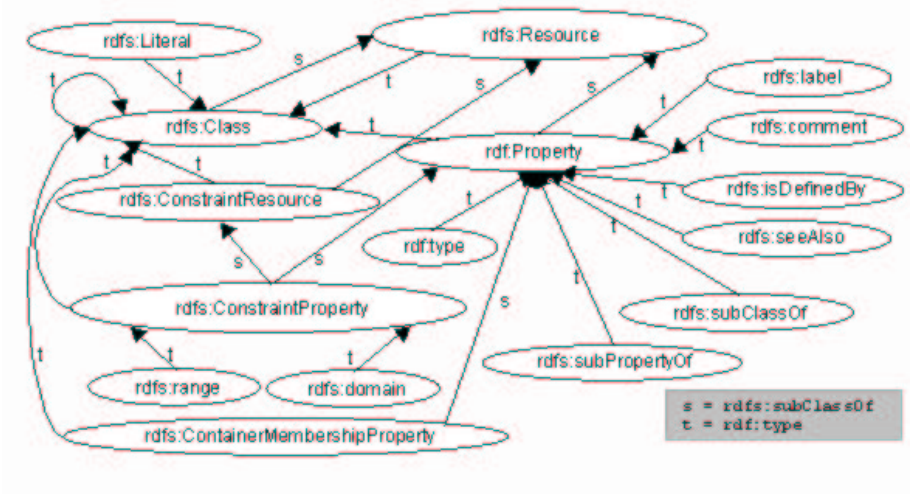

Abbildung 4.5: Klassenhierarchie des RDF-Schemas (aus [13])

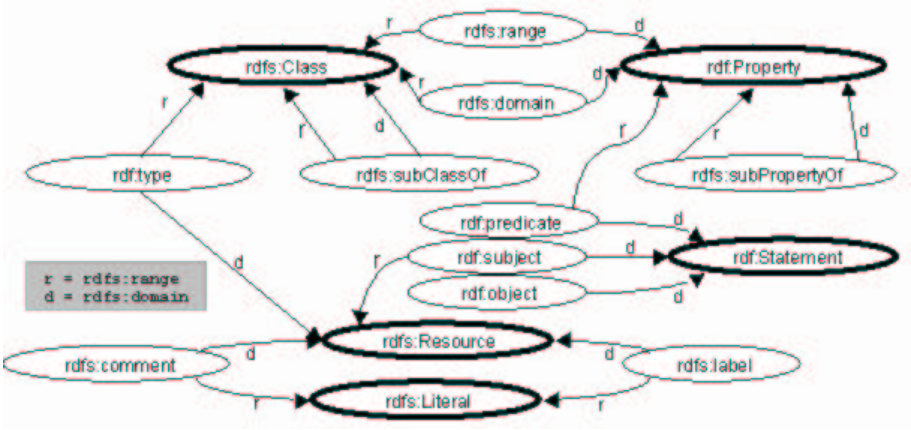

Abbildung 4.6: Constraints des RDF-Schemas (aus [13])

Bisher wurden in dieser Arbeit Meta- und Metadaten-Modelle des KBS Hyperbook Systems beschrieben, auf deren Basis Domänenmodelle und Hypermedia-Dokumente erstellt werden. Ebenso ist es notwendig, bereits existierende Meta-, Metadaten- und Domänenmodelle einschließlich ihrer Hypermedia-Dokumente in Hyperbooks zu integrieren. Insbesondere im WWW finden Meta- und Metadaten-Modelle auf der Basis der RDF Schema Spezifikation und damit beschriebener Hypermedia-Dokumente in zunehmendem Maße Verwendung.

Bevor jedoch darauf eingegangen wird, wie die in RDF formulierten Meta- und

Metadaten-Modelle mittels des KBS Hyperbook Systems in ein Hyperbook integriert werden können (vergl. hierzu auch [20]), sollen kurz die Eigenheiten der RDF Schema Spezifikation erläutert werden.

RDF-Schemata werden verwendet, um die Struktur von Metadaten zu definieren. Die Metadaten beschreiben wiederum WWW-Ressourcen, die durch eine URI eindeutig identifiziert werden. Die RDF Schema Spezifikation besteht aus einigen Basisklassen, kann jedoch erweitert werden, um möglichst viele Domänen zu beschreiben.

Der im RDF verwendete Prefix *rdfs:* besagt, daß das Konstrukt in der RDF Schema Spezifikation definiert ist. Entsprechend besagt der Prefix *rdf:*, daß die Definition des Konstrukts Teil des RDF Datenmodells [116] ist. Die beiden Abbildungen 4.5 und 4.6 sind der RDF Schema Spezifikation entnommen. Sie stellen grafisch die notwendigen RDF Konstrukte, Zusammenhänge und Einschränkungen dar.

Die Klassen werden in der RDF Schema Spezifikation hierarchisch angeordnet, wie auch der Abbildung 4.5 entnommen werden kann. Die root-Klasse der Klassenhierarchie wird `rdfs:Resource` genannt, von der das Konstrukt `rdfs:Class` zur Deklaration von Klassen eine Subklasse ist.

Sogenannte Properties werden durch die Klasse `rdf:Property` dargestellt und entsprechen sowohl Attributen als auch Relationen in anderen Modellierungssprachen. Properties werden auch als Ressourcen angesehen. Sie sind deshalb Subklassen von `rdfs:Resource`. Die Verwendung und Gültigkeit von Properties kann auf bestimmte Klassen eingeschränkt werden. Neben den zur Vereinfachung im RDF existierenden Properties wie `rdfs:seeAlso` und `rdfs:comment` tragen die folgenden Properties besondere Bedeutungen: `rdfs:subClassOf`, `rdf:type`, `rdfs:range` und `rdfs:domain`. Diese vier Properties werden verwendet, um die anderen Klassen des RDF-Schemas zu definieren und sie sind gleichzeitig Teil des Schemas. Sie spielen somit duale Rollen als primitive Konstrukte der RDF Schema Spezifikation und als spezifische Instanzen von `rdf:property`. Aufgrund dieser dualen Rolle wird in der RDF Schema Spezifikation der Selbstbezug von Konstrukten aufeinander ermöglicht, der jedoch von Systemen für die Auswertung der Modelle nicht aufgelöst werden kann. Die duale Rolle der vier Properties erscheint unter anderem als Folge der Bemühung, die RDF-Schemata selbst als explizite Metadaten über RDF-Schemata darzustellen.

Abbildung 4.5 stellt in einer Art “Knoten und Kanten”-Darstellung die Klassenhierarchie der RDF-Schema Spezifikation dar. Dort werden deutlich die Verwendungen von `rdfs:subClassOf` und `rdf:type` aufgezeigt. `rdfs:subClassOf` wird eingesetzt, wenn eine Klasse eine Subklasse einer anderen ist. `rdf:type` deklariert eine Ressource als Instanz einer Klasse. Hier werden wieder die doppelten Rollen dieser beiden Properties deutlich, die sowohl spezifische Properties (siehe die Knoten `rdfs:subClassOf` und `rdf:type` in der Abbildung 4.5) als auch primitive Konstrukte der Modellierung (siehe die mit “s” für `rdfs:subClassOf` und “t” für `rdf:type` bezeichneten Pfeile in der Abbildung 4.5) sind.

Abbildung 4.6 beschreibt die Einschränkungen (engl. Constraints) der möglichen Re-

lationen der Konstrukte aus Abbildung 4.5. Zur Deklaration der Einschränkungen werden die beiden Properties `rdfs:range` und `rdfs:domain` verwendet (dargestellt durch die mit “r” und “d” bezeichneten Pfeile in Abbildung 4.6). Gleichzeitig werden dieses Properties jedoch auch durch die Anwendung auf sich selbst (siehe die Knoten `rdfs:range` und `rdfs:domain` in Abbildung 4.6) eingeschränkt, womit wiederum deren bereits angesprochene duale Rollen deutlich werden.

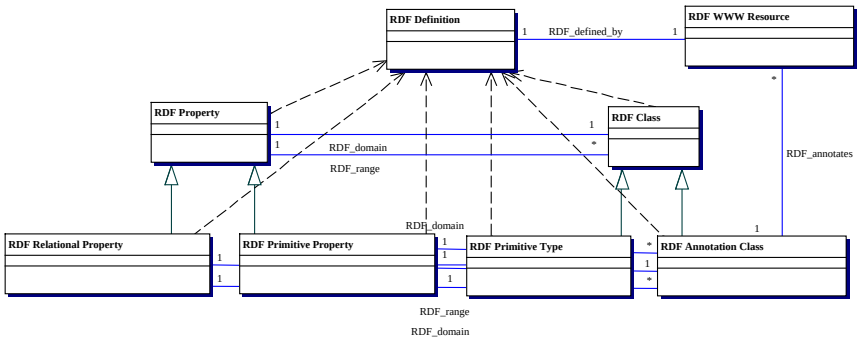

Abbildung 4.7: Basis-RDF-Konstrukte als Instanzen von RDF_Definition

Im Folgenden wird nun das für das KBS Hyperbook System verwendete Meta-Modell vorgestellt, das die Integration von vereinfachten RDF-Schemata in Hyperbooks erlaubt. Die RDF-Schemata sind mittels der Konstrukte der RDF Schema Spezifikation beschriebenen. In diesem Meta-Modell wird eindeutig zwischen primitiven und konventionellen Properties unterschieden. Somit differenziert das vorgestellte Meta-Modell eindeutig zwischen Meta-Modell Konstrukten und deren Instanzierungen, wodurch der im RDF-Schema mögliche Selbstbezug vermieden wird.

Wie in Abbildung 4.7 dargestellt sind die RDF-Schema Konstrukte (`RDF_Class`, `RDF_Property`, etc.) Instanzen der Klasse `RDF_Definition`. Da die Klasse `RDF_Definition` zusammen mit der Klasse `RDF_WWW_Resource` dem Konstrukt `rdfs:Resource` aus Abbildung 4.5 entspricht, wird mittels dieses Meta-Modells der RDF Sichtweise entsprochen, daß RDF-Schemata selbst wieder Daten sind, die durch WWW-Ressourcen definiert werden. Es wird deutlich, daß hier zur Darstellung der Instanzbeziehung das “in”-Konstrukt als Instanzbeziehung verwendet wird. Es ist ein primitives Konstrukt der verwendeten Modellierungssprache O-Telos.

Die in Abbildung 4.7 als Instanzen beschriebenen Konstrukte `RDF_Class`, `RDF_Property`, etc. bilden die Basiskonstrukte des eigentlichen Meta-Modells der RDF-Schema Spezifikation. Abbildung 4.8 beschreibt die Konstrukte und deren Verwendung zur Annotation von WWW-Ressourcen. Einige RDF-Konstrukte wie `rdfs:ConstraintProperty` oder `rdfs:ConstraintResource` sind nicht enthalten, da sie zu dieser Diskussion inhaltlich nichts beitragen.

Die Konstrukte `RDF_Class` und `RDF_Property` entsprechen den Konstrukten `rdfs:Class` und `rdf:Property`. Sie sind mittels der Konstrukte `RDF_range` und `RDF_domain` miteinander in Beziehung gesetzt. `RDF_range`

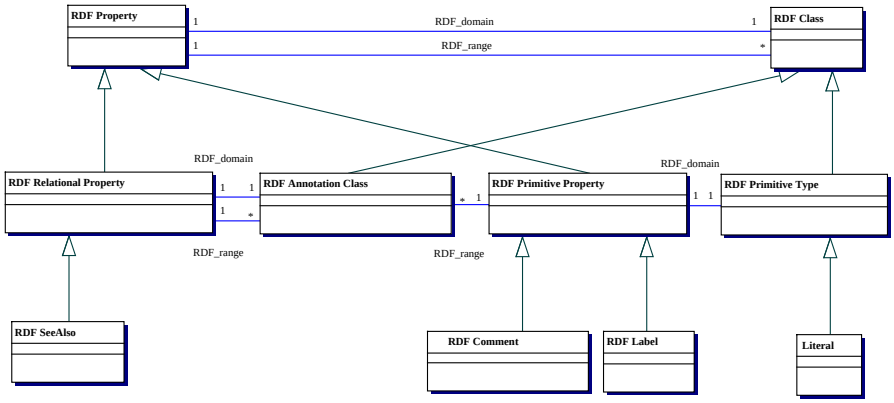

Abbildung 4.8: Basis-RDF-Konstrukte des Meta-Modells zur RDF-Kompatibilität

und `RDF_domain` korrespondieren zu den Konstrukten `rdfs:range` und `rdfs:domain`, stellen aber spezifische Beziehungen dar. Im Gegensatz zur originären RDF Schema Spezifikation sind sie keine Instanzen von `RDF_Property`.

Ähnlich sind die Konstrukte `rdfs:subClassOf` und `rdf:subPropertyOf` durch die Konstrukte `RDF_subClass` und `RDF_subPropertyOf` ersetzt. Sie ermöglichen die Definition von Subklassen- und Subproperty-Beziehungen mit der zugehörigen Vererbungssemantik. Diese können zur Vereinfachung in der praktischen Modellierung durch das entsprechende Konstrukt der Modellierungssprache ersetzt werden (z.B. das “IsA”-Konstrukt von O-Telos).

Die Konstrukte `RDF_annotates` und `RDF_definedBy` ersetzen das Konstrukt `rdf:type`, wenn die Beziehung zwischen RDF-Metadaten und WWW-Ressourcen dargestellt wird. Handelt es sich um eine tatsächliche Instanzierung mit der zugehörigen Semantik, wird das entsprechende Konstrukt der verwendeten Modellierungssprache substituiert (das “In”-Konstrukt von O-Telos).

Das Konstrukt `RDF_Class` hat zwei Subklassen, `RDF_Annotation_Class` und `RDF_Primitive_Type`. Die Instanzen der `RDF_Annotation_Class` und ihrer Subklassen entsprechen den RDF Pendants – z.B. entspricht `RDF_SeeAlso` `rdf:seeAlso` – und werden zur Annotation von WWW-Ressourcen verwendet. Eine solche WWW-Ressource, im Modell durch eine Instanz der Klasse `RDF_WWW_Resource` dargestellt, kann durch mehrere Instanzen der `RDF_Annotation_Class` Klasse in mehreren RDF-Schemata beschrieben werden. Die Instanzen von `RDF_Primitive_Type` und die zugehörigen Subklassen wie `String` oder `Literal` sind spezifische Werte für RDF-Properties.

Das so dargestellte Meta-Modell weist einige weitere Unterschiede im Vergleich zur ursprünglichen RDF Schema Spezifikation auf. So kann in der RDF Schema Spezifikation `rdfs:Class` eine Instanz ihrer selbst sein. Aus der Spezifikation geht nicht hervor, unter welchen Umständen diese Möglichkeit verwendet werden soll, weshalb diese Option im Meta-Modell vernachlässigt wird.

Die Beziehungen `rdfs:subClassOf` zwischen `rdfs:Resource` und `rdfs:Class` bzw. `rdfs:Property` werden durch eine Substitution von `rdfs:Resource` durch spezifischere Klassen wie `rdfs:Class` ersetzt. Dies ist möglich, da diese Beziehungen nur zur Spezifikation von `rdfs:domain` und `rdfs:range` in der RDF Schema Spezifikation verwendet werden.

Außerdem wird im Unterschied zur RDF Schema Spezifikation die `rdf:type` Beziehung zwischen `rdfs:class` und `rdfs:Resource` durch die Beziehungen `RDF_annotates` und `RDF_definedBy` ausgedrückt.

Das vorgestelle Meta-Modell erreicht prinzipiell dieselben Ziele wie die originale RDF Schema Spezifikation, stellt aber die verwendeten Konstrukte übersichtlicher dar und ist aufgrund der Auflösung der Selbstbezüge einfacher zu formalisieren. Die Möglichkeit, RDF-Schemata formalisiert auszudrücken, wird im KBS Hyperbook System zur Erstellung der Hyperbooks verwendet. Im Folgenden wird dargestellt, wie das vorgestellte RDF-Meta-Modell zur Verwendung im Hyperbook transformiert werden muß. Die Transformationen liegen in der verwendeten Modellierungssprache O-Telos und der Kompatibilität zu älteren, bereits entwickelten Hyperbook-Modellen begründet und ändern nichts an der ursprünglichen Aussage des RDF-Meta-Modells. Werden diese Änderungen am RDF-Meta-Modell vorgenommen, können alle entsprechenden RDF-Schemata und zugehörigen Hypermedia-Dokumente in die Hyperbooks des KBS Hyperbook Systems aufgenommen werden.

Die folgenden Änderungen erweisen sich als notwendig: Das Konstrukt `RDF_Class` korrespondiert zum Konstrukt `Concept` im allgemeinen Meta-Modell. Durch das Attribut `id` von `Concept` wird die Beziehung `RDF_annotates` dargestellt, wobei eine `RDF_WWW_Resource` durch ihre als String dargestellte URI im Domänenmodell enthalten ist. Das Konstrukt `RDF_Property` entspricht dem Konstrukt `Relation` und dessen Subklassen. Die O-Telos Vererbungsrelation “IsA” entspricht `RDF_subClassOf` und `RDF_subPropertyOf` und die O-Telos Instanzrelation “In” ersetzt `rdf:type`. Die O-Telos-Sprachelemente “IsA” und “In” werden verwendet, da mit ihnen die Semantik der Vererbung (IsA) und der Instanzierung (In) in zur Verarbeitung von O-Telos geeigneten Softwareprodukten enthalten ist. Werden diese Änderungen am RDF-Meta-Modell vorgenommen, können die zugehörigen RDF-Schemata und Hypermedia-Dokumente in die Hyperbooks des KBS Hyperbook Systems aufgenommen werden.

4.4 Das Hyperbook zur Vorlesung GdI 1

Nachdem in den vorangegangenen Kapiteln unterschiedliche Ansätze und Modelle vorgestellt wurden, soll hier nun beispielhaft anhand der Vorlesung “Grundzüge der Informatik 1” (GdI 1), gehalten von Prof. Nejdl an der Universität Hannover [85] die Entwicklung eines Hyperbooks für eine Vorlesung beschrieben werden. Die GdI 1 Vorlesung wird im Rahmen einer Kooperations-Vorlesung per Videokonferenz auch an die Freie Universität Bozen in Italien übertragen. Das Metadaten-Modell der Vorlesung

GdI 1 muß zur Verdeutlichung dieses Sachverhalts also nicht nur dem Anwendungsfall der Vorlesung in Hannover, sondern auch dem der Vorlesung in Bozen genügen, wobei beide Vorlesungen zum Teil die gleichen Lehrinhalte verwenden.

Das Hyperbook soll mehrere Funktionen erfüllen. Es dient als Nachschlagewerk und als Skriptum für die Vorlesungen. Gleichzeitig informiert es umfassend über die Sachgebiete und Inhalte der Vorlesung. Anhand des Hyperbooks werden die Studenten bei der Erarbeitung ihrer Projekte und bei ihren Vorbereitungen auf Prüfungen unterstützt. Im Hyperbook werden die von den Studenten während der Vorlesung durchgeführten Projekte dokumentiert und integriert.

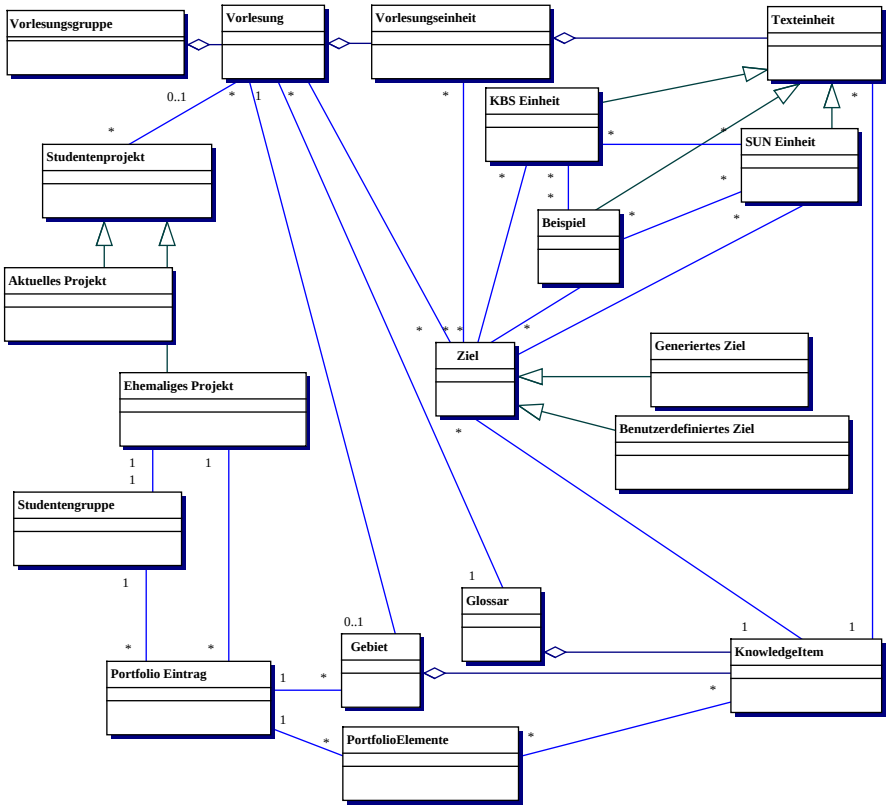

Abbildung 4.9: Ausschnitt aus dem Metadaten-Modell zur Vorlesung GdI 1

Neben der Darstellung mehrerer Vorlesungen in einem Modell soll das System auch die Benutzeradaption mit den zugehörigen Funktionalitäten wie Lernsequenzgenerierung ermöglichen. Die Darstellung der Benutzeradaption ist nicht Bestandteil dieser Arbeit. An dieser Stelle werden nur kurz die für das Metadaten-Modell relevanten Konzepte erläutert. Der geneigte Leser wird zu diesem Thema speziell auf die Dissertation von N. Henze [53] verwiesen, in der dieses Thema im Kontext der Hyperbooks umfassend erläutert wird.

Abbildung 4.9 stellt einen Teil des Meta-Modells dar, das aus den zur Modellierung der Veranstaltungen, der Inhalte und der Benutzeradaption notwendigen Konzepten

besteht.

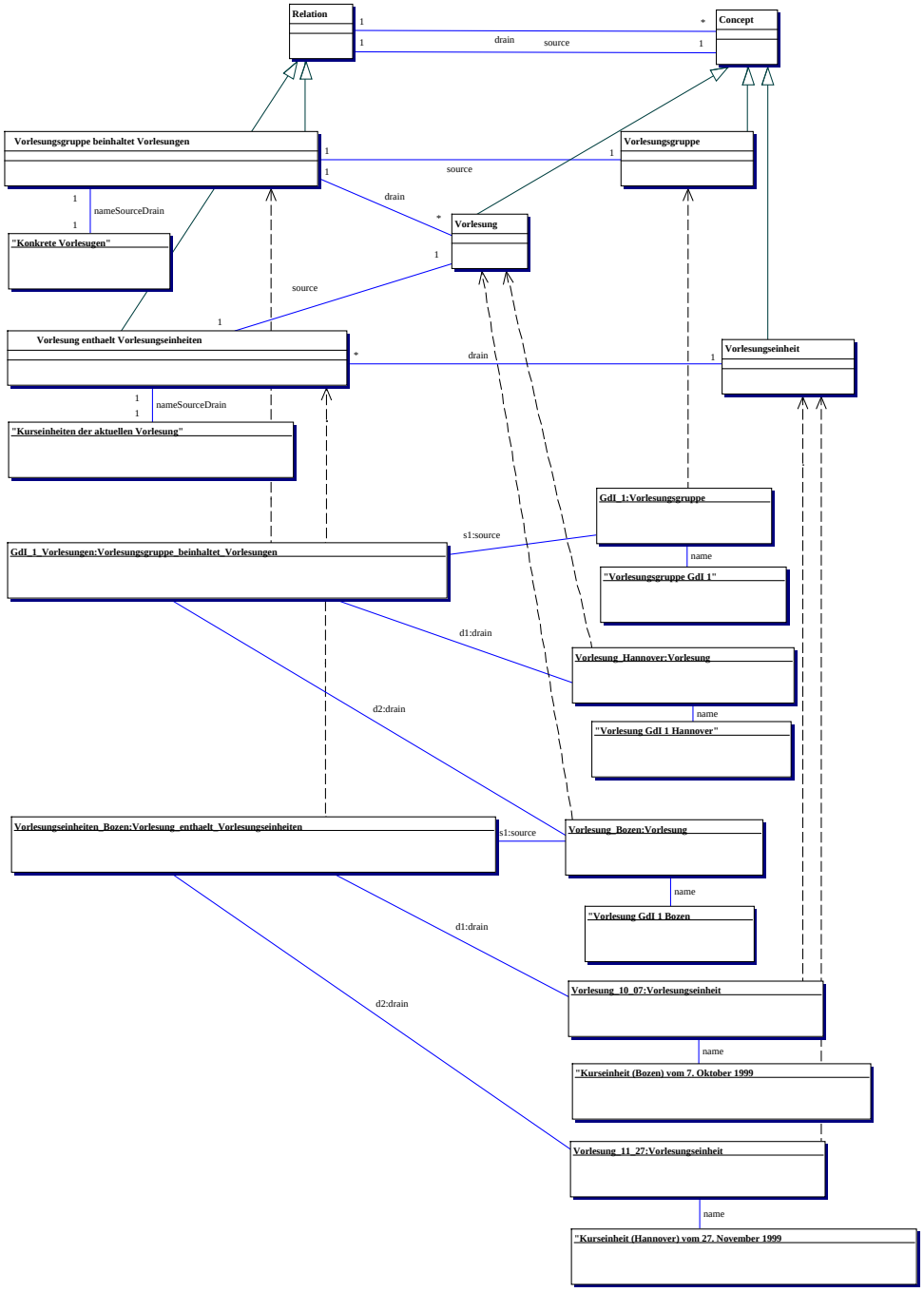

Abbildung 4.10: Ausschnitt aus dem Metadaten-Modell mit Metadaten der Vorlesung GdI 1

Die Konzepte `Vorlesungsgruppe`, `Vorlesung` und `Vorlesungseinheit` modellieren die Vorlesungen. Eine Vorlesungsgruppe besteht aus Vorlesungen

(z.B. besteht die Vorlesungsgruppe GdI 1 aus den Vorlesungen in Hannover und Bozen), die wiederum aus Vorlesungseinheiten bestehen. Die Vorlesungseinheiten bestehen aus Texteinheiten (`Texteinheit`)[1], die die verschiedenen Typen von Dokumenten darstellen. Im Fall der GdI 1 Vorlesung sind dies Texteinheiten aus dem Sun Tutorial [19] für Java (`SUN_Einheit`), das institutseigene Vorlesungsskript (`KBS_Einheit`) und Beispiele. Diese Einheiten können zueinander in Beziehung stehen, z.B. wenn von einer `KBS_Einheit` zum Zweck der weiteren Erläuterung auf eine entsprechende `SUN_Einheit` und ein oder mehrere Beispiele verwiesen werden soll. Die Beziehungen der Texteinheiten untereinander werden durch die Adaptionskomponente des Systems automatisch zur Laufzeit erstellt.

Neben den Vorlesungseinheiten setzt sich eine Vorlesung aus Studentenprojekten, Lernzielen der Benutzer, Glossaren und Sachgebieten zusammen. Die Studentenprojekte, die aktuelle (`Aktuelles_Projekt`) oder ehemalige (`Ehemaliges_Projekt`) Projekte sein können, beschreiben die Aufgaben und die durch die Studenten erarbeiteten Lösungen. Aktuelle Projekte sind in der aktuellen Vorlesung von Studierenden bearbeitete Projekte, während ehemalige Projekte von Studierenden in früheren Semestern erstellt und abgeschlossen wurden. Die ehemaligen, also abgeschlossenen Projekte werden dann in Portfolios [30], bestehend aus Einträgen (`Portfolio_Eintrag`), Portfolio-Elementen (`Portfolio_Element`) und den Homepages der Gruppen (`Studentengruppe`) dokumentiert. Die möglichen Lernziele der Benutzer stellt das Konzept `Ziel` dar. Sie werden aufgrund von Benutzereingaben automatisch zur Laufzeit durch die Adaptionskomponente des Systems generiert. Sie können vom Benutzer oder vom System auf der Basis des Wissensstandes des Benutzers definiert werden. Ebenso automatisiert können die Sequenzen von Texteinheiten, die einen Benutzer bei der Erreichung seiner Ziele unterstützen, automatisch von dieser Komponente erzeugt werden. Die Adaption basiert unter anderem auf der Indizierung der Texteinheiten, Sachgebiete und Portfolio-Elemente durch Schlagworte (`Knowledge_Item`). Die Indizierung beschreibt ein Wissensmodell der Inhalte des Hyperbooks und entspricht einem Schlagwortindex. Dieser wird im `Glossar` abgebildet. Eine inhaltliche Gruppierung der Schlagworte, und damit verbunden der Texteinheiten, wird durch die Sachgebiete (`Sachgebiet`) dargestellt.

Das so beschriebene Metadaten-Modell basiert auf den Konstrukten des allgemeinen Modells, das in Kapitel 4.3 beschrieben wird. Die Konzepte (Klassen in UML-Notation) aus Abbildung 4.9 sind Subklassen des Konzepts `Concept` aus Abbildung 4.1. Die Beziehungen (Relationen in UML-Notation) in Abbildung 4.9 stellen entsprechend Subklassen des Konzepts `Relation` aus Abbildung 4.1 dar.

Das vollständige GdI 1 Repräsentationsmodell besteht aus mehreren Hundert Konzepten und Relationen. Abbildung 4.10 zeigt einen kleinen Ausschnitt des Modells. In diesem Ausschnitt wird ein Teil der Konzepte gezeigt, die für die Erzeugung der in Abbildung 4.11 gezeigten WWW-Seite mit dem Hypermedia-Dokument notwendig sind. Die Konzepte `Vorlesungsgruppe`, `Vorlesung` und `Vorlesungseinheit` stellen als Subklassen von `Concept` die mit Hypermedia-

[1] Der in der Klammer stehende Name ist der Bezeichner des entsprechenden Konzepts.

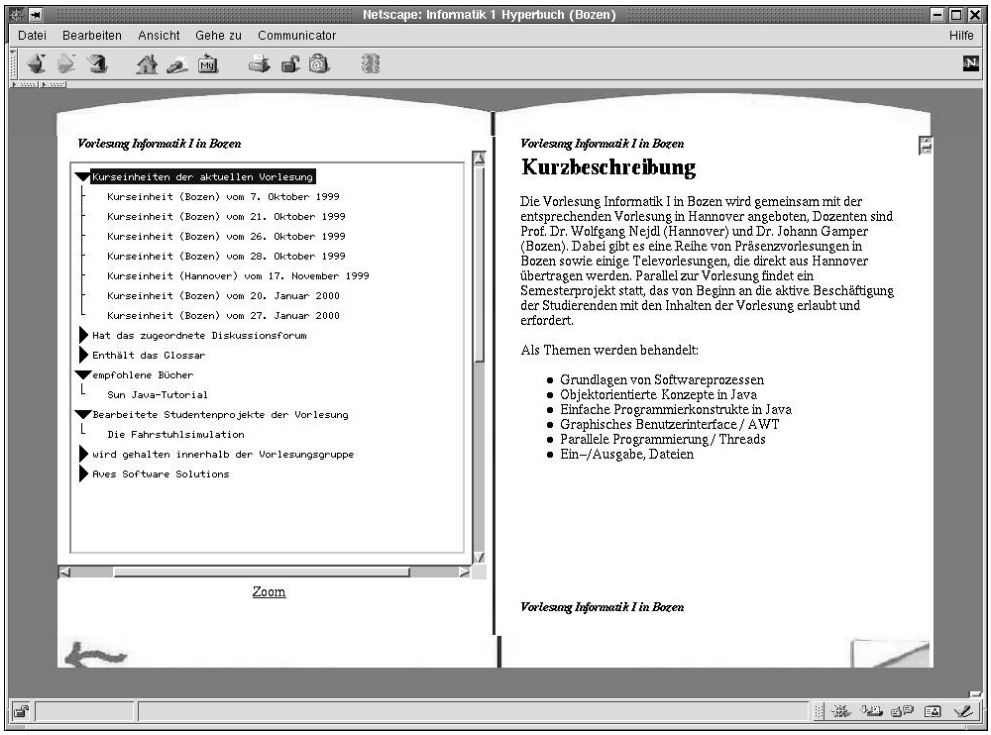

Abbildung 4.11: Ein Hypermedia-Dokument des GdI 1 Hyperbooks

Dokumenten assoziierten Konzepte dar. Die als Subklassen von `Relation` modellierten Konzepte `Vorlesungsgruppe_beinhaltet_Vorlesungen` und `Vorlesung_enthaelt_Vorlesungen` repräsentieren die Struktur, die die Subklassen von `Concept` miteinander in Beziehungen setzt.

In diesem Kontext betrachtet besagt das in Abbildung 4.10 wiedergegebene Metadatenmodell, daß eine Vorlesungsgruppe mittels der Relation `Vorlesungsgruppe_beinhaltet_Vorlesungen` aus einer oder mehreren Vorlesungen besteht. Die Vorlesungen setzen sich wiederum mittels der Relation `Vorlesung_enthaelt_Vorlesungen` aus einer oder mehreren Vorlesungseinheiten zusammen. In diesem Beispiel gibt es als Vorlesungsgruppe die `GdI_1_Gruppe` mit Namen “Vorlesungsgruppe GdI 1”, die via `GdI_1_Vorlesungen` Relation mit der Vorlesung `Vorlesung_Hannover` mit Namen “Vorlesung GdI 1 Hannover” und der Vorlesung `Vorlesung_Bozen` mit Namen “Vorlesung GdI 1 Bozen” verbunden ist. Weiterhin wird gezeigt, daß die `Vorlesung_Bozen` unter anderem aus den Vorlesungseinheiten `Vorlesung_10_07` namens “Kurseinheit (Bozen) vom 7. Oktober 1999” und `Vorlesung_11_27` namens “Kurseinheit (Hannover) vom 27. November 1999” besteht.

In Abbildung 4.11 wird gezeigt, wie die zum Konzept `Vorlesung_Bozen` gehörende, vom GdI 1 Hyperbook generierte WWW-Seite aussehen kann. In der aktuellen Implementierung des KBS Hyperbook Systems setzt sich die WWW-Seite aus mindestens zwei Teilen, rechter und linker Frame genannt, zusammen. Im rechten Frame

wird das mit dem Konzept assoziierte Hypermedia-Dokument gezeigt, das in diesem Fall eine Beschreibung der in Bozen gehaltenen Vorlesung beinhaltet. Im linken Frame werden die vom System generierten Links dargestellt. Hervorgehoben ist eine Relation vom Typ `Vorlesung_enthaelt_Vorlesungen` namens “Vorlesungseinheiten der aktuellen Vorlesung”, die die Kurseinheiten der Vorlesung aus dem Jahr 1999 enthält.

4.5 Das O-Telos-RDF Schema

Im vorhergehenden Kapitel 4.3.5 wurde erläutert, daß RDF-Schemata auf der Modellierungsebene in das KBS Hyperbook System integriert werden können. In diesem Kapitel nun soll eine Variante von RDF(S)[2] vorgestellt werden, die auf der Basis der Modellierungssprache O-Telos im Rahmen des KBS Hyperbook Systems entwickelt wird. Diese Variante, O-Telos-RDF genannt, übernimmt alle Vorzüge und Ansätze des RDF(S) und erweitert diese um Metamodellierungs- und Reifikationsmöglichkeiten.

Von der Modellierungssprache O-Telos verwendet O-Telos-RDF alle Axiome, wie sie in [70] beschreiben werden. Gegebenenfalls werden die Axiome an die Erfordernisse eines zum RDF(S) vergleichbaren Resource Description Framework angepaßt. Die Axiome werden auf der Tupelebene deklariert, weshalb in diesem Kapitel zunächst ein Vergleich der Tupeldarstellungen von RDF(S) und O-Telos aufgestellt und erläutert wird. Im Anschluß daran werden dann die Axiome zur Deklaration von O-Telos-RDF beschrieben.

O-Telos-RDF bietet im Gegensatz zu RDF eine größere Unterstützung der Mechanismen der Metamodellierung, beispielsweise die Fähigkeit der Reifikation und die Fähigkeit zur Bildung der Abstraktionsebenen. Außerdem wird O-Telos-RDF durch Axiome formalisiert, um die Semantik eindeutig zu beschreiben. RDF(S) wird im Gegensatz dazu in syntaktischer Form beschrieben. Jedoch haben Conen und Klapsing in [25] einen Entwurf der Formalisierung der Semantik von RDF vorgelegt, der RDF(S) direkt formalisiert. Ein Vergleich dieser RDF-Formalisierung mit O-Telos-RDF ist in [86] wiedergegeben.

4.5.1 Vergleich von O-Telos und RDF auf der Tupelebene

Die Modellierungssprache O-Telos setzt ebenso wie RDF ein semantisches Netzwerk zur Beschreibung der Informationen ein. Diese Art der Modellierung beruht auf einer langen Tradition, die mit Tarskis Verwendung binärer Beziehungen [109] begann und sich über Abrials semantisches Datenmodell [1] fortsetzte.

Das semantische Netz besteht aus Knoten und Kanten. In O-Telos repräsentieren Knoten alle möglichen Arten von Individuals, z.B. Klassen, Metaklassen, Literale und In-

[2] RDF(S) bezeichnet hier sowohl RDF gemäß der RDF Model and Syntax Specification [116] als auch das RDF-Schema gemäß der RDF Schema Specification [13].

stanzen. Kanten hingegen stellen die Beziehungen der Knoten in Form von Attributen zueinander dar. Knoten und Kanten sind mit eindeutigen Bezeichnern (ID) versehen. Mittels der IDs ist es Kanten möglich, zwei Konstrukte miteinander zu verbinden, wobei es sich dabei sowohl um zwei Knoten, zwei Kanten oder einen Knoten und eine Kante handeln kann.

In RDF(S) stellen Knoten die (WWW) Ressourcen und Kanten die Beziehungen der Knoten, Properties genannt, dar. Sowohl Knoten als auch Kanten werden über ihre Namen identifiziert, wobei die Namen durch URIs dargestellt werden. Die URIs werden unter Verwendung des aus XML bekannten Namensraum-Konzepts [12] gebildet, um ihre Eindeutigkeit zu gewährleisten.

O-Telos verwendet zur Darstellung von Knoten und Kanten eine uniforme Proposition. Eine Proposition ist ein Quadrupel p(pid,x,l,y), aufgebaut aus pid als eindeutiger ID, x und y als zwei pids anderer oder derselben Proposition(en) und l als der Name der Proposition. Das Quadrupel besagt, daß es unter der ID pid eine Beziehung l gibt, die die beiden Propositionen x und y miteinander verbindet. In ähnlicher Weise stellt RDF(S) die Knoten und Kanten durch sogenannte Statements der Form s(x,p,y) als Tripel dar. x und y sind Knotennamen und p ist ein Kantenname. Das Tripel besagt, daß die Ressource x über das Property p mit der Ressource y in Beziehung steht. Im Unterschied zu O-Telos verzichtet RDF(S) also auf die Verwendung einer eindeutigen ID.

O-Telos-Objekte werden durch eine Framesyntax beschrieben, die die Attribute eines Objekts in einem Frame gruppiert (siehe hierzu auch Kapitel 6.2.1). Die Framesyntax wird dann in die Propositionsdarstellung abgebildet, ohne den klassenzentrierten Ansatz zur Darstellung der Informationen aufzugeben. RDF(S) verwendet zur Formulierung der Modelle eine XML-Serialisierung. Die Properties werden als eigenständige Konstrukte unabhängig von den Ressourcen abgebildet, so daß RDF im Gegensatz zu O-Telos einen propertyzentrierten Ansatz in den Modellen verwendet.

Aus dem Vergleich von RDF und O-Telos ergibt sich also, daß die O-Telos Propositionen ähnlich den RDF Statements sind. Ebenso sind die Attribute ähnlich den Properties und die Individuals ähnlich den Ressourcen. Im Unterschied zu RDF haben die Propositionen von O-Telos eine eindeutige ID, über die sie explizit referenziert werden können. Im RDF haben nur die einer URI zugeordneten Ressourcen eben diese URI als eindeute ID. In O-Telos wird ein Objekt durch die in seiner Deklaration enthaltenen Attribute definiert, während es in RDF möglich ist, die Properties einer Ressource unabhängig von der Ressource zu deklarieren.

4.5.2 O-Telos-RDF Formalisierung

Aus dem vorhergehenden Kapitel 4.5.1 geht hervor, daß O-Telos und RDF beide semantische Netzwerke zur Darstellung von Informationen verwenden. Ebenso sind die Daten-Repräsentationen auf Tupelebene sehr ähnlich, wobei der wichtigste Unterschied das Fehlen der IDs der Statements im RDF ist.

O-Telos-RDF kombiniert O-Telos und RDF, indem es die O-Telos Axiome auf die syntaktische Deklaration von RDF(S) anwendet. Unter Zuhilfenahme der von Conen und Klapsing in [25] vorgeschlagenen RDF-Axiomatisierung werden die O-Telos Axiome an RDF(S) angepaßt, so daß ein dem RDF(S) vergleichbares Resource Description Framework Schema entsteht. Dieses alternative RDF-Schema namens O-Telos-RDF ist kompatibel zum originären RDF, ermöglicht aber zudem auch Meta-Modellierung und Reifikation.

O-Telos-RDF verwendet zur Kennzeichnung den mit "otelos:" deklarierten Namensraum, dessen vorläufiges Schema unter der URL "http://www.kbs.uni-hannover.de/2001/01/rdf-otelos-schema" zu finden und im Anhang B.1 aufgeführt ist.

Im Folgenden werden nun die Axiome beschrieben, die das O-Telos-RDF Schema deklarieren. Die Axiome sind entsprechend den Konstrukten geordnet, die sie deklarieren.

4.5.2.1 Das Statement

In O-Telos-RDF werden alle Konstrukte (Knoten und Kanten) in Form von O-Telos-RDF Statements dargestellt. Die Statements entsprechen den RDF-Statements, werden aber um eine eindeutige ID erweitert. Sie haben daher die allgemeine Darstellung s(sid,x,p,y), in der sid die ID, p ein Label und x und y sids anderer Statements sind. Alle sids sind URIs bzw. einer URI so ähnlich wie möglich. Ihr genauer Aufbau wird in den einzelnen folgenden Abschnitten jeweils erläutert.

Alternativ werden die Elemente x als Subjekt, p als Prädikat und y als Objekt des Statements s(sid,x,p,y) bezeichnet, welches über die eindeutige Statement ID sid referenziert wird. Es folgt daraus das erste Axiom:

Axiom 1 *Statement Identifikatoren sind eindeutig:*

$$\forall sid, x1, x2, p1, p2, y1, y2\, s(sid, x1, p1, y1) \wedge s(sid, x2, p2, y2) \\ \Rightarrow (x1 = x2) \wedge (y1 = y2) \wedge (p1 = p2)$$

Das Axiom 1 korrespondiert zum O-Telos Axiom 1. Ebenso wie in O-Telos werden alle Statements in O-Telos-RDF implizit reifiziert, so daß die Existenz aller beteiligten Statements notwendig ist. Im Gegensatz zu RDF sind alle Statements mit einer eindeutigen ID versehen. O-Telos-RDF erlaubt, daß die Labels der Statements URIs oder atomare Werte annehmen, wohingegen RDF für die Labels RFC 2396-Konformität (URI) fordert.

4.5.2.2 Das Individual Statement

Ein Knoten eines O-Telos-RDF Graphen wird Individual genannt. Er kann als Subjekt und/oder Objekt in anderen Statements verwendet werden und wird dabei über seine ID referenziert. Sein Statement hat die Form s(ns:p,ns:p,p,ns:p) bzw. s(o,o,o,o), wobei

o und ns:p URIs sind, ns: den aktuellen namespace und p einen atomaren Wert darstellt. Entsprechend werden die beiden Sets Statement und Individual durch die folgenden Axiome definiert:

Axiom 2 *Das Set aller Statements wird durch das Individual s(otelos:statement,otelos:statement,statement,otelos:statement) mit der Abkürzung otelos:statement dargestellt.*

Axiom 3 *Alle Statements sind elements des Sets otelos:statement.*

$$\forall sid, x, p, y\ s(sid, x, p, y) \Leftrightarrow type(sid, otelos : statement)$$

Axiom 4 *Das Set aller Individuals wird durch s(otelos:individual,otelos:individual,individual,otelos:individual) mit der Abkürzung otelos:individual dargestellt.*

Axiom 5 *Wenn das Label eines Individuals ein atomarer Wert ist, ist es zusammen mit dem aktuellen Namensraum eindeutig. In allen anderen Fällen ist das Label eine URI. Die Statement ID setzt sich aus dem Label und ggf. dem Namensraum zusammen und ist daher in jedem Fall eindeutig. Sie lautet ns:p oder o.*

Das Axiom 2 korrespondiert zu den O-Telos Axiomen 18 und 24, die die Existenz von Statements deklarieren. Die O-Telos Axiome 19 und 25 deklarieren ebenso wie das Axiom 4 die Existenz der Individuals, wobei diese hier ebenso wie im RDF als Ressourcen angesehen werden. Das Axiom 5 beschreibt, daß nicht nur wie im RDF "benannte", sondern alle Ressourcen durch eine URI eindeutig identifiziert werden (korrespondiert zum O-Telos Axiom 2).

4.5.2.3 Literale und andere primitive Datentypen

Ein Literal l wird durch das Statement s(l,l,l,l) dargestellt. Das Set aller Literale wird durch das Individual otelos:literal repräsentiert. Auf diese Weise können beliebige weitere primitive Datentypen eingeführt werden.

Die Statement IDs sind zwar nicht URI-konform, erlauben aber die direkte O-Telos- und RDF(S)-konforme Verwendung der Literale in anderen Statements. In RDF(S) werden alle Objekte, die keine Ressource sind, als Literale deklariert. Im Gegensatz dazu gibt es in O-Telos und O-Telos-RDF vordefinierte Literale, beispielsweise die Klassen String und Integer.

4.5.2.4 Das Class-Individual

In O-Telos-RDF werden sowohl Klassen als auch Objekte durch Individuals ausgedrückt. Die Individuals können instanziert werden, so daß keine Notwendigkeit für ein

explizites otelos:class Statement besteht. Da die Instanzierungen von Individuals und Properties explizit modelliert werden, können beliebig tiefe Instanzierungshierarchien in Metamodellierungskontexten erstellt werden. Dies ist in RDF(S) nicht möglich, weil instanzierte Properties dort kein Label und keine eigene ID haben und somit nicht eindeutig referenzierbar sind.

Zur Erhaltung der Kompatibilität zu RDF(S) kann ein Indivdual otelos:class eingeführt werden. Dieses Set enthält dann alle Individuals, die instanziert werden (entsprechend O-Telos). Eine andere Möglichkeit der Kompatibilitätserhaltung ist die Definition eines Synonyms otelos:class für das Individual otelos:individual. Beide Möglichkeiten sind im O-Telos-RDF-Schema nicht enthalten, denn rdfs:Class wird hier direkt auf otelos:individual abgebildet.

4.5.2.5 Das Property Statement

Die Kanten der semantischen Modelle, Properties genannt, werden durch Statements der Form s(sid,x,p,y) bzw. s(ns:p,otelos:statement,p,y) dargestellt, wobei sid unterschiedlich zu x und y sowie ns:p unterschiedlich zu otelos:statement und y sein muß. Außerdem darf das Label nicht die drei reservierten Werte type, subClassOf und subPropertyOf annehmen:

Axiom 6 *Definition der Properties:*

$$\forall sid, x, p, y\, s(sid, x, p, y) \wedge (sid \neq x) \wedge (sid \neq y) \wedge (p \neq subClassOf) \\ \wedge (p \neq subPropertyOf) \wedge (p \neq type) \Leftrightarrow type(sid, otelos:property)$$

Axiom 7 *Das Set aller Properties wird durch das Statement s(otelos:property,otelos:property,property,otelos:property) dargestellt, dessen Abkürzung otelos:property ist.*

Statements der Form s(sid,x,p,y) werden *Object Scoped Properties* genannt, da das Prädikat p im Rahmen des aktuellen Namensraumes nur zusammen mit dem zugehörigen Subjekt x eindeutig ist. Die sid des Statements lautet dann ns:x_p und ist ebenso im aktuellen Namensraum eindeutig. Diese Definition entspricht dem klassenzentrierten Ansatz der Attributsdeklaration in Frame-basierten Sprachen wie O-Telos.

Axiom 8 *Der Property-Name von Object Scoped Properties ist im Zusammenhang mit dem Subjekt eindeutig.*

$$\forall sid1, sid2, x, p, y1, y2\, s(sid1, x, p, y1) \wedge P(sid2, x, p, y2) \\ \Rightarrow (sid1 = sid2) \vee (p = type) \vee (p = subClassOf) \vee (p = subPropertyOf)$$

Wenn das Label einer Property s(ns:p,otelos:statement,p,y) ein atomarer Wert ist, der im aktuellen Namensraum eindeutig ist, wird das Property *Globally Scoped Property* genannt. Diese Definition entspricht der propertyzentrierten Property-Definition, wie sie im RDF(S) verwendet wird.

Axiom 9 *Für Globally Scoped Properties wird Axiom 8 erweitert, so daß die Properties auch ohne Zusammenhang zum Subjekt eindeutig sind.*

$$\forall sid1, x1, x2, p, y1, y2\, s(ns:p, x1, p, y1) \wedge P(sid1, x2, p, y2)$$
$$\Rightarrow ((sid1 = ns:p) \wedge (x1 = x2) \wedge (y1 = y2)) \vee (p = otelos:type)$$
$$\vee (p = otelos:subClassOf) \vee (p = otelos:subPropertyOf)$$

Die propertyzentrierte Deklaration hat den Nachteil, daß die Deklaration eines Properties in verschiedenen Namensräumen unterschiedlich sein kann. Das kann dann zu einem einwertigen Property führen, dem aber mehr als ein Wert zugewiesen wird, wodurch Dateninkonsistenzen entstehen können.

Die Axiome 7, 8 und 9 entsprechen den O-Telos Axiomen 3, 22 und 26. Die Properties werden wie im RDF(S) als Ressourcen mit gültigen URIs angesehen. Eine Ausnahme hiervon bilden die Properties mit den drei reservierten Labeln type, subClassOf und subPropertyOf, sowie Literale jeglicher Art. Sie werden in den folgenden Unterkapiteln behandelt.

Im Unterschied zu O-Telos und RDF erlaubt O-Telos-RDF die Deklaration und Verwendung von Object Scoped und Globally Scoped Properties. Da jedoch die Globally Scoped Properties ein Sonderfall der Object Scoped Properties sind, werden die Properties objekt-zentriert ähnlich O-Telos in der Objekt-Deklaration definiert.

4.5.2.6 Das type Statement

Statements der Form s(sid,x,type,y) verwenden das reservierte Label type um die Klassenzugehörigkeit bzw. die Instanzierung auszudrücken. Das Statement besagt also, daß x Instanz von y ist, wobei x und y sowohl Individuals als auch Properties sein können.

Axiom 10 *Type Statements können durch das Hilfskonstrukt type(x,y) dargestellt werden:*

$$\forall sid, x, y\, s(sid, x, type, y) \Rightarrow type(x, y)$$

Axiom 11 *Alle Statements, in denen Subjekt und Objekt eine zur Statement ID unterschiedliche ID haben, und die das Label "type" verwenden, sind Instanzen des Sets otelos:type. Das Set otelos:type wird durch das Statement s(otelos:type,otelos:statement,type,otelos:statement) dargestellt.*

$$\forall sid, x, c\, s(sid, x, type, c) \wedge (sid \neq x) \wedge (sid \neq c)$$
$$\Leftrightarrow type(sid, otelos:type)$$

Axiom 12 *Das Label "type" dieser Statements ist in Verbindung mit dem Subjekt x und dem Objekt y eindeutig. Daher wird die Statement ID wie folgt gebildet: sid = x_type_y.*

$$\forall sid1, sid2, x, p, y\, s(sid1, x, p, y) \wedge s(sid2, x, p, y) \wedge (p = type)$$
$$\Rightarrow (sid1 = sid2)$$

Axiom 13 *Property-Instanzen können unter Verwendung des Namens des Properties dargestellt werden (anstelle eines neuen Labels), indem das Hilfsprädikat P(x,m,y) verwendet wird.*

$$\forall sid1, sid2, x, l, y, c, m, d\, s(sid1, x, l, y) \wedge s(sid2, c, m, d) \wedge type(sid1, sid2) \Rightarrow P(x, m, y)$$

Axiom 14 *Properties P des Subjekts x werden immer als Property-Statements dargestellt. Diese Statements sind immer Instanzen der Property-Definitionen einer Klasse, von der x Instanz ist.*

$$\forall x, l, y, c, m, d, sid1, sid2\, type(x, c) \wedge P(x, m, y) \wedge s(sid1, c, m, d) \Rightarrow \exists s(sid2, x, l, y) \wedge type(sid2, sid1)$$

Axiom 15 *Mehrfachinstanzierung: Wenn c Instanz zweier Klassen c und d ist, die beide eine Property m deklarieren, dann muß x auch Instanz einer Klasse g sein, die Subklasse von c und d ist und auch ein Property m deklariert.*

$$\forall x, m, y, c, d, sid1, sid2, e, f (type(x, c) \wedge type(x, d) \wedge s(sid1, c, m, e) \wedge s(sid2, d, m, f) \Rightarrow \exists g, sid3, h\, type(x, g) \wedge s(sid3, g, m, h) \wedge subClassOf(g, c) \wedge subClassOf(g, d))$$

Axiom 16 *Subjekte und Objekte einer Property-Instanz werden durch die zugehörige Property-Definition bestimmt.*

$$\forall sid1, sid2, x, l, y\, s(sid1, x, l, y) \wedge type(sid1, sid2) \Rightarrow \exists c, p, r\ s(sid2, c, p, r) \wedge type(x, c) \wedge type(y, r)$$

Das Axiom 15 beschreibt die Mehrfachinstanzierung / -vererbung für ein Objekt, das Instanz zweier Klassen ist, die beide dasselbe Property deklarieren. In diesem Fall wird, zum O-Telos Axiom 17 korrespondierend, eine dritte Klasse gefordert, die ebenfalls das Property deklariert und so die Instanzierung des Properties durch das Objekt eindeutig macht. Weiterhin fordert das Axiom 16, daß die Klassen des Subjekts und des Objekts einer Property-Instanz durch dessen Property-Definition bestimmt sind. Entsprechend ist der Wertebereich eines Properties auf diese Klasse (wie bei RDF(S)) beschränkt. Dieses zum O-Telos Axiom 14 korrespondierende Axiom ermöglicht im Gegensatz zu RDF(S) jdeoch die Existenz desselben Properties mit unterschiedlichen Wertebereichen für unterschiedliche Klassen. Entsprechend den Axiomen 15 und 16 sind die RDF(S) Konstrukte rdfs:range und rdfs:domain in O-Telos-RDF unnötig und werden vernachlässigt.

Allgemein betrachtet, wird die Instanzierung in O-Telos-RDF, O-Telos und RDF(S) für Individuals auf dieselbe Weise durchgeführt (Axiome 10 und 12 korrespondieren zu den O-Telos Axiomen 4 und 5). Properties hingegen werden in O-Telos-RDF und O-Telos im Gegensatz zu RDF(S) explizit instanziert (Axiome 13 und 14, die den O-Telos Axiomen 7, 8 und 9 entsprechen).

O-Telos-RDF und O-Telos ermöglichen die Deklaration beliebig tiefer Abstraktionshierarchien, da zum einen die Instanzierung von Individuals und Properties explizit angegeben und zum anderen nicht zwischen Klassen, Metaklassen, Meta-Metaklassen,

usw. unterschieden wird. Außerdem bietet das Axiom 13 in O-Telos-RDF die Möglichkeit, RDF(S)-ähnliche Property Statements P(x,m,y) zu übernehmen, indem eine eindeutige Statement ID und ein Label für die Instanz des Properties und ein Statement für die Instanzierung des Properties generiert werden.

Das Axiom 11 korrespondiert zu den O-Telos Axiomen 20 und 27. Es besagt, daß im Gegensatz zu RDF(S) otelos:type kein otelos:property ist, sondern ein eigenes, gleichwertiges Konstrukt darstellt.

4.5.2.7 Das subClassOf Statement

Statements der Form s(sid,x,subClassOf,y) deklarieren die Spezialisierung von y durch x, besagen also, daß x eine Subklasse von y ist.

Axiom 17 *Das Label "subClassOf" in subClassOf-Statements ist in Verbindung mit dem Subjekt und dem Objekt des Statements eindeutig, wobei x und y Individuals sein müssen. Die Statement IDs haben die Form sid = x_subClassOf_y.*

$$\forall sid1, sid2, x, p, y\, s(sid1, x, p, y) \wedge s(sid2, x, p, y) \wedge (p = subClassOf) \Rightarrow (sid1 = sid2)$$

Axiom 18 *Eine subClassOf-Beziehung wird durch das Hilfsprädikat subClassOf(x,y) dargestellt.*

$$\forall sid, c, d\, s(sid, c, subClassOf, d) \Rightarrow subClassOf(c, d)$$

Axiom 19 *Das Set aller subClassOf-Statements wird durch das Statement s(otelos:subClassOf,otelos:individual,subClassOf,otelos:individual) dargestellt.*

Tatsächlich entspricht ein subClassOf-Statement s(otelos:subClassOf,otelos:statement,subClassOf,otelos:statement) der O-Telos Spezialisierung "isA". Da aber in RDF(S) unterschiedliche Konstrukte für die Vererbung von Individuals und Properties existieren, wird in O-Telos-RDF die Gültigkeit der subClassOf-Beziehung auf Individuals eingeschränkt. Für die Properties wird ein dem subClassOf-Statement äquivalentes subPropertyOf-Statement deklariert (siehe das folgende Kapitel 4.5.2.8).

Axiom 20 *Alle Statements mit dem Label "subClassOf", deren sid ungleich Subjekt und Objekt ist, gehören dem Set otelos:subClassOf an.*

$$\forall sid, c, d\, s(sid, c, subClassOf, d) \wedge (sid \neq c) \wedge (sid \neq d) \Leftrightarrow type(sid, otelos : subClassOf)$$

Die subClassOf-Beziehung ist eine partielle Ordnung der Statement IDs. Die Beziehung ist reflexiv und transitiv. Sie beinhaltet keine Zyklen, benutzt jedoch die Reflexivität, um Gleichheit auszudrücken.

Axiom 21 *Reflexivität:*

$$\forall sid\, type(sid, otelos : statement) \Rightarrow subClassOf(sid, sid)$$

Axiom 22 *Transitivität:*

$$\forall sid1, sid2, sid3\ subClassOf(sid1, sid2) \wedge subclassOf(sid2, sid3) \Rightarrow subClassOf(sid1, sid3)$$

Axiom 23 *Keine Zyklen, Statement der Gleichheit:*

$$\forall sid1, sid2\ subClassOf(sid1, sid2) \wedge subClassOf(sid2, sid1) \Rightarrow (sid1 = sid2)$$

Axiom 24 *Klassenzugehörigkeit wird an die Superklassen vererbt.*

$$\forall x, c, d\ type(x, d) \wedge subClassOf(d, c) \Rightarrow type(x, c)$$

Die Axiome 24, 20, 18, 17 und 24 korrespondieren zu den O-Telos Axiomen 28, 21, 6, 4 und 13 und definieren die subClassOf-Beziehung entsprechend RDF(S). Im Gegensatz zu RDF(S) wird sie als partielle Ordnung deklariert. Die Grundaussage von subClassOf bleibt jedoch erhalten, denn die Axiome 21, 22 und 23, die zu den O-Telos Axiomen 10, 11 und 12 korrespondieren, definieren wie auch im RDF(S) die Gleichheit und Zyklenfreiheit.

4.5.2.8 Das Statement subPropertyOf

Das Statement subPropertyOf definiert ähnlich dem subClassOf-Statement für Individuals die Spezialisierung von Properties. Statements der Form s(sid,x,subPropertyOf,y) deklarieren also, daß x eine Subproperty von y ist, wobei x und y Properties sein müssen.

Axiom 25 *Das Label "subPropertyOf" in subPropertyOf-Statements ist in Verbindung mit dem Subjekt und dem Objekt des Statements eindeutig, wobei x und y Properties sein müssen. Die Statement IDs haben daher die Form sid = x_subPropertyOf_y.*

$$\forall sid1, sid2, x, y, l\ s(sid1, x, l, y) \wedge s(sid2, x, l, y) \wedge (l = subPropertyOf) \wedge type(x, otelos : property) \wedge type(y, otelos : property) \Rightarrow (sid1 = sid2)$$

Axiom 26 *Eine subPropertyOf-Beziehung kann auch als Hilfsprädikat subPropertyOf(x,y) dargestellt werden.*

$$\forall c, d, sid\ type(c, otelos : property) \wedge type(d, otelos : property) \wedge s(sid, c, subPropertyOf, d) \Rightarrow subPropertyOf(c, d)$$

Die subPropertyOf-Beziehung ist eine partielle Ordnung der Statement IDs. Die Beziehung ist reflexiv und transitiv. Sie beinhaltet keine Zyklen, benutzt jedoch die Reflexivität, um Gleichheit auszudrücken.

Axiom 27 *Reflexivität:*

$$\forall sid\ type(sid, otelos : statement) \Rightarrow subPropertyOf(sid, sid)$$

Axiom 28 *Transitivität:*

$$\forall sid1, sid2, sid3\ subPropertyOf(sid1, sid2) \wedge subPropertyOf(sid2, sid3)$$
$$\wedge type(sid1, otelos:property) \wedge type(sid2, otelos:property)$$
$$\Rightarrow subPropertyOf(sid1, sid3)$$

Axiom 29 *Keine Zyklen, Statement der Gleichheit:*

$$\forall sid1, sid2\ subPropertyOf(sid1, sid2) \wedge subPropertyOf(sid2, sid1)$$
$$\wedge type(sid1, otelos:property) \wedge type(sid2, otelos:property)$$
$$\Rightarrow (sid1 = sid2)$$

Axiom 30 *Die Klassenzugehörigkeit wird an die Superklassen vererbt.*

$$\forall x, c, d\ type(x, d) \wedge subPropertyOf(d, c) \wedge type(c, otelos:property)$$
$$\wedge type(d, otelos:property) \Rightarrow type(x, c)$$

Axiom 31 *Alle subPropertyOf-Statements sind im Set otelos:subPropertyOf enthalten, das durch das Statement s(otelos:subPropertyOf,otelos:property,subPropertyOf,otelos:property) definiert ist.*

$$\forall sid, c, d\ (s(sid, c, subPropertyOf, d) \wedge (sid \neq c) \wedge (sid \neq d)$$
$$\wedge type(c, otelos:property) \wedge type(d, otelos:property)$$
$$\Leftrightarrow type(sid, otelos:subPropertyOf))$$

Axiom 32 *Subklassen, die Properties mit demselben Namen definieren, den auch Properties der Superklassen haben, müssen diese Properties spezialisieren.*

$$\forall sid1, sid2, c, d, e, f, m$$
$$subClassOf(d, c) \wedge s(sid1, c, m, e) \wedge s(sid2, d, m, f)$$
$$\Rightarrow subClassOf(f, e) \wedge subClassOf(sid2, sid1)$$

Axiom 33 *Subproperties müssen das Subjekt und das Objekt des Superproperties spezialisieren.*

$$\forall sid1, sid2, c, d, e, f, m, l$$
$$subClassOf(sid2, sid1) \wedge s(sid1, c, m, e) \wedge s(sid2, d, l, f)$$
$$\Rightarrow subClassOf(d, c) \wedge subClassOf(f, e)$$

O-Telos-RDF deklariert die subPropertyOf-Beziehung, um die Kompatibilität zu RDF(S) und dessen Beziehung rdfs:subPropertyOf zu erhalten. Die otelos:subPropertyOf Beziehung entspricht weitestgehend der otelos:subClassOf Beziehung, gilt hier aber für Properties. O-Telos an sich unterscheidet nicht zwischen der Spezialisierung von Individuals und Properties. Daher entsprechen die Axiome 25 und 26 den O-Telos Axiomen 4 und 6, und Axiom 31 entspricht den O-Telos Axiomen 21 und 28. Ähnlich der Spezialisierung von Klassen ist auch die Subproperty-Beziehung eine partielle Ordnung ohne Zyklen (Axiome 27, 28 und 29 entsprechen O-Telos Axiomen 10, 11 und 12).

Eine Instanz eines Properties ist natürlich auch Instanz aller Superklassen des Properties, wie im Axiom 30 bzw. O-Telos Axiom 13 gefordert. Die O-Telos-RDF Axiome 32 und 33 korrespondieren zu den O-Telos Axiomen 15 und 16, haben aber keine Entsprechung im RDF(S). Diese Axiome beschreiben das Verhalten der in Individuals deklarierten Properties bei Spezialisierungen der Individuals.

4.5.2.9 Die Statements Sequence und Bag

RDF(S) definiert die Konstrukte rdfs:Seq und rdfs:Bag. rdfs:Bag stellt eine Sammlung von Ressourcen explizit dar. Da diese Fähigkeit in O-Telos und O-Telos-RDF nativ vorhanden ist (Klassenzentriertheit und Mehrwertigkeit aller Attribute), muß dieses Konstrukt in O-Telos-RDF nicht definiert und deklariert werden. rdfs:Seq hingegen stellt eine Sequenz von Ressourcen dar. Anstatt jedoch dessen Definition in O-Telos-RDF direkt zu übernehmen, wird die Fähigkeit von O-Telos und O-Telos-RDF genutzt, den Properties wieder Properties zuzuordnen. Es wird also den Properties, die in einer Sequenz dargestellt werden sollen, eine Ordnungszahl zugeordnet.

In O-Telos-RDF wird zu diesem Zweck ein Individual otelos:ordinal als Subklasse des otelos:literal Statements deklariert, dessen Instanzen die Folge der Integerzahlen mit einem "_"-Prefix als Label darstellen. Außerdem wird das Sequenz-Statement als Set aller Sequenzen deklariert: s(otelos:sequence,otelos:statement,sequence,otelos:ordinal).

Zur Darstellung einer Sequenz wird also jedem Property eine Ordnungszahl zugeordnet. Die Ordnungszahl gibt die Position des Properties in der Sequenz wieder. Auf diese Weise wird das in RDF(S) mögliche Definieren von unklassifizierten Datencontainern wie etwa bei den Instanzen von rdfs:Seq vermieden.

4.5.3 Die Modellierung der Vorlesung Künstliche Intelligenz

Anhand der Modellierung der Vorlesung "Künstliche Intelligenz (KI)", gehalten von Professor Nejdl an der Universität Hannover, werden die Unterschiede und Gemeinsamkeiten von RDF und O-Telos-RDF dargestellt. Das Beispiel wird hier auf einen kleinen Ausschnitt aus dem Modell der Vorlesung beschränkt, dessen vollständige Wiedergabe den Rahmen dieser Arbeit überschreiten würde.

Der Modellausschnitt deklariert auf Klassenebene, daß eine Vorlesungseinheit (`LectureUnit`)[3] einer Vorlesung (`Lecture`) zugeordnet ist. Vorlesungseinheiten haben einen Titel (`title`) und eine Beschreibung (`description`). Sie bestehen aus Theorieseiten (`TheoriePage`), Beispielen, usw.

Der Modellausschnitt wird einmal in der RDF XML-Serialisierung und einmal in der O-Telos-RDF XML-Serialisierung dargestellt. Die unterschiedlichen Serialisierungen und deren Statements werden in den Anhängen B.3.1 bis B.3.4 angegeben. Das RDF-Modell im Anhang B.3.1 und das O-Telos-RDF-Modell im Anhang B.3.2 beschreiben denselben Ausschnitt des Modells der KI-Vorlesung (`ailecture`). Die im Anhang B.3.2 verwendete XML-Serialisierung von O-Telos-RDF ist im Anhang B.2 wiedergegeben.

In der Serialisierung im Anhang B.3.2 sind unter Verwendung des RDF-O-Telos Sche-

[3] In Klammern werden die in den Quelltexten verwendeten Bezeichnungen angegeben.

mas alle Statements Instanzen der Individuals otelos:individual, otelos:property, otelos:type, etc. Diese werden jedoch nicht explizit angegeben, da sie auf der syntaktischen Form der Statements basieren und als gegeben vorausgesetzt werden können. In RDF sind solche Zuordnungen nicht definiert und entfallen daher völlig (vergl. Anhang B.3.1).

Ein weiterer Unterschied der beiden Serialisierungen ist die explizite Deklaration jeder WWW-Seite im Anhang B.3.2. Ebenso explizit werden in diesem Beispiel die type-Beziehungen der Properties der `LectureUnit1` modelliert. Die explizite Darstellung wird durch die Axiomatisierung von O-Telos-RDF gefordert, um sicherzustellen, daß Property-Statements entweder ein Property instanzieren oder ein neues Property deklarieren.

Diese Unterschiede bilden die Basis, die die Fähigkeiten von O-Telos-RDF zur Metamodellierung und Reifikation ermöglichen. Außerdem wird wiederum deutlich, daß RDF-Schemata propertyzentriert sind, während O-Telos-RDF Schemata klassenzentriert sind.

In den Anhängen B.3.3 und B.3.4 werden die zu den XML-Serialisierungen des Modellausschnitts von RDF (Anhang B.3.1) und O-Telos-RDF (Anhang B.3.2) gehörenden Statements dargestellt.

Ein Vergleich der Darstellungsformen des Modellausschnitts der KI-Vorlesung in RDF-Serialisierung und -Statements mit der O-Telos-RDF-Serialisierung und -Statements verdeutlicht die Vorteile, die das O-Telos-RDF-Schema gegenüber dem originären RDF-Schema aufweist. Die Reifikation der Statements wird vereinfacht, Metamodellierung mit mehr als drei Abstraktionsebenen ermöglicht und Properties mit demselben Namen können für unterschiedliche Klassen mit unterschiedlichen Wertebereichen deklariert werden. Außerdem wird O-Telos-RDF in einer formalen Beschreibung deklariert, die stark an die Axiomatisierung von O-Telos angelehnt ist. O-Telos-RDF löst in der aktuellen Version die dualen Rollen der RDF(S) Konstrukte rdf:type, rdfs:subClassOf und rdfs:subPropertyOf auf.

Kapitel 5

Allgemeine Eigenschaften des KBS Hyperbook Systems

Dieses Kapitel beschreibt die Eigenschaften des KBS Hyperbook Systems. Es ist ein offenes System, so daß die im System beschriebenen Hypermedia-Dokumente im WWW aufzufinden sind. Das System ist erweiterbar und flexibel: Die Erweiterbarkeit ist sowohl gegenüber neuen und zusätzlichen Modellen als auch gegenüber neuen Funktionalitäten gegeben. Die Flexibilität ermöglicht den Einsatz des Systems in anderen, neuen Kontexten, neuen Situationen und neuen Szenarien.

Das System sorgt außerdem für die Konsistenz der verwendeten Repräsentationsmodelle, indem es nur Änderungen an diesen zuläßt, die nicht zu einem inkonsistenten Modell führen. Die Wiederverwendbarkeit des Systems und der Repräsentationsmodelle wird durch die modulare Implementierung und die Verwendung von Metadaten gewährleistet.

Mittels des KBS Hyperbook Systems wird ein Hyperbook im WWW realisiert. Dieses verwendet abruf- und darstellbare WWW-Seiten, deren Aufbau aus konventionellen HTML-Sprachelementen besteht. Diese grundlegende Funktionalität wird unter anderem durch den Einsatz bereits existierender Technologien und Tools ermöglicht. Verwendete Tools umfassen die Datenbanksysteme ConceptBase [67] und ObjectStore [35] sowie den Java WebServer [108]. Es werden unter anderem der Servlet Mechanismus [27] im Java WebServer, die objekt-orientierte Speicherung der in der Programmiersprache Java realisierten Repräsentationsmodelle und ein Parser zum Erzeugen der Java-Objekte aus dem O-Telos Modell eingesetzt.

Im Folgenden sollen zunächst die Eigenschaften des KBS Hyperbook Systems näher erläutert werden. Im Anschluß folgt dann eine Beschreibung der eingesetzten Technologien und Tools.

5.1 Das KBS Hyperbook System

Das KBS Hyperbook System ist ein Werkzeug zum Erstellen, Warten, Pflegen und Anzeigen von Hyperbooks. Die Hyperbooks bestehen aus den die jeweiligen Domänen beschreibenden Modellen, den zugehörigen Hypermedia-Dokumenten und der Software, die zur Verarbeitung der Modelle und zur Produktion der jeweils gewünschten Funktionalitäten notwendig ist.

5.1.1 Ein offenes System

Die erstellten Hyperbooks sind offene Systeme, deren Inhalte in Form von Hypermedia-Dokumenten beliebige Lokationen aufweisen können. Die Lokation der Hypermedia-Dokumente wird über deren eindeutige Identifizierung durch die URI angegeben, wobei die Dokumente selbst nicht im Hyperbook gespeichert sein müssen.

Im Hyperbook beschreiben Konstrukte die Dokumente, die alle notwendigen Informationen zum Auffinden und Repräsentieren der Dokumente beinhalten. Diese Art Konstrukte wird Konzept genannt und entspricht dem `Concept`, das in Kapitel 4.3 vorgestellt wird. Entsprechend existieren im Hyperbook Konstrukte, die die Konzepte miteinander in Beziehung setzen. Diese werden als Relation typisiert und entsprechen dem Konstrukt `Relation` aus Kapitel 4.3.

5.1.2 Erweiterbarkeit und Flexibilität

Das KBS Hyperbook System und die damit erstellten Hyperbooks weisen die Eigenschaften der Flexibilität und Erweiterbarkeit auf. Flexibilität bedeutet zum einen, daß ein Hyperbook in mehreren Kontexten eingesetzt werden kann. So kann es einer Vorlesung als Skriptum, als Nachschlagewerk, als Darstellungsmedium der studentischen Projekte und Projektlösungen sowie als Vorbereitungshilfe auf Klausuren dienen. Das in der Vorlesung “Grundzüge der Informatik 1” eingesetzte und im Kapitel 4.4 beschriebene Hyperbook ist ein Beispiel für diese Vielfalt an Verwendungsszenarien.

Die Flexibilität impliziert die Erweiterbarkeit des Hyperbook-Systems. Es können die Konstrukte für Hypermedia-Dokumente und deren Relationen hinzugefügt werden. Neue Konstrukte können auch Metadaten sein, die zu zusätzlichen Strukturen im Modell und somit im Hyperbook führen. Die neuen Konstrukte müssen, wenn sie als Metadaten respektive Annotationen verwendet werden, nicht mit einem Hypermedia-Dokument assoziiert sein. Bestehende Konstrukte können modifiziert werden, d.h. sie können um neue Attribute und Werte erweitert werden, ihre Attribute können neue Werte annehmen oder sie können einem neuen bzw. anderen Typ als Subklasse oder Instanz zugeordnet werden. Natürlich können im Hyperbook-System existierende Konstrukte auch entfernt werden. Es können entweder ganze Konstrukte oder die Attribute eines Konstrukts gelöscht werden.

Das KBS Hyperbook System und die zugehörigen Hyperbooks sind auch im Hinblick auf die Implementierung flexible Systeme. Sie basieren in der Implementierung auf einem modularen Aufbau, wobei die einzelnen Module bezüglich des Gesamtsystems bestimmte Funktionen erfüllen. Es existieren Module für die Visualisierung der Hypermedia Dokumente, für die Auswertung der Daten, zur Benutzerverwaltung, zur Adaption, etc. Die Module können unter Berücksichtigung der entsprechenden Schnittstellenspezifikationen durch andere Module mit gleichen Schnittstellen (vergleiche Anhang A.4) ausgetauscht werden. Auf ähnliche Weise können auch neue Module in das KBS Hyperbook System integriert werden, wobei im Allgemeinen keine Änderung des vollständigen Systems notwendig wird.

5.1.3 Konsistente Daten

Die Modifikation des Inhalts der Hyperbooks durch die beschriebenen Funktionalitäten kann zu Fehlfunktionen des Hyperbooks und zu fehlerhaften Datenmodellen führen. Um dies zu verhindern, führt das System Konsistenzprüfungen durch, wovon einige hier beispielsweise aufgeführt sind:

- Wird z.B. ein neues Konstrukt in das Hyperbook eingeführt, so muß es Subklasse oder Instanz von `Concept` oder `Relation` des allgemeinen Meta-Modells aus Kapitel 4.3.1 sein.
- Um als Relation anzeigbar sein zu können, muß ein Konstrukt Instanz der `DisplayRelation` sein und dem `displayConstraint` genügen, entsprechend dem allgemeinen Modell.
- Der verwendete Identifikator (ein System-interner Bezeichner) muß eindeutig sein, d.h. er darf als Identifikator von keinem anderen Konstrukt verwendet werden.
- Ein als Text definiertes Attribut darf keinen Nicht-Text-Wert annehmen.
- Zum Löschen eines Konzepts müssen dessen Beziehungen zu anderen Konzepten berücksichtigt und ggf. auch gelöscht werden.
- Beim Anlegen von Subklassen und Instanzen mit zugehörigen Beziehungen sind Selbstbezüge zu verhindern.

Zusammenfassend gilt, daß das Hyperbook System bei allen Modifikationen des zugehörigen Modells prüft, ob die Änderungen ein wiederum konsistentes Modell ergeben. Die Änderungen werden vom Hyperbook-System zurückgewiesen, wenn sie zu einem nicht konsistenten Repräsentationsmodell führen würden.

5.1.4 Wiederverwendbarkeit

Die in den Hyperbooks des KBS Hyperbook Systems verwendeten Daten sind nicht fest in die jeweiligen Hyperbooks einprogrammiert, sondern werden diesen in Form separater Datenmodelle, den Repräsentationsmodellen, zugeführt. Da alle Hyperbooks auf der Basis desselben Meta-Modells und der darin definierten Basis-Konstrukte `Concept` und `Relation` arbeiten, sind die Modelle zwischen den Hyperbooks austauschbar. Solch ein Austausch führt normalerweise zu einem Verlust von Teilen der Funktionalität des Hyperbooks, denn die einzelnen Hyperbooks sind mit unterschiedlichen zusätzlichen Modulen ausgestattet, die auf der Basis spezieller Konstrukte in den Modellen funktionieren (siehe z.B. die Adaptionskomponenten aus [53]). Mittels der Modelle werden die eigentlichen Hypermedia-Dokumente beschrieben, so daß die Dokumente an ihren ursprünglichen Speicherorten verbleiben können und für die Verwendung in anderen Modellen und Kontexten unabhängig vom Hyperbook weiter zur Verfügung stehen. Es können mehrere zunächst unabhängige Modelle nebeneinander in einem Hyperbook existieren. Die verschiedenen Modelle werden durch ein gemeinsames Meta-Modell modelliert, wie das auch in [42, 121, 36] beschrieben wird.

Aufgrund dieser modularen Darstellungsweise der Daten in den Hyperbooks, und somit auch im KBS Hyperbook System, wird eine umfassende Wiederverwendbarkeit der Daten und Modelle gewährleistet. Es ist beispielsweise möglich, die Modelle zu exportieren, um sie in anderen Systemen zu verwenden. Voraussetzung hierfür ist, daß die zugrunde liegende Semantik und Grammatik der Modellierungssprachen und die Modellierungskonzeptionen zueinander kompatibel sind.

Das KBS Hyperbook System importiert Daten aus anderen Modellierungen, wie beispielsweise anhand des Imports von RDF-Dokumenten und RDF-Schemata im vorhergehenden Kapitel 4.3.5 verdeutlicht wurde. Auch hier gilt als Voraussetzung, daß die verwendeten Konzeptionen und Modellierungen zueinander kompatibel sein müssen.

Im Hinblick auf die Implementierung werden zur Konstruktion der einzelnen Hyperbooks dieselben Module verwendet. Ein Modul wird dabei durch eine Schnittstellenspezifikation (siehe Anhang A.4) beschrieben. Aufgrund der verwendeten Schnittstellenspezifikationen ist es möglich, nur die Module in einem Hyperbook zu verwenden, die für die notwendigen Funktionalitäten benötigt werden.

5.1.5 Die Darstellung der WWW-Seiten

Die Hyperbooks des KBS Hyperbook Systems erzeugen zur Laufzeit WWW-Seiten, die über das Internet abgerufen werden können. Die WWW-Seiten werden in der im WWW verwendeten Hypertext Markup Language (HTML) [93] formuliert. Sie sollen keine besonderen Anforderungen an handelsübliche Browser wie den Netscape Browser oder den Microsoft Internet Explorer stellen. Der jeweilige Browser muß allerdings in der Lage sein, HTML-Frames wiedergeben zu können.

Die WWW-Seiten stellen neben dem jeweiligen Hypermedia-Dokument die Struktur

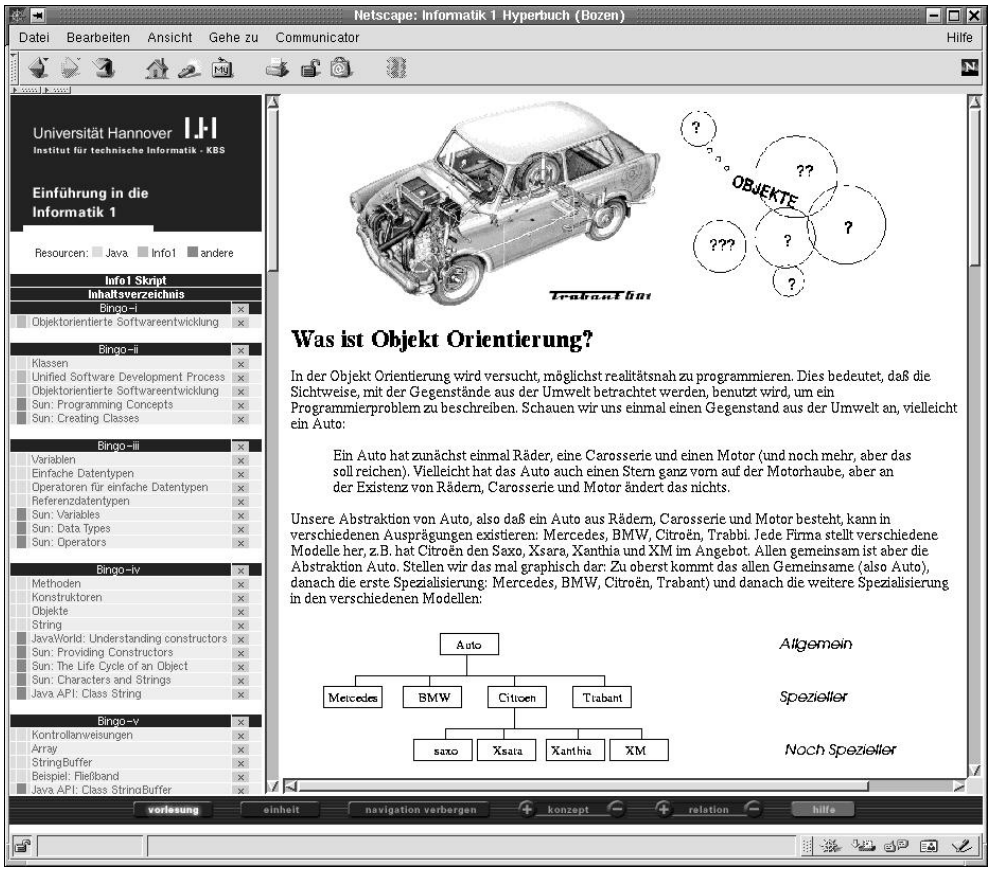

Abbildung 5.1: Eine von einem Hyperbook generierte WWW-Seite

dar, in die das Dokument eingebettet ist. Die Struktur besteht aus den vom System generierten, zum Dokument gehörenden Beziehungen zu anderen Dokumenten. Sie werden als Hypertext-Links in einem ausgezeichneten Bereich der WWW-Seite realisiert.

Der Betrachter der WWW-Seite soll aufgrund des Aufbaus der Seite zwischen dem Hypermedia-Dokument, den semantischen Links und ggf. zusätzlichen Funktionalitäten des Hyperbooks unterscheiden können. Entsprechend wird in Abhängigkeit von der Konfiguration des Hyperbooks die WWW-Seite aus mindestens zwei oder drei Frames aufgebaut (siehe als Beispiel Abbildung 5.1). In einem Frame wird das Hypermedia-Dokument angezeigt, ein weiterer Frame beinhaltet die strukturellen Links und der dritte Frame enthält mögliche zusätzliche Hyperbook-Funktionalitäten. Das Aussehen der vom Hyperbook generierten WWW-Seite kann mit Ausnahme des Hypermedia-Dokument-Frames durch einen einfachen Konfigurationsmechanismus den Vorstellungen des Hyperbook-Autors angepaßt werden. Eine ausführliche Erläuterung folgt im nachfolgenden Kapitel 6.1.

5.2 Verwendete Tools

Im KBS Hyperbook System werden eine Reihe von bereits existierenden Tools verwendet, die im Folgenden kurz erläutert werden.

Zur Modellierung der Repräsentationsmodelle der KBS Hyperbooks wird die Modellierungssprache O-Telos [70] eingesetzt. Sie ist im Datenbanksystem ConceptBase [67, 68] implementiert. Das ConceptBase Datenbanksystem dient der Validierung und der Speicherung der Repräsentationsmodelle des KBS Hyperbook Systems. In der aktuellen Version des KBS Hyperbook Systems wird das ConceptBase Datenbanksystem nur zur Validierung der Modelle eingesetzt. Als zweites Datenbanksystem wird hier das ObjectStore Datenbanksystem als Repository eingesetzt. Die aktuelle Version des ObjectStore Datenbanksystems erscheint unter denselben Bedingungen als Repository des KBS Hyperbook Systems performanter zu sein als das ConceptBase Datenbanksystem.

Der Java WebServer wird als Webserver zur Anbindung der Hyperbook-Systeme an das WWW eingesetzt. Dieser Webserver verwendet die Technologie der Servlets als Alternative zu Common Gateway Interfaces (CGI) [122, 23]. Die Hyperbook-Systeme sind als Servlets im Webserver in der Sprache Java implementiert und verwenden die jeweiligen Datenbanksysteme als Speicher für die Repräsentationsmodelle.

5.2.1 O-Telos und ConceptBase

Die im KBS Hyperbook System eingesetzten Modelle werden in der Modellierungssprache O-Telos formuliert [70, 67, 68]. An dieser Stelle folgt nun zunächst eine kurze Beschreibung der Sprache O-Telos. Diese Erläuterungen sind an die Ausführungen von M. Jeusfeld in [71] angelehnt.

O-Telos ist eine objekt-orientierte, deduktive Modellierungssprache, die an der RWTH Aachen auf der Basis von Telos [83] und anderen Modellierungssprachen entwickelt wurde. Sie ist, wie die Vorgänger auch, eine Sprache zur Wissensdarstellung. Zur Darstellung aller beliebigen Konstrukte wie z.B. Klassen, Metaklassen, Objekte, Vererbungs- und Instanzrelationen werden Tupel einer einzigen Art verwendet. Diese Tupel entsprechen den kleinsten, nicht mehr weiter teilbaren Einheiten/Konstrukten der Modellierungssprache. Auf diesen Einheiten basieren die ca. 30 Axiome, die die Semantik von O-Telos bilden. In Form von Regeln (Rules) und Einschränkungen (Constraints) beschreiben die Axiome z.B. die Vererbungs- und Instanzierungsmechanismen. Die Semantik von O-Telos basiert auf der Sprache DATALOG [112], wobei DATALOG eine Version von Prolog für Datenbanksysteme ist.

Die Sprache O-Telos eignet sich für die Darstellung der in dieser Arbeit verwendeten Modelle, da sie eine Datenmodellierungssprache einer Metadatenbank darstellt [71]. Sie ist in der Lage, die semantischen Eigenschaften anderer Modellierungssprachen, beispielsweise ER-Diagramme oder UML, darzustellen.

O-Telos wird in einem System namens ConceptBase (CB) [67] implementiert. CB ist ein multi-benutzerfähiges, deduktives und objekt-orientiertes Datenbanksystem für den Einsatz in der konzeptionellen Modellierung, das die Datenmodellierungssprache O-Telos verwendet. Zusätzliche Funktionalitäten wie ein Query- oder ein View-Mechanismus für Data-Retrieval werden im ConceptBase Datenbanksystem als Erweiterungen zu O-Telos (z.B. query- und view-Klassen) realisiert (s. [68, 24]). Auf der Basis von O-Telos stellt CB alle Konstrukte, wie etwa Klassen, Metaklassen oder Vererbungsbeziehungen als Objekte dar. Diese Objekte können während ihres Lebenszyklusses modifiziert werden. Rules und Constraints, als Objekte modelliert, können unter anderem zur Konsistenzwahrung eingesetzt werden.

CB ist als Client-Server-Architektur realisiert. Zur Kommunikation von Server und Clients wird die Interprozess-Kommunikation des Internet-Protokolls eingesetzt. Die Clients können in den Sprachen Prolog, C oder Java implementiert sein.

Zu Forschungszwecken implementiert das KBS Hyperbook System auch eine Schnittstelle zur CB-Datenbank als Objektspeicher. Die Schnittstelle zwischen CB und den Hyperbook Systemen wird in Java realisiert. Als einfach realisierte Schnittstelle bietet sie die gleichzeitige Nutzung durch mehrere voneinander unabhängige Clients. CB stellt nicht die Möglichkeit des Multi-Threadings zur Verfügung, läßt also den gleichzeitigen, mehrmaligen Zugriff eines Programms auf CB nicht zu. Das Multi-Threading wird in den Hyperbook Systemen eingesetzt, um eine schnellere Bearbeitung der Anfragen zu erreichen. So kann CB zwar für Forschungszwecke im KBS Hyperbook System eingesetzt werden, genügt aber nicht der, wenn auch subjektiven, Anforderung nach hoher Verarbeitungsgeschwindigkeit. Die Verarbeitungsgeschwindigkeit beschreibt hier die Geschwindigkeit, mit der eine zu einem KBS Hyperbook gehörende WWW-Seite in einem Browser nach ihrer Anforderung durch den Browser aufgebaut wird.

5.2.2 Der Java-Webserver, Servlets und Java

Mit dem KBS Hyperbook System werden Hyperbooks erstellt, die über das Internet im WWW veröffentlicht werden. Das WWW und das darin verwendete Hypertext-Transmission-Protokoll (HTTP) werden genutzt, denn sie sind nahezu weltweit verfügbar. Außerdem existieren für eine große Anzahl von Plattformen Browser zum Betrachten der WWW-Seiten. Der Benutzer soll zum Betrachten der Hyperbooks keine zusätzliche Software auf seinem Rechner installieren müssen.

Die WWW-Seiten der Hyperbooks werden aus diesem Grund in der im WWW gebräuchlichen Sprache HTML zur Laufzeit vom Hyperbook System erstellt. Da zwischen den einzelnen Browsern geringe Unterschiede in der Interpretation der HTML-Spezifikation [93] existieren (vergleiche z.B. Netscape Navigator und Microsoft Internet Explorer und deren unterschiedliche Darstellung von Style Sheets), werden nur solche HTML-Sprachelemente verwendet, die von den verbreiteteren Browsern Netscape Navigator und Microsoft Internet Explorer in gleicher Weise unterstützt werden.

Der Microsoft Internet Explorer und der Netscape Navigator werden aufgrund der unter [37, 79] veröffentlichten Statistiken in dieser Arbeit als die verbreiteteren Browser angesehen. Nach diesen Statistiken wird der Microsoft Internet Explorer von ca. 73%, der Netscape Navigator von ca. 26% der Internetbenutzer verwendet. Eine genauere Aussage ist unter Berücksichtigung von [46] aufgrund der im WWW verwendeten Cache-Architekturen nur eingeschränkt möglich und für diese Arbeit nicht notwendig.

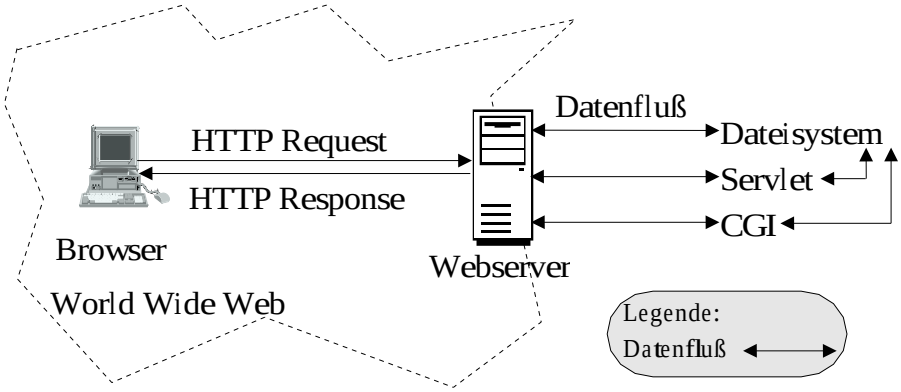

Abbildung 5.2: Der Datenfluß im WWW zur Darstellung einer WWW-Seite

Die WWW-Seiten werden von Webservern im Internet bereitgestellt. Auf Anfrage versucht ein Webserver, die angeforderte Seite zu laden und an den anfragenden Browser zu versenden. Der Webserver lädt die WWW-Seiten aus dem lokalen Dateisystem, von einem anderen Webserver oder er generiert die WWW-Seite selbst. Möglich ist auch, daß er die Generierung der WWW-Seite in einem anderen, im lokalen Netz befindlichen Server, z.B. einem Datenbanksystem wie ConceptBase oder ObjectStore, anstößt und von dort lädt. Die Webserver bilden also die Schnittstelle zwischen den im lokalen Netz agierenden Servern, beispielsweise den File- oder Datenbankservern und dem über das Internet kommunizierenden Browser. Abbildung 5.2 gibt diesen Zusammenhang wieder.

Die Hyperbooks übernehmen die Funktion von Webservern. Sie erhalten die Anfragen nach den WWW-Seiten, erstellen diese und führen deren Lieferung an die anfragenden Browser durch. Das KBS Hyperbook System verwendet als Webserver den Java WebServer [108] der Firma JavaSoft/Sun. Der Java WebServer ist implementiert die Java-Server-Architektur [108], die die Servlet Technologie [27] spezifiziert.

Der Java WebServer leitet die an die Hyperbooks gerichteten Anfragen an die Servlets weiter, wie Abbildung 5.3 darstellt. Die Servlets holen die zur Bearbeitung der Anfragen notwendigen Daten der Modelle vom Datenbanksystem ObjectStore (wird im Kapitel 5.2.3 näher erläutert). Aus den Daten der Modelle berechnen die Servlets dann, welche Hypermedia-Dokumente aus dem Internet geholt werden müssen.

Im KBS Hyperbook System sind zu Forschungszwecken verschiedene Möglichkeiten enthalten, wie die Daten des zu einer WWW-Seite gehörenden Hypermedia-Dokuments zum anfragenden Browser gelangen. Entweder lädt das System die Seite

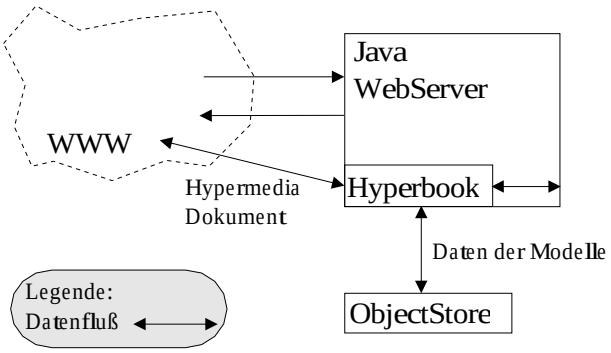

Abbildung 5.3: Der Datenfluß im KBS Hyperbook System

erst selbst, um sie zu manipulieren und leitet sie dann weiter (vergl. Abbildung 5.3), oder das System veranlaßt den Browser, das Hypermedia-Dokument selbst zu laden (Redirect-Mechanismus). Im KBS Hyperbook System wird die zweite Variante eingesetzt, da die Manipulation fremder Hypermedia-Dokumente datenschutzrechtlich sehr fragwürdig ist. Die Verwendung des Redirect-Mechanismus verdeutlicht die Trennung der Daten von dem Repräsentationsmodell, welches sie beschreibt.

Die im KBS Hyperbook System eingesetzten Servlets bieten im Vergleich zu Common Gateway Interface (CGI)-Skripten [122, 23] eine Reihe von Vorteilen. Im Gegensatz zu CGI-Skripten sind Servlets in der Programmiersprache Java geschrieben, wodurch sie unter anderem Plattform-unabhängig einsetzbar sind. Sie werden vom Server einmal geladen und existieren auch zwischen der Bearbeitung zweier Anfragen, während CGI-Skripte nach jeder Abarbeitung einer Anfrage terminieren. Der Server legt vom Servlet für jede Anfrage eine Instanz an, wobei nur die erste Instanz initialisiert werden muß. Alle weiteren Instanzen desselben Servlets erhalten die initialen Eigenschaften der ersten Instanz sofort bei ihrer Instanzierung. Als Folge davon reagieren Servlets bei komplexen Aufgaben schneller auf Anfragen als CGI-Skripte, die zur Bearbeitung jeder Anfrage neu geladen und initialisiert werden müssen [82]. Unter Ausnutzung dieser Servletseigenschaft wird in einem Hyperbook des KBS Hyperbook Systems zur Laufzeit nur einmal die Datenbankverbindung aufgebaut, um dann von allen Servletinstanzen verwendet zu werden.

Servlets sind darüber hinaus sichere Programme. Sie unterliegen den Sicherheitsmechanismen der Sprache Java [40], weshalb sie ausschließlich von dem Webserver, der sie erzeugt, verwendet werden können. Ein Client bzw. Browser im Internet hat keine Möglichkeit, direkt auf ein Servlet zuzugreifen. Im Gegensatz dazu ist ein CGI-Skript immer ein Programm, das auch ohne Webserver-Einsatz vollständig lauffähig ist. Als Folge kann ein CGI-Skript von unberechtigter Seite für unerlaubte Zwecke eingesetzt werden. Wenn der Webserver ungenügend konfiguriert ist, sind solche CGI-Skripte auch über das Internet ausführbar und stellen dann Sicherheitslücken für die lokalen Netzwerke dar.

Ein weiterer Vorteil der Servlets ist die Absturzsicherheit des Servers. Ein Absturz

eines Servlets führt im Gegensatz zu CGI-Skripten nicht zum Absturz des gesamten Serverprozesses. Diese Eigenschaft beruht auf der Art der Einbindung des Servlets in den Webserver, der für jedes Servlet einen eigenen Subprozess verwendet, während CGI-Skripte als Teil des Webserverprozesses vom Webserver ausgeführt werden. Stürzt ein Servlet eines Hyperbooks z.B. durch Fehlbedienung ab, muß der Browser die gewünschte WWW-Seite einfach noch einmal anfordern, ohne das ein Serverneustart oder ähnliches notwendig ist. Auch wird die Arbeit anderer Servlets im Webserver durch den Absturz eines der Servlets nicht behindert. Es ist also davon auszugehen, daß der Betrieb eines Webservers durch die Verwendung von Servlets weniger störanfällig verläuft.

Die Servlets werden im Internet aufgrund ihrer URI (siehe Kapitel 2.1) identifiziert. Der als vierter Bestandteil in einer URI vorhandene Textstring wird vom Webserver aus der URI extrahiert und ist dem Servlet zugänglich, das ihn dann auswertet. In Bezug auf das KBS Hyperbook System enthält der Textstring Informationen zur Identifzierung des jeweiligen Benutzers, zum gewünschten Hypermedia-Dokument und zur gewünschten Verhaltensweise des Hyperbooks.

Die Servlets werden in der Programmiersprache Java auf der Basis des Java Development Kits (JDK) in der Version 1.2.2 erstellt. Die objekt-orientierte Programmiersprache Java [48] wird von der Firma Sun/Javasoft entwickelt und verbreitet.

Java ist eine objekt-orientierte und plattform-unabhängige Programmiersprache für den Einsatz in heterogenen Netzwerken. Die Sprache wurde als objekt-orientierte Programmiersprache entwickelt, so daß objekt-orientierte Konzepte wie Vererbung und Instanzierung direkt in die Programmiersprache und deren Entwicklungsumgebung integriert sind. Der Vererbungsmechanismus von Java erlaubt die Einfachvererbung von Klassen. Mehrfachvererbung und Meta-Klassenbildung werden nicht unterstützt, so daß in O-Telos formulierte Modelle nicht direkt in Java abgebildet werden können.

Die Basisklassen der Programmiersprache Java sind im Java Application Programming Interface (API) [106] definiert. Sie ermöglichen neben den grundlegenden Funktionalitäten einer Programmiersprache, wie einfache Datenstrukturen, Flußkontrollanweisungen, Ein- und Ausgabemöglichkeiten, auch die Netzwerkunterstützung, das Multi-Threading mit Synchronisationsmechanismen, grafische Oberflächen, Fehlerbehandlung und Serialisierung von Objekten. Ein Beispiel für eine Java Applikation ist das Software Modellierungstool TogetherJ von TogetherSoft [110].

5.2.3 Das ObjectStore Datenbanksystem

Das ObjectStore Datenbanksystem [35] der Firma Excelon ist ein objekt-orientiertes Datenbanksystem, das Objekte der Programmiersprachen Java und C++ verwalten kann. Über ein Java API [34] wird das Datenbanksystem direkt in das Programm, hier das Hyperbook Servlet, integriert. Im Servlet muß somit die Netzwerkanbindung des Datenbanksystems nicht weiter berücksichtigt werden. Den Aufbau, die Verwaltung und den Betrieb von Netzwerkanbindung und der Kommunikation darüber implemen-

tieren die im ObjectStore Java API bereitgestellten Klassen automatisch.

ObjectStore speichert die Java-Objekte direkt, d.h. die Objekte existieren als vollständige Objekte in der Datenbank, wobei auch deren Referenzen zu anderen Objekten erhalten bleiben. Bei einer Datenbankanfrage werden das gefragte Objekt und alle damit zusammenhängenden Objekte als Antwort geliefert. Es entfällt die bei relationalen Datenbanken notwendige Konvertierung der Objektdarstellung in die Tabellendarstellung und vice versa.

Das Datenbanksystem wird in einem lokalen Netzwerk betrieben, so daß ein Client des Datenbanksystems nicht auf demselben Rechner arbeiten muß, auf dem das Datenbanksystem arbeitet. Diese Eigenschaft von ObjectStore wird im KBS Hyperbook System zur Lastverteilung und somit für Performance-Gewinne durch parallelisierte Abarbeitung eingesetzt. ObjectStore startet zu diesem Zweck auf dem Rechner, auf dem das Datenbanksystem arbeitet sowie auf dem Clientrechner je ein Cachemanager-Programm (Cache-Forward Architektur genannt). Diese Cachemanager haben die Aufgabe, Anfragen an die jeweilige Datenbank entweder aus den von ihnen verwalteten Caches oder aus der Datenbank zu beantworten. Wenn der Cachemanager eines Clienten die Anfrage nicht beantworten kann, fordert der Cachemanager die notwendigen Daten von dem Cachemanager an, der auf demselben Rechner läuft wie der Datenbankprozess. Dieser beantwortet die Anfrage entweder aus seinem Cache oder aus der Datenbank. Die zwischen den Cachemanagern im lokalen Netzwerk ausgetauschten und in der ObjectStore Datenbank gespeicherten und verwalteten Daten sind serialisierte Java-Objekte [34].

Aufgrund des Einsatzes von lokal auf Clientseite vorhandene und durch Cachemanager verwaltete Caches ist das ObjectStore Datenbanksystem deutlich schneller in der Bearbeitung der Client-Aufträge als das ConceptBase Datenbanksystem.

Unter Ausnutzung dieser auf Cachemanagern beruhenden Systemarchitektur implementiert ObjectStore die Möglichkeit des gleichzeitigen Zugriffs mehrerer Clients auf dasselbe Objekt in einer Datenbank (Concurrency Control). Das Hyperbook-Servlet generiert z.B. mit Hilfe der Datenbank WWW-Seiten, während gleichzeitig am Repräsentationsmodell von einem anderen Client Veränderungen vorgenommen werden. Concurrency Control wird von ObjectStore auch für die Implementierung des Multi-Threadings eingesetzt, das den gleichzeitigen Zugriff eines Programms auf ein Objekt der Datenbank ermöglicht. Das KBS Hyperbook System verwendet Multi-Threading zur Bearbeitung der WWW-Seitenanfragen, indem es mehrere Instanzen der Hyperbook-Servlets gleichzeitig einsetzt, die gleichzeitig auf die Datenbank zugreifen.

Außerdem stellt ObjectStore Mechanismen für die Datensicherung zur Verfügung. Kommt es im Rahmen von technischen Pannen zu einem Systemabsturz, können die Daten aus den vom Datenbanksystem angelegten Backup-Sicherungskopien wiederhergestellt werden.

Kapitel 6

Aspekte der Implementierung des KBS Hyperbook Systems

In diesem Kapitel wird die Implementierung des KBS Hyperbook Systems und darauf basierender Hyperbooks vorgestellt. Zunächst wird eine von einem Hyperbook generierte WWW-Seite beschrieben und erläutert. Darauf aufbauend wird die Generierung der WWW-Seiten anhand eines Repräsentationsmodells erläutert. Auf der Basis der verwendeten Datenstrukturen wird geschildert, wie die Repräsentationsmodelle im O-Telos Framesyntax in Java-Objekten abgebildet werden. Das Kapitel endet mit einer kurzen Beschreibung der einzelnen Module des Hyperbooks.

6.1 Beschreibung der WWW-Seiten des Hyperbooks

Hypermedia-Dokumente werden im Rahmen der generierten WWW-Seiten des Hyperbooks mit zusätzlichen Informationen und Navigationshilfen sowie ggf. mit zusätzlichen Funktionalitäten wie Adaption, Benutzerverwaltung und Annotationsmöglichkeiten ausgestattet. Hyperbooks arbeiten im Internet, wo ihre WWW-Seiten mittels dafür geeigneter Browser (moderne Browser der Firmen Microsoft, Netscape, usw.) betrachtet werden können.

Damit in einem Hyperbook die visuelle Trennung von Hypermedia-Dokument und zusätzlichen Informationen und Funktionen eindeutig ist, wird die generierte WWW-Seite aus mindestens zwei Fenstern im Browserdisplay aufgebaut. Die Fenster werden als HTML-Frames [93] realisiert. In einem Frame wird das Hypermedia-Dokument angezeigt. Ein weiterer Frame enthält die Navigationshilfen. Ein möglicher und standardmäßig auch verwendeter dritter Frame stellt die zusätzlichen Funktionen eines Hyperbooks wie z.B. das Verbergen der Navigationshilfen zur Verfügung. Der Zusammenhang der Frames einer WWW-Seite wird durch ein Frameset beschrieben, das in der WWW-Seite ausgedrückt wird. Die URI des Framesets entspricht der URI des Konzepts im Hyperbook, welche das angeforderte Hypermedia-Dokument beschreibt.

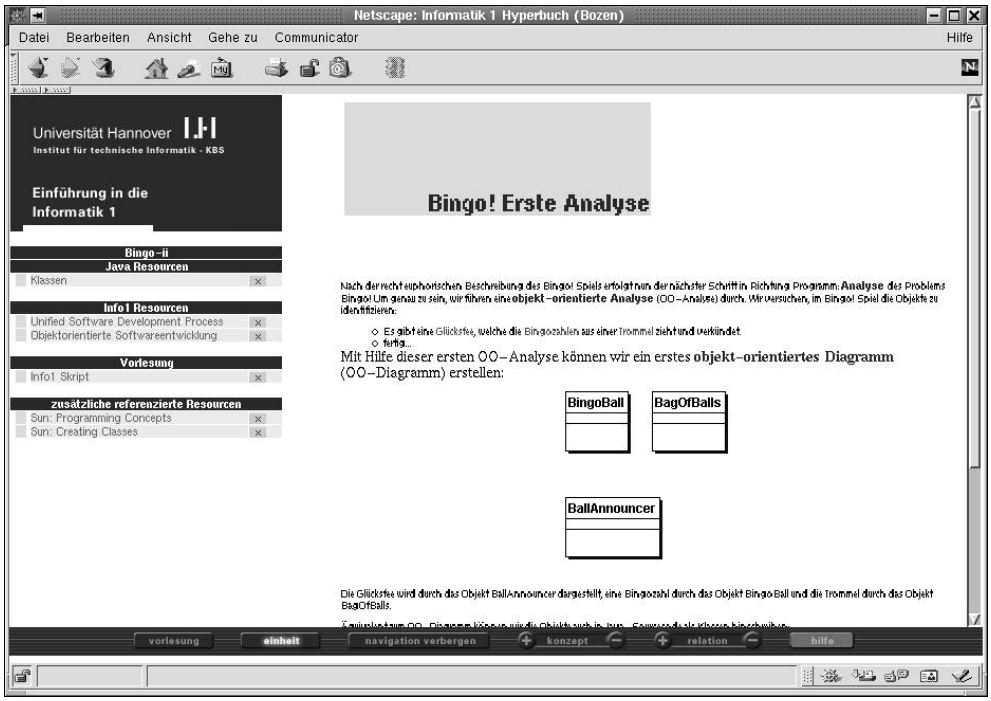

Abbildung 6.1: Aufbau einer WWW-Seite des GdI 1 Hyperbooks

Im Frameset werden die einzelnen Frames deklariert und durch ihre URI beschrieben. Die Frames werden dann aufgrund der zugeordneten URIs geladen und im Browser dargestellt.

Eine WWW-Seite des Hyperbooks besteht also immer aus dem sie beschreibenden Frameset und den zugehörigen mindestens zwei Frames (Frame des Hypermedia-Dokuments und Frame der Navigationselemente, evtl. ein dritter Frame mit zusätzlicher Funktionalität). In Abbildung 6.1 wird eine der generierten WWW-Seiten gezeigt. Der rechte Frame enthält neben der URI des Hypermedia-Dokuments die Anweisung an den Browser, das Hypermedia-Dokument aus dem Internet unter der gegebenen URI zu laden und im Frame anzuzeigen. Im unteren Frame werden zusätzliche Funktionalitäten dargestellt, beispielsweise die “navigation verbergen”-Funktion aus Abbildung 6.1, die das Hypermedia-Dokument im ganzen Browserdisplay darstellt. Desweiteren sind auch Funktionen zum Hinzufügen oder Löschen von Hypermedia-Dokumenten und Relationen zum Hyperbook möglich.

Der linke Frame stellt die Links dieses Hypermedia-Dokuments zu anderen Hypermedia-Dokumenten des Hyperbooks dar. Die Links basieren auf den Relationen, die das das Hypermedia-Dokument beschreibende Konzept zu anderen Konzepten hat. Die Links werden nach den Namen ihrer Relationen gruppiert, so daß alle Links mit demselben Relationsnamen in einer Gruppe angezeigt werden. Neben der URI des Hypermedia-Dokuments bestehen die Links aus einem Namen und optional einer kurze Zusammenfassung des referenzierten Hypermedia-Dokuments. Alle dargestellten Informationen sind im Repräsentationsmodell enthalten oder werden zur Laufzeit auf

dessen Basis erstellt.

Neben den im Repräsentationsmodell enthaltenen Relationen können durch zusätzliche Module des KBS Hyperbook Systems, z.B. durch das Adaptionsmodul, generierte Relationen als Links im linken Frame dargestellt werden. Das Adaptionsmodul fügt Links hinzu, die dem Benutzer helfen sollen, ein Themengebiet zu erlernen. Ebenso ist es möglich, die existierenden Relationen aus dem Repräsentationsmodell als Links mit zusätzlichen, von anderen Modulen generierten Informationen zu versehen. So versieht das Adaptionsmodul die Links etwa mit einer farbigen Markierung. Die farbige Markierung im Stil der Ampelfarben stellt dar, ob ein Link durch einen Benutzer verfolgt werden kann (grüne Farbe), ob es für den Benutzer evtl. zu schwierig ist, dem Link zu folgen (gelbe Farbe) oder ob es für den Benutzer nicht ratsam ist, dem Link zu folgen (rote Farbe) [55].

Die Layouts der Frames werden durch das KBS Hyperbook System beeinflußt. Das Hypermedia-Dokument spezifiziert sein eigenes Layout, so daß das Layout dieses Frames von Hyperbook nicht berücksichtigt wird. Die Layouts für den Frame der Navigationshilfen und den Frame der zusätzlichen Funktionalitäten werden durch Konfigurationsdateien spezifiziert. Eine genauere Erläuterung dieses Sachverhalts findet sich im folgenden Kapitel 6.3.3 über das Modul Visualisierung. Reichen diese Anpassungen dem Autoren nicht aus, kann das zur Generierung der WWW-Seite eingesetzte Modul Visualisierung durch ein anderes ausgetauscht werden, das das gewünschte Layout erzeugt.

Das Layout der Frames für die zusätzlichen Funktionalitäten und die Navigation wird in der aktuellen Implementierung des KBS Hyperbook Systems uniform gehalten. Das Layout ändert sich für ein Hyperbook unabhängig vom dargestellten Konzept nicht. Auf diese Weise wird erreicht, daß nicht das Hyperbook-System für den Benutzer in den Vordergrund rückt, sondern der Fokus des Benutzers auf den Inhalt des Hypermedia-Dokuments gelenkt wird.

Die generierte WWW-Seite verdeutlicht durch ihren Aufbau aus mindestens drei voneinander getrennt und uniform dargestellten Frames, daß die Inhalte der Hypermedia-Dokumente getrennt von den sie beschreibenden Daten der Konzepte und Relationen der Hyperbooks existieren. Es wird also auch visuell der Inhalt der Dokumente von den zugehörigen Navigationsstrukturen getrennt.

6.2 Datenstrukturen im KBS Hyperbook System

Dieses Kapitel beschreibt die Datenstrukturen, die das KBS Hyperbook System nutzt. Nach einer Erläuterung der Modellierungssprache O-Telos wird die Abbildung der O-Telos Modelle in Java-Objekte beschrieben. Anhand eines Beispiels wird der Vorgang der Abbildung verdeutlicht.

6.2.1 Kurze Vorstellung der Modellierungssprache O-Telos

Das KBS Hyperbook System verwendet Java-Klassen, um die Daten der Repräsentationsmodelle darstellen, speichern und verarbeiten zu können (vergleiche auch die Beschreibung des Datenbanksystems ObjectStore in Kapitel 5.2.3). Die Repräsentationsmodelle werden in der Sprache O-Telos [71, 70] erstellt, die eine Framedarstellung aller Objekte der Modelle verwendet. Die Frames stellen textuell die Klassen, Objekte und weiteren Konstrukte der Sprache O-Telos dar. In den folgenden Beispielen werden einige O-Telos Frames kurz vorgestellt.

```
Class Concept with
  attribute
  name:     String;
  abstract: String;
  id:       String
end
```

Beispiel 6.2.1: Quelltext einer Framedarstellung in O-Telos

Der O-Telos Frame im Beispiel 6.2.1 stellt die Deklaration einer Klasse `Concept` dar. Durch das Schlüsselwort `Class` zu Beginn des Frames wird dieser als Klasse deklariert. Die Deklaration der Attribute dieser Klasse wird durch das Schlüsselwort `with` eingeleitet, wobei das darauf folgende Schlüsselwort `attribute` die folgenden Konstrukte als Attribute und nicht als Ausprägungen von Attributen kennzeichnet. Die Klasse hat drei Attribute `name`, `abstract` und `id` je vom Datentyp `String`. Durch den uniformen Datentyp `String` geht aus der Framedarstellung die eigentliche Bedeutung der Attribute nicht hervor, da diese für das KBS Hyperbook System nicht wichtig ist. Während die Bedeutung der Attribute für den menschlichen Leser durch deren Namen verdeutlicht wird, benutzt das KBS Hyperbook System die Namen der Attribute zum Auffinden der Werte. Das Attribut `name` beinhaltet den anzuzeigenden Namen des jeweiligen Konzepts, `abstract` stellt eine kurze Zusammenfassung des Konzepts dar, und `id` beschreibt die URI der durch das Konzept repräsentierten Ressource bzw. des Hypermedia-Dokuments. Das Schlüsselwort `end` am Ende des Frames schließt die Deklaration des Frames.

Die im Beispiel 6.2.2 dargestellten O-Telos Frames ergänzen den Frame aus dem Beispiel 6.2.1 um weitere Frames für die Klasse `Vorlesung` und die Instanz `Vorlesung_Bozen`. Die Frames sind ein Teil des Repräsentationsmodells aus Kapitel 4.4, das zur Erzeugung der in Abbildung 4.11 gezeigten WWW-Seite der "Vorlesung Informatik 1 in Bozen" dient. Der Frame, der die Klasse `Vorlesung` deklariert, verwendet zur Darstellung der Vererbungsbeziehung das Schlüsselwort `isA`. Die Klasse `Vorlesung` wird somit als Subklasse von `Concept` deklariert.

Der Frame `Vorlesung_Bozen` wird durch das Schlüsselwort `Individual` als Instanz deklariert. Die Klasse der Instanz, in diesem Fall `Vorlesung`, wird nach dem Schlüsselwort `in` angegeben. Das Schlüsselwort `with` leitet die Attributdeklaration und die Attributausprägung ein. Der folgende Name

```
Class Concept with
  attribute
    name:     String;
    abstract: String;
    id:       String
end

Class Vorlesung isA Concept
end

Individual Vorlesung_Bozen in Vorlesung with
  name
    n1: "Vorlesung Informatik I in Bozen"
  id
    i1: "http://www.kbs.uni-hannover.de/bozen/
                                          vorlesung.html"
end
```

Beispiel 6.2.2: Beispiel der Vererbung und Instanzierung in O-Telos

`n1` des Attributs `name` ordnet den String `“Vorlesung Informatik I in Bozen”` als Ausprägung des Attributs diesem zu. Ebenso wird der String `“http://www.kbs.uni-hannover.de/bozen/vorlesung.html”` unter dem Namen `i1` dem Attribut `id` als dessen Ausprägung zugeordnet.

Zur Verdeutlichung des beschriebenen Sachverhalts zeigt Abbildung 6.2 zeigt die im Beispiel 6.2.2 als Frames dargestellten O-Telos Objekte in der grafischen UML Notation.

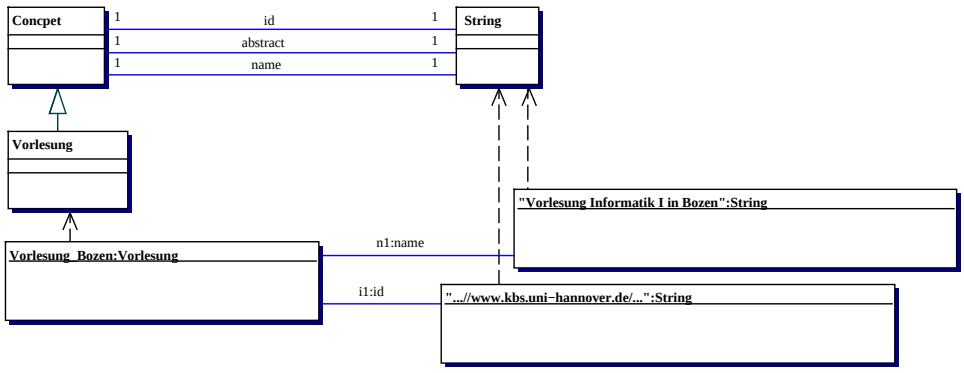

Abbildung 6.2: Grafische Notation der Frames aus Beispiel 6.2.2

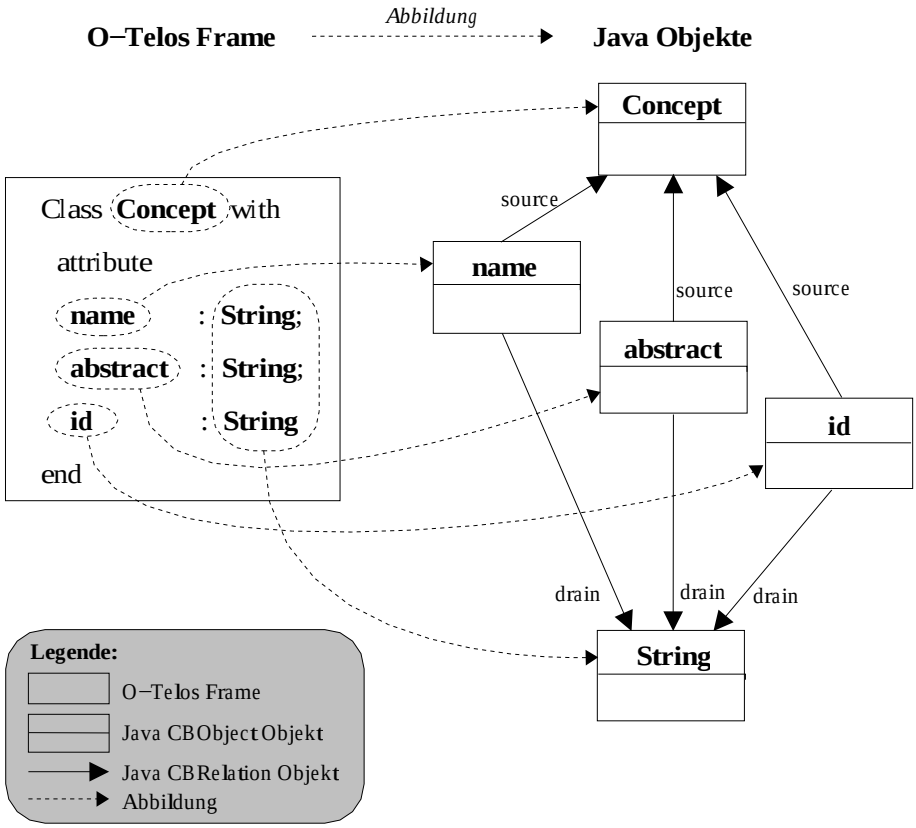

Abbildung 6.3: Abbildung der O-Telos Frames in Java Objekte

6.2.2 Abbildung von O-Telos in Java Objekte

Die Repräsentationsmodelle werden dem KBS Hyperbook System als in Textdateien deklarierte O-Telos Frames zugeführt. Das System lädt die Textdateien und bildet die O-Telos Frames auf Java-Objekte ab. Diese Abbildung basiert auf der Framedarstellung von O-Telos [68], wobei eine Abbildung der Darstellung von O-Telos Frames auf Java-Objekte beschrieben wird.

Die Abbildung erstellt für jeden Frame ein Java-Objekt. Ebenso wird für jedes Attribut eines Frames ein weiteres Java-Objekt erstellt, wobei diese über entsprechende Relationen mit dem Java-Objekt des Frames verbunden sind. Die Grafik 6.3 stellt die Abbildung grafisch dar. Der durch das Repräsentationsmodell dargestellte Sachverhalt ist für die Abbildung ohne Bedeutung.

Alle Mechanismen der Framedarstellung wie Attributierung der Klassen, Vererbung und Instanzierung werden in der Konvertierung berücksichtigt und erhalten. Java erlaubt die in O-Telos vorhandenen Mechanismen der Mehrfachvererbung und Instanzierung von Objekten [87] nicht. Sie bleiben in der Abbildung von in O-Telos formulierten Modellen in Java-Modelle trotz der Einschränkungen der Sprache Java (siehe Kapitel 5.2.2) erhalten [62], da sie in der Tokenebene explizit modelliert werden. Sie

stehen somit in der Modellierung uneingeschränkt zur Verfügung, wobei eine mögliche spätere Konvertierung unberücksichtigt bleiben kann.

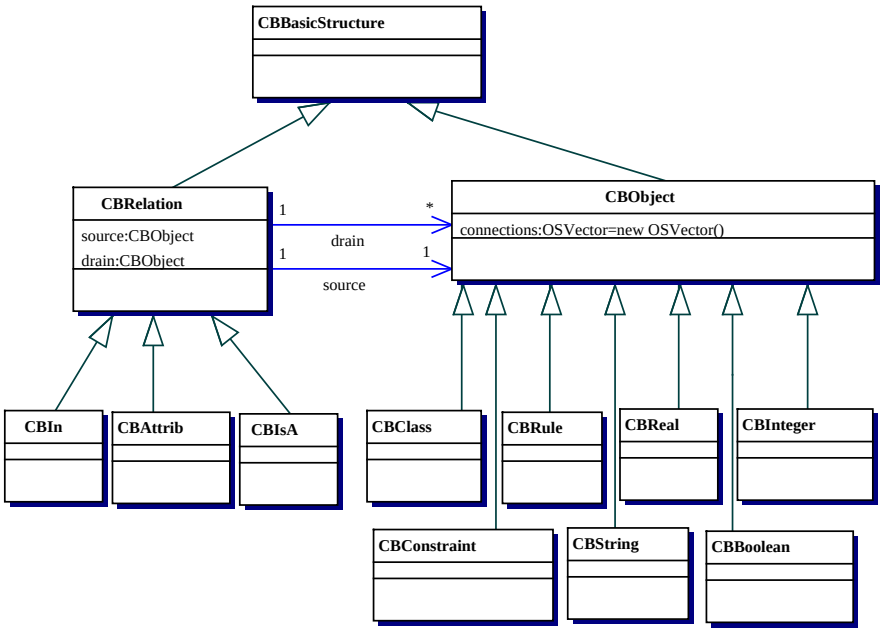

Abbildung 6.4: Die Java-Klassen zur Abbildung der O-Telos Frames

Die Vererbungshierarchie der verwendeten Java-Klassen wird in Abbildung 6.4 wiedergegeben. Die Klasse `CBBasicStructure` stellt das Basisdatenelement für die Konvertierung dar. Die Klasse `CBBasicStructure` ist Superklasse der Klassen `CBRelation` und `CBObject`. Die Klasse `CBRelation` deklariert zwei Attribute `source` und `drain` jeweils vom Typ `CBObject`. Die Klasse `CBObject` deklariert ein Attribut `connections` vom Typ `OSVector`. Die Einteilung der O-Telos Objekte in die Klasse der Objekte und die der Beziehungen der Objekte verdeutlicht, daß Beziehungen durch eigene Objekte dargestellt werden. Als Beziehungen von O-Telos Objekten werden hier nur die drei Objekt-Beziehungen Attribut, Instanz und Vererbung verstanden. Alle weiteren möglichen Beziehungen werden in O-Telos durch explizite Framedarstellungen modelliert.

Die Klasse `CBRelation` und deren Subklassen `CBAttrib`, `CBIsA` und `CBIn` dienen der Darstellung der (1:1)-Beziehung von O-Telos Frame und dem zugehörigen Attribut. Ein `CBRelation` Objekt ordnet einem Frame Objekt ein anderes als Attribut deklariertes Objekt zu. Das `source` Attribut des `CBRelation` Objekts enthält das Frame Objekt. Das `drain` Attribut desselben `CBRelation` Objekts enthält das Attribut Objekt. Indem auch die O-Telos Vererbungs- und Instanzierungsmechanismen `IsA` und `In` durch `CBIsA` und `CBIn` Klassen dargestellt werden, werden diese Mechanismen von der zugehörigen Funktionalität der Vererbung und Instanzierung gelöst. Deren Funktionalität ist im KBS Hyperbook System implementiert. Auf diese Weise werden die in Java nicht möglichen Funktionalitäten der Mehrfachvererbung

und Meta-Klassen-Bildung von O-Telos aufgelöst und in Java-Objekten dargestellt.

Die Klasse `CBObject` und deren Subklassen bilden die unterschiedlichen, in O-Telos möglichen und für die Hyperbooks notwendigen, Frame-Typen ab, z.B. als Class, Rule oder Constraint Typen. Die `CBClass` Klasse stellt die O-Telos Frames an sich dar, während die Klassen `CBString`, `CBReal`, `CBInteger`, `CBBoolean`, `CBConstraint` und `CBRule` für die Darstellung der Attribute von Frames verwendet werden.

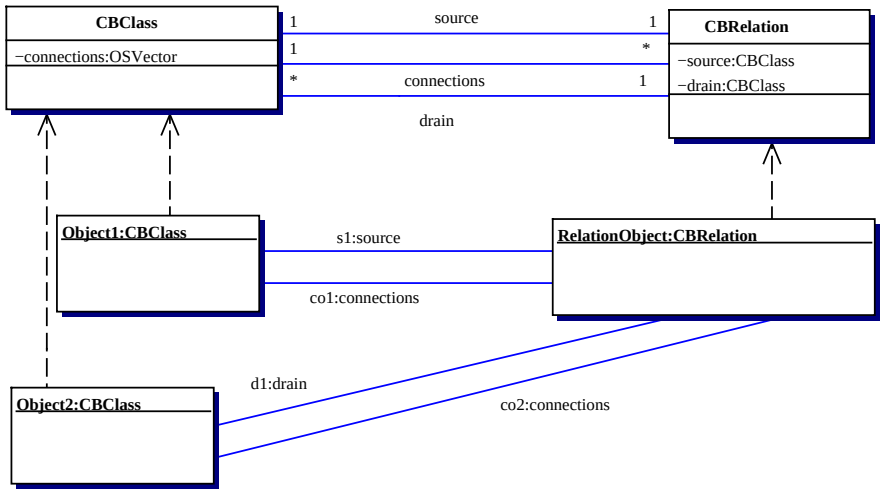

Abbildung 6.5: Beispielhafte Darstellung der Beziehungen zwischen zwei Objekten

Beziehungen zwischen den als Java-Objekte dargestellen O-Telos-Objekten werden im KBS Hyperbook System immer durch die Speicherung der Beziehungen (Instanzen von `CBRelation`) in den Attributen `connections` der beiden beteiligten Objekte und die Speicherung der Objekte in den Attributen `source` und `drain` der Beziehung repräsentiert. Abbildung 6.5 zeigt, daß `CBClass` durch das eigene Attribut `connections` und die Attribute `source` und `drain` der Klasse `CBRelation` mit dieser Klasse verbunden ist. Außerdem sind zwei `CBClass` Instanzen `Object1` und `Object2` durch eine Instanz `RelationObject` der Klasse `CBRelation` miteinander verbunden. Deutlich wird in Abbildung 6.5 die Verwendung der Attribute `source` und `drain` der Klasse `CBRelation`, die je eines der Objekte aufnehmen. Die Attribute `connections` der Objekte `Object1` und `Object2` nehmen beide dasselbe `RelationObject` auf.

Diese redundante Darstellung, wie sie auch Abbildung 6.5 zeigt, wird zur beschleunigten Verarbeitung der Objekte in Hyperbooks herangezogen. Wenn z.B. das Attribut `connections` nicht verwendet würde, müßten zur Bestimmung der Beziehungen zwischen Objekten erst alle zugehörigen Beziehungen aus dem Objektpool herausgesucht werden. Dieser Schritt entfällt durch den Einsatz des Attributs `conncections`, in dem die zugehörigen Beziehungen gespeichert sind. Diese Redundanz hat den Nachteil des erhöhten Speicherplatzbedarfs, der aber bei modernen Rechnern ver-

nachlässigbar ist. Werden andererseits die Attribute `source` und `drain` der Klasse `CBRelation` nicht berücksichtigt, muß das zu einer Beziehung und einem Objekt gehörende zweite Objekt aus dem Objektpool herausgesucht werden. Durch die Verwendung der beiden Attribute entfällt das Heraussuchen des Objekts aus dem Objektpool, wodurch das KBS Hyperbook System zwar mehr Speicherplatz benötigt, aber auch schneller arbeitet. Natürlich muß das System die Datenkonsistenz bei Modifizierung der Modelle gewährleisten.

6.2.3 Die Abbildung der Objekte am Beispiel einer Vorlesung

In der bisherigen Diskussion wurde darauf eingegangen, wie Objekte und deren Beziehungen durch die Java-Klassen dargestellt werden. Eine Instanzierung wird z.B. durch eine entsprechende Beziehung zwischen den zwei beteiligten Objekten geschaffen. Die Instanzierungssemantik besagt, daß eine Instanz der Klasse ein in dieser Klasse deklariertes Attribut ausprägt. Diese Ausprägung wird wiederum durch eine Beziehung dargestellt, die neben den beteiligten beiden Objekten auch beinhalten muß, welche Attributdeklaration der Klasse zur Ausprägung des Attributs verwendet wird. Die Attributdeklaration wird durch ihren Namen beschrieben: in einem O-Telos Frame folgt der Attributsdeklarationsname nach dem Schlüsselwort `with` (siehe z.B. den eine Instanz darstellenden Frame `Vorlesung_Bozen` im Beispiel 6.2.2). Die Klasse `CBRelation` verwendet zur Darstellung der ausgeprägten Attributsdeklaration ein Attribut `category`. Im Attribut `category` sind die Attribute festgehalten, die durch die Beziehung und die zugehörigen Objekte ausgeprägt werden. In `category` kann mehr als ein Attribut enthalten sein, wenn die Instanzierung z.B. im Rahmen einer Meta-Klasse zu Klasse zu Objekt Beziehung über mehrere Instanzebenen hinweg vorgenommen wird.

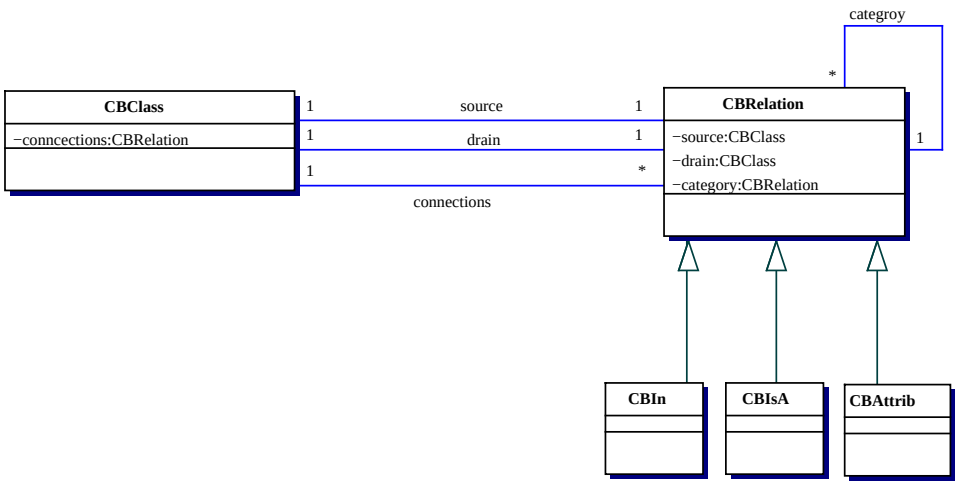

Abbildung 6.6: Vereinfachtes Klassendiagramm von `CBClass` und `CBRelation` und deren Subklassen

In Abbildung 6.6 werden noch einmal zusammenfassend zwei zur Abbildung der O-Telos Frames verwendeten Java-Klassen `CBClass` und `CBRelation` mitsamt den zugehörigen Attributen vereinfacht dargestellt. `CBClass` deklariert ein Attribut `connections`, das je `CBClass` Instanz genau einmal existiert, aber mehrere `CBRelation` Instanzen aufnehmen kann. Die `CBRelation` Klasse deklariert zwei Attribute `source` und `drain` jeweils vom Typ `CBClass`. Alle drei Attribute werden zur redundanten Darstellung von O-Telos Objekten und deren Beziehungen zueinander verwendet. Die Klasse `CBRelation` deklariert ein weiteres Attribut namens `category`, um die Namen der Attributdeklarationen aufnehmen zu können. Als Subklassen von `CBRelation` werden in Abbildung 6.6 die Klassen `CBIn` zur Darstellung der Instanzbeziehung, `CBIsA` zur Darstellung der Vererbungsbeziehung und `CBAttrib` zur Darstellung der Attributbeziehungen verwendet.

```
Individual Info1 in Vorlesungsgruppe with
  name n1: "Vorlesungsgruppe Informatik I"
  id   i1: "http://www.kbs.uni-hannover.de/bozen
                                        /info1_gruppe.html"
end

Individual Vorlesung_Bozen in Vorlesung with
  name n1: "Vorlesung Informatik I in Bozen"
  id   i1:
    "http://www.kbs.uni-hannover.de/bozen/vorlesung.html"
end

Individual Vorlesung_Hannover in Vorlesung  with
  name n1: "Vorlesung Informatik I in Hannover"
  id   i1: "http://www.kbs.uni-hannover.de/bozen
                                  /vorlesung_hannover.html"
end

Individual Info1_Vorlesungen
  in Vorlesungsgruppe_beinhaltet_Vorlesungen with
  source s1: Info1
  drain  d1: Vorlesung_Bozen;
         d2: Vorlesung_Hannover
end
```

Beispiel 6.2.3: Eine Vorlesungsgruppe besteht aus mehreren Vorlesungen, ausgedrückt in O-Telos Frames

Das folgende einfache Beispiel zeigt, wie O-Telos Frames in Java-Objekte im KBS Hyperbook System abgebildet werden.

Im Beispiel in Beispiel 6.2.3 wird der Sachverhalt ausgedrückt, daß es in der Vorlesungsgruppe Info1 zwei Vorlesungen, je eine in Hannover und Bozen gibt: Es gibt zwei Instanzen `Vorlesung_Bozen` und `Vorlesung_Hannover` der Klasse `Vorlesung`. Außerdem gibt es eine Instanz `Info1` der Klasse

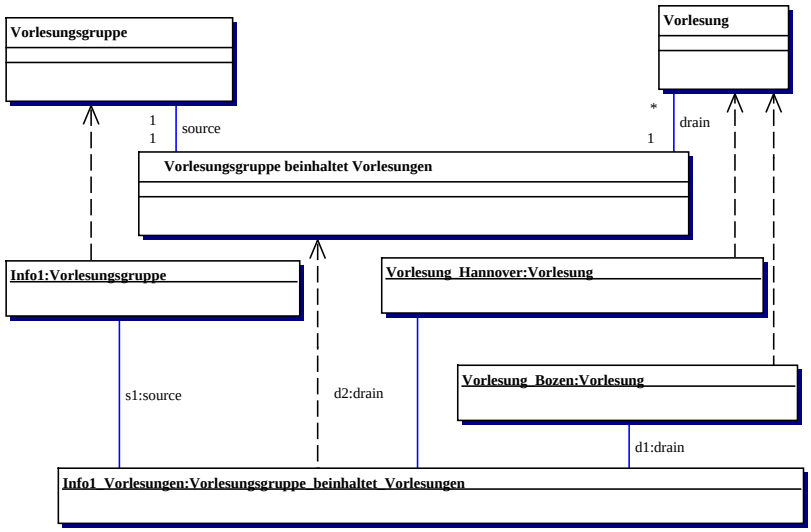

Abbildung 6.7: Klassendiagramm zum Quelltext in Beispiel 6.2.3

`Vorlesungsgruppe`, die über die Instanz `Info1_Vorlesungen` der Relation `Vorlesungsgruppe_beinhaltet_Vorlesungen` die beiden Vorlesungen `Vorlesung_Bozen` und `Vorlesung_Hannover` mit der Vorlesungsgruppe `Info1` in Beziehung setzt.

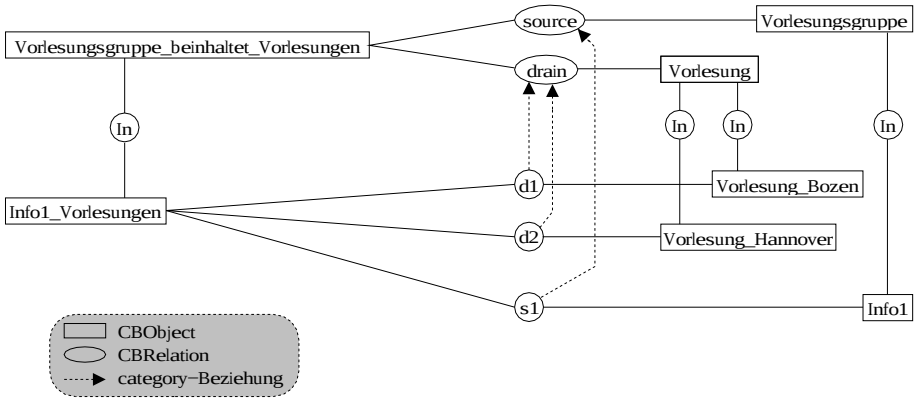

Abbildung 6.8: Die Java-Objekte zum Quelltext in Beispiel 6.2.3

In Abbildung 6.7 wird der durch die O-Telos Frames aus dem Beispiel 6.2.3 beschriebene Sachverhalt in UML Notation grafisch dargestellt. In der Abbildung 6.7 wird aus Gründen der Übersichtlichkeit auf die Darstellungen der Attribute `name` und `id` vom Typ String sowie auf deren Ausprägungen verzichtet. Die Ausprägungen stellen die Werte dieser Attribute aus dem Quelltext in Beispiel 6.2.3 dar.

Durch die Abbildung der O-Telos-Frames aus dem Quelltext in Beispiel 6.2.3 entstehen eine Reihe von Java-Objekten. Diese Objekte und ihre Zusammenhänge werden in

Abbildung 6.8 wiedergegeben. Die O-Telos Frames werden auf Instanzen der Klassen `CBObject` nebst zugehörigen Subklassen abgebildet. Die O-Telos Attribute werden auf Instanzen der Klassen `CBRelation` bzw. deren Subklassen abgebildet. In Abbildung 6.8 sind die dargestellten Instanzbeziehungen (beschrieben durch "In" in einem Kreis) Instanzen der Klasse `CBIn`. Die Attribute `source`, `drain`, `s1` (als Instanz von `source` durch das Attribut zugehörige `category` gekennzeichnet), `d1` und `d2` (beide als Instanzen von `drain` durch die zugehörigen `category` Attribute gekennzeichnet) werden als Instanzen der Klasse `CBAttrib` dargestellt.

In diesem Beispiel wird deutlich, wie durch den Einsatz der Abbildung der O-Telos Framedarstellung auf Java-Objekte die Bedeutung des Repräsentationsmodells aufgelöst und dessen Konstrukte durch wenige Datentypen vereinheitlicht beschrieben werden. Aufbauend auf den Konzepten `Concept` und `Relation` kann ein Repräsentationsmodell eine große Menge an Inhalten darstellen. Die diversen möglichen Repräsentationsmodelle können durch das KBS Hyperbook System verarbeitet, dargestellt und gespeichert werden. Diese Funktionalitäten werden durch die Vereinheitlichung der Darstellung der Modelle im System erreicht.

6.3 Die Module des KBS Hyperbook Systems

Das KBS Hyperbook System besteht aus verschiedenen Modulen, die in der Sprache Java implementiert sind. Der in dieser Arbeit verwendete Modulbegriff basiert auf den Ausführungen in [99] und [10] zum Modulbegriff im objekt-orientierten Software Engineering. Demnach ist ein Modul ein Programm, dessen Funktionalität durch eine definierte Schnittstelle nach außen festgelegt und zugänglich ist. Die Implementierung des Moduls ist von außerhalb der Grenzen des Moduls nicht sichtbar. Ein Modul kann sowohl als allein stehendes Programm ausgeführt oder durch andere Module aufgerufen werden. Module können miteinander über Schnittstellen kommunizieren. In diesem Fall werden der Datenaustausch und der Funktionsaufruf der Module untereinander als Kommunikation bezeichnet. Ein Modul kann auch Teil eines anderen Moduls sein, wobei es ausschließlich über seine Schnittstelle verwendet wird.

In der Sprache Java besteht ein Modul aus mindestens einer Klasse (siehe auch die Java Package-Deklaration im JDK API [106]). Durch den Einsatz von Modulen wird die Wiederverwendbarkeit des Systems, seiner Teile und Funktionen erhöht, denn die Module lassen sich aufgrund ihrer festen Schnittstellenspezifikationen auch im Kontext anderer Systeme einsetzen. Ebenso wird durch die Verwendung von Modulen die Erweiterbarkeit des Systems unterstützt, denn die Funktionalität der Module kann durch einen Austausch derselben bzw. die Integration neuer Module in das System erweitert und an die jeweilige Aufgabenstellung angepaßt werden. Die Schnittstellenspezifikationen des KBS Hyperbook Systems finden sich im Anhang A.4.

Die Module des KBS Hyperbook Systems umfassen einen Parser zum Wandeln von O-Telos in Java-Klassen, ein Ladeprogramm zum Laden der in O-Telos formulierte Repräsentationsmodelle in die ObjectStore (OS) Datenbank, ein Programm zum Ab-

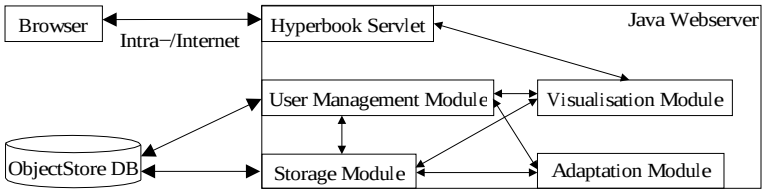

Abbildung 6.9: Die Module und die Datenflüsse im System

speichern des Datenbankinhalts in einer O-Telos Textdatei, ein Datenbank Abfrage- und Verwaltungsmodul, ein Konvertierungsmodul zur Erzeugung von HTML-Seiten aus den Daten der Datenbank und das Servlet zur Bereitstellung der Hyperbook-Funktionalitäten. Abbildung 6.9 stellt als Übersicht die einzelnen Module und deren Datenflüsse dar. Die Module werden durch das Servlet zu einem Hyperbook im Rahmen des Java-Webservers kombiniert. Zusätzliche Funktionalitäten wie das Adaptions- oder Benutzerverwaltungsmodul werden über entsprechende Schnittstellen der Module Visualisierung und Storage in das System integriert. Die Kommunikation des KBS Hyperbook Systems mit den umgebenden Systemen beschränkt sich auf die folgenden beiden Zugriffsweisen: Auf die Datenbank hat nur das Storage Modul Zugriff. Ein Browser ist mit dem KBS Hyperbook System ausschließlich über das Servlet und den Webserver via Intra-/Internet verbunden. Die anderen Module können nicht direkt via Intra-/Internet angesprochen werden.

6.3.1 Überblick über die Java-Klassen des Systems

Die Abbildung 6.10 beschreibt die Klassen der Module des KBS Hyperbook Systems. Auf die Darstellung der Methoden der Klassen wird aus Platzgründen verzichtet. Das Klassendiagramm stellt die bereits aus den vorstehenden Unterkapiteln bekannten Klassen `Hyperbook`, `VisualisationModule`, `StorageModule`, `CBOSConnectionModule`, `SmlToOs` und `OsToSML` dar. Daneben sind Klassen aufgeführt, die im System zur Unterstützung der Module verwendet werden. Zu diesen Klassen gehören `UtilPage` und `RelationPage`, die der Klasse `VisualisationModule` bei der Erstellung der Layouts von Navigations- und zusätzlichen Funktionen-Frames mittels der Konfigurationsdateien dienen. Die Klassen `FrameStruct`, `AttributeInstance`, `AttribContainer` und `AxiomCheckException` unterstützen das Modul Storage in dessen Tätigkeit. Neben den Klassen sind auch die statischen Beziehungen der Klassen, und somit auch der Module untereinander wiedergegeben.

Die in Abbildung 6.10 dargestellte Klassenhierarchie wird durch einen Parser und dessen Klassen ergänzt. Die Klassen des Parsers werden in Abbildung 6.11 beschrieben. Der Parser wird mit Hilfe des Tools Java Compiler Compiler (JavaCC) [107] unter Verwendung der Diplomarbeit von K. Hudalla [62] entwickelt. JavaCC ist ein Tool zur Generation von in Java implementierten Parsern. Es verwendet eine um Java-Methoden ergänzte Backus-Naur-Form (BNF oder EBNF) [2] des zu erstellenden Parsers als

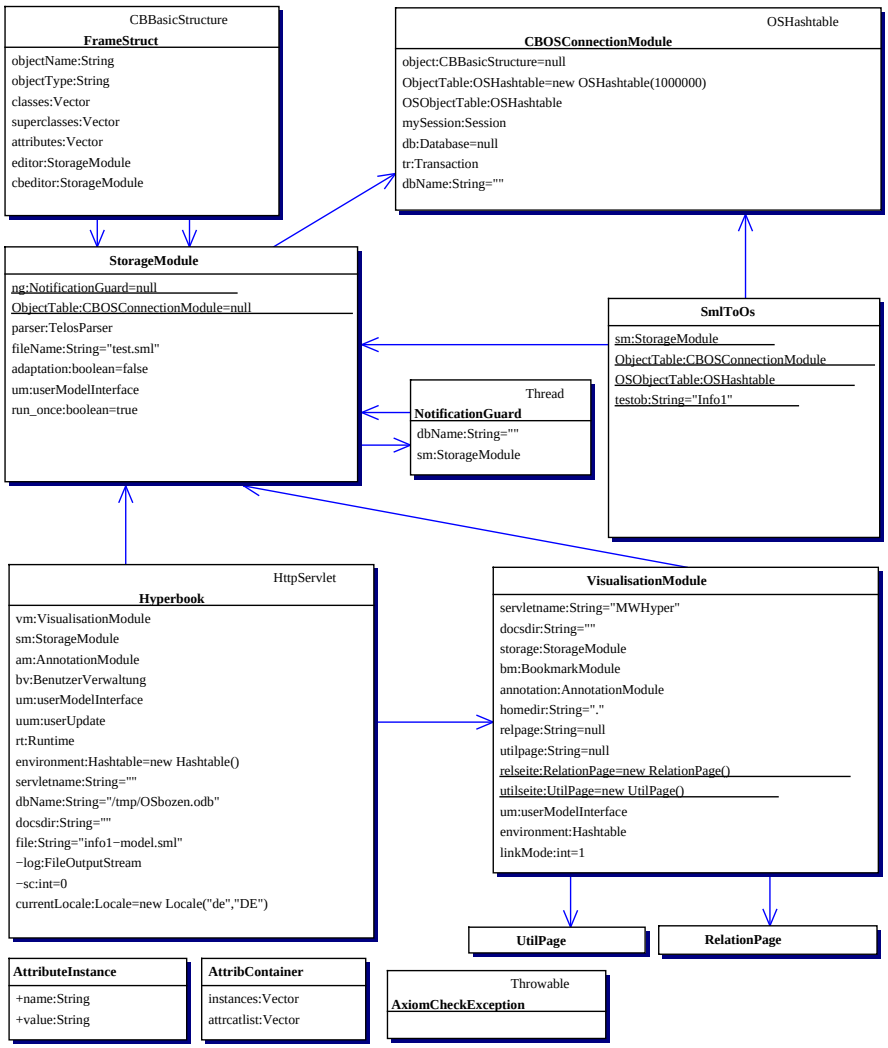

Abbildung 6.10: Die Klassen der Module des KBS Hyperbook Systems

Konfiguration.

Der Parser erzeugt aus den O-Telos Frames mittels der in Abbildung 6.16 beschriebenen Klassenhierarchie die Java-Objekte. In der Klassenhierarchie werden auch die Attribute und Methoden dargestellt, um den Unterschied zu den in der Datenbank gespeicherten Objekten zu verdeutlichen. Die zur Speicherung in der Datenbank verwendete Klassenhierarchie wird in Abbildung 6.12 wiedergegeben. Verglichen mit der Abbildung 6.16 wird deutlich, daß sich beide Hierarchien nur in dem Vorhandensein bzw. Nicht-Vorhandensein von Methoden unterscheiden. Die Methoden der Klassen in Abbildung 6.16 implementieren die Axiome der Sprache O-Telos, die in der Erstellung neuer Objekte, der Veränderung und Löschung der bestehenden Objekte zur Konsistenzwahrung eingesetzt werden. Beispielsweise implementieren sie den Vererbungs- und Instanzierungsmechanismus, aber auch die Konvertierung von O-Telos-Frames zu

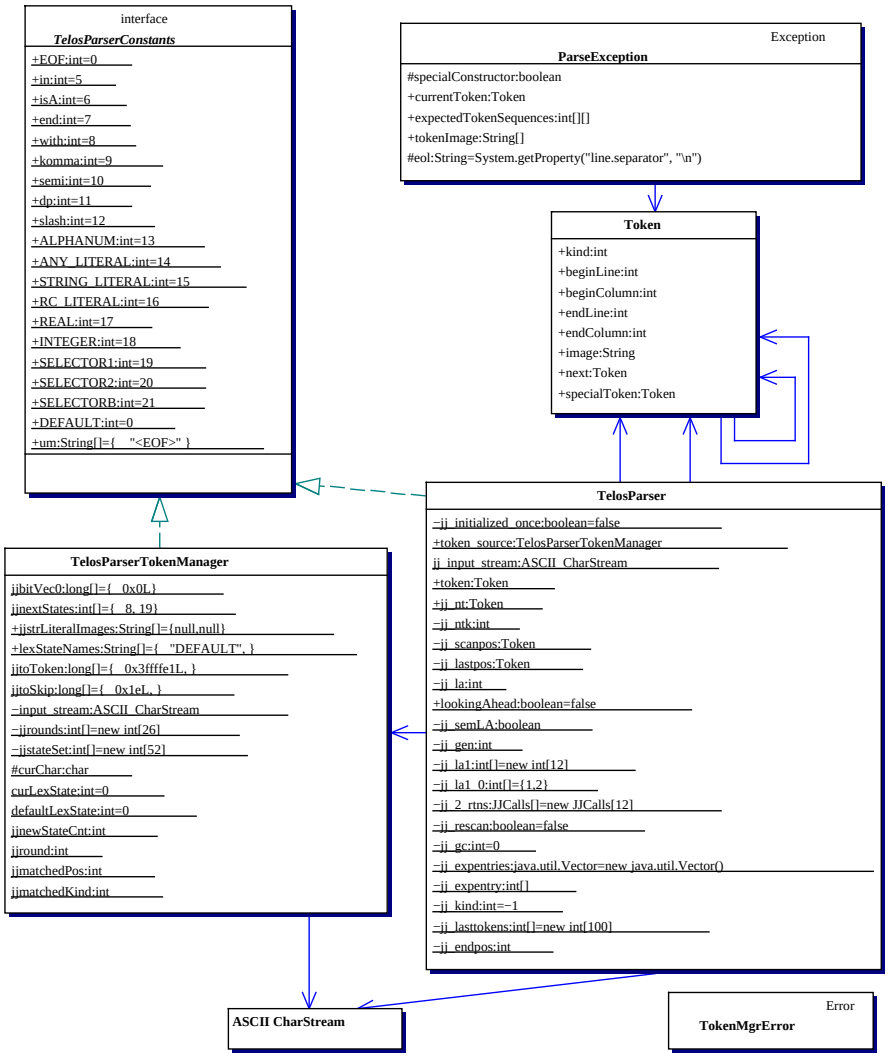

Abbildung 6.11: Die Java-Klassen des Parsers

Java-Objekten und die Konvertierung von Klassen mit Methoden in Klassen ohne Methoden und vice versa.

In Abbildung 6.13 werden die von den Modulen zur Kommunikation untereinander eingesetzen Klassen gezeigt. Auf die Darstellung der Methoden wird aus Platzgründen verzichtet. Diese Klassen werden von den Modulen über entsprechende Schnittstellen ausgetauscht und vereinfachen die Kommunikation unter den Modulen. Sie dienen als Datenträger, die die im Modul Storage gespeicherten Token zu bedeutungsvollen Objekten wie Konzepten und Relationen mit allen zugehörigen Daten zusammenfassen.

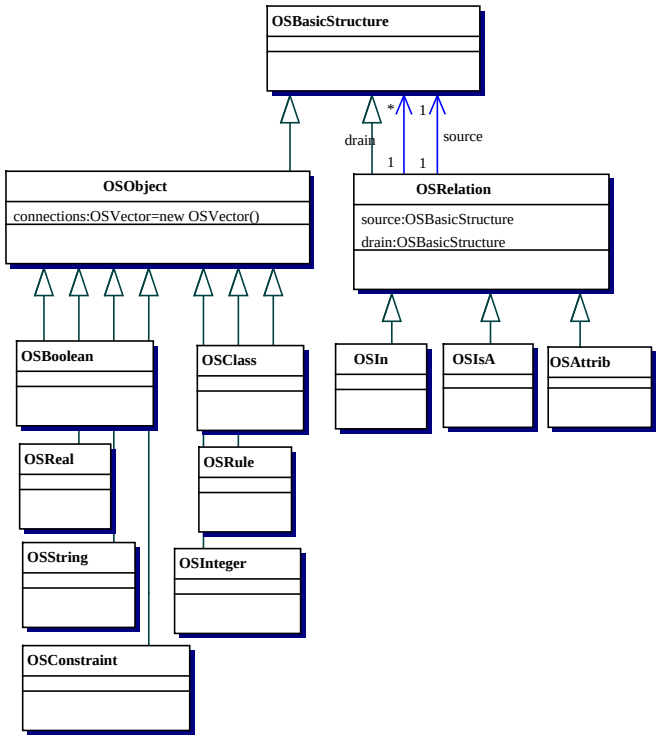

Abbildung 6.12: Die Java-Klassen der O-Telos Frames zur Speicherung in der Datenbank

6.3.2 Das Hyperbook Servlet

Das Hyperbook Servlet residiert im WWW-Server, hier dem Java WebServer. Die Aufgaben des Servlets umfassen die Initialisierung der Module des Hyperbooks, die Vermittlung von Anfragen aus dem WWW an das Modul Visualisierung und die von dort gelieferten Antworten an den WWW-Server zur Verteilung an den aufrufenden Browser.

Zur Initialisierung der Module wird dem Servlet eine Reihe von Parametern vom Webserver übergeben. Die Parameter nehmen grundsätzliche Einstellungen des Systems vor, die sich zur Laufzeit nicht ändern und daher nur einmal zu Beginn der Laufzeit notwendig sind. Es handelt sich dabei um die folgenden Parameter:

1. `Docsdir`: das Verzeichnis der statischen HTML-Dokumente des Systems; wird im Modul Visualisierung verwendet.
2. `RelationPageLayout`: Vorgabedatei für das Layout des Frames mit den Navigationshilfen und Informationen; wird im Modul Visualisierung verwendet.
3. `UtilPageLayout`: Vorgabedatei für das Layout des Frames für die zusätzlichen Funktionen; wird im Modul Visualisierung verwendet.

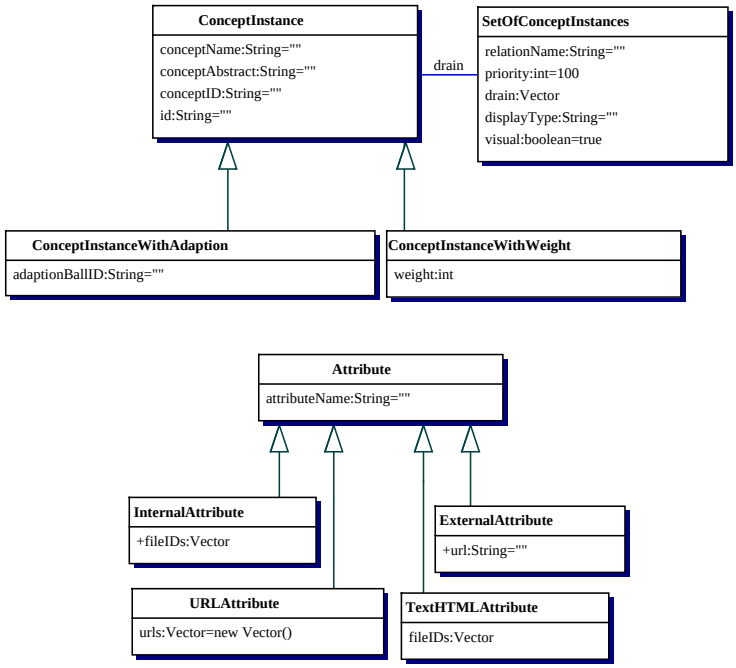

Abbildung 6.13: Die System-internen Java-Klassen zum Datenaustausch

4. `Servlethome`: das home-Verzeichnis des Servlets; wird im Modul Visualisierung verwendet.
5. `Servletname`: der aktuelle Servletname, der vom realen Servletnamen abweichen kann (das Feature Servletalias des Java WebServers); wird im Modul Visualisierung verwendet.
6. `SMLfile`: der Name der zu verwendenden Datenbank; wird im Modul Storage verwendet.
7. `Useradaptation`: bool'scher Schalter zum Ein-/Ausschalten der Adaption; wird im Modul Visualisierung zur Aktivierung des Moduls Adaptation verwendet.
8. `Userdir`: Verzeichnis zur Speicherung der Benutzerdaten; wird im Modul Adaptation verwendet.
9. `VisualisationModule`: Name der Klasse, die als Modul Visualisierung verwendet werden soll - Möglichkeit zum Einbinden von Subklassen des Moduls Visualisierung, so daß die originären Klassen nicht überschrieben werden müssen.

Die Parameter werden vom Servlet an die Module weitergegeben. Dann wird die Initialisierung der Module durchgeführt.

Das Servlet erhält vom Webserver die Anfragen in der Form des Textstrings als viertem Teil einer URI (siehe auch Kapitel 2.1). Der Textstring wird im Browser erstellt und durch Verwendung der HTTP-Get-Methode an die URI angefügt und zum Webserver übertragen. Dieser Textstring beinhaltet Variablenpaare, wobei ein Paar jeweils aus einem Variablennamen und den zugeordneten Werten besteht. Die Variablen, die das Verhalten des Systems steuern, werden im Folgenden kurz erläutert:

- DomainObject `do`: Name des Konzepts, welcher das angeforderte Hypermedia-Dokument im System repräsentiert/modelliert.
- Modus `m`: Modus, in dem das System zur Beantwortung dieser Anfrage arbeiten soll.
- User `u`: Name des Benutzers im System.
- Group `g`: Name der Gruppe im System, der der Benutzer zugeordnet ist.

Der Benutzername und die Gruppenzugehörigkeit werden zur Identifikation des Benutzers im System verwendet. Sie werden auch in der Benutzerverwaltung zur Charakterisierung der Rechte des Benutzers im System herangezogen.

Mittels des Namens des Konzepts, das das angeforderte Hypermedia-Dokument modelliert, wird die WWW-Seite mit den zum Konzept zugehörigen zusätzlichen Funktionen, Navigationshilfen und Informationen generiert. Die Variable `do` bestimmt also, welches Hypermedia-Dokument in der generierten WWW-Seite enthalten ist.

Die Variable `m` beschreibt den Modus, in dem das Hyperbook betrieben wird, um die WWW-Seite oder deren Teile zu generieren und wie das System auf die Anfrage reagiert. Durch einen eigenen Modus je Frame wird die Generation des Framesets und der Frames einer WWW-Seite gesteuert. Auch zusätzliche Funktionen wie das Hinzufügen oder Entfernen eines Konzepts oder einer Relation werden durch die Angabe des entsprechenden Modus gesteuert. Die den Funktionen zugeordneten zusätzlichen Informationen im Textstring werden dann in den entsprechenden Modulen aus diesem extrahiert. Die Verarbeitung und Steuerung des Hyperbooks durch die Modi ist im Modul Visualisierung realisiert.

6.3.3 Modul Visualisierung

Das Modul Visualisierung hat die Aufgabe, auf Anfrage des Servlet Moduls die entsprechende WWW-Seite zu erstellen und an das Servlet weiterzuleiten. Außerdem stellt dieses Modul eine Schnittstelle für weitere Module zur Verfügung. Diese Schnittstelle wird genutzt, um im Hyperbook zusätzliche Funktionalitäten zu integrieren.

Das Modul Visualisierung erhält vom Servlet die Anfrage, eine WWW-Seite zu erstellen. Aufgrund des in der Anfrage übergebenen Modus wird die Art der WWW-Seite und somit die Art der Erstellung derselben festgelegt. Es werden im Minimum

vier unterschiedliche Seitentypen erstellt: das Frameset zur Beschreibung des WWW-Seitenaufbaus, der Frame mit der URI des Hypermedia-Dokuments, der Frame mit den zusätzlichen Funktionalitäten und der Frame mit der Navigation. Das Layout des Frames, der das Hypermedia-Dokument enthält, wird durch das Dokument spezifiziert. Die Layouts des Frames für die Navigation und für zusätzliche Funktionalitäten werden durch die in den Parametern `RelationPageLayout` und `UtilPageLayout` benannten Dateien beschrieben. Beide Dateien verwenden neben den üblichen HTML-Befehlen auch zusätzliche, vom KBS Hyperbook System zur Verfügung gestellte Befehle. Die Befehle werden vom KBS Hyperbook System zur Laufzeit ausgewertet. Die Befehle entsprechen Platzhaltern für Textdaten, die vom System zur Laufzeit bestimmt werden, z.B. den aktuellen Namen des Konzepts. Eine Aufstellung dieser Befehle findet sich im Anhang A.2.

Die für die Erstellung der Frames eines Konzepts notwendigen Modellinformationen erhält das Modul Visualisierung auf Anfrage vom Modul Storage. Auf der Basis der vom Modul Storage und anderen Modulen erhaltenen Daten erstellt das Modul Visualisierung die WWW-Seiten. In der Erstellung der WWW-Seite der Navigation werden die Daten verwendet, um die Links zu generieren. Als Basis der Links des Frames mit den Navigationshilfen dienen die Beziehungen des aktuellen Konzepts und die so mit dem Konzept verbundenen Konzepte. Zusätzlich werden durch die Daten der `DisplayRelation`, von der jede Relation eine Instanz sein kann, weitere Eigenschaften der Darstellung der Beziehungen als Links festgelegt. Durch das Attribut `priority` wird die Reihenfolge der Darstellung der Beziehungen bestimmt. Das Attribut `visualisation` bestimmt, ob eine Beziehung angezeigt werden soll, während die Attribute `nameSourceDrain` bzw. `nameDrainSource` den angezeigten Namen der Beziehung deklarieren. Der angezeigte Name der Beziehung hängt davon ab, ob das aktuelle Konzept im Attribut `source` oder in der Liste der `drain` Attribute enthalten ist.

Die solchermaßen erstellten WWW-Seiten werden an das Servlet übergeben und von dort weiterverarbeitet.

Die beiden Module Visualisierung und Storage (vergleiche dessen Erläuterung im anschließenden Kapitel 6.3.5) nutzen eigene Datentypen zum Austausch von Informationen. Diese Datentypen werden als Vereinfachung genutzt, denn sie fassen die in der Datenbank gespeicherten Token (siehe Kapitel 2.4) zu bedeutungsvollen Objekte zusammen. Abbildung 6.14 beschreibt die Klassenhierarchie der Datentypen mitsamt den zugehörigen Attributen. So besteht ein Konzept beispielsweise nicht mehr aus einzelnen Objekten und Relationen, sondern wird als ein Konzept mit den Attributen `name`, `id` und ggf. `abstract` beschrieben. Die Klasse `ConceptInstance` und deren Subklassen werden zur Darstellung eines Konzepts eingesetzt.

Ein Konzept kann Beziehungen gleichen Namens zu mehreren Konzepten besitzen (neben (1:1)- auch (1:n)-Beziehungen). Diese Beziehungen werden zusammengefaßt, indem die Objekte der einzelnen Beziehungen mit den jeweils dazugehörigen Konzepten in einer gemeinsamen Datenstruktur namens `SetOfConceptInstances` gruppiert werden. Über die Schnittstelle des Moduls Visualisierung stehen diese Da-

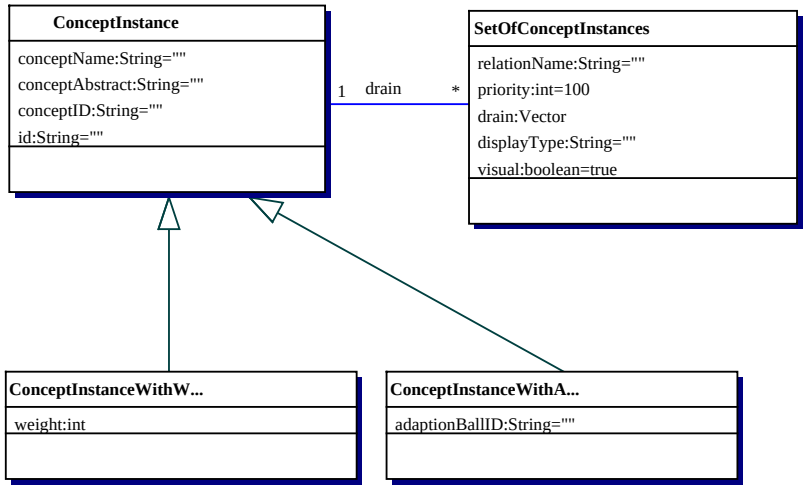

Abbildung 6.14: Die Klassenhierarchie für die intermodulare Kommunikation

tenrepräsentationen auch anderen Modulen zur Verfügung. Die Daten fließen dabei vom Modul Visualisierung zu den anderen Modulen und umgekehrt.

Das Modul Visualisierung kann durch andere Module ersetzt, bzw. durch Subklassen, die dieselben Schnittstellen implementieren, erweitert und verändert werden. So kann das Aussehen der vom System generierten WWW-Seiten an die Wünsche der Autoren der Hyperbooks angepaßt werden. Beispielsweise lassen sich einzelnen Konzepten oder Konzepttypen eigene Darstellungsweisen zuordnen. Die Modulerweiterung wird im GdI 1 Hyperbook angewendet, um die Umschaltung von Navigation für ein Konzept auf ein Inhaltsverzeichnis zu ermöglichen (siehe Kapitel 4.4).

6.3.4 Modul Adaption

Das Modul Adaption ist von N. Henze im Rahmen ihrer Dissertation [53] für das KBS Hyperbook System erstellt worden.

Das Modul Adaption hat zur Aufgabe, mittels der Annotation der Navigation und der Addition neuer Navigationselemente den Benutzer eines Hyperbooks durch dessen Informationsvielfalt zu begleiten oder führen. Die Adaption zeigt auf der Basis des Kenntnisstandes des Benutzers und seiner Ziele sinnvolle Lernschritte auf, ermittelt geeignete Übungsprojekte, schlägt eine Lesereihenfolge vor und verweist auf relevante Informationen.

Das Modul verwendet zur Beschreibung einer Domäne ein Indizierungsmodell, das auch im KBS Hyperbook System enthalten sein kann. Die Adaption der Domäneninhalte basiert dann auf der Indizierung, indem diese von einem Bayes'schen Schlußmechanismus zur Abschätzung des Wissens des Benutzers angewandt wird.

Im Rahmen des KBS Hyperbook Systems bestimmt das an das Servlet übergebene Attribut `Useradaptation`, ob das Modul Adaption aktiviert wird. Erhält das Modul Visualisierung die Anforderung, den Frame mit der Navigation zu erstellen, fragt es nach Erhalt der vom Modul Storage benötigten Daten das Modul Adaption um dessen Daten für die Erstellung des Navigationsframes. Das Modul Adaption annotiert dann die bereits existierenden Daten im Modul Visualisierung, generiert unter Zuhilfenahme des Moduls Storage aber auch neue Daten und übergibt diese dem Modul Visualisierung.

6.3.5 Modul Storage

Das Modul Storage hat die Aufgabe, die Daten der Repräsentationsmodelle zu verwalten, also sie aufzunehmen, zu speichern, auszulesen, zu modifizieren und zu löschen. Außerdem ist das Modul Storage für die Datenbankanbindung und die damit zusammenhängenden Zugriffe auf die Datenbank zuständig. Die Daten werden im Modul Storage in der Tokendarstellung gespeichert, deren Eigenschaften in Kapitel 2.4 beschrieben wurden. Entsprechend der Darstellungen in diesem Kapitel werden die Token durch Objekte und deren Relationen dargestellt. Die Token werden durch die Klasse `CBBasicFrameStructure` und deren Subklassen beschrieben. Die Subklassen beschreiben eine Klassifizierung der Objekte und Relationen gemäß der Darstellungen in Kapitel 6.2. Das Modul Storage verwendet zur effizienten Speicherung die Datenstruktur Hashtable, die sich durch geringe Auslese-Zeiten auszeichnet [47, 60]. Diese Datenstruktur ist in Java nativ implementiert. Über eine Schnittstelle stehen die Daten des Moduls anderen Modulen zur Verfügung.

Auf Anfrage eines anderen Moduls, z.B. des Moduls Visualisierung, stellt das Modul Storage die gewünschten Daten in für diese Kommunikation vorgesehenen Datenstrukturen zur Verfügung. Die Tokendarstellungen der Objekte werden in diesen Datenstrukturen zu bedeutungsvolleren Objekten zusammengefaßt. Die Objekte repräsentieren z.B. ein Konzept oder eine Beziehung mit den zugehörigen Attributen und den zugehörigen Konzeptdaten. Somit werden dem anfragenden Modul nicht einzelne Token geliefert, sondern schon entsprechend der Anfrage zusammengestellte Objekte. Auf diese Weise wird eine Reduktion der Anfragen an das Module Storage und somit des Verarbeitungsaufwands erreicht.

Das Modul verwendet seinerseits eine Java-Klasse namens `CBOSConnection-Module` zur Kapselung des Zugriffs auf die Datenbank. Zum Wechsel einer Datenbank muß nur diese Klasse ausgetauscht werden, nicht aber das ganze Modul Storage.

Im Fall der ObjectStore Datenbank wird der Hashtable des Moduls Storage in der Datenbank gespeichert. In Versuchen hat sich gezeigt, daß die Datenbank Anfragen schneller beantwortet, wenn die im Hashtable gespeicherten Java-Objekte keine Methoden besitzen. Aus diesem Grund werden die Objekte des Hashtables im Modul Storage durch die kapselnde Klasse in Objekte ohne Methoden kopiert, die dann in der Datenbank gespeichert werden.

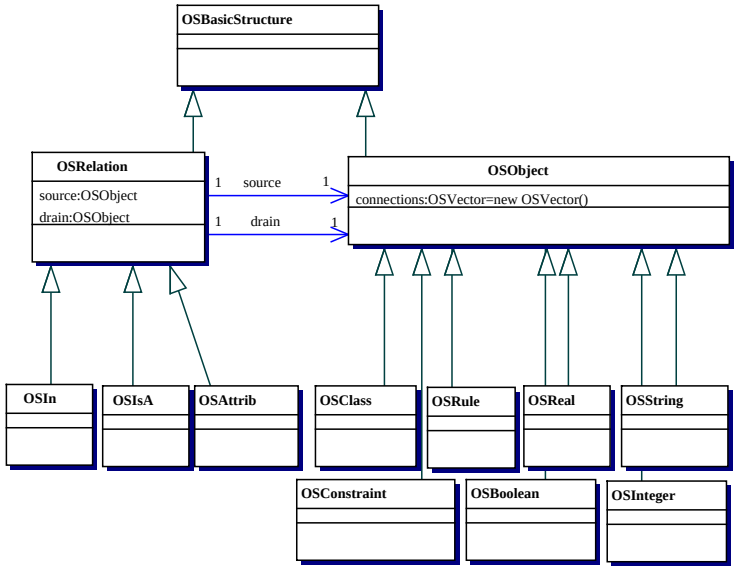

Abbildung 6.15: Die Java-Klassen zur Speicherung

Die im Hashtable des Moduls Storage gespeicherten Objekte werden in der Initialisierung des Moduls in den Hashtable geladen. Sie werden in diesem Vorgang von Objekten ohne Methoden zu Objekten mit den notwendigen Methoden konvertiert. Die Java-Klassen ohne Methoden zur Abbildung der O-Telos Frames entsprechen in ihrem Aufbau und ihrer Hierarchie den Java-Klassen mit Methoden. Diese Äquivalenz der Klassen wird auch durch einen Vergleich der Abbildung 6.15 mit Abbildung 6.4 deutlich. Die beiden Klassenhierarchien müssen äquivalent sein, denn sie sind beide das Repräsentationsmodell in Tokendarstellung. Die Methoden der Klassen sind für alle Objekte derselben Klasse gleich, weshalb deren Speicherung nicht notwendig ist. Die Methoden implementieren einerseits die Instanzierungs- und Vererbungsmechanismen des Repräsentationsmodells sowie andererseits die Axiome der Sprache O-Telos unter Berücksichtigung der Eigenschaften des Repräsentationsmodells.

Nach der Initialisierung des Moduls wird die Verbindung zur Datenbank beendet und nur wieder aufgebaut, wenn sich im Hashtable oder in der Datenbank Änderungen in den Daten ergeben. Änderungen werden vom KBS Hyperbook System und den Hyperbooks festgestellt, indem die Benachrichtigungseinrichtung der Datenbank (Notification Facility [34]) eingesetzt wird. Diese Einrichtung überwacht die in der Datenbank gespeicherten Objekte und benachrichtigt bei deren Veränderungen das Modul Storage, das zu diesem Zweck die Klasse `NotificationGuard` implementiert. Ändern sich die Objekte im Modul Storage, werden diese Veränderungen ohne Zeitverlust in der Datenbank reflektiert.

Im Rahmen des Moduls Storage wird ein Parser eingesetzt, der eine Textdatei, die O-Telos Frames enthält, einlesen und in Java-Objekte abbilden kann. Der Parser wurde mittels des Tools Java CC [107] entwickelt. Er ist in der Lage, die in Anhang A.1 wie-

dergegebene Syntax der O-Telos Frames auf Java-Objekte abzubilden. Der Parser wird genutzt, um die in O-Telos modellierten Repräsentationsmodelle durch Java-Objekte auszudrücken.

Das Modul Storage kann auch als eigenständiges Programm ausgeführt werden. Es bietet eine Reihe von Kommandozeilenparametern zur Steuerung seiner Verhaltensweisen. Haupteinsatzgebiet als Programm ist die Verifikation und der Test der Datenbank und der darin repräsentierten Modelle.

In der Kommandozeile hat das Modul die folgende Syntax:
`java StorageModule Datenbankpfad Befehl Objekt`.
Der `Datenbankpfad` enthält den Pfad, unter der die Datenbank im Datenbanksystem ObjectStore gefunden wird. Der `Befehl` beschreibt die vom Modul auszuführende Aktion und `Objekt` benennt das aktuelle Konzept, auf das sich die Befehle `Befehl` beziehen können. Das Modul implementiert die folgenden `Befehle`: `rm` löscht das in `Objekt` genannte Konzept. `tc` holt das in `Objekt` benannte Konzept als `ConceptInstance` und gibt es aus. `t` stellt alle Beziehungen des in `Objekt` genannten Konzepts als `CBRelation` Objekte dar.

6.3.6 Modul SmlToOs

Das Modul SmlToOs wird als eigenständiges Programm verwendet. Es lädt ein mittels O-Telos Frames formuliertes Repräsentationsmodell in das Datenbanksystem ObjectStore. Außerdem wird es zur Verifizierung der Funktionsweise des Datenbanksystems und des Moduls Storage eingesetzt.

Das Repräsentationsmodell wird aus einer Datei geladen. Unter Verwendung des Moduls Storage wird die Java-Objektrepräsentation des Modells erzeugt. Die Objektrepräsentation wird mittels der Klasse `CBOSConnectionModule` in einer Datenbank des Datenbanksystems gespeichert.

Der Syntax des Programmaufrufs dieses Moduls lautet:
`java SmlToOs Option/Datei Datenbankpfad Objekt`.
Das `Objekt` beschreibt hierbei das zu Testzwecken aus der Datenbank geholte und ausgegebene Objekt. Der `Datenbankpfad` enthält den Pfad, unter dem die Datenbank im Datenbanksystem ObjectStore gefunden wird. `Option/Datei` ist Platzhalter für eine von drei möglichen Werten: Der Wert `help` veranlaßt die Ausgabe einer Hilfenachricht, die die Handhabung des Programms kurz beschreibt. Die Wert `test` testet die Datenbank, indem das `Objekt` aus der in `Datenbankpfad` beschriebenen Datenbank geholt und ausgegeben wird. Alle anderen Werte der `Option/Datei` werden als Quelldatei des Repräsentationsmodells interpretiert. Dieses wird aus der Datei geladen, die es beschreibenden Java-Objekte werden erzeugt und in die in `Datenbankpfad` spezifizierte Datenbank geschrieben. Daraufhin versucht das Programm, das in `Objekt` genannte Objekt aus der im `Datenbankpfad` spezifizierten Datenbank zu laden und auszugeben. Alle auftretenden Fehler während der Laufzeit des Programms werden ausgegeben und führen zu seiner Terminierung.

6.3.7 Modul OsToSml

Das Modul SmlToOs wird wie das Modul SmlToOS als eigenständiges Programm verwendet. Es speichert die in der Datenbank OS enthaltenen Repräsentationsmodelle in Form von O-Telos Modellen in Dateien im Filesystem.

Unter Verwendung des `CBOSConnectionModuls` wird die Datenbank ausgelesen und in O-Telos Frames gewandelt. Die Frames werden dann in einer Datei gespeichert.

Die Syntax des Programmaufrufs lautet wie folgt:
`java OsToSml Datenbankpfad Datei.sml`.
Der `Datenbankpfad` ist in diesem Fall der vollständige Pfad der gewünschten Datenbank im Datenbanksystem. `Datei.sml` steht für den Dateinamen der Datei, in der die erzeugten O-Telos Frames gespeichert werden. Sie sollte das Prefix `.sml` haben, damit die Weiterverarbeitung im ConceptBase-System direkt möglich ist.

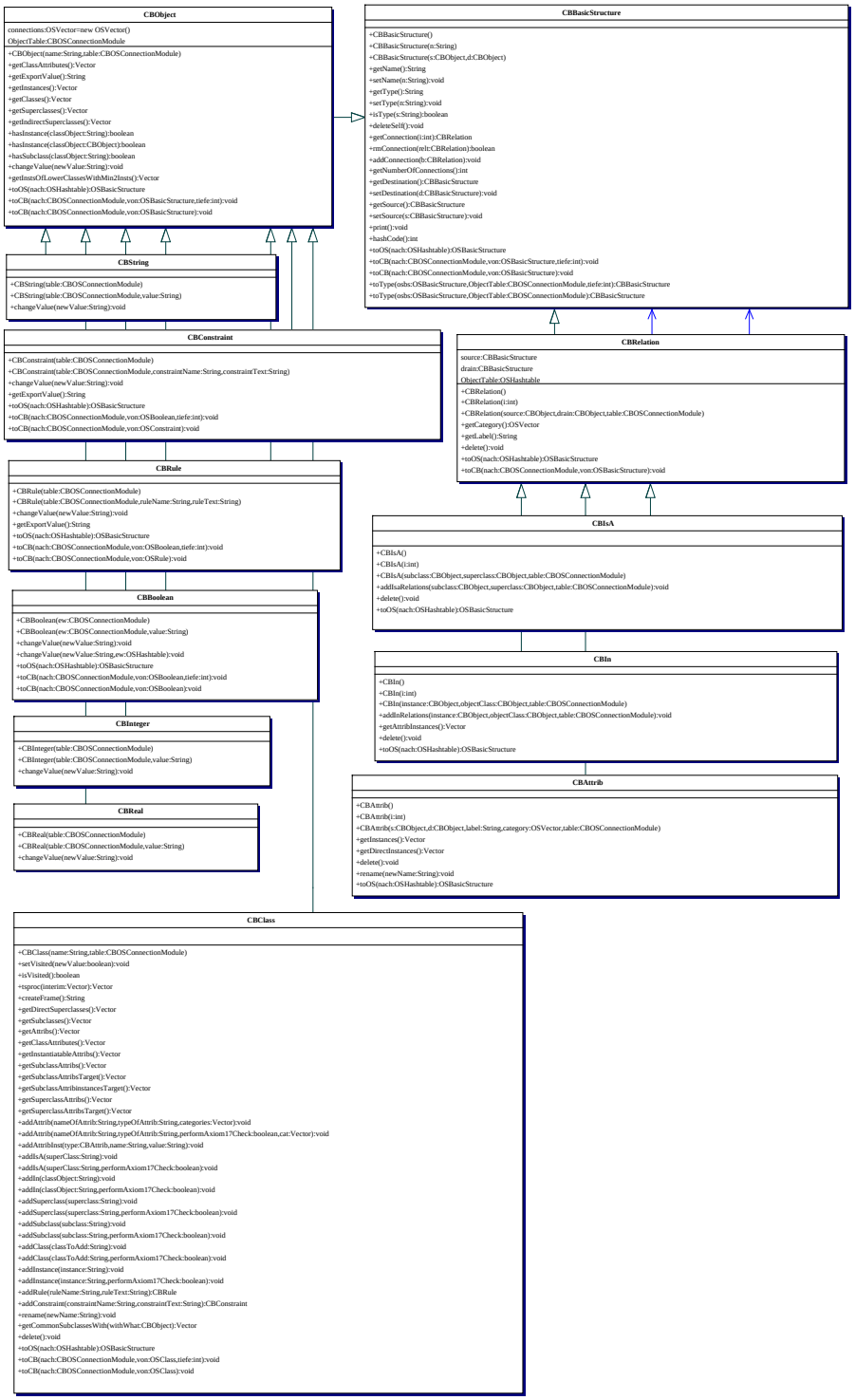

Abbildung 6.16: Die Java-Klassen der O-Telos Frames zur Verarbeitung im KBS Hyperbook System

Kapitel 7

Zusammenfassungen

7.1 Übersicht (Deutsch)

Im Rahmen dieser Arbeit wurde das KBS Hyperbook System zur Verwaltung, Speicherung und Bereitstellung von Hypermedia-Dokumenten entwickelt und implementiert. Dieses WWW-basierte Hypermedia-System bildet als Forschungssystem die Basis für weitere Arbeiten, unter anderem in den Themenbereichen Modellierung und Adaption der Inhalte des Systems. Darüber hinaus wird das System beispielsweise in der Lehre als Vorlesungsskript, Nachschlagewerk und zur Prüfungsvorbereitung von Lehrenden und Lernenden eingesetzt.

Zu Beginn der Arbeit wurden zunächst grundlegende Anforderungen auf der Basis der Aufgabenstellungen der Systeme an ein solches Hypermedia-System hergeleitet. Die Aufgabenstellungen umfassen die Erweiterbarkeit des Systems um neue Inhalte und neue Funktionalitäten, dessen Offenheit gegenüber den Hypermedia-Dokumenten, die Flexibilität des Systems gegenüber unterschiedlichen Verwendungsszenarien und Modifikationen des Inhalts, die Wiederverwendbarkeit der Hypermedia-Dokumente und ihrer Beschreibungen. Diese Anforderungen werden vorteilhaft durch den Einsatz von in der Modellierungssprache O-Telos geschriebenen Meta-, Metadaten- und Datenmodellen im KBS Hyperbook System erfüllt. Das KBS Hyperbook System kombiniert die Modellierungsansätze existierender Systeme, indem gleichzeitig die Wissensgebiete (Domänen) und die Benutzer durch Datenmodelle beschrieben werden. Dies wird in verschiedenen Hyperbooks aus den Bereichen Lehre, Terminologien und Metamodellierung (RDF) beispielhaft dargestellt.

Das KBS Hyperbook Systems ist in der Programmiersprache Java modular implementiert. Der modulare Aufbau des KBS Hyperbook Systems ermöglicht den Austausch der Module oder den Einbau neuer Module, wobei die Kompatibilität durch Schnittstellenspezifikationen sichergestellt wird.

7.2 Übersicht (Englisch)

This work describes the development and implemention of the KBS Hyperbook System. It is a system for administration, storage and provision of hypermedia documents. Also, this WWW-based hypermedia system is a platform for further research activities, e.g. in the area of data modeling and user adaptation. Furthermore, it is used by teachers and students as a lecture script, as a reference book and for exam preparations.

The basic requirements for this kind of hypermedia systems are derived from the main tasks of the systems. The requirements comprise extensibility for new contents and new functionalities, openness regarding the hypermedia documents, flexibility regarding the systems different use cases and modifications of the content and reusability of the hypermedia documents and their models. These requirements are fulfilled by the usage of meta-, metadata- and datamodels stated in the modeling language O-Telos. The KBS Hyperbook System combines modeling approaches of existing systems by describing the knowledge domains and the users in datamodels at the same time. As examples several existing hyperbooks from the various areas of teaching, terminologies and meta modeling (RDF) are given.

The present KBS Hyperbook System uses a modular implemention approach in the programming language Java. The modular architecture of the system enables the replacement of the modules or the implementation of new modules while the compatibility is guaranteed by interface specification.

Kapitel 8

Ausblick

Das KBS Hyperbook System stellt eine Basis für die Erforschung und Entwicklung von weiteren Hypermedia-Systemen mit spezifischen Aufgabenstellungen dar. Diese Systeme können ihren Fokus beispielsweise auf Modellierungsaspekte oder domänenspezifische Aspekte richten. Erforschbare Modellierungsaspekte sind etwa Untersuchungen über die Kombination von Domänenmodellen und Ontologien, die Verwendung neuer Modellarten (z.B. der Prozessmodelle zur Domänenmodellierung) oder die Spezifikation diverser Meta-Modelle. Zu den domänenspezifischen Aspekten gehören z.B. Lehr-/Lernsysteme und deren Implementierungen diverser Lehr- und Lernmethoden, die Dokumentation wissenschaftlicher Untersuchungen und der Einsatz im Wissensmanagement.

Ein für die Modellierung interessanter Aspekt ist auch die Modellierung der Benutzer, ihrer Gruppen und Rechte sowie die Kombination dieses Benutzermodells mit den Domänenmodellen. Eine auf diesen Modellen basierende Verwaltung der Benutzer und der Gruppen von Benutzern bestimmt die Rechte, die die Benutzer im System haben, z.B. das Recht zum Anlegen, Lesen und Löschen von Konzepten und Relationen. Solch ein Annotationssystem kann beispielsweise unter Berücksichtigung der Diplomarbeit von K. Naceur [84] auf dem KBS Hyperbook System beruhen. Zum Anlegen von Relationen und Konzepten werden im aktuellen KBS Hyperbook System HTML-Formulare ohne Berücksichtigung der Rechte der Benutzer verwendet. Denkbar ist, daß diese textuell basierte Lösung durch ein grafisches Editorsystem erweitert wird, das auch die Rechte der Benutzer berücksichtigt. Das Editorsystem würde es ermöglichen, zur Laufzeit bestehende Modelle zu erweitern oder neue Modelle dem System zur Verfügung zu stellen. Die im KBS Hyperbook System verwendeten Modelle umfassen viele Konzepte, so daß das grafische Editorsystem fortgeschrittene Entwicklungen zur Datenvisualisierung implementieren müßte. Ein Ansatz für ein solches System wurde in der Diplomarbeit von E. E. Steen [104] für eine frühere Version des KBS Hyperbook Systems vorgeschlagen.

Das KBS Hyperbook System ermöglicht aufgrund der modularen Aufbauweise auch Erweiterungen der Funktionalität. So wird die Visualisierung der Navigationsstruktu-

ren im KBS Hyperbook System hauptsächlich durch Links realisiert. Hier können fortgeschrittene grafische Darstellungen der Strukturen dem Benutzer zusätzliche Navigationshilfen zur Verfügung stellen. Vor diesem Hintergrund kann auch der Dualismus zwischen Link und Relation erörtert werden, so daß alternative Darstellungsweisen für Relationen gefunden werden können.

Desweiteren kann das System um eine Art Warenkorbsystem erweitert werden. Ein Warenkorbsystem ermöglicht dem Benutzer das Anlegen und Verwalten von eigenen Sequenzen von WWW-Seiten des aktuellen Hyperbooks. Im Rahmen dieser Erweiterung ist die Kombination des KBS Hyperbook Systems mit Systemen aus dem eCommerce-Bereich zu untersuchen.

Auf der Basis des KBS Hyperbook Systems kann ein Manager für RDF-Schemata und deren Hypermedia-Dokumente entwickelt und implementiert werden. Dieser Manager ermöglicht es Autoren, dort ihre Seiten automatisiert zu registrieren. Aufgrund der von den Autoren verwendeten RDF-Schemata ordnet das System die Dokumente in eine oder mehrere Domänen ein und verwaltet sie. Auf diese Weise entstehen große Mengen strukturierter Informationen, die in Domänenmodellen erfaßt sind und mittels spezialisierter Hyperbooks abgerufen werden können.

Anhang A

Implementierungsdetails

A.1 Framesyntax des Parsers

In der in [68] angegebenen Syntax werden alle vom Datenbanksystem ConceptBase eingesetzten möglichen O-Telos Konstrukte beschrieben. Die Syntax wird weitestgehend vom Parser implementiert und ermöglicht unter anderem den Zugriff auf die Attribute des Objekts. Diese Funktionalität unterstützt der in dieser Arbeit unter Berücksichtigung der Diplomarbeit von K. Hudalla [62] entwickelte Parser hingegen nicht explizit, da die in den Repräsentationsmodellen enthaltenen Attribute, Instanzierungen und Spezialisierungen in den zu ihren Quellklassen erzeugten Frames definiert werden.

<object>	-->	<objectname> <objectname> <inspec> <isaspec> <withspec> <endspec> |<objectname> <inspec> <isaspec> <withspec> <endspec>
<objectname>	-->	(<objectname>) |<label>
<inspec>	-->	<empty> |in <classlist>
<isaspec>	-->	<empty> |isA <classlist>
<classlist>	-->	<objectname> |<objectname> , <classlist>
<withspec>	-->	<empty> |with <decllist>
<decllist>	-->	<empty> |<declaration> |<declaration> <decllist>
<declaration>	-->	<attrcatlist> <proplist>
<attrcatlist>	-->	<label> |<label> , <attrcatlist>
<proplist>	-->	<property>

	|	|<property> ; <proplist>
<property>	-->	<label> : <objectname> |<label> : <complexref> |<label> : <enumeration> |<label> : <pathexpression>
<complexref>	-->	<objectname> <withspec> <endspec>
<enumeration>	-->	[<classlist>]
<pathexpression>	-->	<objectname> SELECTORB <pathargument>
<pathargument>	-->	<label> |<label> SELECTORB <pathargument> |<restriction> |<restriction> SELECTORB |<pathargument>
<restriction>	-->	(<label> : <enumeration>) |(<label> : <pathexpression>) |(<label> : <objectname>)
<endspec>	-->	end
<label>	-->	ALPHANUM |ANY_LITERAL |STRING_LITERAL |RC_LITERAL |REAL |INTEGER
ALPHANUM	-->	[a-zA-Z0-9]+
ANY_LITERAL	-->	everything except .|"'$:;!^ ->=,()[]{}/, newlines, tabs, carriage returns
STRING_LITERAL	-->	everything enclosed in " except " and \
RC_LITERAL	-->	everything enclosed in $ except $ and \
REAL	-->	[-]?([0-9]+\.[0-9] *|[0-9]* \.[0-9]+)([Ee] [-+]?[0-9]+)?
INTEGER	-->	[-]?[0-9]+
SELECTORB	-->	"."|"|"

A.2 Zusätzliche Layout-Befehle

A.2.1 Frame für die Navigation

Im Folgenden werden die möglichen Befehle erläutert, die in der HTML-Seite verwendet werden können, um das Layout des Frames der Navigation zu beschreiben.

- `<RelationBlock>` und `</RelationBlock>`
  Markiert Beginn und Ende der Spezifikation und gibt an, wie die einzelnen Re-

lationen dargestellt werden. Wird nur einmal angegeben und gilt dann für alle Relationen des Frames der Navigations.

- `<ConceptBlock>` und `</ConceptBlock>`
  Markiert Beginn und Ende der Spezifikation und zeigt, wie die einzelnen Konzepte einer Relation dargestellt werden. Wird einmal angegeben und gilt dann für alle Konzepte aller Relationen des Frames der Navigation.
- `<RelationName>`
  Gibt den Namen der aktuellen Relation an. Der Name der Relation ist einer der Werte der Attribute `nameSourceDrain` oder `nameDrainSource` der aktuellen `Relation`, abhängig davon, ob das aktuelle Konzept im `source` Attribut oder im `drain` Attribut enthalten ist.
- `<ActualConcept>`
  Stellt den eindeutigen Identifier des Konzepts dar.
- `<ConceptName>`
  Gibt den Namen des Konzepts an, hier also den Wert des Attributs `name` der Klasse `Concept`.
- `ConceptID`
  Enthält die URI des Hypermedia-Dokuments, das durch das Konzept beschrieben wird, d.h. der Wert des Attributs `id` der Klasse `Konzept`.
- `<ConceptAbstract>`
  Gibt die kurze Zusammenfassung des Konzepts an, hier also den Wert des Attributs `abstract` der Klasse `Konzept` an.
- `<User>`
  Enthält die ID des Benutzers.
- `<Group>`
  Gibt die Gruppe des Benutzers an.

Die folgende Tabelle listet den Befehl und den Ort seiner Verwendung in Bezug auf die HTML-Seite auf.

Befehlsname	**Verwendung**
`<ActualConcept>`	an jeder Stelle
`<RelationBlock>`	an jeder Stelle
`<RelationName>`	an jeder Stelle im `RelationBlock`
`<ConceptBlock>`	an jeder Stelle im `RelationBlock`
`<ConceptName>`	an jeder Stelle im `ConceptBlock`
`<ConceptID>`	an jeder Stelle im `ConceptBlock`
`<ConceptAbstract>`	an jeder Stelle im `ConceptBlock`
`</ConceptBlock>`	definiert das Ende eines `ConceptBlocks`
`</RelationBlock>`	definiert das Ende eines `RelationBlocks`
`<User>`	an jeder Stelle
`<Group>`	an jeder Stelle

A.2.2 Frame für die zusätzlichen Funktionalitäten

In der folgenden Tabelle sind die Befehle aufgeführt, die für die Spezifikation des Layouts des Frames der zusätzlichen Funktionalitäten zur Verfügung stehen. Die Befehle können in der HTML-Seite deklariert werden, die zur Definition des Layouts des Frames verwendet wird. Sie entsprechen Platzhaltern für die Verweis-URI (`href`-Komponente) des Links, der im Hyperbook eine bestimmte Funktionalität auslöst.

Der Text (durch die Symbole *text1* und *text2* dargestellt) kann der übliche, in einer Linkdefinition nach der HTML-Spezifikation [93] erlaubte Inhalt gemäß `<a` *text1* `..>` *text2* `</a>` sein.

Befehlskonstrukt	**Erläuterung**
`<a <Zoom>` *text1* `>` *text2* `</a>`	Deklaration eines Links, der im Hyperbook die “navigation verbergen”-funktion auslöst.
`<a <AddConcept>` *text1* `>` *text2* `</a>`	Deklaration eines Links, der im Hyperbook die Funktion zum Einfügen eines Konzepts auslöst.
`<a <AddRelation>` *text1* `>` *text2* `</a>`	Deklaration eines Links, der im Hyperbook die Funktion zum Einfügen einer Beziehung zwischen Konzepten auslöst.
`<a <RmConcept>` *text1* `>` *text2* `</a>`	Deklaration eines Links, der im Hyperbook die Funktion zum Löschen eines Konzepts auslöst.
`<a <RmRelation>` *text1* `>` *text2* `</a>`	Deklaration eines Links, der im Hyperbook die Funktion zum Löschen einer Beziehung zwischen Konzepten auslöst.
`<a <ViewContent>` *text1* `>` *text2* `</a>`	Deklaration eines Links, der im Hyperbook die Darstellung einer Inhaltsübersicht auslöst.
`<a <ViewSingle>` *text1* `>` *text2* `</a>`	Deklaration eines Links, der im Hyperbook die Darstellung der Navigationsstrukturen eines Konzepts auslöst.

A.3 Verwendete Tools

Im Rahmen dieser Arbeit werden zur Erstellung und zum Betrieb des KBS Hyperbook Systems und der dazugehörigen Hyperbooks Tools von Fremdfirmen verwendet. Die folgende Auflistung gibt Aufschluß über diese eingesetzten Tools und zugehörige Informationen.

- Webserver JavaWebServer [108]
  Webserver der Firma Sun, der in Java programmiert ist. Der Webserver implementiert ein API, mit dem das Erstellen von Servlets in der Sprache Java möglich ist. In dieser Arbeit werden die Versionen 1.0 und 2.0 des Webservers für

das KBS Hyperbook System verwendet.

- Datenbanksystem ObjectStore [35]
  Ein objekt-orientiertes Datenbanksystem der Firma Excelon, das Java Objekte und Klassen direkt verarbeiten kann. Das KBS Hyperbook System muß die von ObjectStore zur Verfügung gestellten APIs implementieren, um das Datenbanksystem zu nutzen. Das KBS Hyperbook System wurde mit Hilfe der Versionen 5.0, 6.0 und 6.0 Service Pack 1 bis 3 dieser Datenbank entwickelt.

- Datenbanksystem ConceptBase [67]
  Ein objekt-orientiertes Datenbanksystem, das an der RWTH Aachen entwickelt wird. Zur Verwendung in Java Programmen wird als Teil dieser Arbeit ein eigenes Java API entwickelt, das die für das KBS Hyperbook System benötigte Funktionalität zur Anbindung des Datenbanksystems implementiert. Das KBS Hyperbook System wurde mit Hilfe der Versionen 4.0, 5.0, 5.0.1 und 5.0.2 des ConceptBase Datenbanksystems entwickelt.

- Java [48, 106]
  Das KBS Hyperbook System wird in der Programmiersprache Java implementiert. Diese objekt-orientierte Sprache wird von der Firma Sun zur Verfügung gestellt. Sie wird im sogenannten Java Development Kit (JDK) [106] verbreitet und kann auf vielen Plattformen ohne umfangreiche Anpassungen ausgeführt werden, z.B. auf Unix, Linux, Macintosh und Windows Systemen. Das KBS Hyperbook System wurde mit Hilfe der Version 1.2.2 des JDK der Firma Sun entwickelt.

- JavaCC [107]
  Der Java Compiler Compiler (JavaCC) ist ein in Java geschriebenes Tool zur Erstellung von Parsern. Auf der Basis der Bakus Naur Form (BNF) bzw. der Erweiterten BNF (EBNF) [2] wird eine vom JavaCC verstandene Grammatik der zu verarbeitenden Daten entworfen. Diese Grammatik verarbeitet der JavaCC zu einem in Java implementierten Parser. In der Grammatik können neben der Beschreibung der EBNF auch Java-Methoden aufgeführt werden, die im Parser implementiert werden. Der JavaCC wird in der Version 0.8 pre 1 im KBS Hyperbook System verwendet, um einen Parser für die Abbildung der O-Telos Frames in Java-Objekten zu implementieren.

- TogetherJ [110]
  Das Tool TogetherJ der Firma TogetherSoft ist eine Entwicklungsumgebung für objekt-orientierte Programmierprojekte, die in der Sprache Java implementiert werden sollen. Zur Darstellung des Designs wird die Unified Modelling Language (UML) [39] verwendet. TogetherJ wird in der Version 4.2 zur Weiterentwicklung und zur Dokumentation des KBS Hyperbook Systems eingesetzt.

A.4 Schnittstellenspezifikationen

Die genauen Schnittstellenspezifikationen können direkt den entsprechenden Java Interface Deklarationen entnommen werden. Die Interface Deklarationen existieren für die folgenden Module:

Modulbezeichnung	**Klassenname**	**Interfacename**
Visualisierung	VisualisationModule	VisualisationModuleInterface
Storage	StorageModule	StorageModuleInterface
CBOSConnectionModule	CBOSConnection-Module	CBOSConnection-ModuleInterface

An dieser Stellen sollen nun die Methoden der Schnittstellen kurz erläutert werden.

VisualisationModuleInterface	
Methodenname	**Erläuterung**
setUserModel	übergibt das Benutzeradaptionsmodul an das VisualisationModule
setStorageModule	übergibt das StorageModule an das VisualisationModule
setEnvironment	übergibt den Environment Hashtable mit den Umgebungsvariablen an das VisualisationModule
init	initialisier das VisualisationModule
createPage	erzeugt die jeweils geforderte WWW-Seite

StorageModuleInterface	
Methodenname	**Erläuterung**
init	initialisieren des Moduls
initDB	initialisieren der Datenbank
appendItem	ein Objekt der Datenbank hinzufügen
getItem	ein Objekt aus der Datenbank lesen
getNumberOfItems	Anzahl der Objekte in der Datenbank
isInstanceOf	liefert boolschen Ausdruck, der besagt, ob ein Objekt Instanz eines anderen Objekts ist
getConceptInstances	liefert alle Instanzen eines Konzepts
getConceptInstance	liefert ein ConceptInstance Objekt mit Namen, URI und Abstrakt
getConceptAttribute	liefert die URIs bzw. die Dateinamen von Konzepten; wird benötigt, wenn ein Modell dieses Daten explizit modelliert
getRelation	liefert ein Relation Objekt
getSCIs	liefert alle evtl. adaptierten Relationen eines Konzepts als eine Menge von SetOfConceptInstances
getRelatedConceptInstances	liefert zu einer ID und einem Relationsnamen alle Instanzen der Relation - also alle Instanzen der Relation, die vom Konzept ID ausgehen
getRestrictedSetsOf-ConceptInstances	liefert zu einer ID und einem Relationsnamen alle Instanzen der Relation, wobei eine Bedingung erfüllt sein muß
getRelatedConcept-InstancesWithWeight	liefert zu einer ID und einem Relationsnamen alle Instanzen der Relation mit zugehörigen Gewichten
makeConcept	erzeugt ein Konzept in der Datenbank
makeRelation	erzeugt eine Relation in der Datenbank
removeConcept	löschen eines Konzepts oder einer Relation

CBOSConnectionModuleInterface	
Methodenname	**Erläuterung**
init	initialisiere das Modul
get	hole ein Objekt aus der Datenbank
remove	lösche ein Objekt aus der Datenbank
removeObjects	lösche eine Reihe von Objekten aus der Datenbank
put	schreibe ein oder mehrere Objekte in die Datenbank
size	Anzahl der Objekte in der Datenbank
keys	liefert eine Aufzählung der Objekte aus der Datenbank
containskey	liefert einen boolschen Ausdruck, der besagt, ob ein key in der Datenbank enthalten ist

A.5 Beispiele der O-Telos Rules und Constraints im KBS Hyperbook System

Im Folgenden werden einige Beispiele gezeigt, wie Einschränkungen (Constraints) und Regeln (Rules) der Modellierungssprache O-Telos in den Meta-, Metadaten- und Datenmodellen eingesetzt werden, um die Konsistenz der Modelle sicherzustellen.

```
Class Relation11 isA Relation with
    attribute, single
        source : Concept;
        drain : Concept
    constraint
    singleConstraint : $ forall p/Class!single
    c,d/Class x,m/VAR In(p,Class!single)
    and P(p,c,m,d) and In(x,c)
    ==> forall y1,y2/VAR In(y1,d) and In(y2,d)
                    and A(x,m,y1) and A(x,m,y2)
    ==> IDENTICAL(y1,y2) $
end
```

Beispiel A.5.1: Ein möglicher O-Telos Quelltext der (1:1) Relation im Meta-Modell

Das Beispiel A.5.1 deklariert eine Klasse `Relation11` als eine (1:1)-Beziehung zwischen zwei Konzepten. Als Beziehung ist sie Subklasse von `Relation` und spezialisiert die Attribute `source` und `drain` als vom Typ `Concept`. Auf beide Attribute wird das Constraint `singleConstraint` angewendet, ausgedrückt durch das im Constraint deklarierte und in der Attributdeklaration verwendete Schlüsselwort `single`. Das `singleConstraint` beschränkt das Attribut, auf das es angewendet wird, auf einen Wert. Es definiert also einwertige Attribute. Das Constraint kann auf alle Attribute im Meta-, Metadaten- und Datenmodell angewendet werden, muß aber nur einmal deklariert werden. Es stellt sicher, daß eine (1:1)-Beziehung nur zwei Konzepte über eine `Relation11` verbindet.

```
  constraint
    c1 : $ forall c/Concept a/ConceptAttribute
    ((c properties a)
      and (c in SemanticInformationUnit))
    ==> ((a name "Beschreibung")
          or (a name "Motivation")) $
```

Beispiel A.5.2: Ein möglicher O-Telos Constraint zur Sicherstellung der Konsistenz der Daten

Das Beispiel A.5.2 deklariert einen Constraint, der sicherstellt, daß alle `ConceptAttribute` ein Attribut `name` haben, das entweder den Wert `Beschreibung` oder den Wert `Motivation` hat. Das `ConceptAttribute`

muß dabei über ein Attribut `properties` mit einem `Concept` vom Typ `SemanticInformationUnit` in Verbindung stehen. Das Constraint schränkt den Wertebereich des Attributs `name` aller Konzepte vom Typ `SemanticInformationUnit` auf dieselben zwei Werte ein. Der Wert des Attributs wird vom KBS Hyperbook System während der Generierung der WWW-Seite als Überschrift des entsprechenden Hypermedia-Dokumentteils (durch das `ConceptAttribute` beschrieben) verwendet. Das Constraint wird üblicherweise im Datenmodell für die entsprechenden Konzepttypen deklariert.

```
Class DisplayRelation with
  attribute
    nameSourceDrain: String;
    nameDrainSource: String;
    visualisation:   Boolean;
    priority:        Integer
    rule
      displayConstraint : $ forall r/DisplayRelation
                            ((r in Relation)
                              or (r isA Relation))$
end
```

Beispiel A.5.3: O-Telos Rule zur Anzeige von Relationen

Das Beispiel A.5.3 deklariert die zur Anzeige von Relationen verwendete Klasse `DisplayRelation`. Diese Klasse deklariert eine Reihe von Attributen, die für die Anzeige der Relationen notwendig sind. Außerdem wird hier die Rule `displayConstraint` deklariert, die besagt, daß jede Instanz von `DisplayRelation` entweder eine Instanz oder eine Subklasse von Relation sein muß. Diese Regel wird im Meta-, Metadaten- oder Datenmodell deklariert und stellt sicher, daß die zur Anzeige von Relationen notwendigen Attribute in der Relation ausgeprägt sind.

Anhang B

O-Telos-RDF

B.1 O-Telos-RDF-Schema

In diesem Kapitel wird das von O-Telos-RDF verwendete Schema vorgestellt. Das Schema ist an die Darstellung des RDF-Schemas aus der RDF Schema Spezifikation [13] angelehnt.

Das Schema gibt einige der in 4.5 dargestellten Axiome direkt wieder. Zu diesen Axiomen gehören die folgenden: 2,4, 6, 7, 11, 12, 17, 20, 25, 31

Alle anderen Axiome lassen sich nur in First Order Logik ausdrücken, so daß sie nicht in dieser XML-Serialisierung enthalten sind.

Die Konstrukte `literal`, `ordinal` und `class` sind im Schema aufgenommen worden, da sie allgemein zur Verfügung stehen sollen. Sie werden durch die beschriebenen Axiome ermöglicht, so daß die Konstrukte nicht durch eigene Axiome deklariert werden müssen.

```
<otelos:OTELOS
  xmlns:otelos="http://www.kbs.uni-hannover.de/otelos/2001/06
                                                  /otelos-rdf-schema">
  <otelos:Description ID="statement">
    <otelos:type s="otelos:individual"/>
    <subClassOf s="otelos:statement"/>
    <otelos:property s="otelos:statement"/>
  </otelos:Description>

  <otelos:Description ID="individual">
    <subClassOf s="otelos:statement"/>
  </otelos:Description>

  <otelos:Description ID="literal">
    <subClassOf s="otelos:individual"/>
```

```
  </otelos:Description>

  <otelos:Description ID="ordinal">
    <subClassOf s="otelos:literal"/>
  </otelos:Description>

  <otelos:Description ID="class">
    <subClassOf s="otelos:individual"/>
  </otelos:Description>

  <otelos:Descritpion ID="property>
    <subClassOf s="otelos:statement"/>
    <subPropertyOf s="otelos:property"/>
  </otelos:Description>

  <otelos:Description ID="type">
    <subClassOf s="otelos:statement"/>
  </otelos:Description>

  <otelos:Description ID="subClassOf">
    <subClassOf s="otelos:statement"/>
  </otelos:Description>

  <otelos:Description ID="subPropertyOf">
    <subClassOf s="otelos:statement"/>
  </otelos:Description>

</otelos:OTELOS>
```

B.2 O-Telos-RDF XML-Serialisierung

Die hier wiedergegebene XML-Serialisierung von O-Telos-RDF ist der XML-Serialisierung von RDF aus der RDF Syntax und Modell Spezifikation [116] sehr ähnlich. Beispielsweise werden alle WWW-Seiten durch eigene Statements deklariert, was durch einen intelligenten Parser übernommen werden könnte, wodurch die Angabe dieser Statements entfallen würde. Ebenso ließe sich “resource=” als Synonym für “s=” verwenden, wodurch soviel RDF XML-Serialisierung wie möglich beibehalten werden könnte.

Die O-Telos-RDF XML-Serialisierung wird in der aktuellen Form zur Darstellung der O-Telos-RDF Statements in XML, hauptsächlich der primitiven Typen “otelos:literal” und “otelos:statement” und deren Subklassen verwendet. Diese Serialisierung gruppiert mehrere Statements derselben Ressource in einem “Description” Element unter Verwendung des XML-Syntax.

Wie in der RDF Syntax und Modell Spezifikation das RDF Element wird hier das OTELOS Element zum Deklarieren von Anfang und Ende der O-Telos-RDF statements in einem XML Dokument verwendet.

Die EBNF der O-Telos-RDF XML-Serialisierung sieht wie folgt aus:

1)	OTELOS	::=	['<otelos:OTELOS>'] description* ['</otelos:OTELOS>']
2)	description	::=	'<otelos:Description' idAboutAttr? '>' propertyElt* '</otelos:Description>' | '<otelos:Description' idAboutAttr? '/ >'
3)	idAboutAttr	::=	idAttr | aboutAttr
4)	aboutAttr	::=	'about=" statement-reference "'
5)	idAttr	::=	'ID=" IDsymbol "'
6)	propertyElt	::=	'<' label '>' value '</' label '>' | '<' label statementAttr '/>'
7)	label	::=	name
8)	value	::=	description | string
9)	statement-Attr	::=	's=" statement-reference "'
10)	statement -reference	::=	string, interpreted as statement-ID, includes NSprefix
11)	IDsymbol	::=	(any legal XML name symbol)
12)	name	::=	(any legal XML name symbol)
13)	NSprefix	::=	(any legal XML namespace prefix)
14)	string	::=	(any XML text, with '<', '>', and '&' escaped)

B.3 Beispiele der Verwendung von RDF und O-Telos-RDF

In diesem Anhang finden sich die XML-Serialisierungen des Modellausschnitts der Vorlesung "Künstliche Intelligenz", einmal mittels des RDF-Schemas und einmal mittels des RDF-O-Telos Schemas geschrieben. Außerdem werden zu diesen beiden Serialisierungen auch die entsprechenden Statement-Darstellungen mit angegeben.

Zur Vereinfachung der Lesbarkeit der Beispiele sind die URIs der Namensräume durch die Namen der Namensräume abgekürzt. Aus demselben Grund werden die angegebenen URIs der WWW-Ressourcen um die Server- und Pfadangabe reduziert dargestellt.

B.3.1 RDF XML-Serialisierung der KI-Vorlesung

An dieser Stelle wird das Teilmodell der AI-Vorlesung in Form der RDF XML-Serialisierung wiedergegeben. Es gelten alle oben beschriebenen Vereinfachungen.

```
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

```
xmlns:s="http://www.kbs.uni-hannover.de/otelos/2001/06
                                          /example-schema#">

<rdf:Description ID="Lecture">
  <rdf:type resource="rdfs:Class"/>
</rdf:Description>

<rdf:Description ID="LectureUnit">
  <rdf:type resource="rdfs:Class"/>
</rdf:Description>

<rdf:Description ID="title">
  <rdf:type resource="rdf:Property"/>
  <rdfs:range resource="rdfs:Literal"/>
  <rdfs:domain resource="s:LectureUnit"/>
</rdf:Description>

<rdf:Description ID="description">
  <rdf:type resource="rdf:Property"/>
  <rdfs:range resource="rdfs:Literal"/>
  <rdfs:domain resource="s:LectureUnit"/>
</rdf:Description>

<rdf:Description ID="parentCourse">
  <rdf:type resource="rdf:Property"/>
  <rdfs:range resource="s:Lecture"/>
  <rdfs:domain resource="s:LectureUnit"/>
</rdf:Description>

<rdf:Description ID="theoryPage">
  <rdf:type resource="rdf:Property"/>
  <rdfs:range resource="s:TheoryUnit"/>
  <rdfs:domain resource="s:LectureUnit"/>
</rdf:Description>

<rdf:Description ID="parentUnit">
  <rdf:type resource="rdf:Property"/>
  <rdfs:range resource="s:LectureUnit"/>
  <rdfs:domain resource="s:TheoryUnit"/>
</rdf:Description>

<rdf:Description ID="TheoryUnit">
  <rdf:type resource="rdfs:Class"/>
</rdf:Description>

<rdf:Description ID="AILecture">
  <rdf:type resource="s:Lecture"/>
</rdf:Description>

<rdf:Description ID="LectureUnit1">
  <rdf:type resource="s:LectureUnit"/>
  <title> Lecture Unit 1</title>
  <description>
```

```
      Introduction to intelligent agents
    </description>
    <parentCourse resource="s:AILecture"/>
    <theoryPage resource="http://.../Definitions.htm"/>
    <theoryPage resource="http://.../Characterisation.htm"/>
    <theoryPage resource="http://.../Structure.htm"/>
    <theoryPage resource="http://.../Types.htm"/>
  </rdf:Description>

  <rdf:Description about="http://.../Definitions.htm">
    <rdf:type resource="s:TheoryUnit"/>
    <parentUnit resource="s:LectureUnit1"/>
  </rdf:Description>

  <rdf:Description about="http://.../Characterisation.htm">
    <rdf:type resource="s:TheoryUnit"/>
    <parentUnit resource="s:LectureUnit1"/>
  </rdf:Description>

  <rdf:Description about="http://.../Structure.htm">
    <rdf:type resource="s:TheoryUnit"/>
    <parentUnit resource="s:LectureUnit1"/>
  </rdf:Description>

  <rdf:Description about="http://.../Types.htm">
    <rdf:type resource="s:TheoryUnit"/>
    <parentUnit resource="s:LectureUnit1"/>
  </rdf:Description>
</rdf:RDF>
```

B.3.2 O-Telos-RDF XML-Serialisierung der KI-Vorlesung

Hier folgt nun die Darstellung der O-Telos-RDF XML-Serialisierung des Teilmodells der AI-Vorlesung. Es gelten alle oben beschriebenen Vereinfachungen.

```
<otelos:OTELS xml:lang="en"
  xmlns:otelos="http://www.kbs.uni-hannover.de/otelos/2001/06
                                              /otelos-rdf-schema#"
  xmlns:t="http://www.kbs.uni-hannover.de/otelos/2001/06
                                         /example-otelos-schema#">

  <otelos:Description ID="Lecture"/>

  <otelos:Description ID="LectureUnit">
    <title s=otelos:literal/>
    <description s=otelos:literal/>
    <parentCourse s=t:Lecture/>
    <theoryPage s=t:TheoryUnit/>
  </otelos:Description>
```

```
<otelos:Description ID="TheoryUnit">
  <parentUnit s="t:LectureUnit"/>
</otelos:Description>

<otelos:Description ID="AILecture">
  <type s="t:Lecture"/>
</otelos:Description>

<otelos:Description ID="LectureUnit1">
  <type s="t:LectureUnit"/>
  <title>Lecture Unit 1</title>
  <description>
    Introduction to intelligent agents
  </description>
  <parentCourse s="t:AILecture"/>
  <theoryPage1 s="http://.../Definitions.htm"/>
  <theoryPage2 s="http://.../Characterisation.htm"/>
  <theoryPage3 s="http://.../Structure.htm"/>
  <theoryPage4 s="http://.../Types.htm"/>
</otelos:Description>

<otelos:Description ID="LectureUnit1_title">
  <type s="t:LectureUnit_title"/>
</otelos:Description>

<otelos:Description ID="LectureUnit1_description">
  <type s="t:LectureUnit_description"/>
</otelos:Description>

<otelos:Description ID="LectureUnit1_parentCourse">
  <type s="t:LectureUnit_parentCourse"/>
</otelos:Description>

<otelos:Description ID="LectureUnit1_theoryPage1">
  <type s="t:LectureUnit_theoryPage"/>
</otelos:Description>

<otelos:Description ID="LectureUnit1_theoryPage2">
  <type s="t:LectureUnit_theoryPage"/>
</otelos:Description>

<otelos:Description ID="LectureUnit1_theoryPage3">
  <type s="t:LectureUnit_theoryPage"/>
</otelos:Description>

<otelos:Description ID="LectureUnit1_theoryPage4">
  <type s="t:LectureUnit_theoryPage"/>
</otelos:Description>

<otelos:Description about="http://.../Definitions.htm">
  <type s="t:TheoryUnit"/>
  <parentUnit s="t:LectureUnit1"/>
</otelos:Description>
```

```
  <otelos:Description about="http://.../Characterisation.htm">
    <type s="t:TheoryUnit"/>
    <parentUnit s="t:LectureUnit1"/>
  </otelos:Description>

  <otelos:Description about="http://.../Structure.htm">
    <type s="t:TheoryUnit"/>
    <parentUnit s="t:LectureUnit1"/>
  </otelos:Description>

  <otelos:Description about="http://.../Types.htm">
    <type s="t:TheoryUnit"/>
    <parentUnit s="t:LectureUnit1"/>
  </otelos:Description>

</otelos:OTELOS>
```

B.3.3 RDF Statements der KI-Vorlesung

Tabelle B.2: RDF-Tupel des Ausschnitts aus dem AI-Vorlesungsmodell

Nr.	Statement (Subjekt,Prädikat,Objekt)
1	s(online:Lecture,rdf:type,rdfs:Class)
2	s(online:LectureUnit,rdf:type,rdfs:Class))
3	s(online:title,rdf:type,rdf:Property))
4	s(online:genid2,resource,rdfs:Literal)
5	s(online:title,rdfs:range,online:genid2)
6	s(online:genid5,resource,s:LectureUnit)
7	s(online:title,domain,online:genid5)
8	s(online:description,type,rdf:Property)
9	s(online:genid8,resource,rdfs:Literal)
10	s(online:description,range,online:genid8)
11	s(online:genid11,resource,s:LectureUnit)
12	s(online:description,domain,online:genid11)
13	s(online:parentCourse,type,rdf:Property)
14	s(online:genid14,resource,s:Lecture)
15	s(online:parentCourse,range,online:genid14)
16	s(online:genid17,resource,s:LectureUnit)
17	s(online:parentCourse,domain,online:genid17)
18	s(online:parentUnit,type,rdf:Property)
19	s(online:genid20,resource,s:LectureUnit)
20	s(online:parentUnit,range,online:genid20)
21	s(online:genid23,resource,s:TheoryUnit)
22	s(online:parentUnit,domain,online:genid23)
23	s(online:theoryPage,type,rdf:Property)
24	s(online:genid26,resource,s:TheoryUnit)
25	s(online:theoryPage,range,online:genid26)
26	s(online:genid29,resource,s:LectureUnit)
27	s(online:theoryPage,domain,online:genid29)

Nr.	Statement (Subjekt,Prädikat,Objekt)
28	s(online:TheoryUnit,type,rdfs:Class)
29	s(online:AILecture,type,s:Lecture)
30	s(online:LectureUnit1,type,s:LectureUnit)
31	s(online:LectureUnit1,online:title,Lecture Unit 1)
32	s(online:LectureUnit1,online:description,Introduction to intelligent agents)
33	s(online:genid34,resource,s:AILecture)
34	s(online:LectureUnit1,online:parentCourse,online:genid34)
35	s(online:genid37,resource,http://.../Definitions.htm)
36	s(online:LectureUnit1,online:theoryPage,online:genid37)
37	s(online:genid40,resource,http://.../Characterisation.htm)
38	s(online:LectureUnit1,online:theoryPage,online:genid40)
39	s(online:genid43,resource,http://.../Structure.htm)
40	s(online:LectureUnit1,online:theoryPage,online:genid43)
41	s(online:genid46,resource,http://.../Types.htm)
42	s(online:LectureUnit1,online:theoryPage,online:genid46)
43	s(http://.../Definitions.htm,type,s:TheoryUnit)
44	s(online:genid50,resource,s:LectureUnit1)
45	s(http://.../Definitions.htm,online:parentUnit,online:genid50)
46	s(http://.../Characterisation.htm,type,s:TheoryUnit)
47	s(online:genid54,resource,s:LectureUnit1)
48	s(http://.../Characterisation.htm,online:parentUnit,online:genid54)
49	s(http://.../Structure.htm,type,s:TheoryUnit)
50	s(online:genid58,resource,s:LectureUnit1)
51	s(http://.../Structure.htm,online:parentUnit,online:genid58)
52	s(http://.../Types.htm,type,s:TheoryUnit)
53	s(online:genid62,resource,s:LectureUnit1)
54	s(http://.../Types.htm,online:parentUnit,online:genid62)

In Tabelle B.2 werden die vom SiRPAC Parser [97] erzeugten RDF-Statements dargestellt, die auf der im Beispiel B.3.1 angegebenen RDF XML-Serialisierung basieren. Der Prefix online: stellt Konstrukte dar, die keine URI haben. Der Prefix online:genid stellt dagegen die Konstrukte dar, für die automatisch eine ID erzeugt wurde.

B.3.4 O-Telos-RDF Statements der KI-Vorlesung

Hier wird die Darstellung der O-Telos-RDF Statements der O-Telos-RDF XML-Serialisierung aus Anhang B.3.2 wiedergegeben. Zur Vereinfachung werden die Statement IDs nur symbolisch dargestellt. Die eigentlichen IDs werden entsprechend der in den vorhergehenden Kapiteln beschriebenen Regeln ggf. automatisch erzeugt. Daher werden in der XML-Serialisierung in Anhang B.3.2 die IDs nur in dem Fall direkt angegeben, in dem ein Statement über ein Statement gemacht wird.

Ebenso werden aus Gründen der Vereinfachung die type-Statements nicht dargestellt, die aus den Axiomen 1 bis 33 folgen. Diese Statements können von den entsprechenden Axiomen abgeleitet werden, würden aber im Beispiel nur verwirrend und unübersichtlich wirken.

```
s(sid1,sid1,Lecture,sid1)

s(sid2,sid2,LectureUnit,sid2)
s(sid3,sid2,title,otelos:literal)
s(sid4,sid2,description,otelos:literal)
s(sid5,sid2,parentCourse,sid1)
s(sid6,sid2,theoryPage,sid7)

s(sid7,sid7,TheoryUnit,sid7)
s(sid8,sid7,parentUnit,sid2)

s(sid9,sid9,AILecture,sid9)
s(sid10,sid9,type,sid1)

s(sid11,sid11,LectureUnit1,sid11)
s(sid12,sid11,type,sid2)
s(sid13,sid11,title,"Lecture Unit 1")
s(sid14,sid13,type,sid3)
s(sid15,sid11,description,"Introduction to intelligent agents")
s(sid16,sid15,type,sid4)
s(sid17,sid11,parentCourse,sid9)
s(sid18,sid17,type,sid5)
s(sid19,sid11,theoryPage1,sid27)
s(sid20,sid19,type,sid6)
s(sid21,sid11,theoryPage2,sid31)
s(sid22,sid21,type,sid6)
s(sid23,sid11,theoryPage3,sid35)
s(sid24,sid23,type,sid6)
s(sid25,sid11,theoryPage4,sid39)
s(sid26,sid25,type,sid6)

s(sid27,sid27,"http://.../Definition.htm",sid27)
s(sid28,sid27,type,sid7)
s(sid29,sid21,parentUnit,sid11)
s(sid30,sid29,type,sid8)

s(sid31,sid31,"http://.../Characterisation.htm",sid31)
s(sid32,sid31,type,sid7)
s(sid33,sid31,parentUnit,sid11)
s(sid34,sid33,type,sid8)

s(sid35,sid35,"http://.../Structure.htm",sid35)
s(sid36,sid35,type,sid7)
s(sid37,sid35,parentUnit,sid11)
s(sid38,sid37,type,sid8)

s(sid39,sid39,"http://.../Types.htm",sid39)
s(sid40,sid39,type,sid7)
s(sid41,sid39,parentUnit,sid11)
s(sid42,sid41,type,sid8)
```

Literaturverzeichnis

[1] J. R. Abrial. Data semantics. In Klimbie and Koffeman, editors, *Data Base Management*. North-Holland Publ., 1974.

[2] H.-J. Appelrath. *Starthilfe Informatik*. B. G. Teubner Verlag, Stuttgart; Leipzig, Deutschland, 1998.

[3] A. Bapat, M. Löhr, and J. Wäsch. Hypermedia Support for the Integrated Systems Engineering Environment MUSE. In *Proceedings of the 5th International Conference on Data and Knowledge Systems for Manufacturing and Engineering (DKSME'96)*, 1996.

[4] V. R. Benjamins, D. Fensel, and Asunción Gómez Pérez. Knowledge management through Ontologies. In *Proc. of the 2nd Int. Conf. on Practical Aspects of Knowledge management (PAKM-98)*, Basel, Switzerland, Oct. 1998.

[5] T. Berners-Lee. *World Wide Web: An illustrated seminar*, 1991. http://www.w3.org/pub/WWW/Talks/General.html.

[6] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. In *Internet Official Protocol Standards (STD 1)*, RFC. The Internet Society, rfc 2396 edition, 1998. http://www.ietf.org/rfc2396.txt.

[7] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). In *Internet Official Protocol Standards (STD 1*, RFC. The Internet Society, rfc 1738 edition, 1994. http://www.ietf.org/rfc1738.txt.

[8] Bibliomania: The Network Library, 1997. http://www.bibliomania.com.

[9] M. Bichler and S. Nusser. Modular design of Complex Web-Applications with W3DT. In *Proc. of the IEEE WET ICE'96*, Stanford, California, USA, 1986.

[10] G. Booch. *Object-Oriented Analysis and Design with Applications*. Addison-Wesly-Longman, Menlo Park, Kalifornien, USA, 1994.

[11] C. Bothur. Das Chamäleon. *Internet World*, Dezember 2000.

[12] T. Bray, J. Paoli, C.M. Sperberg-McQueen, and E. Maler. Extensible markup language (xml) 1.0 (second edition). W3c recomendation, World Wide Web Consortium W3C, October 2000. http://www.w3.org/TR/REC-xml.

[13] D. Brickley and R. V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. Technical report, World Wide Web Consortium (W3C), 2000. http://www.w3.org/TR/rdf-schema.

[14] P. Brusilovsky. Adaptive Hypermedia: an attempt to analyze and generalize. In *Workshop on Adaptive hypertext and hypermedia at UM'94*, Hyannis, Cape Cod, MA, USA, August 1994.

[15] P. Brusilovsky and L. Pesin. ISIS-Tutor: An Intelligent Learning Environment for CDS/ISIS Users. In *Proc. of CLCE'94*, Joensuu, Finland, 1994.

[16] P. Brusilovsky and E. Schwarz. User as Student: Towards an Adaptive Interface for Advanced Web-Based Applications. In *Proceedings of the Sixth International Conference on User Modeling, UM97*, Sardinia, Italy, 1997.

[17] P. Brusilovsky, E. Schwarz, and G. Weber. ELM-ART: An intelligent tutoring system on World Wide Web. In C. Frasson, G. Gauthier, and A. Lesgold, editors, *Intelligent Tutoring Systems (Lecture Notes in Computer Science, Vol. 1086)*, pages 261–269, Berlin, 1996. Springer.

[18] P. Brusilovsky, E. Schwarz, and G. Weber. A Tool for Developing Adaptive Electronic Textbooks on WWW. In *Proceedings of WebNet'96 - World Conference of the Web Society*, Boston, MA, USA, June 1996.

[19] M. Campione and K. Walrath. *The Java Tutorial, Second Ediion: Object-Oriented Programming for the Internet*. Addision-Wesley, 1998. http://java.sun.com/docs/books/tutorial.

[20] C. Capelle. Semantische Strukturierung und Annotation von Informationsangeboten durch Ontologien. Master's thesis, Universität Hannover, Hannover, Deutschland, 1999. http://www.kbs.uni-hannover.de/Arbeiten/Diplomarbeiten/.

[21] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins. The Ontology of Tasks and Methods. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, April 1998.

[22] P. Chen. Database desing based on entity and relationships. In S. Bing Yao, editor, *Principles of Database Design*, pages 174–210. Prentice-Hall, Englewood Cliffs, NJ, USA, 1990.

[23] K.A.L. Coar and D.R.T. Robinson. The WWW Common Gateway Interface Version 1.1. Internet-draft, NCSA, W3C, June 1999. http://CGI-Spec.Golux.com/draft-coar-cgi-v11-03-clean.html.

[24] ConceptBase Team. *ConceptBase Tutorial (CB 5.1)*. Informatik V, RWTH Aachen, Aachen, Deutschland, 5.1 edition, 2000. http://www-i5.informatik.rwth-aachen.de/CBdoc/tutorial/.

[25] W. Conen and R. Klapsing. A logical interpretation of RDF. In *Linköping Electronic Articles in Computer and Information Science, Semantic Web Area*, volume 5. Linköping University Electronic Press, December 2000. http://www.ep.liu.se/ea/cis/2000/013/.

[26] T. A. Dahn. Slicing Information Technology (SIT). Technical report, Slicing Information Technology (SIT), 1999. http://www.slicing-infotech.de.

[27] J.D. Davidson and D. Coward. Java Servlet Specification 2.2. Technical report, Sun Microsystems, Inc., December 1999. http://java.sun.com/products/servlet/.

[28] P. de Bra. Hypermedia Structures and Systems: Online Course at Eindhoven University of Technology, 1997. http://wwwis.win.tue.nl/2L690/.

[29] J.L. Dionne. Redefining the textbook: the impact of electronic custom publishing. *Logos*, 2(4), 1991.

[30] R. A. Duschl and D. H. Gitomer. Epistemological Perspectives on Conceptual Change: Implications for Educational Practice. *Journal of Research in Science Teaching*, 26(9):839–858, 1991.

[31] J. Eklund. Knowledge-Based Navigation Support in Hypermedia Courseware using WEST. In *Australian Educational Computing*, volume 11(2), 1996.

[32] J. Eklund, P. Brusilovsky, and E. Schwarz. Adaptive Textbooks on the World Wide Web. In *Third Australian World Wide Web Conference*, Queensland, Australia, July 1997.

[33] Y. Arens et.al. Retrieving and Integrating Data from Multiple Information Sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2), 1993.

[34] eXcelon Corp., Inc. *ObjectStore Java Interface Release 6.0 Service Pack 3 Bookshelf*. eXcelon Corporation, Inc, Massachusetts, USA, März 2000.

[35] Excelon Corp., Inc. ObjectStore white papers. Technical report, eXcelon Corp., Inc., 2001.
http://www.exceloncorp.com/formapp/createform.asp?formnum=135.

[36] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. *International Journal of Human-Computer Studies*, 46:707–728, 1997.

[37] Firma WebHits. Web-Statistiken. Technical report, Fa. WebHits, 2001.
http://www.webhits.de/datsh/webstats.html.

[38] A. Forst. Wissen als betriebliche Ressource. In *Aufbruch ins Wissensmanagement, 21. Online-Tagung der DGI*, pages 175–182, Frankfurt am Main, 1999. Deutsche Gesellschaft für Informationswissenschaft und Informationspraxis.

[39] M. Fowler and K. Scott. *UML konzentriert*. Addison-Wesley-Longmann, Bonn, Deutschland, 1998.

[40] J.S. Fritzinger and M. Mueller. Java Security. Technical report, Sun Microsystems, Inc., 1995. http//java.sun.com/security/whitepaper.ps.

[41] J. Gamper, W. Nejdl, and M. Wolpers. Combining Ontologies and Terminologies in Information Systems. In *5th International Congress on Terminology and Knowledge Engineering*, Innsbruck, Austria, August 1999.

[42] A. Gangemi, D. M. Pisanelli, and G. Steve. Ontology Integration: Experiences with Medical Terminologies. In Nicola Guarino, editor, *Proceedings of the First International Conference on Formal Ontology in Information Systems*, pages 163–178, Trento, Italy, June 1998. IOS Press.

[43] F. Garzotto, D.Schwabe, and P. Paolini. HDM - A Model Based Approach to Hypermedia Application Design. *ACM Transactions on Information Systems*, 11(1), 1993.

[44] F. Garzotto, L.Mainetti, and P. Paolini. Hypermedia Design, Analysis, and Evaluation issues. *Communications of the ACM*, 38(8), 1995.

[45] P. Giaretta and N. Guarino. Ontologies and Knowledge Bases - Towards a Terminological Clarification. In N. Mars, editor, *Towards Very Large Knowledge Bases*. IOS Press, Amsterdam, 1995.

[46] J. Goldberg. Why web usage statistics are (worse than) meaningless. Technical report, Cranfield Computer Center, Cranfield University, Cranfield University, UK, 2001.
http://www.cranfield.ac.uk/docs/stats/.

[47] M. T. Goodrich and R. Tamassia. *Data Structures and Algorithms in Java*. John Wiley and Sons, Inc., New York, USA, 1998.

[48] J. Gosling and H. McGilton. The Java Language Environment – a white paper. Technical report, JavaSoft/Sun, Mountain View, California, USA, May 1996.
http://java.sun.com/docs/white/langenv.

[49] T. Gruber. Towards Principles for the Design of Ontologies used for Knowledge Sharing. In R. Poli, editor, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1994.

[50] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[51] N. Guarino. Understanding, Building, and using Ontologies. *Internaltional Journal of Human and Computer Studies*, 46(2/3), 1997.
http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html.

[52] N. Guarino. Formal Ontology and Information Systems. In N. Guarino, editor, *Formal Ontology in Information Systems*. IOS Press, 1998.

[53] N. Henze. *Adaptive Hyperbooks: Adaptatin for Project-Based Learning Resources*. PhD thesis, Fachbereich Mathematik und Informatik der Universität Hannover, Hannover, 2000.

[54] N. Henze and W. Nejdl. Constructivism in Computer Science Education: Evaluating a Teleteaching Environment for Project Oriented Learning. In *Workshop on Interactive Computer Aided Learning - Concepts and Applications*, Villach, Österreich, October 1998.

[55] N. Henze and W. Nejdl. Student Modeling in an Active Learning Environment using Bayesian Networks. Technical report, University of Hannover, November 1998.

[56] N. Henze and W. Nejdl. Adaptivity in the KBS Hyperbook System. In *2nd Workshop on Adaptive Systems and User Modeling on the WWW*, Toronto, Canada, Mai 1999.

[57] N. Henze, W. Nejdl, and M. Wolpers. Modeling constructivist teaching functionality and structure in the KBS Hyperbook System. In *CSCL'99: Computer Supported Collaborative Learning*, Stanford, USA, Dezember 1999. Erschien auch in einer Vorversion auf dem AIED'99 Workshop on Ontologies for Intelligent Educational Systems, July 1999, Le Mans, Frankreich.

[58] N. Henze, W. Nejdl, and M. Wolpers. Modeling constructivist teaching functionality and structure in the KBS Hyperbook System. In *AIED'99 Workshop on Ontologies for Intelligent Educational Systems*, Le Mans, Frankreich, 1999.

[59] M. E. Hodges and R. M. Sasnett, editors. *Multimedia Computing: Case Studies from MIT Project ATHENA*. Addison-Wesley, 1993.

[60] E. Horowitz, S. Sahni, and S. Anderson-Freed. *Fundamentals of Data Structures in C*. Computer Science Press, New York, USA, 1993.

[61] W. Horton. *Designing and Writing Online Documentation*. John Wiley and Sons, 1994.

[62] K. Hudalla. Ein Modellierungs- und Entwicklungssystem für Hyperbooks. Master's thesis, Universität Hannover, Hannover, Deutschland, 1999. http://www.kbs.uni-hannover.de/Arbeiten/Diplomarbeiten/.

[63] T. Isakowitz, E. A. Stohr, and P. Balasubrahmanian. RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8), August 1995.

[64] ISO/IEC. *Information technology - Information Resource Dictionary System (IRDS) framework, ISO/IEC 10027*, 1990 (E).

[65] ISO/IEC. *Information technology - Information Resource Dictionary System (IRDS) Services Interface, ISO/IEC 10728*, 1993 (E).

[66] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley-Longman, Reading, Massachusetts, Januar 1999.

[67] M. Jarke, R. Gallersdörfer, M. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase - a deductive object base for meta data management. *Journal on Intelligent Information Systems*, 4(2):167 – 192, 1995.

[68] M. Jarke, M. Jeusfeld, and C. Quix. ConceptBase V5.1 User Manual. Technical report, RWTH Aachen, 1999. http://www-i5.informatik.rwth-aachen.de/CBdoc/userManual/.

[69] Java Programming Language, 2001. http://www.javasoft.com.

[70] M. Jeusfeld. *Änderungskontrolle in deduktiven Objektbanken*. Infix-Verlag, St. Augustin, Deutschland, 1992.

[71] M. A. Jeusfeld. What is O-Telos? Technical report, Tilburg University, Infolab, Niederlande, April 1999. http://www-i5.informatik.rwth-aachen.de/CBdoc/O-Telos.html.

[72] J. Kay and R. J. Kummerfeld. An Individualised Course for the C Programming Language. In *Proc. of the 2nd International World Wide Web Conference*, Chicago, USA, Oktober 1994.

[73] C. Keep and T. McLaughlin. The Electronic Labyrinth. Technical report, Institute for Advanced Technology in the Hunanities, University of Virginia, Virginia, 1997.
http://jefferson.village.virginia.edu/elab/.

[74] Knowledge Interchange Format (KIF). http://logic.stanford.edu/kif/kif.html.

[75] G. P. Landau. Course Assignments using hypertext: The example of INTERMEDIA. *Journal of Research on Computing in Education*, 3(21):349–363, 1989.

[76] G. P. Landow and P. Kahn. Where's the hypertext? The Dickens web as a system-independent hypertext. In *Proceedings of the 4th ACM ECHT Conference on Hypertext*, pages 149–160, Mailand, Italien, Dezember 1992. ACM.

[77] O. Lassila. Web Metadata: A Matter of Semantics. *IEEE Internet Computing*, 2(4):30–37, 1998.

[78] D. Lowe and W. Hall. *Hypermedia and the Web*. J. Wiley and Sons, 1999.

[79] MetaGer Suchmaschine. Der Browserkrieg. Technical report, Regionales Rechenzentrum Niedersachsen (RRZN), Niedersachsen, Deutschland, 2001.
http://www.metager.de/browser.html.

[80] M. Milosavljevic, A. Tulloch, and R. Dale. Text Generation in a Dynamic Hypertext Environment. In *Proceedings of the 19th Australasian Computer Science Conference*, Melbourne, Australia, January 31 - February 2 1996.

[81] P. Müller. Writing hypertext books. Technical report, FU Berlin, 1995.
http://www.inf.fu-berlin.de/tec/Mosaic/HTB/.

[82] K. Moss. *Java Servlets*. McGraw-Hill Comp., New York, USA, 1999.

[83] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: A language for representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4), 1990.

[84] K. Naceur. Annotationssysteme für adaptive Hyperbooks. Master's thesis, Universität Hannover, Hannover, Deutschland, 1998.
http://www.kbs.uni-hannover.de/Arbeiten/Diplomarbeiten/.

[85] W. Nejdl. Grundzüge der Informatik 1 (GdI 1) – Programmieren 2, 2001.
http://www.kbs.uni-hannover.de/Lehre/Info1/informatik1.html.

[86] W. Nejdl, H. Dhraief, and M. Wolpers. O-Telos-RDF: A resource description format with enhenced meta-modeling functionalities based on o-telos. Technical report, Institut f. Technische Informatik, FG Rechnergestützte Wissensverarbeitung, Universität Hannover, 2001.
http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2001/kcap01-workshop.ps.

[87] W. Nejdl and M. Wolpers. KBS Hyperbook – a data-driven information system on the web. Technical report, University of Hannover, November 1998.
http://www.kbs.uni-hannover.de/paper/99/www8.

[88] W. Nejdl, M. Wolpers, and C. Capelle. The RDF Schema Specification Revisited. In *Proc. Modellierung 2000*, St. Goar, Germany, April 2000.
http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2000/modeling2000/wolpers.pdf.

[89] T. Nelson. Replacing the printed word: A complete literary system. In S. Lavington, editor, *Proceedings IFIP Congress*, pages 1013–1023, Niederlande, 1980.

[90] C. Niederée, U. Steffens, H.-W. Sehring, F. Matthes, and J. W. Schmidt. On Indexing in Digital Libraries: Cooperation, Personalization, and Evolution. Technical report, Technical University Hamburg-Harburg, June 1998.

[91] J. M. Ockerbloom. *The on-line books page*. University of Pennsylvania, Pennsylvania, USA, 1993.
http://digital.library.upenn.edu/books/.

[92] L. Pazzi. Three Points of view in the Characterization of Complex Entities. In N. Guarino, editor, *Formal Ontology in Information Systems*. IOS Press, 1998.

[93] D. Raggett, A. Le Hors, and I. Jacobs. Hypertext Markup Language (HTML) 4.01 Specification. Technical report, World Wide Web Consortium (W3C), 1999.
http://www.w3.org/TR/1999/REC-html401-19991224/.

[94] P. Reiman, K. Müller, and P. Starkloff. Kognitiv kompatibel? - Wissensmanagement: Brückenschlag zwischen Technik und Psyche. *c't*, 4:274, 2000.

[95] D. Renoult. The digital collections of the bibliotheque nationale de france: An experiment on internet. In *IEEE International Conference on the Advances in Digital Libraries*, Washington, USA, 1997.

[96] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *[Object-Oriented Modeling and Design (OMT)*. Prentice Hall, 1991.

[97] J. Saarela. *SiRPAC - simple RDF parser & compiler*. World Wide Web Consortium (W3C), 2001. http://w3c.org/RDF/Implementations/SiRPAC.

[98] J. C. Sager. Terminology: Custodian of knowledge and means of knowledge transfer. *Terminology*, 1(1):7–16, 1994.

[99] S. R. Schach. *Classical and Object-Oriented Software Engineering*. Richard D. Irwin Inc., Chicago, USA, 1996.

[100] R. Schank and C. Cleary. *Engines for Education*. Lawrence Erlbaum Associates, 1994.

[101] S. Schäfer. *Objektorientierte Entwurfsmethoden: Verfahren zum objektorientierten Softwareentwurf*. Addison-Wesley GmbH, 1994.

[102] K. Schubert. Das VINETA-Projekt, Januar 2000. http://www.wi.fh-flensburg.de/tue/schubert/KSProj/KSProjVineta.htm.

[103] D. Schwabe, G. Rossi, and S. D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Proceedings of the Seventh ACM Conference on Hypertext*, Washington DC, March 1996.

[104] E. E. Steen. Ein System zur Visualisierung und zur Verwaltung von Objektmodellen. Master's thesis, Universität Hannover, Hannover, Deutschland, 1999. http://www.kbs.uni-hannover.de/Arbeiten/Diplomarbeiten/.

[105] A. Steinacker, C. Seeberg, K. Reichenbacher, S. Fischer, and R. Steinmetz. Dynamically generated tables of contents as guided tours in adaptive hypermedia systems. In *Proceedings of the ED-Media Conference*, Seattle, USA, 1999.

[106] Sun Microsystems, Inc. *Java Developement Kit (JDK) Applciation Programming Interface*. Sun Microsystems, Inc., jdk 2, version 1.3 edition. http://java.sun.com/j2se/1.3/docs/api/.

[107] Sun Microsystems, Inc. *Java Compiler Compiler (JCC) Documentation*. Sun Microsystems, Inc. and WebGain Co., October 2000. http://www.webgain.com/products/metamata/java_doc.html.

[108] Sun Microsystems, Inc. The JavaWebServer 2.0. Technical report, Sun Microsystems, Inc., 2001. http://www.sun.com/jwebserver/techinfo/doc/.

[109] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6(3), September 1941.

[110] TogetherSoft Corp., Inc. *Together Version 4.2 Documentation*. TogetherSoft Corp., Inc., November 2000. http://www.togethercommunity.com.

[111] M.A.S. Turine, M.C.F. de Oliveira, and P.C. Masiero. A Navigation-Oriented Hypertext Model Based on Statecharts. In *Proceedings of the Eighth ACM International Hypertext Conference*, Southampton, UK, 1997.

[112] J. D. Ullmann. *Principles of database and knowledgebase systems*. Principles of computer science series. Computer Science Press, Inc., Maryland, USA, 1988.

[113] P. E. van der Vet. Bottom-up Construction of Ontologies. *IEEE Trans. on Knowledge and Data Engineering*, 10(4), July/August 1998.

[114] Verlag McGraw-Hill. Primis, 1989. http://www.mhhe.com/primis.

[115] Projektverbund Virtual Campus (VC). http://www.uni-hildesheim.de/zfw/vc.

[116] W3C Working Group. W3C Resource Description framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/REC-rdf-syntax/, February 1999.

[117] B. T. Watkins. San Diego campus and McGraw-Hill create custom texts. *The Chronicle of Higher Education*, 38, November 1991.

[118] G. Weber. Episodic Learner Modeling. *Cognitive Science*, 20, 1996.

[119] G. Weber and A. Möllenberg. ELM programming environment: A tutoring system for lisp beginners. In K. Wender, F. Schmalhofer, and H.-D. Böcker, editors, *Cognition and Computer Programming*. Ablex Publishing Corp., 1995.

[120] G. Weber and M. Specht. User Modeling and Adaptive Navigation Support in WWW-Based Tutoring Systems. In *Proceedings of the Sixth International Conference on User Modeling, UM97*, Sardinia, Italy, 1997.

[121] G. Wiederhold. Interoperation, Mediation, and Ontologies. In *Proc. International Symposium on 5th Generation Computer Systems (FGCS94), Workshop on heterogeneous Cooperative Knowledge-Bases*, volume W3, Japan, December 1994. ICOT.

[122] World Wide Web Consortium (W3C). Common Gateway Interface (CGI). http://www.w3c.org/CGI.

[123] World Wide Web Consortium (W3C). http://www.w3c.org.

Anhang C

Lebenslauf
Martin Wolpers

Geboren
14. Februar 1969 in Langenhagen

Wissenschaftlicher Werdegang

Mai 1989	Abitur am Gymnasium Langenhagen
Okt. 1989 - Juni 1997	Studium der Elektrotechnik an der Universität Hannover
23. Juni 1997	Diplom der Elektrotechnik
seit August 1997	Wissenschaftlicher Mitarbeiter am Institut für Technische Informatik, Abteilung Rechnergestützte Wissensverarbeitung, Universität Hannover
seit August 2001	Wissenschaftlicher Mitarbeiter am Learning Lab Lower Saxony, Universität Hannover
30. November 2001	Promotion im Fachbereich Elektrotechnik und Informationstechnik der Universität Hannover