Umsetzung eines semantischen Modells in DAML+OIL zur Steuerung einer Web-Anwendung

Baccalaureatsarbeit

Fachbereich Informatik
Universität Hamburg

Moritz Kleine
8kleine@informatik.uni-hamburg.de
Matrikel-Nr.: 5110496

Betreuer: Dr. Ralf Klischewski
Arbeitsbereich Softwaretechnik

Hamburg, 21. August 2003

Inhaltsverzeichnis

0 Einleitung

Diese Baccalaureatsarbeit ist im Rahmen des Hauptstudienprojektes *Software, Wissen und Organisationen* entstanden. Eine der Zielsetzungen des Projektes war es, einen Prototyp eines Bürger-Informations-Services unter dem Gesichtspunkt zu entwickeln, "Benutzern möglichst einfach und flexibel Sachinformationen bezogen auf den von ihnen offengelegten persönlichen Kontext zu vermitteln" [FI03].
Die Vorlage für den Prototyp war der *Direkte Bürger-Informations-Service* der Hansestadt Hamburg (DiBIS)[1].

Während des Projektverlaufs wurde entschieden, exemplarisch für Benutzeranliegen, die der Bürger-Informations-Service darstellen soll, das Benutzeranliegen *Umzug* zu modellieren. Als Grundlage für die Modellierung wurde das Resource Description Framework (RDF)[2] gewählt. In dieser Arbeit wird der Ansatz verfolgt, dass ein Benutzerkontext einen RDF-Graphen darstellt, der jedoch durch eine Ontologie (siehe Kapitel 1.2) beschrieben werden muss. Zur Formulierung der Ontologie wird in dieser Arbeit die Markup-Sprache DAML+OIL[3] verwendet.

Das Systemverhalten des Prototypen während einer Benutzersitzung wird durch die Auswertung des semantischen Modells gesteuert. Dafür muss der Prototyp das semantische Modell des Umzugs im Hinblick auf Zusammenhänge zwischen Informationseinheiten und vom Benutzer eingegebenen Daten auswerten. Die Systemausgaben sollen wie eingangs erwähnt kontextabhängig sein und auch die Vollständigkeit von Informationseinheiten wahren.

Damit die Ontologie, die das Benutzeranliegen (die *Umzugsontologie*) beschreibt, von dem Systemkern interpretiert werden kann, ist eine weitere Ontologie (die *Systemontologie*) nötig, die auf die erstgenannte Ontologie Bezug nimmt. Anhand der Systemontologie wird ein RDF-Graph erzeugt, dessen Knoten Elemente der Umzugsontologie referenzieren und auf diese Weise mögliche Systemabläufe (wie z.B. das Weiterleiten von Kontextdaten) oder die Semantik spezieller Klassen der Umzugsontologie definieren. Dieser RDF-Graph, zusammengenommen mit der Umzugsontologie und der Systemontologie, beschreibt das *semantische Modell*, das zur Steuerung der Web-Anwendung genutzt wird.

Die zentralen Punkte dieser Arbeit sind

1. die Architektur des Prototypen im Hinblick auf diejenigen Komponenten zu diskutieren, die das semantische Modell verarbeiten und

2. die Umzugsontologie und die Systemontologie, sowie denjenigen RDF-Graphen vorzustellen, der gemeinsam mit den beiden Ontologien das semantische Modell bildet.

Die hierbei verfolgte Fragestellung lautet:

[1] http://dibis.dufa.de
[2] http://www.w3.org/RDF/
[3] http://www.daml.org/

> Wie sind die Systemontologie und der Systemkern des Prototypen zu strukturieren, damit die Auswertung des semantischen Modells zu aus der Sicht eines Redakteurs korrekten kontextabhängigen Systemausgaben führt?

Um die verwendeten Begrifflichkeiten zu klären, wird im Kapitel 1 auf Web-Anwendungen und Ontologien eingegangen. Im Abschnitt 1.1 werden die Begriffe Web Site, Web Service und Web-Anwendung voneinander abgegrenzt und der Prototyp der Web-Anwendung in diese Begrifflichkeiten eingeordnet. Anschliessend wird erläutert, was semantische Modelle und Ontologien sind und der Unterschied zwischen einer Ontologie und einem semantischen Modell hervorgehoben (Abschnitt 1.2). Dieses Kapitel schliesst im Unterabschnitt 1.3 mit einer Einführung in DAML+OIL.

Im Abschnitt 2 wird der Prototyp vorgestellt, der im Projektverlauf entwickelt wurde. Anforderungen, die die Web-Anwendung erfüllen sollte, werden im Unterabschnitt 2.1.1 vorgestellt. Dabei wird darauf eingegangen, welche dieser Punkte durch den Prototyp umgesetzt wurden. Daraufhin wird ein Überblick über die Architektur des Prototypen gegeben (Abschnitt 2.1.2). Der nächste Unterabschnitt (2.2) stellt grundlegende Eigenschaften des Prototypen vor. Beispielsweise wird in 2.2.3 skizziert, wie die Steuerung der Web-Anwendung als Prozess des Systemkerns funktioniert.

Im Kapitel 3 wird das semantische Modell des Benutzeranliegens "Umzug" anhand seiner Darstellung in DAML+OIL vorgestellt. Es wird auf die Aufteilung des Modells in Ontologien und Instanzen eingegangen und gezeigt, wie das Systemverhalten durch das semantische Modell beeinflusst werden kann. Im Unterabschnitt 3.3 wird ein Ausschnitt des semantischen Modells vorgestellt, der eine Datenweiterleitung anhand der Umzugs- und der Systemontologie festlegt.

Den Abschluss dieser Arbeit (Abschnitt 4) bildet eine Einschätzung dieses Ansatzes hinsichtlich der Einsetzbarkeit in einem Produktivsystem.

1 Semantische Modelle zur Unterstützung von Web-Anwendungen

Sowohl der Begriff der *Web-Anwendung* als auch die Begriffe *semantisches Modell* und *Ontologie* sind zur Zeit viel zitierte Begriffe, die jedoch häufig gar nicht und wenn doch, dann mitunter sehr unterschiedlich definiert werden.

In diesem Abschnitt wird zunächst auf Web Sites, Web Services und Web-Anwendungen eingegangen und der Zusammenhang zu dem Prototypen hergestellt. Darauf folgt eine Beschreibung des Begriffs des semantischen Modells. Anschliessend werden Ontologien vorgestellt und darauf eingegangen, wie Ontologien in diesem Fall eingesetzt werden. Das Kapitel schliesst mit einer Einführung in die Markup-Sprache DAML+OIL, in der semantische Modelle umgesetzt werden können.

1.1 Web Sites, Web Services und Web-Anwendungen

In dieser Arbeit werden die Begriffe *Bürger-Informations-Service*, *Web Service* und *Web-Anwendung* verwendet. Damit diese Begriffe klar voneinander abgegrenzt werden können, muss zunächst Einigkeit darüber bestehen, was eine *Web Site* ist.

1.1.1 Web Sites

Unter *Web Site* wird der WWW-Auftritt einer Firma, Person, Organisation oder Körperschaft verstanden, der unter einer bestimmten Internetadresse aufrufbar ist. Bei whatis.com ist die folgende Definition zu finden:

> "A Web site is a related collection of World Wide Web (WWW) files that includes a beginning file called a home page. A company or an individual tells you how to get to their Web site by giving you the address of their home page." [WSia]

Bei wikipedia.com ist eine ähnliche Definition zu finden:

> "A website or web site is a collection of web pages, that is, documents accessible via the World Wide Web on the Internet. The pages of a website will be accessed from a common root URL, the homepage [...]." [WSib]

Der Begriff Web Site ist also sehr allgemein. Die wichtigen Charakteristika einer Web Site sind:

1. Die Web Site kann von einem Browser dargestellt werden.
2. Die Web Site ist über einen URL adressierbar.
3. Sie kann einer juristischen Person zugeordnet werden.

1.1.2 Web Services

Das IBM Web Services Architecture Team versteht unter *Web Services* modular aufgebaute Anwendungen, die in einem Netzwerk plattformübergreifend angesprochen werden können [Tea00]. Zwischen *Web Services* sind die drei Operationen *Veröffentlichen*, *Auffinden* und *Binden* definiert, sowie die drei Rollen *Service-Anbieter*, *Service-Nutzer* und *Service-Vermittler*, wobei ein *Web Service* eine oder mehrere dieser Rollen einnehmen kann. Im Zusammenhang mit *Web Services* werden meistens die Technologien SOAP und WSDL genannt. WSDL ist eine XML-basierte Beschreibungssprache für Web Services, in welcher ein Web Service durch eine abstrakte Darstellung seiner Methoden, deren Parameter und Rückgabewerte und schliesslich der konkreten Bindung von Netzwerkendpunkten an Übertragungsprotokolle beschrieben wird. Als Übertragungsprotokoll wird im Zusammenhang mit WSDL in der Regel das *Simple Object Access Protocol* (SOAP) genannt, das auf den weit verbreiteten offenen Standards HTTP und XML beruht.
Im Gegensatz zu *Web Sites* besitzen *Web Services* also üblicherweise keine graphisch darstellbare Benutzungsoberfläche, sondern eine xml-basierte Schnittstelle zur Aussenwelt.

1.1.3 Bürger-Informations-Service

Die Bezeichnung *Bürger-Informations-Service* ist aus dem Namen der DiBIS-Web Site[4] abgeleitet und bezeichnet in dieser Arbeit eine Web Site, die Bürger bei behördlichen Anliegen oder bei Anliegen, die behördliche Aspekte umfassen, informiert. Diese Bezeichnung ist kein Fachbegriff der Informatik. Als Definition für die weitere Verwendung in dieser Arbeit soll gelten:

Bürger-Informations-Service Web Site, die für Bürger aus behördlicher Sicht relevante Informationen bezüglich eines vom Bürger vorgebrachten Anliegens ausliefert.

1.1.4 Web-Anwendung

Der Begriff *Web-Anwendung* wird zur Zeit unter verschiedenen Sichtweisen verwendet. Aus der technischen Perspektive eines Java-Programmierers ist eine Web-Anwendung ein Paket aus Servlets und Java-Server-Pages, das von einem Web-Server wie z.B. Tomcat[5] verarbeitet wird und eine von einem Browser darstellbare Ausgabe liefert [Weba]. Der in dieser Arbeit vorgestellte Prototyp ist in diesem Sinne eine Web-Anwendung.

Ausserdem hat es sich in den letzten Jahren durchgesetzt, den Begriff aus Benutzersicht zu definieren und all diejenigen Web Sites *Web-Anwendung* zu nennen, die mit der Intention entwickelt wurden, einen oder mehrere Benutzeraufträge zu erfüllen. Dabei werden Web-Anwendungen also von den Web Sites unterschieden, deren alleiniger Zweck es ist, Benutzern Texte oder Bilder anzuzeigen. Die folgenden Charakteristika für Web-Anwendungen, beschreiben den Begriff so, wie er in dieser Arbeit verwendet wird:

[4] http://dibis.dufa.de
[5] http://jakarta.apache.org/tomcat

- Die Benutzungsschnittstelle wird von einem Browser dargestellt. [Win00]
- Die Web-Site ist interaktiv und führt Aufträge des Benutzers aus. [Webb]

Baxley beschreibt Web-Anwendungen als Unterklasse von Web Sites. Er formuliert als das wichtigste Charakteristikum von Web-Anwendungen:

> "The fundamental purpose of all web applications is to facilitate the completion of one or more tasks." [Bax03]

Für die weitere Verwendung in dieser Arbeit wird definiert:

Web-Anwendung Anwendung, die mit einem Web Browser über ein Netzwerk aufgerufen werden kann. Sie kann interaktiv Benutzeraufträge entgegennehmen und diese ausführen.

In dieser Arbeit geht es um den Prototyp eines Bürger-Informations-Services, der es anbietet, personenbezogene Daten von Benutzern zu speichern oder zu manipulieren und diese auf Wunsch des Benutzers an Web Services weiterzuleiten. Ein Anwendungsaspekt des Prototypen wird also dadurch umgesetzt, dass er Web Services integriert.

1.2 Semantische Modelle und Ontologien

Semantische Modelle werden in vielen Wissenschaften eingesetzt um Sachverhalte zu veranschaulichen oder abstrakt darzustellen. Der Begriff Ontologie stammt aus der Philosophie und bezeichnet die Lehre des Seienden, also die Lehre dessen, was ist und was nicht ist.[6] "Im Gegensatz dazu handeln die faktischen, die Natur- und Sozialwissenschaften davon, *wie* etwas ist."[SN00] Die Informatik hat sich den Begriff aus der Philosophie entlehnt und seine Bedeutung leicht verändert.

1.2.1 Semantische Modelle

Bevor definiert wird, was ein semantisches Modell ist, soll ein Beispiel zur Veranschaulichung gegeben werden:
Der Begriff des semantischen Modells wird häufig im Zusammenhang mit Informationsmodellierung bei der Erstellung von Datenbank-Schemata genannt. In dem Zusammenhang bezeichnet ein semantisches Modell die Abbildung des Bereichs, der in der Datenbank dargestellt werden soll, auf ein abstraktes Modell, z. B. das Entity-Relationship-Model (ERM). Das semantische Modell beinhaltet also ein allgemeines Modell und eine konkrete Umsetzung des allgemeinen Modells.

In diesem Text gelten die folgenden Definitionen:

Instanz Eine Instanz ist eine konkrete Ausprägung bzw. ein konkretes Auftreten einer allgemein beschriebenen Sache oder eines Sachverhalts.[7]

[6] Dem interessierten Leser sei an dieser Stelle die Web Site http://www.formalontology.it empfohlen.
[7] Siehe zu Instanzen auch Abschnitt 1.3.1

semantisches Modell Ein semantisches Modell bezeichnet eine Abbildung eines wohldefinierten Ausschnittes der Welt in eine textuelle, symbolische oder räumliche Darstellung, deren Elementen aufgrund eines abstrakten Modells eine bestimmte Bedeutung zugeordnet ist. Ein semantisches Modell beinhaltet also

1. ein abstraktes Modell und
2. eine Instanz des abstrakten Modells.

Im weiteren Verlauf dieser Arbeit wird auch von *dem* semantischen Modell gesprochen. Damit ist dann die auf das Fallbeispiel bezogene Definition von Seite 9 gemeint.

1.2.2 Ontologien

In der Informatik bezeichnet eine Ontologie eine formale Spezifikation eines repräsentativen Vokabulars für einen gemeinsamen Wissensbereich – also einen wohldefinierten Ausschnitt der Welt betreffend – verbunden mit einer Sammlung von Aussagen über diese Vokabeln, die die Vokabeln zueinander in Beziehung setzen. Tom Gruber definiert dies mit den Worten:

> "A specification of a representational vocabulary for a shared domain of discourse – definitions of classes, relations, functions, and other objects – is called an ontology." [Gru93]

Eine allgemeinere aber weithin anerkannte und oft zitierte Definition des Begriffs Ontologie ist die Folgende von Uschold und Grüninger:

> "An Ontology is an explicit account or representation of some part of a conceptualisation. [UG96]"

Uschold fügt dieser Definition erklärend hinzu:

> "Broadly, a conceptualisation is a world view; it corresponds to a way of thinking about some domain. [Usc96]"

Etlichen Definitionen von Ontologien liegt die oben genannte Definition von Gruber zugrunde. Beispielsweise schreibt Dieckmann:

> "Unter einer Ontologie wird [...] eine Sammlung und Strukturierung zusammengehöriger Begriffe verstanden. Die in einer Ontologie zusammengeführten Begriffe werden [...] in definierte Beziehungen gebracht."[Die03]

Diesen Definitionen ist gemeinsam, dass eine Ontologie ein Vokabular und Zusammenhänge zwischen den definierten Vokabeln beinhaltet. Die geforderten Zusammenhänge werden meistens benannt als eine Einteilung der Vokabeln in Klassen und Eigenschaften bzw. Zusammenhängen zwischen den Klassen. Von den Klassen ist meist gefordert, dass sie in einer hierarchischen Ordnung mit multipler Vererbung angeordnet sind [SN00]. Instanzen, also das konkrete Vorkommen eines Objektes einer Klasse oder die konkrete Ausprägung einer beschriebenen Eigenschaft, sind nicht Teil einer Ontologie [NM].

Eine Ontologie bezeichnet in dieser Arbeit ein Vokabular für eine Diskurswelt, das die in der folgenden Definition genannten Eigenschaften besitzt:

Ontologie Ein für eine bestimmte Diskurswelt entwickeltes Vokabular mit den Eigenschaften

- Das Vokabular teilt sich in Klassen und Eigenschaften.
- Die Klassen sind hierarchisch (mit Mehrfachvererbung) angeordnet.
- Die Eigenschaften setzen Klassen zueinander in Beziehung.

1.2.3 Einsatzbereiche für Ontologien

Uschold und Grünginger nennen Kommunikation, Interoperationabilität und Systementwicklung als Einsatzbereiche für Ontologien [UG96]:

1. Im Bereich der zwischenmenschlichen Kommunikation ermöglichen es Ontologien, eine gemeinsame Terminologie zu verwenden.
2. Ontologien können von verschiedenen Systemen zur "Datenübersetzung" genutzt werden und in diesem Sinne Interoperationabilität unterstützen.
3. Ontologien unterstützen Systementwicklung dadurch, dass sie
   (a) die Systemspezifikation repräsentieren können.
   (b) Verifizierbarkeit des System auch bei evolutionärer Entwicklung ermöglichen.
   (c) Beziehungen und Prozesse innerhalb des Systems repräsentieren können.

Der Prototyp, der im Folgenden vorgestellt wird, setzt Ontologien in dem zu letzt genannten Bereich ein. Die Ontologien legen die möglicherweise auftretenden Daten und deren Beziehungen zueinander fest. Ausserdem wird eine Ontologie verwendet, um es zu ermöglichen, Abläufe über den Daten zu formulieren.

1.2.4 Ontologien und Modelle

Nach dieser kurzen Einführung von Ontologien mag man sich fragen, ob zwischen einem Modell und einer Ontologie überhaupt ein Unterschied besteht, da unter einem Modell im Allgemeinen eine Abstraktion eines wohldefinierten Ausschnittes der Welt bzw. der Realität verstanden wird. Für diese Arbeit wichtige Unterschiede sind, dass Modelle nicht notwendiger Weise Vokabulare definieren und durchaus Instanzen mit einbeziehen können. Im Bezug auf Modellierung kann eine Ontologie als Metamodell aufgefasst werden, das Eigenschaften des Modells definiert [SN00].

Dem semantische Modell des Umzugs liegt eine Ontologie zugrunde, die eine Klassenhierarchie und Beziehungen zwischen den Klassen definiert. Diese wird in dieser Arbeit **Umzugsontologie** genannt. Sie dient in erster Linie der Beschreibung von Kontextdaten und deren Eigenschaften. Zur Steuerung der Web-Anwendung erfordert es jedoch nicht nur Modelle der Benutzeranliegen, sondern auch eine systemintere Ontologie, die für das System

die Semantik der Ontologien der Benutzeranliegen darstellt. Diese wird in dieser Arbeit als **Systemontologie** bezeichnet.

Das semantische Modell des Umzugs enthält sowohl die Umzugsontologie als auch die Systemontologie und zusätzlich eine Menge von Instanzen, die zu Klassen gehören, die in der Systemontologie definiert sind, die jedoch gleichzeitig auf Klassen der Umzugsontologie verweisen. Anfallende Kontextdaten sind Instanzen von Klassen der Umzugsontologie. Diese werden hier jedoch nicht zu dem semantischen Modell des Umzugs gezählt.

Es sollen im weiteren Verlauf dieses Textes die folgenden Definitionen gelten:

Anliegenontologie: redaktionell erzeugte Ontologie, die die Begrifflichkeiten eines Benutzeranliegens beschreibt.

Umzugsontologie: eine Anliegenontologie. Die einzige, die im Rahmen des Projektes erzeugt worden ist.

Systemontologie: dem Systemkern bekannte Ontologie, die vom Systementwickler erzeugt wird.

Das semantische Modell besteht aus einem RDF-Modell, das auf Grundlage der Systemontologie über Begriffe der Umzugsontologie erzeugt wurde, in Verbindung mit der Umzugs- und der Systemontologie.

1.3 Einführung in DAML+OIL

Dieser Abschnitt setzt Grundkenntnisse über die Extensible Markup Language (XML), XML-Schema, das Resource Description Framework[8] (RDF) sowie RDF-Schema (RDFS) voraus. Da DAML+OIL auf RDF und RDFS aufbaut, werden in dem ersten Unterabschnitt die wesentlichen Merkmale von RDF und RDFS beschrieben, und in dem zweiten Unterabschnitt wird schliesslich DAML+OIL vorgestellt.

1.3.1 RDF und RDFS

Die elementaren Elemente von RDF sind Statements, die sich informell ausgedrückt wie natürlichsprachliche Sätze durch die Aneinanderreihung von Subjekt, Verb und Objekt formulieren lassen. Standardbeispiele hierfür sind Sätze wie: "Diese Person heisst Tom". Dabei wären sowohl "Diese Person", "heisst" und "Tom" Ressourcen, in diesem konkreten Statement "diese Person" das Subjekt, "heisst" das Verb und "Tom" das Objekt. Lassila und Swick definieren dies in einer W3C Empfehlung [LS99] genauer:

Resource All things being described by RDF expressions are called resources.

Property A property is a specific aspect, characteristic, attribute, or relation used to describe a resource.

[8] http://www.w3.org/RDF/

Statement A specific resource together with a named property plus the value of that property for that resource is an RDF statement.

Eine Sammlung RDF-Statements bildet *ein* RDF-Modell. Dabei ist die Bezeichnung *ein* RDF-Modell von der Bezeichnung *das* RDF-Modell zu unterscheiden. Wenn nicht aus dem Kontext hervorgeht, welches konkrete RDF-Modell gemeint ist, bezeichnet *das* RDF-Modell allgemein die Art und Weise, in der Aussagen in RDF gemacht werden. Ein RDF-Modell kann als gerichteter Graph aufgefasst werden, in dem die Statements die Subjekt- und Objekt-Ressource mit einer vom Subjekt zum Objekt gerichteten (und benannten) Kante, der Property, verbinden. Abbildung 1 zeigt einen RDF-Graphen, der das oben genannte Beispiel dargestellt.

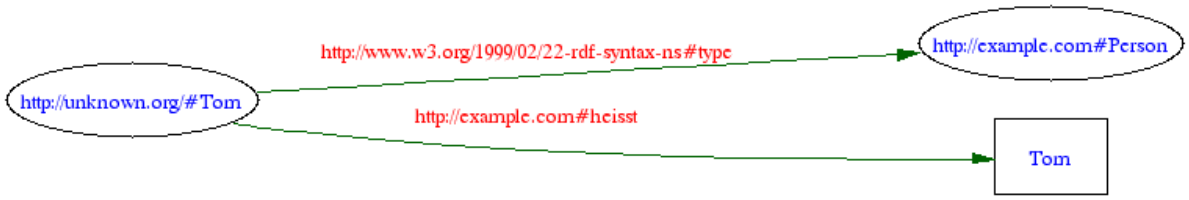

Abbildung 1: Darstellung des RDF/RDFS Beispiels durch den W3C RDF-Validator

Ein wichtiger Punkt, den Lassila und Swick in [LS99] nennen, ist der, dass RDF unabhängig von der Syntax ist, in der ein RDF-Modell repräsentiert wird. Sie führen zwar eine XML-Syntax für RDF ein, heben jedoch hervor:

> "It is also important to understand that this XML syntax is only one possible syntax for RDF [...]"

Die Bedeutung von einem RDF-Model kann durch ein RDF-Schema beschrieben werden. In einem RDF-Schema können Klassen durch Angabe der möglichen Properties beschrieben werden. Durch Subklassenbildung können Properties vererbt und Klasseneigenschaften ergänzt werden. In [BG02] werden wichtige Klassen wie z. B. Resource, Literal, Class, Statement und Container beschrieben und wichtige Properties vorgestellt, unter denen subClassOf, subPropertyOf, domain und range zu finden sind. Brickley und Guha schreiben in [BG02] über RDFS:

> "The RDF vocabulary description language defines classes and properties that can be used to describe other classes and properties. [...] RDF's vocabulary description language, RDF Schema, is specified in terms of the basic RDF information model - a graph structure describing resources and properties."

Ein RDF-Schema stellt selbst ein RDF-Modell [LS99] dar, das die Menge alle RDF-Graphen beschreibt, die passend zu ihm erzeugt werden können. RDF-Recourcen, die vom Typ einer Klasse sind, werden Instanzen genannt. Mike Dean schreibt in [Dea02]:

> "an instance is related to its **Class** using an **rdf:type** predicate"

In Abbildung 1 ist die Ressource `http://unknown.org/#Tom` eine Instanz vom Typ `http://example.com#Person`.

Die Zugehörigkeit einer Ressource zu einem bestimmten Schema wird in RDF wie in XML über Namensräume festgelegt. Lassila und Swick formulieren:

> "RDF also requires the XML namespace facility to precisely associate each property with the schema that defines the property [...]"

1.3.2 DAML+OIL

```
<?xml version='1.0'?> <rdf:RDF
    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
    xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
    xmlns:daml='http://www.daml.org/2001/03/daml+oil#'
    xmlns:xsd='http://www.w3.org/2000/10/XMLSchema#'
    xmlns:bsp='http://example.com#'
    xmlns='http://example.com#'>
    <daml:Ontology
         daml:versionInfo='1.0'>
        <daml:comment>Ein Beispiel</daml:comment>
    </daml:Ontology>
    <daml:Class rdf:ID='Person'/>
    <daml:DatatypeProperty rdf:ID='heisst'>
        <daml:domain rdf:resource='#Person'/>
        <daml:range
          rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
    </daml:DatatypeProperty>
    <bsp:Person rdf:ID='Tom'>
        <bsp:heisst>Tom</bsp:heisst>
    </bsp:Person>
</rdf:RDF>
```

Abbildung 2: Typischer Aufbau eines DAML+OIL-Dokuments

Die *DARPA Agent Markup Language + Ontology Inference Layer*[9] (DAML+OIL) baut auf dem RDF-Modell auf. Dieses beinhaltet die wichtige Feststellung, dass gültige DAML+OIL Beschreibungen ebenfalls als Graphen aufgefasst werden können, die demzufolge mit denselben graphentheoretischen Mitteln ausgewertet werden können, wie die von ihnen beschriebenen RDF-Graphen.

Konzepte, die RDFS zur Beschreibung von RDF-Graphen definiert, werden von DAML+OIL erweitert. Beispielsweise ist jede DAML+OIL-Klasse Subklasse einer RDFS-Klasse. In der

[9] http://www.daml.org/

DAML+OIL Sprachbeschreibung [DAMb] werden unter anderem die folgenden wichtigen Unterschiede zu RDFS genannt:

- Properties können auf Datenwerte (aus dem XML-Schema Typsystem) verweisen.
- Es können multiple Urbild- und Bildbereiche für Properties angegeben werden.
- Properties können mit Mengenangaben und -einschränkungen versehen werden.
- Klassen können durch Mengenoperationen auf Klassen erzeugt werden.
- Klassen können durch die abgeschlossene Aufzählung aller möglichen Instanzen definiert werden.

DAML+OIL bietet also Möglichkeiten, die denen von erweitertem Entity-Relationship-Model[10] ähnlich sind, nämlich eine Klassenhierarchie durch Mehrfachvererbung aufzubauen und Eigenschaften der Klassen mit Kardinalitätsrestriktionen zu versehen.

Formal ist die Syntax von DAML+OIL in XML beschrieben und unter dem XML-Namensraum `http://www.daml.org/2001/03/daml+oil#` zusammengefasst. Abweichend von der auf Seite 8 gegebenen Definition des Begriffs Ontologie, wird ein DAML+OIL-Modell auch als DAML+OIL-Ontologie bezeichnet. Eine DAML+OIL-Ontologie besteht aus einem Header, in dem Versionsnummer, eine natürlichsprachliche Beschreibung der Ontologie und Importanweisungen für andere DAML+OIL-Ontologien angegeben werden können, und aus den Class- und Property-Elementen, sowie Instanzen. [DAMa]

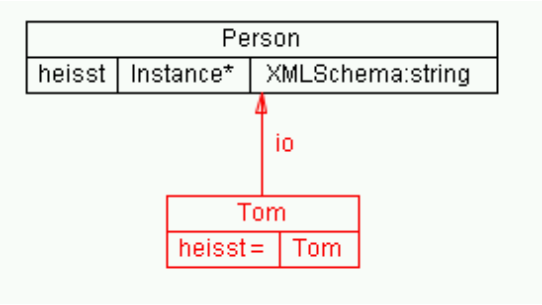

Abbildung 3: Darstellung des DAML+OIL Beispiels durch Protege2000

Da DAML+OIL eigene, RDF und RDFS-Elemente verwendet, sowie die Einbindung von XML-Schema-Datentypen vorsieht, werden diese Namensräume typischerweise zu Beginn der XML-Darstellung einer DAML+OIL-Ontologie bekanntgemacht. Werden weitere Ontologien importiert, kann ein Namensraum angegeben, der den Default-Namensraum der importierten Ontologie ersetzt. Durch diesen Mechanismus können die jeweiligen Elemente auch nach dem Import eindeutig den verschiedenen Ontologien zugeordnet werden.

[10] ERM erweitert um Abstraktionskonzepte und Kardinalitätsangaben.

Ein typischer Grundaufbau einer DAML+OIL-Ontologie in XML-Notation ist in Abbildung 2 dargestellt. Dieses Beispiel greift das Beispiel aus Abschnitt 1.3.1 auf. Es wird die Klasse Person definiert und die Eigenschaft von Personen, einen Namen zu haben, festgelegt. In dem Beispiel wird "Tom" sowohl als Referenz auf eine Instanz der Klasse Person, als auch als Wert für eine Eigenschaft der Instanz verwendet. Abbildung 3 stellt dies graphisch dar.

2 Fallbeispiel: Der Prototyp eines Bürger-Informations-Services

Die Web-Anwendung, die es zu entwickeln galt, ist als Erweiterung bestehender Internetportale wie z.B. Hamburg.de[11] gedacht und muss also ohne allzugrossen Aufwand in bereits existierende Systeme integrierbar sein. Die Vorlage für den Prototyp ist der *Direkte Bürger-Informations-Service* der Hansestadt Hamburg (DiBIS)[12], der in das Angebot von Hamburg.de integriert ist. Anhand der Vorlage wurde im Projektverlauf ein typisches Anliegen, welches Bürger in bezug auf behördliche Auskünfte bzw. An-, Um- und Abmeldungen haben können, gewählt. Die Wahl fiel hierbei auf das Anliegen *Umzug*, welches stellvertretend für andere Anliegen modelliert und von dem Prototyp dargestellt werden sollte.

Der Prototyp ist in Java implementiert worden.

In diesem Kapitel wird zunächst der allgemeine Funktionsumfang und ein Überblick über die Architektur des Prototypen gegeben. Anschliessend wird auf diejenigen Komponenten des Prototypen eingegangen, die für die Verwaltung und Auswertung des semantischen Modells relevant sind.

2.1 Funktionsumfang und Aufbau des Prototypen

Der Prototyp stellt einen Bürger-Informations-Service[13] dar, dessen Systemausgaben kontextualisiert sind. Das bedeutet, dass anzuzeigende Informationen unter Einbeziehung derjenigen Daten ausgewählt werden, die dem System vom Benutzer bekannt gemacht worden sind.

In diesem Abschnitt wird auf Anforderungen eingegangen, die der Protoyp erfüllen sollte und die die Entwicklung des Prototypen massgeblich beeinflusst haben und wird die Architektur des Prototypen vorgestellt.

2.1.1 Anforderungen an den Prototyp

Die Architektur des Prototypen ist darauf ausgelegt, die Anwendung unabhängig von eventuell verwendeten Content-Management-Systemen einsetzen zu können. Im Projekt wurde zwar CoreMedia[14] als Content-Management-System eingesetzt, die Java-Implementation des Systemkerns ist davon aber unabhängig. Die einzige Voraussetzung ist, dass der Webserver des Content-Management-Systems Servlets verarbeiten kann.

Zusätzlich zu einer möglichst grossen Unabhängigkeit von der Umgebung, in der das System eingesetzt werden könnte, ist der Prototyp des Bürger-Informations-Services im Hinblick auf die nachstehenden Anforderungen entwickelt worden:

1. Darzustellende Informationen werden in einem Redaktionsprozess eingegeben und strukturiert.

[11] http://www.hamburg.de
[12] http://dibis.dufa.de
[13] Siehe Definition auf Seite 5
[14] http://www.coremedia.com

2. Aus allen vorhandenen Informationen sollen bei einer Benutzeranfrage diejenigen Informationen herausgesucht und präsentiert werden, die zu dem von dem jeweiligen Benutzer offengelegten Kontext passen.

3. Kontextdaten[15] sollen zur Erledigung von Teilaufgaben des Benutzeranliegens an (externe) Web Services weitergeleitet werden können.

4. Benutzer sollen die Möglichkeit haben, diejenigen Daten einzusehen, zu speichern oder zu löschen, die sie im Laufe der Benutzung des Systems eingegeben oder gespeichert haben.

Nicht alle Anforderungen konnten letztendlich umgesetzt werden. Bezüglich Anforderung 1 ist zu bemerken, dass der Prototyp keine Unterstützung für einen Redaktionsprozess bietet. Sie ist jedoch als massgebliche Richtlinie in den Erzeugungsprozess für das semantische Modell eingegangen. Dieses hat dazu geführt, dass es die Systemontologie[16] gibt, denn sie kann es einem Nicht-Programmierer ermöglichen, Systemverhalten anhand der Umzugsontologie[17] festzulegen.

Die Umsetzung der Punkte 2 und 3 wird in dieser Arbeit beschrieben. Jedoch ist Punkt 3 nur teilweise umgesetzt worden. Die Systemontologie stellt Mechanismen bereit, eine Datenweiterleitung an externe Web Services zu beschreiben, es fehlt jedoch eine Komponente für die nötige Datenaufbereitung, genauso wie die SOAP-Schnittstelle, über die die Daten letztendlich weitergeleitet werden könnten.

Die letzte Anforderung (4) wurde mit der Einschränkung umgesetzt, dass Kontextdaten nicht persistent gespeichert werden.

2.1.2 Architektur des Prototypen

Einen schematischen Überblick über die Architektur gibt Abbildung 4. Dabei handelt es sich um den Stand der Architektur, der am Ende des ersten Semesters der Projektlaufzeit als vorläufiges Ergebnis präsentiert wurde. Diese Schichtenarchitektur wird im Folgenden kurz vorgestellt, zur korrekten Steuerung des Prototypen ist sie jedoch noch nicht detailliert genug. Im Rahmen dieser Arbeit ist eine verfeinerte Architektur entwickelt worden, die im Abschnitt 2.2.3 vorgestellt wird.

Die *Anzeige*-Komponente repräsentiert ein bereits bestehendes Internetportal. Middleware-Schichten und Datenbanken, auf denen ein Internetportal üblicherweise aufbaut, werden in dieser Darstellung ausgeblendet. Für den hier diskutierten Prototyp ist bezüglich der *Anzeige*-Komponente nur von Bedeutung, dass sie die durch einen Browser gestellte Anfrage entgegennimmt und an den Systemkern weiterreicht.

[15] Als Kontextdaten werden in diesem Zusammenhang all diejenigen Daten angesehen, die der Benutzer während der Nutzung des Informationsservices in Formularfelder einträgt oder die durch Auswertung des Klickpfades des Benuzters gewonnen werden können.

[16] Siehe Definition auf Seite 9

[17] Siehe Definition auf Seite 9

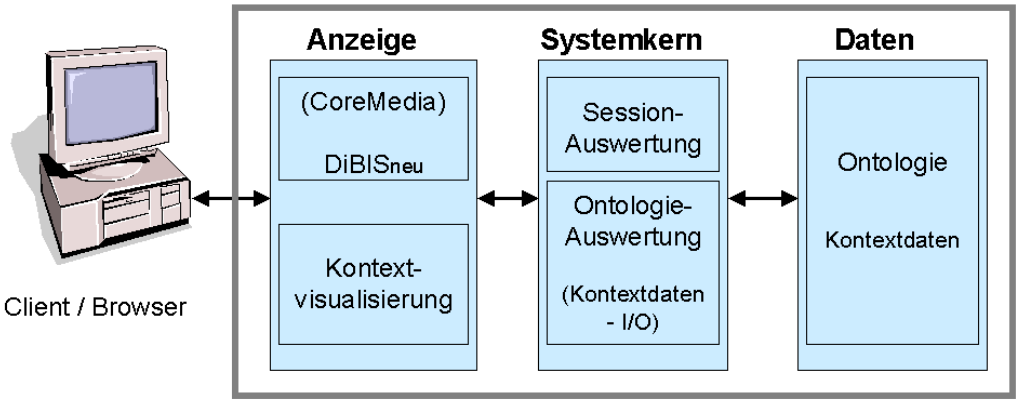

Abbildung 4: Architektur des Prototypen

Der Systemkern lädt für jede Sitzung, die von einem Benutzer eröffnet wird, die Systemontologie, die Umzugsontologie und die dazugehörigen Instanzen und ordnet dieses Modell der Sitzung zu. Wenn bereits ein Modell in der Sitzung vorhanden ist, extrahiert der Systemkern für ihn relevante Parameter aus der Anfrage und wertet das semantische Modell anhand dieser Parameter aus.
Dies geschieht in der Weise, dass das vom Web Server aufgerufene `HttpServlet`[18] das entsprechende `HttpServletRequest`-Objekt an den Systemkern weiterleitet. Das `HttpServletRequest`-Objekt stellt sowohl eine Methode bereit, die eine Referenz auf das jeweilige `HttpSession`-Objekt zurückgibt, als auch Methoden um die HTTP-Get- und HTTP-Post-Parameter zurückgeben. Diese Parameter enthalten eventuell eingegebene Formulardaten, also diejenigen Daten, die den Kontextdaten der aktuellen Session, also dem oben genannten Modell, hinzugefügt werden.

Da das `HttpServletRequest`-Objekt die Information beinhaltet, welcher Link oder welcher Button vom Benutzer angeklickt wurde, kann anhand der Kontextdaten, der Umzugsontologie und der Systemontologie bestimmt werden, welche Information als nächstes angezeigt werden soll. Der Systemkern wertet also das semantische Modell im Hinblick auf die gewünschten Informationen aus und generiert eine Sammlung an Verweisen. Diese gibt der Systemkern an die mit *Anzeige* beschriftete Komponente zurück, welche daraufhin die durch die Verweise referenzierten Informationen zusammenstellt und eine entsprechende Antwort an den Browser zurückliefert.

Diese Interaktion der drei Komponenten ist durch die bidirektionalen Pfeile vereinfachend dargestellt. Eine detailliertere Beschreibung des Zusammenspiels der Systemkern- und der Datenkomponente folgt in dem nächsten Unterabschnitt.

[18] enthalten im Java-Paket javax.servlet

2.2 Die Funktionsweise des Systemkerns

Der Einsatz von Ontologien wurde im Projektverlauf beschlossen, um Zusammenhänge von für einen Bereich relevanten Informationen modellieren zu können. Es wurde festgelegt, dass für jedes darzustellende Thema eine separate Ontologie vorhanden ist, welche auch die Grundlage für das Systemverhalten bezüglich des jeweiligen Themas darstellt. Allgemein ausgedrückt folgt daraus, dass die Ontologie zur Laufzeit als "Wegweiser" durch die Informationszusammenhänge genutzt wird.

2.2.1 Sitzungsverwaltung

Das Jena-Framework ermöglicht es, ein DAML+OIL-Modell zu erzeugen, in das die Ontologie und darauf basierende Instanzen geladen werden können. Dieses Modell ist dynamisch; es können also Daten hinzugefügt, verändert oder entfernt werden. Als Graph betrachtet, bedeutet dies, dass die Beschreibung des Graphen mit dem Graphen in einem gemeinsamen Speicherbereich gehalten wird. Grundsätzlich stehen unter diesen Vorraussetzungen zwei Möglichkeiten für die Sitzungsverwaltung zur Auswahl:

1. Für jeden Kontext wird ein eigenes Modell erzeugt.
2. Alle Kontexte werden in einem Modell verwaltet.

In diesem Fall geschieht die Sitzungsverwaltung nach der ersten Variante. Dafür kann die Session-Verwaltung des Web Servers genutzt werden. Das DAML+OIL-Modell kann als Attribut an die HTTP-Session der Benutzersitzung angehängt werden. Das hat den Vorteil, dass keine eigene Sitzungsverwaltung innerhalb der Modelle implementiert werden muss und dass die Modelle relativ klein gehalten werden.

2.2.2 Anforderungen an den Systemkern bezüglich der Modellauswertung

Der Systemkern muss bei der Navigation in dem Modell einerseits darzustellende Elemente identifizieren und andererseits feststellen, welche Daten eingegeben werden müssen oder können und auch Funktionen erkennen, die mit einigen Klassen des Modells verbunden sind. Es müssen also folgende Fähigkeiten garantiert werden:

1. Das Auffinden aller im nächsten Schritt darstellbaren Einheiten zur Darstellung von Links, die auf diese Einheiten verweisen.
2. Anzuzeigende Daten müssen von zu erfragenden Daten unterschieden werden.
3. Alle vom Redakteur festgelegten Informationseinheiten, die mit einem angeklickten Schlüsselwort verbunden sind, müssen kontextabhängig ermittelt werden, bzw. es muss klar sein, welche mit dem Schlüsselwort verbundenen Informationen im aktuellen Kontext relevant sind.
4. Es muss erkannt werden, ob zusammengehörige Informationseinheiten vollständig oder unvollständig sind.

5. Wenn bereits Daten eingegeben wurden, die den neu zu erfragenden Daten entsprechen könnten, sind diese zu finden und als Vorauswahl anzubieten.

2.2.3 Umsetzung des Systemkerns

Im Abschnitt 2.1.2 wurde bereits besprochen, dass der Systemkern ein HttpServletRequest-Objekt übergeben bekommt. Darin enthaltene Parameter werden in die Auswertung des semantischen Modells einbezogen und eine Ausgabe generiert, die es der Anzeigekomponente ermöglicht, die passenden Informationen zu einer Antwort zusammenzusetzen.
Eine verfeinerte Sicht dieser Zusammenhänge ist in Abbildung 5 dargestellt.

Bevor das Verhalten des Systemkerns beschrieben wird, sollen zwei wichtige Aspekte der Abbildung hervorgehoben werden.

- Das DAML+OIL-Modell ist keine Komponente des Systemkerns, sondern ein Container, der Ontologien und Kontextdaten enthält.[19]
- Die Unterteilung des Datenbereichs verdeutlicht, dass Eigenschaften des Systemverhaltens durch Instanzen zur Systemontologie beschrieben werden. Die Instanzen werden im Redaktionsprozess anhand der Anliegenontologien passend zur Systemontologie erzeugt.

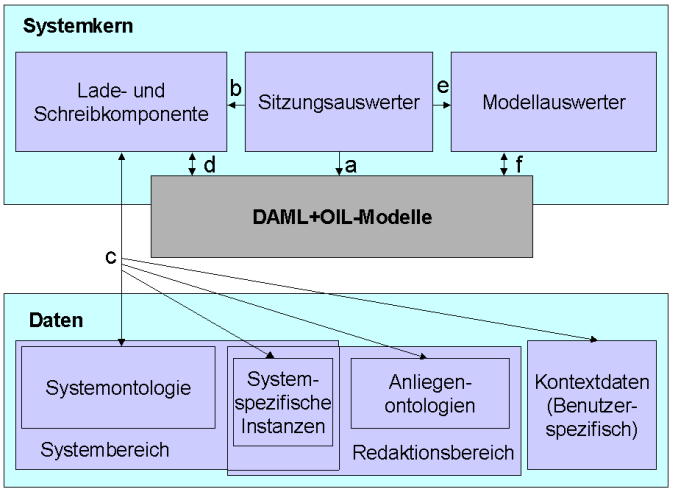

Abbildung 5: Überblick über die Zusammenhänge der Systemkern- und der Datenkomponente

Wenn Informationen bezüglich eines Anliegens angefordert sind, für das eine Ontologie vorhanden ist, ruft der Systemkern zunächst den Sitzungsauswerter auf. Dieser überprüft, ob der jeweiligen HTTP-Session bereits ein DAML+OIL-Modell zugeordnet ist. Besteht bereits ein Modell und sind diesem Daten hinzuzufügen, so übernimmt er dies. Dargestellt sind diese

[19] Siehe dazu die Beschreibung des Jena-Frameworks (com.hp.hpl.jena.daml.DAMLModel)

Zugriffe durch Pfeil *a*. Ist der HTTP-Session noch kein DAML+OIL-Modell zugeordnet, dann wird die Ladekomponente benachrichtigt (Pfeil *b*).

Der Pfeil *c* stellt dar, dass die Lade- und Schreibkomponente die Erzeugung des passenden DAML+OIL-Modells zu dem angeforderten Anliegen übernimmt. Sie lädt die Systemontologie, die Anliegenontologie und die hierfür definierten Instanzen. Falls sich der Benutzer am System angemeldet hat und bereits persistent gespeicherte Daten dieses Benutzers zu diesem Anliegen vorhanden sind, werden diese ebenfalls in das Modell geladen. Die Komponente übernimmt auch die Speicherung der Ontologien, Instanzen. Bezüglich der Instanzen sind die systemspezifischen, die Teil des semantischen Modells sind und auf eine Anliegenontologie verweisen von den anliegenspezifischen zu unterscheiden, die die Kontextdaten darstellen. Bei dem Speicherungsvorgang ist zu beachten, dass für Benutzer nur Kontextdaten, im Redaktionsprozess nur Anliegenontologien und systemspezifische Instanzen und bei einer Systemerweiterung nur die Systemontologie gespeichert werden dürfen.
Die Lade- und Schreibkomponente stellt den anderen Komponenten des Systemkerns die zusammengeführten Modelle zur Verfügung (Pfeil *d*).

Wenn in einer Sitzung bereits ein DAML+OIL-Modell mit Kontextdaten verwaltet wird, ruft der Sitzungsauswerter, Pfeil *e* entsprechend, den Modellauswerter mit den relevanten Parametern aus dem `HttpServletRequest`-Objekt auf.

Der Modellauswerter übernimmt schliesslich diejenigen Aufgaben, die durch die Anforderungen im Abschnitt 2.2.2 adressiert sind (Pfeil *f*):

Anforderung 1: Wie in Abschnitt 1.3 vorgestellt wurde, kann jede Ontologie als Graph aufgefasst werden. Die Knoten des Graphen bilden die Klassen, die Kanten stellen die Properties dar. Der Modellauswerter wertet also das Modell aus, indem er zunächst den aktuellen (den anzuzeigenden) Knoten aus der Umzugsontologie heraussucht. Ausgehend von diesem Knoten findet er alle Knoten, die direkt mit diesem Knoten über Properties verbunden sind und auch diejenigen, die direkte Subklassen des Ausgangsknotens sind. Anhand der Instanzen über der Systemontologie kann der Modellauswerter diejenigen Knoten der gefundenen Knotenmenge auswählen, die auf anzuzeigende Informationen verweisen.

Anforderung 2: Durch einen Vergleich der Klassen der Umzugsontologie mit den Instanzen der Systemontologie, kann festgestellt werden, ob für eine Klasse der Umzugsontologie Benutzereingaben erwartet werden oder ob mit dieser Klasse anzuzeigende Informationen verknüpft sind.

Anforderungen 3, 4: Hat der Modellauswerter eine Instanz der Systemontologie zu einer Klasse der Umzugsontologie gefunden, kann er von diesem Knoten aus den RDF-Graphen der Instanzen durchsuchen. Vorhandene Pfade bilden die Zusammengehörigkeit, Abhängigkeit und Vollständigkeit von Informationen in das Modell ab. Ein Beispiel hierfür wird im Abschnitt 3.3 vorgestellt.

Anforderung 5 Dadurch, dass ObjectProperties und DatatypeProperties in DAML+OIL unterschieden und damit auch im Jena-Framework als verschiedene Klassen repräsen-

tiert werden, kann anhand des Typs einer Propery entschieden werden, ob eine Dateneingabe erforderlich ist. Wenn bereits eingegebene Daten durch die gefundenen DatatypeProperties referenziert werden, gibt der Modellauswerter diese zurück, damit der Benutzer diese Daten entweder bestätigen kann oder neue Daten eingibt.

3 Umsetzung des semantischen Modells *Umzug* in DAML+OIL

Das semantische Modell des Umzugs enthält die Umzugsontologie, die Systemontologie und zusätzlich eine Menge systemspezifischer Instanzen. Damit ist eine Menge von Instanzen bezeichnet, deren Elemente vom Typ in der Systemontologie definierter Elemente sind und die auf Elemente der Umzugsontologie verweisen.
In diesem Abschnitt werden zunächst die für den Prototyp erzeugten Ontologien – die Umzugsontologie und die Systemontologie – vorgestellt und wichtige Aspekte ihrer Verwendung vorgestellt. Darauf folgen Beispiele von Mengen systemspezifischer Instanzen und deren Bedeutung.

Es ist vorweg zu nehmen, dass die Umzugsontologie von Informatikstudenten und nicht wie im Abschnitt 2.1.1 gefordert, von Experten erzeugt wurde. Das Ergebnis ist eine Ontologie, die zwar als vollwertiges Beispiel für eine Anliegenontologie[20] angesehen werden kann, die jedoch keinen Anspruch auf Vollständigkeit oder Korrektheit hat.

Diese Unzulänglichkeit beeinflusst natürlich auch die Systemontologie und die dazu erzeugten Instanzen. Das Wissen von Experten bezüglich behördlichen Abläufe wäre nötig gewesen, um Abläufe zu identifizieren, die von dem Prototyp nachgebildet werden sollten. Da es die Aufgabe der Systemontologie ist, Konzepte zu formulieren und bereitzustellen, mit denen diese Abläufe nachgebildet werden können, mussten im Projektverlauf vorstellbare Abläufe gewählt werden, die umgesetzt werden sollten. So ist die Systemontologie auf den Datenweiterleitungsaspekt konzentriet.

In diesem Kapitel wird der Konvention gefolgt, dass Namen von Klassen gross und Namen von Properties klein geschrieben werden. Ausserdem wurden für Klassen und Properties nach Möglichkeit sprechende Namen verwendet.

Zur Unterstützung der Modellierung wurde Protege2000[21] verwendet. Um DAML+OIL-Ontologien mit Protege2000 bearbeiten und graphisch darstellen zu können, sind das DAML+OIL-[22] und das OntoViz-Plugin[23] benutzt worden. Die graphischen Darstellungen der Ontologien und der RDF-Graphen sind mit dem OntoViz-Plugin erstellt worden.

3.1 Die Umzugsontologie

All diejenigen Begriffe, die aus behördlicher Sicht für den Bürger bei dem Anliegen *Umzug* relevant sind, werden in dieser Ontologie aufgeführt, beschrieben und zueinander in Beziehung gesetzt. Die Umzugsontologie beschreibt sowohl diejenigen Daten und deren Zusammenhänge, die vom Benutzer während einer Sitzung eingegeben werden können, als auch deren Beziehung zu reinen Informationselementen.

[20] Siehe Definition auf Seite 9
[21] http://protege.stanford.edu, Version 1.8
[22] http://www.ai.sri.com/daml/DAML+OIL-plugin/
[23] http://protege.stanford.edu/plugins/ontoviz/ontoviz.html

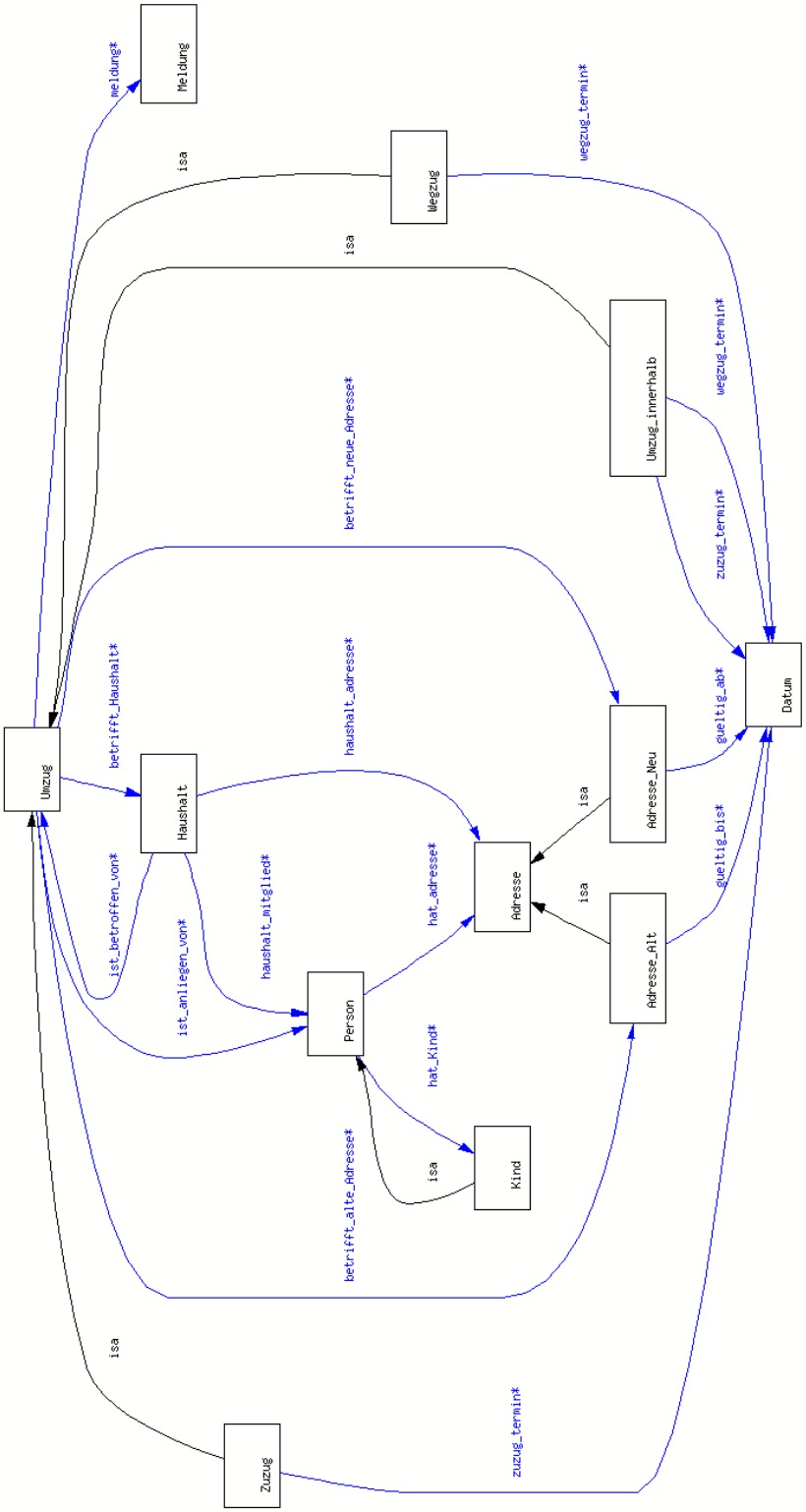

Abbildung 6: Vereinfachte Darstellung der Umzugsontologie

Einen vereinfachten Überblick über die Umzugsontologie gibt Abbildung 6. Die Abbildung ist gegenüber der XML-Darstellung (Anhang A) in der Weise vereinfacht, dass die Subklassen von `Meldung` und alle DatatypeProperties weggelassen wurden.

3.1.1 Die Klassenhierarchie der Umzugsontologie

Die Klassenhierarchie der Umzugsontologie baut auf den Klassen `Umzug`, `Person`, `Adresse`, `Haushalt`, `Datum` und `Meldung` auf.

Die Klasse `Umzug` steht für das Anliegen eines Benutzers umzuziehen. Der `Umzug` ist über ObjectProperties mit den Klassen `Person`, `Adresse_Alt`, `Adresse_Neu`, `Haushalt` und `Meldung` verbunden. Die Subklassen `Umzug_innerhalb`, `Zuzug` und `Wegzug` stellen Spezialisierungen des Umzugs dar, die beschreiben, ob der Benutzer in Hamburg umzieht, von ausserhalb nach Hamburg oder aus Hamburg fort zieht. Sie erben die oben genannten ObjectProperties vom `Umzug` und haben zusätzliche ObjectProperties, die sie mit der Klasse `Datum` verbinden. Die Datumsangabe steht jeweils dafür, wann der Umzug erfolgen soll.

Diejenigen Eigenschaften, die von Personen bei dem Anliegen Umzug relevant sind, werden als Properties der Klasse `Person` festgelegt. Dies sind grösstenteils DatatypeProperties wie z. B. `vorname`, `name`, `geburtsort` und `geburtsdatum` aber auch die ObjectProperties `hat_Kind` und `hat_Adresse`. Die einzige Subklasse von `Person` ist bei dieser Modellierung `Kind`, wobei anzumerken ist, dass diese Modellierung stark von der Sichtweise des Redakteurs abhängig ist. Nach dieser Modellierung kann ein Informationselement mit `Kind` verknüpft werden und für Kinder könnten dieselben Daten eingegeben werden wie für Erwachsene. Wenn ein Redakteur für Kinder andere Daten erwartet als bei Erwachsenen, würde er `Kind` nicht als Subklasse von `Person`, sondern als eigenständige Klasse modellieren.

Adressangaben, die vom Benutzer verlangt werden, sind als DatatypeProperties an die Klasse `Adresse` gehängt worden. Die beiden Subklassen `Adresse_Alt` und `Adresse_Neu` sind noch zusätzlich mit über die ObjectProperty `gueltig_bis` bzw. `gueltig_ab` mit der Klasse `Datum` verknüpft.

Beliebig viele Personen und eine Adresse werden über die Klasse `Haushalt` zu einem Haushalt zusammengefasst. Die Angabe, dass die ObjectProperty `haushalt_adresse` vom Typ UniqueProperty ist, bedeutet, dass einem Haushalt höchstens eine Adresse zugeordnet werden kann.

Die Klasse `Datum` ist lediglich ein Stellvertreter für ein Datum, an den das eigentliche Datum über eine DatatypeProperty angehängt wird. Das ist im Grunde nicht notwendig. ObjectProperties, die auf die Klasse `Datum` verweisen, könnten auch durch DatatypeProperties ersetzt werden. Es ist eine Modellierungsentscheidung, welche dieser beiden Propery-Arten benutzt werden soll, die davon abhängt, ob mit einem allgemeinen Datum ein wichtiges Informationselement verknüpft ist. Die Verknüpfung mit Informationen ist nach dem hier beschriebenen Ansatz nur mit Klassen einer Anliegenontologie möglich, die über Instanzen der Systemontologie referenziert werden.

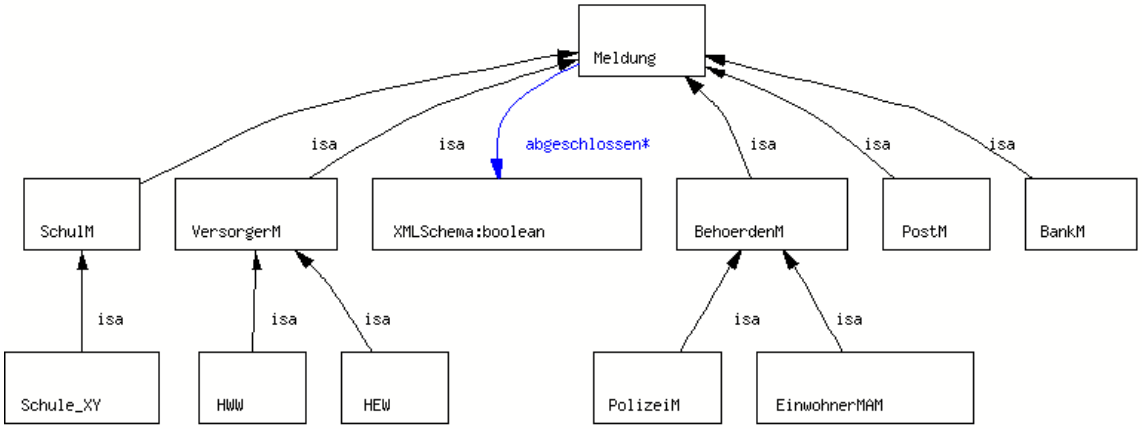

Abbildung 7: Subklassen und Properties der Klasse *Meldung*

Wie Abbildung 7 darstellt, ist die Klasse `Meldung` mit der DatatypeProperty `abgeschlossen` versehen, die angibt, ob eine Meldung bereits erfolgreich durchgeführt worden ist. Die Property `abgeschlossen` wird an alle Subklassen vererbt und ist nur für diejenigen Subklassen relevant, für die – über Instanzen der Systemontologie – Datenweiterleitungen definiert sind. Die Abbildung zeigt ausserdem alle Subklassen von `Meldung`, die in der Umzugsontologie definiert worden sind.

Die Subklassen von `Meldung` sind jeweils mit einem Informationselement verknüpft. Wenn Datenweiterleitungen für Firmen, Behörden oder Körperschaften definiert werden sollen, sind entsprechende Klassen anzulegen. In diesem Beispiel sind dies die Klassen `HEW`, `HWW` und `Schule_XY`.

3.1.2 Verwendung der Umzugsontologie

Die Umzugsontologie wird von dem Modellauswerter auf zwei Arten genutzt.

1. Verwendung als Graph, der Pfade durch das Anliegen *Umzug* vorgibt.
2. Verwendung als Schema, das Zusammenhänge und Aufbau der Kontextdaten beschreibt.

Bei der Verwendung als Graph stellt die Klasse `Umzug` den Einstiegspunkt in das Benutzeranliegen dar. Es wird jeweils die Klasse der Umzugsontologie herausgesucht, die der Anfrage des Benutzers entspricht. Von dieser Klasse wird eine Instanz in den Kontextdaten erzeugt, deren Bedeutung es ist, dass der Benutzer das zu der jeweiligen Klasse gehörende Informationselement angesehen hat. Ausserdem werden alle Subklassen und alle Properties der jeweiligen Klasse herausgesucht, um Verweise auf die nächsten direkt erreichbaren Informationselemente zu generieren.

Unter Hinzunahme der Systemontologie und deren Instanzen wird festgestellt, für welche der gefundenen Klassen Informationselemente vorhanden sind und entsprechende Verweise von dem Modellauswerter generiert.

Wenn also ein Benutzer mit dem Anliegen Umzug aktiv wird, wird eine Instanz der Klasse `Umzug` in dem Kontextdatenmodell des Benutzers erzeugt. Ausgehend von der Klasse `Umzug` werden von dem Modellauswerter, wie in Abschnitt 2.2.3 beschrieben, alle direkten Subklassen und alle Properties, die für diese Klasse definiert sind, gefunden. Das sind zum Einen die Klassen `Umzug_innerhalb`, `Zuzug`, `Wegzug` und zum Anderen Properties, die den allgemeinen Umzug mit `Person`, `Adresse_Alt`, `Adresse_Neu`, `Haushalt` und `Meldung` verbindet. Dieses sind diejenigen Elemente, die für den Benutzer als weiterführende Informationselemente zur Verfügung stehen.

Nach der Auswahl eines der weiterführenden Informationselemente, verfährt der Modellauswerter ebenso. Er sucht die entsprechende Klasse der Umzugsontologie, erzeugt eine Instanz dieser Klasse, erzeugt ein Statement im Graph der Kontextdaten, der der ausgewählten Property entspricht und stellt alle Subklassen und alle Properties fest, um die möglichen nächsten Knoten zu identifizieren. Der Graph der Kontextdaten zeichnet also den Weg des Benutzers durch die Umzugsontologie nach.

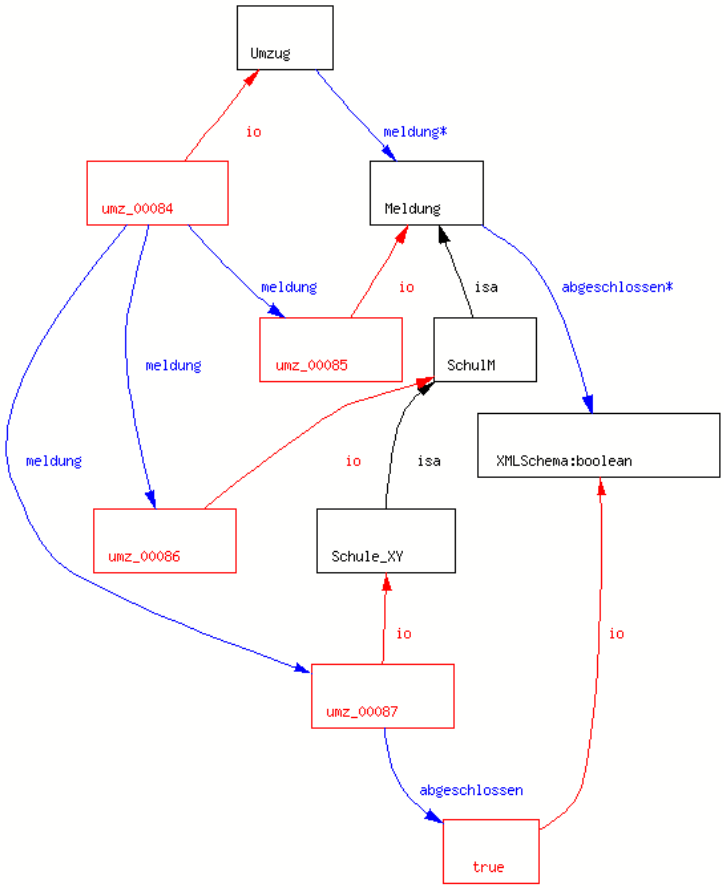

Abbildung 8: Ein einfaches Beispiel von Kontextdaten.

Die Verwendung der Klasse `Meldung` soll an einem Beispiel verdeutlicht werden, das in Abbildung 8 dargestellt ist: Die Klasse `Meldung` ist mit der DatatypeProperty abgeschlossen versehen, die angibt, ob eine Meldung bereits erfolgreich durchgeführt worden ist. Eine Instanz von `Meldung` in dem Graphen der Kontextdaten gibt an, dass eine Person das Informationselement angesehen hat, das mit `Meldung` verknüpft ist. Wenn der Modellauswerter feststellt, dass für eine Subklasse von `Meldung` eine Datenweiterleitung definiert ist, kann diese durchgeführt werden und es kann vom Benutzer die DatatypeProperty abgeschlossen gesetzt werden. In den Kontextdaten sind nach Ansicht von `Umzug`, `Meldung`, `SchulM` und `Schule_XY` und Abschluss der hierfür definierten Datenweiterleitung vier Statements. Die erste verbindet die Instanz der Klasse `Umzug` (`umz_00084`) über die Property `meldung` mit der Instanz der Klasse `Meldung` (`umz_00085`). Die zweite Aussage stellt eine Spezialisierung der ersten dar, Objekt ist die Instanz `umz_00086` von `SchulM`, die dritte wiederum eine Spezialisierung, bei der das Objekt die Instanz `umz_00087` von `Schule_XY` ist. Das vierte Statement verbindet die Instanz von `SchulM` über die Property `abgeschlossen` mit dem entsprechenden Wahrheitswert (`true`).

Der Modellauswerter verwendet die Umzugsontologie als Schema für die Kontextdaten, wenn eine Klasse mit DatatypeProperties versehen ist. Er generiert Verweise für Eingabefelder, so dass die Daten nach Eingabe durch den Benutzer von dem Sitzungsauswerter dem Modell der Kontextdaten hinzugefügt werden können.

3.2 Die Systemontologie

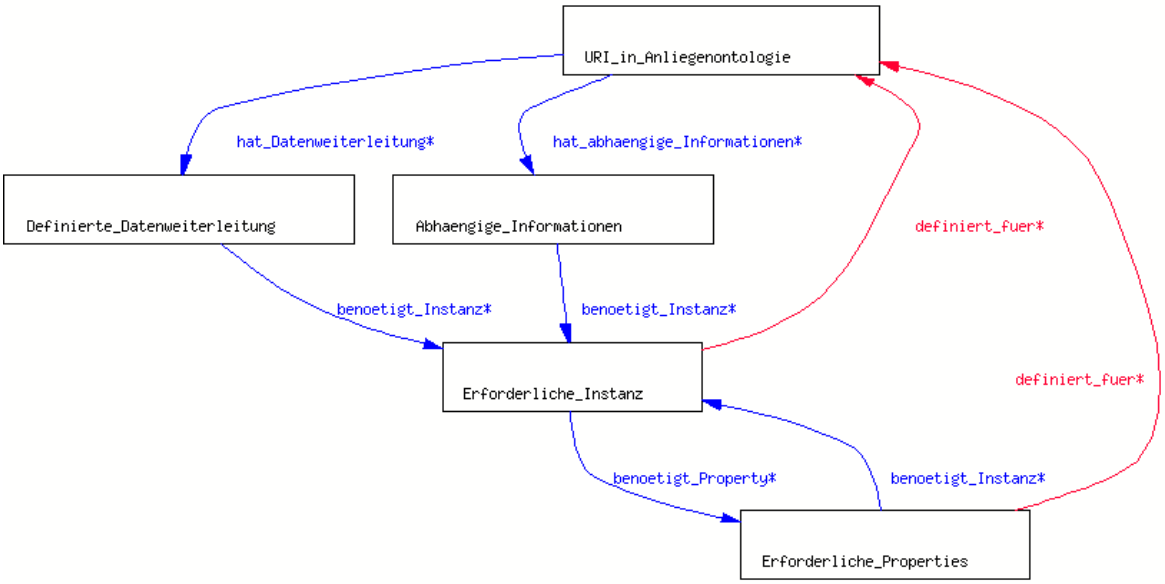

Abbildung 9: Vereinfachte Darstellung der Systemontologie

Die Systemontologie wurde aus dem Grund eingeführt, dass das System mit den verschiedenen Elementen der Umzugsontologie unterschiedlich verfahren sollte.

Wie im Abschnitt 2.2.3 erläutert wurde, ist die Unterscheidung von DatatypeProperties und ObjectProperties bereits in DAML+OIL und damit auch in das Jena-Framework eingebaut. Wenn jedoch Klassen voneinander unterschieden werden sollen, da mit Ihnen ein jeweils unterschiedliches Systemverhalten verknüpft ist, setzt dies voraus, dass dem System die Bezeichner dieser Klassen bekannt sind. Bei der Implementierung sollte jedoch eine grösstmögliche Datenunabhängigkeit erreicht werden, damit ein Redaktionsprozess ermöglicht werden kann, bei dem sämtliche Bezeichner vom Redakteur frei wählbar sind. Es musste also ein Mechanismus entwickelt werden, wie in dem System spezielle Klassen aus Anliegenontologien mit speziellem Systemverhalten verknüpft werden können, ohne dass die Namen dieser Klassen im Quelltext des Systems hart kodiert werden. Dieser Mechanismus ist durch die Systemontologie umgesetzt worden. In ihr werden Klassen und Properties definiert, die dem System bekannt sind, deren Bezeichner also hart kodiert sind. Instanzen dieser Klassen können Klassen der Umzugsontologie referenzieren und auf diese Weise das mit den Klassen der Systemontologie verknüpfte Systemverhalten auf die Klassen der Umzugsontologie übertragen.

In Abbildung 9 ist die Systemontologie vereinfacht dargestellt. In der Abbildung fehlen die DatatypeProperties. Im Anhang B ist die XML-Darstellung der Systemontologie komplett einzusehen.

Die ObjectProperty `definiert_fuer` ist in der Abbildung farblich hervorgehoben, da ihr die besondere Bedeutung zukommt, Instanzen der Systemontologie Elemente einer Anliegenontologie zuzuordnen. Diese Property ist nicht vom Typ UniqueProperty, da es in DAML+OIL Mehrklassenzugehörigkeit und Mehrfachvererbung gibt. Es ist also möglich, von einer Instanz aus den Kontextdaten zu fordern, dass sie mehrere Typen hat.

Die Systemontologie ist von den Entwicklern des Prototypen vorgegeben. Da URIs in jedem DAML+OIL Dokument eindeutig sein müssen, sind die Bezeichner, die in der Systemontologie verwendet werden, für die Verwendung in einer Anliegenontologie gesperrt.

3.2.1 Die Klassen der Systemontologie

Die Systemontologie besteht aus den Klassen `URI_in_Anliegenontologie`, `Abhaengige_Informationen`, `Definierte_Datenweiterleitung`, `Erforderliche_Instanz` und `Erforderliche_Properties`. Zu keiner dieser Klassen sind Subklassen definiert.

Die Klasse `URI_in_Anliegenontologie` stellt einen Mechanismus bereit, die Klassen einer Anliegenontologie zu referenzieren. Sämtliche Instanzen dieser Klasse referenzieren Elemente einer Anliegenontologie.
Über die DatatypeProperty `hat_internen_Verweis` kann eine Instanz der Klasse `URI_in_Anliegenontologie` mit dem Verweis auf ein Informationselement des Content-Management-Systems verknüpft werden. Die Verweise sind den Klassen eindeutig zugeordnet, also ist diese Property vom Typ UniqueProperty. Die DatatypeProperty `hat_externen_Verweis` kann als Wert z. B. URLs externer Web Sites oder Dateien enthalten. Die ObjectProperty `hat_Datenweiterleitung` kann eine Datenweiterleitung für eine Instanz von `URI_in_Anliegenontologie` festlegen. Wenn diese Property gesetzt ist, ist für die zugehörige URI der Anliegenontologie eine Datenweiterleitung definiert. Die ObjectProperty

`hat_Datenweiterleitung` ist vom Typ UnambiguousProperty und vom Typ UniqueProperty (vergleiche Anhang B). Es kann für eine Klasse der Umzugsontologie also höchstens eine Datenweiterleitung definiert sein (UniqueProperty) und diese Datenweiterleitung kann nicht für eine zweite Klasse der Umzugsontologie definiert sein (UmambiguousProperty).
Informationselemente, die für eine Klasse in Abhängigkeit von bestimmten Bedingungen angezeigt werden sollen, können über die Klasse `Abhaengige_Informationen` festgelegt werden. Ist die Property `hat_abhaengige_Informationen` für eine Instanz der Klasse `URI_in_Anliegenontologie` gesetzt, wird das abhängige Informationselement genau dann angezeigt, wenn diejenigen Instanzen und Properties in den Kontextdaten gesetzt sind, deren Typen in dem Teilgraphen referenziert werden, der durch die Property `hat_abhaengige_Informationen` bezeichnet ist.

Die Bedeutung der Klasse `Definierte_Datenweiterleitung` ist für den Modellauswerter, dass Instanzen dieser Klasse für die Auswahl und Weiterleitung von Kontextdaten stehen. Welche Kontextdaten benötigt werden, wird über die Property `benoetigt_Instanz` festgelegt.

Für die Beschreibung von abhängigen Informationen oder Datenweiterleitungen ist die Angabe einer oder mehrerer Klassen erforderlich, von denen Instanzen in den Kontextdaten vorhanden sein müssen, damit die abhängigen Informationen angezeigt werden oder Daten an externe Web Services weitergeleitet werden können. Eine Instanz der Klasse `Erforderliche_Instanz` der Systemontologie wird durch setzen der ObjectProperty `definiert_fuer` über eine Instanz der Klasse `URI_in_Anliegenontologie` einer Klasse der Anliegenontologie zugeordnet. Abhängige Informationen können intern oder extern sein. Dafür wird jeweils die entsprechende DatatypeProperty gesetzt.

Instanzen der Klasse `Erforderliche_Properties` verweisen ebenfalls über die ObjectProperty `definiert_fuer` auf eine Instanz der Klasse `URI_in_Anliegenontologie` und können auf diese Weise festlegen, welche Properties einer Instanz gesetzt sein müssen, damit abhängige Informationen angezeigt oder Datenweiterleitungen durchgeführt werden können. Über die ObjectProperty `benoetigt_Instanz` kann bei einer Property, für die mehrere Range-Klassen angegeben sind, genau eine der gültigen Klassen oder eine der möglichen Unterklassen spezifiziert werden, von der eine Instanz vorhanden sein muss.

3.2.2 Verwendung der Systemontologie

Die Lade- und Schreibkomponente des Systemkerns[24] stellt sicher, dass die Anliegenontologie, die für die jeweilige Sitzung geladen wird, passend zu denjenigen Instanzen ist, die mit der Systemontologie in ein DAML+OIL-Modell geladen werden. Der Zusammenhang zwischen Instanzen der Klasse `URI_in_Anliegenontologie` der Systemontologie und Elementen der jeweiligen Anliegenontologie kann also wie folgt sichergestellt werden:

Während der Erstellung der Anliegenontologie werden sämtliche URIs der Ontologie durch Austausch der Namespace-Komponente der URIs in URIs der Systemontologie umgewandelt

[24] Siehe Kapitel 2.2.3

und die so erhaltenen Ressourcen werden Instanzen der Klasse `URI_in_Anliegenontologie`. Wenn also umgekehrt bei Instanzen dieser Klasse der Namespace der Systemontologie (http://example.com/system#) durch den Namespace der jeweiligen Anliegenontologie (z. B. http://example.com/umzug#) ersetzt wird, ist das Ergebnis der URI einer Klasse oder Property der Anliegenontologie.

Der Modellauswerter verwendet die Systemontologie also in der Weise, dass zunächst geprüft wird, ob eine Instanz vom Typ `URI_in_Anliegenontologie` ist. Dann wird festgestellt, ob mit diesem Element ein internes Informationselement, externe Informationselemente, eine Datenweiterleitung oder abhängige Informationselemente verknüpft sind.

Die Verweise auf die internen und externen Informationselemente sind direkt als Wert (String) über die jeweiligen DatatypeProperties mit den Instanzen verknüpft und können also leicht gefunden werden.

Das Abarbeiten einer Datenweiterleitung oder das Feststellen, ob all diejenigen Instanzen und Properties in den Kontextdaten gesetzt sind, die für abhängige Informationen erforderlich sind, ist etwas aufwendiger:

Da die Klassen `Erforderliche_Properties` und `Erforderliche_Instanz` mit den ObjectProperties `benoetigt_Instanz` und `benoetigt_Property` einen Zyklus bilden (Siehe Abbildung 9) und für diese beiden Properties keine Kardinalitätsbeschränkungen gelten, können zu ihnen passende RDF-Graphen Bäume beliebiger Tiefe darstellen. An jedem Knoten können mehrere Äste ansetzen, wenn die weiterführende Property mehrfach belegt ist. Die ObjectProperty `definiert_fuer` bezieht jeden Knoten dieser Bäume auf Elemente der Anliegenontologie. Diese Bäume werden für die Datenweiterleitung und Abhängigkeitsbestimmung per Tiefensuche und Backtracking durchsucht. Die Datenweiterleitung kann erst dann erfolgen, wenn der gesamte Baum erfolgreich abgearbeitet wurde. Abhängige Daten werden nur dann angezeigt, wenn alle geforderten Instanzen und Properties gesetzt sind.

Bei der Implementierung des Modellauswerters wurde bezüglich der Abarbeitung solch eines Baumes die folgende Strategie angewendet: Wenn die Instanz einer Klasse gefordert wird und keine Angaben bezüglich geforderter Properties gemacht worden sind, sind alle DatatypeProperties erforderlich und die ObjectProperties irrelevant. ObjectProperties werden nur dann verfolgt, wenn sie explizit aufgeführt sind. Wenn zu einer geforderten Instanz DatatypeProperties gefordert werden, werden genau diese weiterverfolgt, alle anderen DatatypeProperties sind irrelevant.

Dieser Mechanismus ermöglicht es, Pfade in dem Graphen der Kontextdaten vorzugeben, die vorhanden sein müssen, bzw. erzeugt werden müssen. Hierbei ist darauf zu achten, dass für eine Instanz nur Properties gefordert werden dürfen, die für die zugehörige Klasse definiert sind. Ein Redaktionstool muss also das Auftreten solcher Inkonsistenzen vermeiden.

3.3 Ausschnitt des semantischen Modells *Umzug*: Definition einer Datenweiterleitung

Als Beispiel für einen Ausschnitt des semantischen Modells *Umzug* soll in diesem Abschnitt die Definition einer Datenweiterleitung vorgestellt werden.

Damit die Weiterleitung von Kontextdaten an externe Web Services funktionieren kann, muss dem Modellauswerter folgendes mitgeteilt werden

- an welchem Punkt in dem semantischen Modell des Umzugs die Datenzusammenführung beginnt und
- welche Kontextdaten weitergeleitet werden sollen.

Der Modellauswerter muss also in die Lage versetzt werden, diejenigen Teile des RDF-Graphen der Kontextdaten zu identifizieren, welche weitergeleitet werden sollen. Dieses kann durch Instanzen der Systemontologie umgesetzt werden.

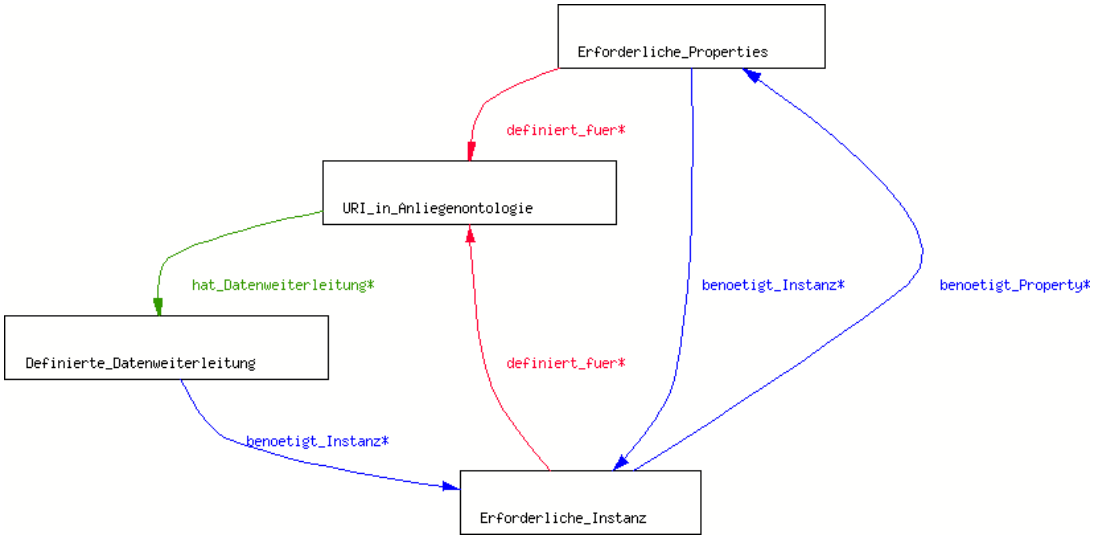

Abbildung 10: Ausschnitt der Systemontologie: Schema der Datenweiterleitung

Abbildung 10 zeigt den für die Datenweiterleitung relevanten Teil der Systemontologie, das Schema für einen RDF-Teilgraphen der Systemontologie, der wiederum Ausschnitte der Umzugsontologie und damit Teilgraphen des RDF-Kontextdatengraphen beschreibt. Wie im Abschnitt 3.2.1 beschrieben, legt dieses Schema fest, dass eine Datenweiterleitung eindeutig einer Klasse der Umzugsontologie zugeordnet ist.
Der RDF-Teilgraph der Systemontologie enthält Instanzen, die Klassenbezeichner der Umzugsontologie referenzieren, von denen Instanzen in dem Kontextdatengraphen enthalten sein müssen. Ausserdem enthält er Instanzen, die die Bezeichner aller von diesen Instanzen ausgehenden Properties referenzieren, die für die Datenzusammenführung erforderlich sind.

Wenn der Modellauswerter also von einer Klasse aus der Umzugsontologie feststellt, dass für die zu ihr gehörige Instanz der Systemontologie die ObjectProperty `hat_Datenweiterleitung` gesetzt ist, dann setzt er das Objekt dieser Property als Wurzel des abzuarbeitenden Baumes.

Ausgehend von der gefundenen Instanz der Klasse `Definierte_Datenweiterleitung` werden die Objekte der Property `benoetigte_Instanz` die Söhne der Wurzel. Die Söhne geben also an, an welchen Klassen der Umzugsontologie die Zusammenführung der Kontextdaten ansetzt. Für jeden der gefundenen Söhne werden alle Objekte der Property `benoetigt_Property` wiederum die Söhne und auf diese Weise rekursiv der Suchbaum aufgebaut. Trifft der Modellauswerter hierbei auf eine Klasse, zu der noch keine Instanz existiert, muss der Benutzer eine Eingabe machen. Existiert eine Instanz, muss der Benutzer diese bestätigen. Die auf diese Weise ausgewählten Daten werden gesammelt und können, nachdem der gesamte Baum abgearbeitet ist, weitergeleitet werden.

Abbildung 11 auf Seite 32 zeigt den RDF-Graph einer definierten Datenweiterleitung für Schule_XY. Die Abbildung verdeutlicht, dass die ObjectProperties `benoetigt_Instanz` und `benoetigt_Property` einen Baum aufspannen, dessen Knoten über die ObjectProperty `definiert_fuer` einem Element der Umzugsontologie zugeordnet wurden.

Konkret bedeutet diese Definition der Datenweiterleitung, dass in den Kontextdaten des Benuzters eine Instanz der Klasse `Person` vorhanden sein muss, für die die Properties `hat_geburtstag`, `hat_Adresse` und `hat_Kind` gesetzt sind. Dabei soll die Property `hat_Adresse` doppelt belegt sein, nämlich erstens mit einer Instanz der Klasse `Adresse_Alt` und zweitens mit einer Instanz der Klasse `Adresse_Neu`. Bei der alten und der neuen Adresse müssen alle DatatypeProperties der Adressen gesetzt sein, zusätzlich ist die ObjectProperty `gueltig_bis` bzw. `gueltig_ab` zu setzen. Von dem Kind sind lediglich ein Geburtsdatum und die Angabe gefordert, in welche Klasse es geht.

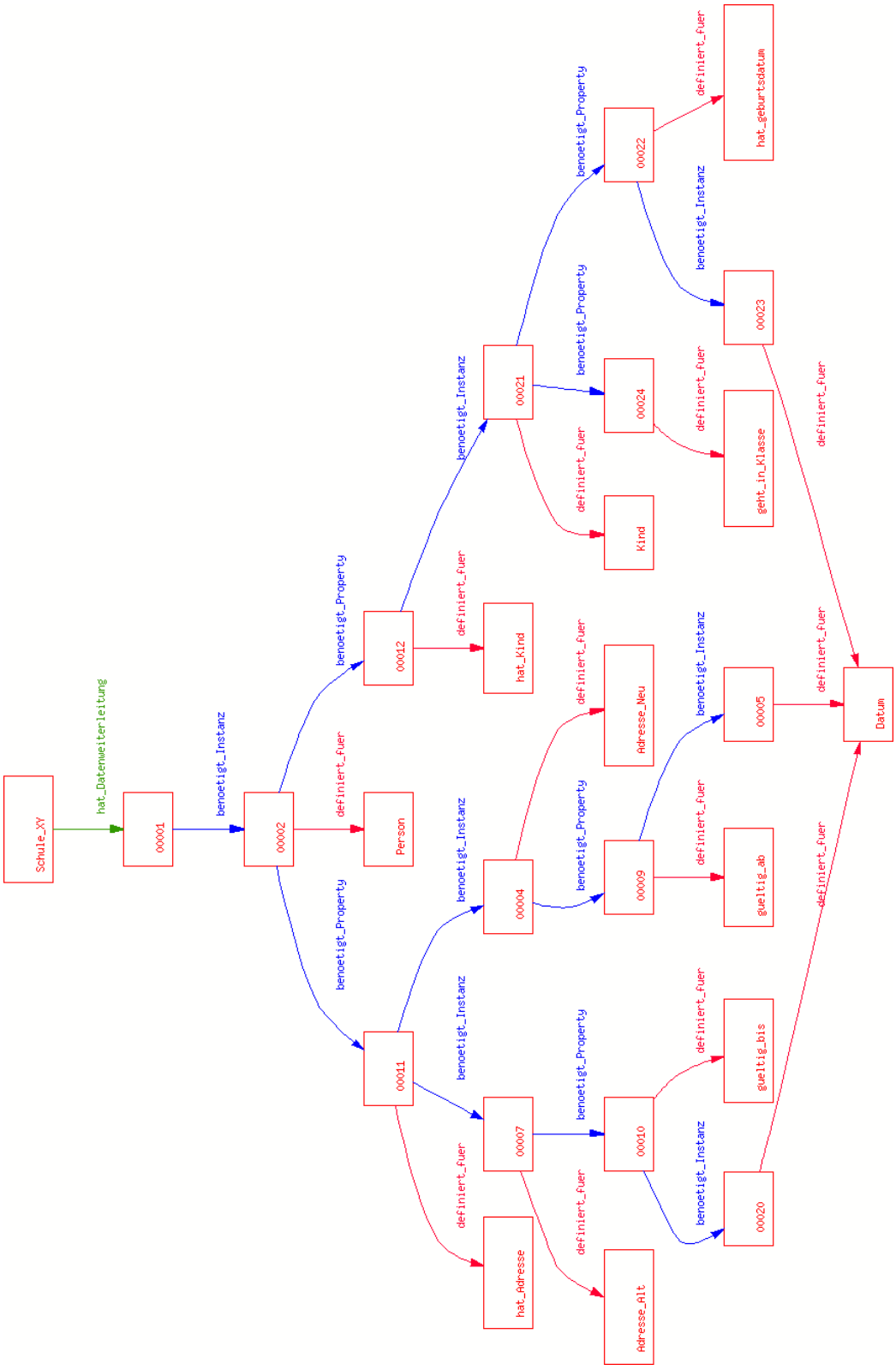

Abbildung 11: Beispiel einer Datenweiterleitungs-Definition

4 Einschätzung dieses Ansatzes zur Steuerung einer Web-Anwendung

In dieser Arbeit ist der Aufbau und die Funktionsweise einer Web-Anwendung vorgestellt worden, die anhand eines in DAML+OIL umgesetzten semantischen Modells gesteuert wird. Der Zugriff auf das semantische Modell, seine Auswertung und Verwaltung ist mit dem Jena-Framework umgesetzt worden, welches sich in der Version 1.6.0 als übersichtlich und stabil erwies. Allerdings unterstützt das Jena-Framework in der Version 1.6.0 nicht alle Konzepte von DAML+OIL effizient und aus diesem Grund ist das semantische Modell des Umzugs nur mit den grundlegenden Konzepten von DAML+OIL formuliert worden. Dennoch ist gezeigt worden, dass dieser Ansatz zur Steuerung einer Web-Anwendung umsetzbar ist.

Im Abschnitt 2.1.1 ist darauf hingewiesen worden, dass der Protoyp derzeit noch keine Unterstützung für einen Redaktionsprozess bietet. Die Möglichkeit, Systemverhalten anhand der Systemontologie zu formulieren, ist jedoch im Hinblick auf einen Redaktionsprozess entwickelt worden. Ein Redaktionstool, das es einem Redakteur ermöglicht, ohne Kenntnis der Sprache DAML+OIL eine Anliegenontologie und ein dazu passendes Modell zu erzeugen, ist jedoch eine wichtige Vorraussetzung für den produktiven Einsatz eines Bürger-Informations-Services, der anhand eines semantischen Modells gesteuert wird.

Der Prototyp bietet die Möglichkeit, Kontextdaten für die Übermittlung an externe Web Services auszuwählen. Allerdings existiert noch keine Schnittstelle für die Übermittlung. In der Systemontologie sind daher auch keine Klassen und Properties angelegt worden, die es ermöglichen würden, Aspekte einer Datenweiterleitung wie z. B. Adresse, Port und Formatierung der Daten festzulegen.

Die oben genannten Punkte, also die Unterstützung des Redaktionsprozesses, eine Schnittstelle zu externen Web Services und eine vollständige Syststemontologie, müssen umgesetzt werden, damit auf Grundlage des beschriebenen Prototypen ein produktiv einsetzbarer Bürger-Informations-Service implementiert werden kann.

Literatur

[Bax03] Bob Baxley. What is a web application? *http://www.boxesandarrows.com/archives/print/003223.php*, 2003.

[BG02] Dan Brickley and R.V. Guha. Rdf vocabulary description language 1.0: Rdf schema. *http://www.w3.org/TR/2002/WD-rdf-schema-20021112*, 2002.

[DAMa] Annotated daml+oil ontology markup. *http://www.w3.org/TR/daml+oil-walkthru/*.

[DAMb] Daml+oil (march 2001) reference description. *http://www.daml.org/2001/03/reference.html*.

[Dea02] Mike Dean. Daml+oil for application developers: An introduction to the semantic web. Technical report, SPAWAR Systems Center, 2002.

[Die03] Jörg Dieckmann. DAML+OIL und owl. XML-Sprachen für Ontologien. 2003.

[FI03] Universität Hamburg Fachbereich Informatik. Kommentiertes Veranstaltungsverzeichnis, WS02/03.

[Gru93] Tom Gruber. A translation approach to portable ontology specifications. Technical report, Stanford University, Knowledge Systems Laboratory, 1993.

[LS99] Ora Lassila and Ralph R. Swick. Resource description framework (rdf) model and syntax specification. *http://www.w3.org/TR/1999/REC-rdf-syntax-19990222*, 1999.

[NM] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford University.

[SN00] F. Steimann and W. Nejdl. Modellierung und ontologie. Technical report, Universität Hannover, Institut für Rechnergestützte Wissensverarbeitung, 2000.

[Tea00] IBM Web Services Architecture Team. *Web Services architecture overview*. ftp://www6.software.ibm.com/software/developer/library/w-ovr.pdf, September 2000.

[UG96] M. Uschold and M. Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[Usc96] Mike Uschold. Building ontologies: Towards a unified methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK, 1996.

[Weba] What is a web application? *http://manuals.sybase.com/onlinebooks/group-eag/eag0420e/jagpg/@Generic__BookTextView/25755*.

[Webb] What is a web application? *http://support.asna.com/kb/documentation/Help_Files/Web_Smarties_35/webtutfp_what_is_a_web_application_.htm*.

[Win00] Dave Winer. What is a web application? *http://davenet.userland.com/2000/03/12/whatIsAWebApplication*, 2000.

[WSia] whatis.com. *http://searchwebservices.techtarget.com/sDefinition/0 sid26_gci213352 0.html*.

[WSib] wikipedia.com. *http://www.wikipedia.org/wiki/Web_site*.

Alle oben genannten URLs wurden im Juli 2003 überprüft.

A DAML+OIL-Darstellung der Umzugsontologie

Dieses ist die DAML+OIL Ausgabe der Umzugsontologie von Protege2000:

```
<rdf:RDF
  xmlns:XMLSchema ="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:daml_oil ="http://www.daml.org/2001/03/daml+oil#"
  xmlns:umzug ="http://example.com/umzug#"
  xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
  xmlns ="http://example.com/umzug#"
  >

<daml_oil:ObjectProperty rdf:ID="betrifft_neue_Adresse">
  <daml_oil:range rdf:resource="#Adresse_Neu"/>
  <daml_oil:domain rdf:resource="#Umzug"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="Wegzug">
  <rdfs:comment>Benutzer zieht aus Hamburg weg.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Umzug"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="BehoerdenM">
  <rdfs:comment>Vielleicht Überflüssig?</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Meldung"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="VersorgerM">
  <rdfs:comment>Meldung an Versorger wie z.B. HEW, HWW.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Meldung"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Adresse">
  <rdfs:comment></rdfs:comment>
</daml_oil:Class>

<daml_oil:Class rdf:ID="HEW">
  <rdfs:comment>Meldung an Hamburger Elektrizitäts Werke.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#VersorgerM"/>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="zuzug_termin">
  <rdfs:comment></rdfs:comment>
  <daml_oil:range rdf:resource="#Datum"/>
  <daml_oil:domain rdf:resource="#Umzug_innerhalb"/>
  <daml_oil:domain rdf:resource="#Zuzug"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
</daml_oil:ObjectProperty>
```

```
<daml_oil:DatatypeProperty rdf:ID="ausgewaehlte_ID">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#EinwohnerMAM"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="wegzug_termin">
  <rdfs:comment></rdfs:comment>
  <daml_oil:range rdf:resource="#Datum"/>
  <daml_oil:domain rdf:resource="#Umzug_innerhalb"/>
  <daml_oil:domain rdf:resource="#Wegzug"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="Zuzug">
  <rdfs:comment>Benutzer zieht nach Hamburg.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Umzug"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="HWW">
  <rdfs:comment>Meldung an Hamburger Wasserwerke.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#VersorgerM"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Meldung">
  <rdfs:comment>
    Umzugsmeldungen sollten an alle Behörden oder Firmen gehen, bei
    denen Daten des Benutzers verwaltet und genutzt werden.
  </rdfs:comment>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="etage">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Adresse"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="haushalt_mitglied">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Haushalt"/>
  <daml_oil:range rdf:resource="#Person"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UnambiguousProperty"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="Umzug">
  <rdfs:comment>Einstieg in das Benutzeranliegen.</rdfs:comment>
```

```
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="familienstand">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Person"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="meldung">
  <rdfs:comment></rdfs:comment>
  <daml_oil:range rdf:resource="#Meldung"/>
  <daml_oil:domain rdf:resource="#Umzug"/>
</daml_oil:ObjectProperty>

<daml_oil:ObjectProperty rdf:ID="gueltig_bis">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="http://example.com/umzug#Adresse_Alt"/>
  <daml_oil:range rdf:resource="#Datum"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="EinwohnerMAM">
  <rdfs:comment>Um- oder Abmeldung des Wohnsitzes.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#BehoerdenM"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Haushalt">
  <rdfs:comment>
    Zusammenwohnende Personen können zu einem Haushalt zusammengefasst
    werden.
  </rdfs:comment>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="hausnummer">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Adresse"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#integer"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="PolizeiM">
  <rdfs:comment>
    Umzugsmeldung an die Polizei (Parkplatzreservierung).
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#BehoerdenM"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="staatsangehoerigkeit">
  <rdfs:comment></rdfs:comment>
```

```
  <daml_oil:domain rdf:resource="#Person"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="gueltig_ab">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Adresse_Neu"/>
  <daml_oil:range rdf:resource="#Datum"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="BankM">
  <rdfs:comment>Umzugsmeldung an Banken.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Meldung"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Adresse_Neu">
  <rdfs:comment></rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Adresse"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Kind">
  <rdfs:comment>
    Es müssen nicht alle Properties gesetzt sein. Man sollte vielleicht
    geht_in_Klasse und hat_Kind x-or verknüpfen.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Person"/>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="ist_anliegen_von">
  <rdfs:comment></rdfs:comment>
  <daml_oil:range rdf:resource="#Person"/>
  <daml_oil:domain rdf:resource="#Umzug"/>
</daml_oil:ObjectProperty>

<daml_oil:ObjectProperty rdf:ID="hat_Kind">
  <daml_oil:range rdf:resource="#Kind"/>
  <daml_oil:domain rdf:resource="#Person"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UnambiguousProperty"/>
</daml_oil:ObjectProperty>

<daml_oil:DatatypeProperty rdf:ID="familienname">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Person"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="ist_betroffen_von">
```

```
  <daml_oil:domain rdf:resource="#Haushalt"/>
  <daml_oil:range rdf:resource="#Umzug"/>
  <daml_oil:inverseOf rdf:resource="#betrifft_Haushalt"/>
</daml_oil:ObjectProperty>

<daml_oil:DatatypeProperty rdf:ID="ort">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Adresse"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="haushalt_adresse">
  <rdfs:comment></rdfs:comment>
  <daml_oil:range rdf:resource="#Adresse"/>
  <daml_oil:domain rdf:resource="#Haushalt"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="Person">
  <rdfs:comment>
    Sollte alle Properties haben, die bei irgendeiner Weiterleitung
    relevant sein könnten.
  </rdfs:comment>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="geht_in_Klasse">
  <daml_oil:domain rdf:resource="#Kind"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Umzug_innerhalb">
  <rdfs:comment>Umzug innerhalb Hamburgs.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Umzug"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="vorname">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Person"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Datum">
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="geburtsort">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Person"/>
```

```
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Adresse_Alt">
  <rdfs:comment></rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Adresse"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="SchulM">
  <rdfs:comment>Umzugsmeldung an Schulen.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Meldung"/>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="hat_adresse">
  <rdfs:comment></rdfs:comment>
  <daml_oil:range rdf:resource="#Adresse"/>
  <daml_oil:domain rdf:resource="#Person"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="Schule_XY">
  <rdfs:comment>Meldung an die Schule XY.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#SchulM"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="wohungsnummer">
  <daml_oil:domain rdf:resource="#Adresse"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:DatatypeProperty rdf:ID="abgeschlossen">
  <daml_oil:domain rdf:resource="#Meldung"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#boolean"/>
</daml_oil:DatatypeProperty>

<daml_oil:DatatypeProperty rdf:ID="strasse">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Adresse"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:DatatypeProperty rdf:ID="geburtstag">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Person"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
```

```
</daml_oil:DatatypeProperty>

<daml_oil:DatatypeProperty rdf:ID="postleitzahl">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Adresse"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#integer"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="betrifft_alte_Adresse">
  <daml_oil:range rdf:resource="http://example.com/umzug#Adresse_Alt"/>
  <daml_oil:domain rdf:resource="#Umzug"/>
</daml_oil:ObjectProperty>

<daml_oil:DatatypeProperty rdf:ID="datum">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Datum"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="PostM">
  <rdfs:comment>Umzugsmeldung an die Post.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Meldung"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="geburtsname">
  <daml_oil:domain rdf:resource="#Person"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="betrifft_Haushalt">
  <daml_oil:range rdf:resource="#Haushalt"/>
  <daml_oil:domain rdf:resource="#Umzug"/>
</daml_oil:ObjectProperty>

<daml_oil:DatatypeProperty rdf:ID="geschlecht">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Person"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Ontology rdf:ID="">
  <daml_oil:versionInfo>3.0</daml_oil:versionInfo>
  <rdfs:comment></rdfs:comment>
</daml_oil:Ontology>

</rdf:RDF>
```

B Systemontologie und ihre Instanzen in DAML+OIL-Darstellung

Dieses ist die DAML+OIL Ausgabe der Systemontologie mit den Instanzen einer Datenweiterleitungsdefinition von Protege2000. Die Instanzen sind durch das Präfix *system* zu erkennen.

```
<rdf:RDF
  xmlns:XMLSchema ="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:daml_oil ="http://www.daml.org/2001/03/daml+oil#"
  xmlns:system ="http://example.com/system#"
  xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
  xmlns ="http://example.com/system#"
  >

<system:URI_in_Anliegenontologie rdf:ID="Kind">
</system:URI_in_Anliegenontologie>

<system:Erforderliche_Instanz rdf:ID="00005">
  <system:definiert_fuer rdf:resource="#Datum"/>
</system:Erforderliche_Instanz>

<system:Erforderliche_Instanz rdf:ID="00004">
  <system:benoetigt_Property rdf:resource="#00009"/>
  <system:definiert_fuer rdf:resource="#Adresse_Neu"/>
</system:Erforderliche_Instanz>

<system:Erforderliche_Instanz rdf:ID="00002">
  <system:benoetigt_Property rdf:resource="#00011"/>
  <system:benoetigt_Property rdf:resource="#00012"/>
  <system:definiert_fuer rdf:resource="#Person"/>
</system:Erforderliche_Instanz>

<system:Definierte_Datenweiterleitung rdf:ID="00001">
  <rdfs:comment></rdfs:comment>
  <system:benoetigt_Instanz rdf:resource="#00002"/>
</system:Definierte_Datenweiterleitung>

<daml_oil:Class rdf:ID="Abhaengige_Informationen">
  <rdfs:comment>
    Wenn zu einer Klasse unter bestimmten Voraussetzungen spezielle
    Informationen angegeben werden sollen, kann dies über diese Klasse
    realisiert werden.
  </rdfs:comment>
</daml_oil:Class>
```

```
<system:URI_in_Anliegenontologie rdf:ID="geht_in_Klasse">
</system:URI_in_Anliegenontologie>

<system:URI_in_Anliegenontologie rdf:ID="hat_Kind">
</system:URI_in_Anliegenontologie>

<daml_oil:ObjectProperty rdf:ID="benoetigt_Property">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Erforderliche_Instanz"/>
  <daml_oil:range rdf:resource="#Erforderliche_Properties"/>
</daml_oil:ObjectProperty>

<system:Erforderliche_Properties rdf:ID="00024">
  <system:definiert_fuer rdf:resource="#geht_in_Klasse"/>
</system:Erforderliche_Properties>

<system:Erforderliche_Instanz rdf:ID="00023">
  <system:definiert_fuer rdf:resource="#Datum"/>
</system:Erforderliche_Instanz>

<system:Erforderliche_Properties rdf:ID="00022">
  <system:benoetigt_Instanz rdf:resource="#00023"/>
  <system:definiert_fuer rdf:resource="#hat_geburtsdatum"/>
</system:Erforderliche_Properties>

<system:Erforderliche_Instanz rdf:ID="00021">
  <system:benoetigt_Property rdf:resource="#00022"/>
  <system:benoetigt_Property rdf:resource="#00024"/>
  <system:definiert_fuer rdf:resource="#Kind"/>
</system:Erforderliche_Instanz>

<system:Erforderliche_Instanz rdf:ID="00020">
  <system:definiert_fuer rdf:resource="#Datum"/>
</system:Erforderliche_Instanz>

<system:URI_in_Anliegenontologie rdf:ID="Schule_XY">
  <system:hat_Datenweiterleitung rdf:resource="#00001"/>
</system:URI_in_Anliegenontologie>

<daml_oil:Class rdf:ID="Erforderliche_Instanz">
  <rdfs:comment></rdfs:comment>
</daml_oil:Class>

<system:URI_in_Anliegenontologie rdf:ID="hat_geburtsdatum">
</system:URI_in_Anliegenontologie>

<daml_oil:ObjectProperty rdf:ID="definiert_fuer">
  <daml_oil:domain rdf:resource="#Erforderliche_Instanz"/>
```

```
  <daml_oil:domain rdf:resource="#Erforderliche_Properties"/>
  <daml_oil:range rdf:resource="#URI_in_Anliegenontologie"/>
</daml_oil:ObjectProperty>

<system:URI_in_Anliegenontologie rdf:ID="Person">
</system:URI_in_Anliegenontologie>

<daml_oil:ObjectProperty rdf:ID="benoetigt_Instanz">
  <rdfs:comment></rdfs:comment>
  <daml_oil:domain rdf:resource="#Abhaengige_Informationen"/>
  <daml_oil:domain rdf:resource="#Definierte_Datenweiterleitung"/>
  <daml_oil:range rdf:resource="#Erforderliche_Instanz"/>
  <daml_oil:domain rdf:resource="#Erforderliche_Properties"/>
</daml_oil:ObjectProperty>

<system:URI_in_Anliegenontologie rdf:ID="gueltig_bis">
</system:URI_in_Anliegenontologie>

<daml_oil:Class rdf:ID="Definierte_Datenweiterleitung">
  <rdfs:comment>
    Instanzen dieser Klasse müssen auch Instanzen in der Umzugsontologie sein.
    Diese Instanz ist der Ausgangsknoten für die Datenzusammenstellung
    für die Weiterleitung an andere Webservices.
  </rdfs:comment>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="hat_Datenweiterleitung">
  <daml_oil:range rdf:resource="#Definierte_Datenweiterleitung"/>
  <daml_oil:domain rdf:resource="#URI_in_Anliegenontologie"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UnambiguousProperty"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
</daml_oil:ObjectProperty>

<system:URI_in_Anliegenontologie rdf:ID="Datum">
</system:URI_in_Anliegenontologie>

<system:Erforderliche_Properties rdf:ID="00014">
</system:Erforderliche_Properties>

<daml_oil:DatatypeProperty rdf:ID="hat_internen_Verweis">
  <rdfs:comment>
    Gibt den Wert für den internen Verweis auf das zugehörige
    Informationselement an.
  </rdfs:comment>
  <daml_oil:domain rdf:resource="#Abhaengige_Informationen"/>
  <daml_oil:domain rdf:resource="#URI_in_Anliegenontologie"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UnambiguousProperty"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
```

```
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<system:Erforderliche_Properties rdf:ID="00012">
  <system:benoetigt_Instanz rdf:resource="#00021"/>
  <system:definiert_fuer rdf:resource="#hat_Kind"/>
</system:Erforderliche_Properties>

<system:Erforderliche_Properties rdf:ID="00011">
  <system:benoetigt_Instanz rdf:resource="#00004"/>
  <system:benoetigt_Instanz rdf:resource="#00007"/>
  <system:definiert_fuer rdf:resource="#hat_Adresse"/>
</system:Erforderliche_Properties>

<system:URI_in_Anliegenontologie rdf:ID="Adresse_Neu">
</system:URI_in_Anliegenontologie>

<system:Erforderliche_Properties rdf:ID="00010">
  <system:benoetigt_Instanz rdf:resource="#00020"/>
  <system:definiert_fuer rdf:resource="#gueltig_bis"/>
</system:Erforderliche_Properties>

<daml_oil:Class rdf:ID="URI_in_Anliegenontologie">
  <rdfs:comment>
    Alle Instanzen dieser Klasse stellen nach Austausch der
    Namespace-Komponente des URIs einen URI einer Anliegenontologie dar.
    Hierbei wird nicht zwischen Klassen oder Properties unterschieden.
  </rdfs:comment>
</daml_oil:Class>

<system:URI_in_Anliegenontologie rdf:ID="hat_Adresse">
</system:URI_in_Anliegenontologie>

<daml_oil:Class rdf:ID="Erforderliche_Properties">
  <rdfs:comment></rdfs:comment>
</daml_oil:Class>

<system:URI_in_Anliegenontologie rdf:ID="gueltig_ab">
</system:URI_in_Anliegenontologie>

<daml_oil:ObjectProperty rdf:ID="hat_abhaengige_Informationen">
  <rdfs:comment>
    Informationselemente können abhängig von vorhandenen Instanzen sein.
  </rdfs:comment>
  <daml_oil:range rdf:resource="#Abhaengige_Informationen"/>
  <daml_oil:domain rdf:resource="#URI_in_Anliegenontologie"/>
  <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UnambiguousProperty"/>
</daml_oil:ObjectProperty>
```

```
<system:Erforderliche_Properties rdf:ID="00009">
  <system:benoetigt_Instanz rdf:resource="#00005"/>
  <system:definiert_fuer rdf:resource="#gueltig_ab"/>
</system:Erforderliche_Properties>

<system:URI_in_Anliegenontologie rdf:ID="Adresse_Alt">
</system:URI_in_Anliegenontologie>

<daml_oil:DatatypeProperty rdf:ID="hat_externen_Verweis">
  <rdfs:comment>
    Gibt Werte für externe Verweise an. Dies können z.B. Links zu
    externen Web Sites sein.
  </rdfs:comment>
  <daml_oil:domain rdf:resource="#Abhaengige_Informationen"/>
  <daml_oil:domain rdf:resource="#URI_in_Anliegenontologie"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Ontology rdf:ID="">
  <daml_oil:versionInfo>2.0</daml_oil:versionInfo>
  <rdfs:comment>Die Systemontologie</rdfs:comment>
</daml_oil:Ontology>

<system:Erforderliche_Instanz rdf:ID="00007">
  <system:benoetigt_Property rdf:resource="#00010"/>
  <system:definiert_fuer rdf:resource="#Adresse_Alt"/>
</system:Erforderliche_Instanz>

</rdf:RDF>
```