

Diplomarbeit-Nr.: 2180
Extraktion einer OWL-Ontologie
Marc Eichler

Studiengang:	Informatik
Prüfer:	Prof. Dr.-Ing. B. Mitschang
Betreuer:	Christoph Mangold
Beginn am:	13.1.2004
Beendet am:	13.7.2004
CR-Nummer:	I 2.4

Inhalt

1	Einleitung	5
1.1	Beschreibung der Aufgabe	5
1.2	Begriffe.....	5
1.2.1	Objektorientierung allgemein.....	6
1.2.2	Java.....	6
1.2.3	OWL.....	6
1.2.4	föederal-Informationsarchitektur	6
2	Die föederal-Informationsarchitektur	8
2.1	Zweck der föederal-Informationsarchitektur	8
2.2	Datenmodell	9
2.2.1	Die Grundlage semantisches Netz.....	9
2.2.2	Instance-Entities	10
2.2.3	Class-Entities.....	10
2.2.4	Meta-Entities	12
2.2.5	Doppelte Instanziierung	12
2.2.6	Units	13
2.3	Implementierung	13
2.3.1	Schichtenarchitektur.....	13
2.3.2	Die „mind“-Schicht	14
3	OWL.....	20
3.1	Grundlagen	21
3.1.1	Beziehungen zu anderen Sprachen.....	21
3.1.2	Die drei Versionen von OWL	23
3.1.2.1	OWL Lite	23
3.1.2.2	OWL DL	23
3.1.2.3	OWL Full	24
3.2	Beschreibung der wichtigsten Konstrukte.....	24
3.2.1	Abkürzungen	24
3.2.2	Namespaces.....	25
3.2.3	Einfache Klassen	26
3.2.3.1	Klassennamen.....	26
3.2.3.2	Subklassen	27
3.2.4	Benannte Individuen	27
3.2.4.1	Definition	27
3.2.4.2	Identität.....	28
3.2.4.3	Unterschiedlichkeit zweier Individuen.....	28
3.2.4.4	Unterschiedlichkeit mehrerer Individuen.....	29
3.2.5	Eigenschaften	29
3.2.5.1	Instanziierung	31
3.2.5.2	Domain und Range.....	31

3.2.5.3 Hierarchien	32
3.2.5.4 Charakteristiken	33
3.2.6 Beschränkungen	36
3.2.6.1 allValuesFrom	37
3.2.6.2 someValuesFrom.....	38
3.2.6.3 hasValue	38
3.2.6.4 minCardinality.....	39
3.2.6.5 maxCardinality	40
3.2.6.6 cardinality.....	40
3.2.7 Komplexe Klassen.....	40
3.2.7.1 Aufzählung aller Individuen.....	40
3.2.7.2 Schnitt von Klassen	41
3.2.7.3 Vereinigung von Klassen	42
3.2.7.4 Komplement von Klassen	42
3.2.7.5 Äquivalente Klassen.....	43
3.2.7.6 Klassen ohne gemeinsame Individuen	43
3.2.8 Anonyme Individuen.....	44
3.3 Äquivalenz von Klassen und Individuen in OWL Full.....	45
3.4 Reasoning	46
4 Extraktion von OWL-Ontologien aus semantischen Netzen der föderal- Informationsarchitektur	48
4.1 Allgemeiner Vergleich der Beschreibungsmittel von OWL und föderal- Informationsarchitektur	49
4.2 Umsetzung der Extraktion.....	50
4.2.1 Beispielnetze	50
4.2.2 Identifizierung relevanter Informationen	52
4.2.3 Leitlinien beim Design der OWL-Ontologien	58
4.2.4 Modellierungsansätze für grundlegende Konstrukte.....	58
4.2.4.1 Entities.....	58
4.2.4.2 Assoziationen und Assoziationstypen.....	59
4.2.4.3 Constraints.....	59
4.2.4.4 Eindeutige Namen	60
4.2.5 Extraktion der OWL Lite-Ontologie	62
4.2.5.1 Grundsätzliches	62
4.2.5.2 Parameter von Methoden	63
4.2.5.3 Definitionen von Klassen	66
4.2.5.4 Definitionen von Eigenschaften	68
4.2.5.5 Individuen und Instanzen von Eigenschaften.....	74
4.2.6 Extraktion der OWL DL-Ontologie	77
4.2.6.1 Grundsätzliches	77
4.2.6.2 Definitionen von Eigenschaften	77
4.2.6.3 Definition von Klassen.....	80
4.2.6.4 Individuen und Instanzen von Eigenschaften.....	83
4.2.7 Extraktion der OWL Full-Ontologie	84
4.2.7.1 Grundsätzliches	84
4.2.7.2 Doppelte Instanziierung	85
4.2.7.3 Attribute von Assoziationstypen	88

5 Fazit.....	89
6 Abbildungsverzeichnis.....	91
7 Literaturverzeichnis.....	92
8 Anhang.....	93
A Aus dem Automobilbeispiel extrahierte OWL Lite-Ontologie.....	93
B Aus dem Automobilbeispiel extrahierte OWL DL-Ontologie.....	110
C Aus dem Automobilbeispiel extrahierte OWL Full-Ontologie.....	119

1 Einleitung

1.1 Beschreibung der Aufgabe

Informationen lassen sich mit Hilfe von sogenannten "semantischen Netzen" darstellen. Dabei handelt es sich um Graphen, die aus Knoten und Kanten besteht. Die Knoten stehen für Konzepte und die Kanten für Relationen zwischen diesen Konzepten.

Im Rahmen des föderal-Projekts entstand eine Informationsarchitektur, die ein semantisches Netz nutzt, um Informationen über den Aufbau von Maschinen zu verwalten. Die Knoten werden dabei von Maschinen bzw. Maschinenteilen gebildet, über deren Beziehungen zueinander mit Hilfe der Relationen (in der föderal-Informationsarchitektur "Assoziationen" genannt) Aussagen gemacht werden können.

Mit Hilfe der von der Semantic Web-Initiative des World Wide Web Consortiums entwickelte Sprache OWL [1] [2] [3] lassen sich ähnlich wie mit semantischen Netzen, Informationen über Konzepte und die Relationen, die zwischen ihnen bestehen darstellen. Ihre Aufgabe ist es, den derzeit im World Wide Web vorherrschenden menschenlesbaren Informationen, von Maschinen auswertbare Informationen zur Seite stellen zu können. Es gibt drei unterschiedlich mächtige Versionen von OWL, die "OWL Lite", "OWL DL" und "OWL Full" genannt werden.

Im Rahmen dieser Arbeit werden die Beschreibungsmittel eines semantischen Netzes der föderal-Informationsarchitektur [4] mit denen von OWL verglichen. Es werden Konzepte vorgestellt, wie konkrete Datenmodelle des semantischen Netzes in OWL beschrieben werden können. Diese werden anhand von OWL-Ontologien verdeutlicht, die mit Hilfe eines Software-Prototypen erzeugt wurden, der OWL-Ontologien aus semantischen Netze der föderal-Informationsarchitektur extrahiert.

1.2 Begriffe

Alle im Zusammenhang mit dieser Arbeit verwendeten Technologien bedienen sich des Paradigmas der Objektorientierung [5]. Das hat zur Folge, dass bei der Verwendung von Begriffen wie „Klasse“ und „Instanz“, ohne weitere Erläuterungen oft nicht klar ist, auf was sie sich beziehen. Dieses Problem tritt zum Beispiel bei der Beschreibung der Umsetzung von Klassen der föderal-Informationsarchitektur in OWL-Klassen mit Hilfe eines Javaprogramms

auf, das seinerseits wieder aus verschiedenen Klassen besteht. Im Rahmen dieser Arbeit werden darum im Zweifelsfall die im Folgenden angegebenen, spezifischeren Begriffe verwendet. Auf weitere Begriffe, die innerhalb der Erläuterungen verwendet werden, wird im Rahmen der entsprechenden Kapitel näher eingegangen. An dieser Stelle geht es zunächst darum, die Zuordnung von Begriffen zu Technologien darzustellen.

1.2.1 Objektorientierung allgemein

- Klasse: Der Begriff „Klasse“, ohne weiteren Zusatz, wird verwendet, wenn es keinen Bezug zu auf eine speziellen Sprache gibt oder wenn die Sprache aus dem Zusammenhang zweifelsfrei hervorgeht.
- Instanz: Der Begriff „Instanz“, ohne weiteren Zusatz, wird verwendet, wenn es keinen Bezug zu auf eine speziellen Sprache gibt oder wenn die Sprache aus dem Zusammenhang zweifelsfrei hervorgeht.

1.2.2 Java

- Java-Klasse: Eine Java-Klasse ist Teil eines Javaprogramms. Sie wird statisch mit Hilfe des Java-Schlüsselworts `class` definiert.
- Java-Instanz: Eine Java-Instanz ist Teil eines Javaprogramms. Sie wird zur Laufzeit mit Hilfe des Java-Operators `new` erzeugt.

1.2.3 OWL

- OWL-Klasse: Eine OWL-Klasse ist Teil einer OWL-Ontologie. Sie wird durch die Tags `<owl:Class>` oder `<owl:Restriction>` definiert.
- OWL-Individuum: Die Instanzen von OWL-Klasse heißen OWL-Individuen. Ein OWL-Individuum ist Teil einer OWL-Ontologie. Es wird durch ein Tag definiert, dessen Name dem Namen der instanziierten Klasse entspricht.

1.2.4 föderal-Informationsarchitektur

- Meta-Entity: Eine Meta-Entity ist eine Entity im semantischen Netz der föderal-Informationsarchitektur, welche sich in der „Meta-Schicht“ befindet.
- Class-Entity: Eine Class-Entity ist eine Entity im semantischen Netz der föderal-Informationsarchitektur, welche sich in der „Class-Schicht“ befindet.

Instance-Entity: Eine Instance-Entity ist eine Entity im semantischen Netz der föderal-
Informationsarchitektur, welche sich in der „Instance-Schicht“ befindet.

2 Die föderal-Informationsarchitektur

Dieses Kapitel beschreibt die föderal-Informationsarchitektur, aus deren Datenmodell OWL-Ontologien extrahiert werden sollen.

Im Abschnitt "2.1 Zweck der föderal-Informationsarchitektur" wird zunächst dargestellt, was die Motivation für die Entwicklung der föderal-Informationsarchitektur war und welche Probleme durch sie gelöst werden.

Es folgt unter "2.2 Datenmodell" die Erläuterung des abstrakten Konzepts, das der Strukturierung der Informationen in der föderal-Informationsarchitektur zu Grunde liegt. Es werden die verschiedenen Konstrukte vorgestellt, die für die Darstellung von Informationen zur Verfügung stehen.

Schließlich wird im Abschnitt "2.3 Implementierung" die Software, das "föderal Engineering-Framework", beschrieben, die das abstrakte Konzept implementiert.

2.1 Zweck der föderal-Informationsarchitektur

Im Maschinenbau werden Maschinen aus sogenannten "mechatronischen" Komponenten zusammengesetzt. Mechatronische Komponenten verknüpfen mechanische, elektronische und softwaretechnische Komponenten und realisieren damit innerhalb der Maschine eine bestimmte Funktion.

Die verschiedenen an der Entwicklung beteiligten Abteilungen wie Mechanikkonstruktion, Elektrokonstruktion, Softwareentwicklung und Dokumentationsabteilung erstellen Ihre Unterlagen dabei mit unterschiedlichen IT-Werkzeugen. Diese verfügen oft über einen "Baukasten", in dem einmal erstellte disziplinspezifische Unterlagen abgelegt werden können und dann für die Wiederverwendung in anderen Projekten zur Verfügung stehen.

Die Wiederverwendung von mechatronischen Komponenten, die aus mehreren disziplinspezifischen Unterlagen zusammengesetzt sind, ist im Allgemeinen sehr aufwändig, da es zwischen den verschiedenen IT-Werkzeugen der an der Entwicklung beteiligten Abteilungen keine Kommunikation gibt. Es fehlt eine Systematik zur gemeinsamen Verwaltung von Unterlagen. Dokumente aus bereits existierenden Komponenten werden durch schlichtes Kopieren in neue Projekte übernommen. Hierbei kann es zur Übernahme von Fehlern aus alten Projekten kommen. Durch fehlende projektübergreifende Koordination

besteht außerdem die Gefahr, dass Komponenten neu entwickelt werden, an deren Stelle auch Komponenten aus dem Baukasten eines anderen Projekts wiederverwendet werden könnten. Die "föederal-Informationsarchitektur" stellt ein firmenweites, abteilungs- und projektübergreifendes Datenmodell für Maschinen- und Anlagenbauer zur Verfügung, das es ermöglicht, Informationen über entwickelte Maschinen beziehungsweise deren Komponenten, funktional strukturiert zu verwalten. Das bedeutet, dass alle Komponenten nicht auf Disziplinen, sondern auf technische Funktionen bezogen sind. Es gibt also zu jeder Funktionseinheit einer Maschine, zum Beispiel einer Achse, eine Komponente im Informationsmodell, welche die verschiedenen zugehörigen disziplinspezifischen Dokumente bündelt. Im Falle der Achse würden hierzu etwa eine Zeichnung der räumlichen Ausdehnung der Achse gehören, ein Stromlaufplan des Antriebsmotors, die Software zur Ansteuerung und eine technische Dokumentation, die die Funktion beschreibt. Die zusammenfassende Komponente stellt einen zentralen Ort zur Verwaltung der zu der Funktionseinheit gehörenden Informationen bereit.

Wird eine derartige mechatronische Komponente in ein neues Projekt übernommen, werden damit automatisch auch immer die aktuellsten Versionen der verschiedenen disziplinspezifischen Informationen übernommen. Hierdurch wird das Kopieren veralteter Unterlagen als Fehlerquelle eliminiert. Da das Modell projektübergreifend ist, stehen jedem Projekt alle von anderen Projekten entwickelten Komponenten zur Verfügung, was die Wahrscheinlichkeit einer doppelten Entwicklung zu verringern hilft.

2.2 Datenmodell

2.2.1 Die Grundlage semantisches Netz

Die Grundlage der föederal-Informationsarchitektur bildet ein semantisches Netz, in dem alle Informationen über in der Firma entwickelte Maschinen bzw. Komponenten von Maschinen und deren Beziehungen zueinander abgebildet sind. Ein mögliches Beispiel für einen Ausschnitt aus einem solchen Netz zeigt Abb. 1.

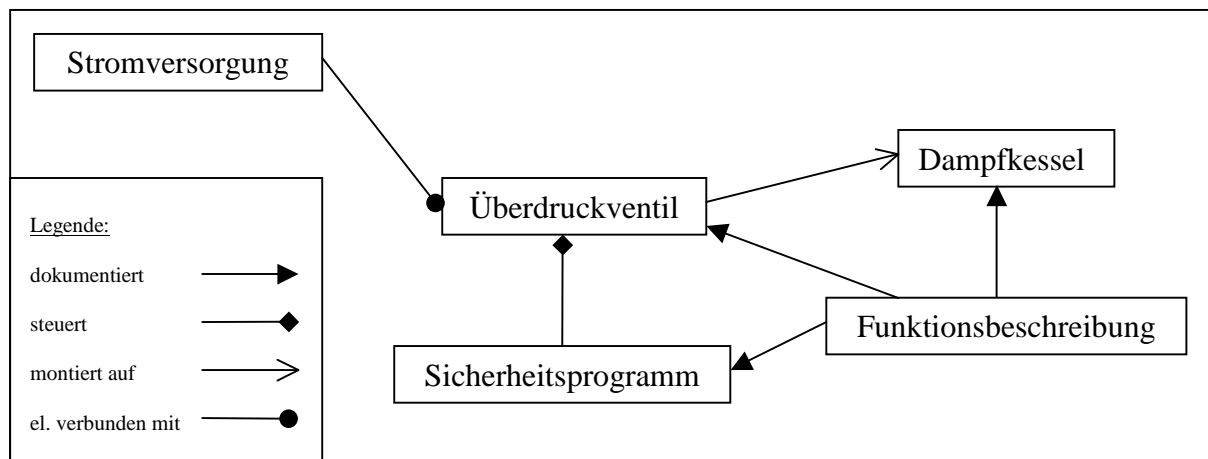


Abb. 1: Beispielausschnitt eines semantischen Netzes der föderal-Informationsarchitektur

Im semantischen Netz der föderal-Informationsarchitektur werden die Kanten „Assoziationen“ und die Knoten „Entities“ genannt. Um die im Netz enthaltenen Informationen zu strukturieren, werden die Entities in drei Untergruppen namens „Instance-Entity“, „Class-Entity“ und „Meta-Entity“ unterteilt, deren Unterschiede im Folgenden erläutert werden.

2.2.2 Instance-Entities

Instance-Entities repräsentieren tatsächlich gebaute Maschinen oder Maschinenteile. Zu ihnen gibt es eine physische Entsprechung in der realen Welt. Die Menge aller Instance-Entities eines Netzes wird als die „Instance-Schicht“ bezeichnet.

2.2.3 Class-Entities

Class-Entities modellieren Gemeinsamkeiten zwischen verschiedenen Instance-Entities. Für Maschinenbauer stellen die Class-Entities einen Baukasten von bereits entwickelten, wiederverwendbaren Komponenten dar, die sich durch die bloße Änderung von Parameterwerten (die in der föderal-Informationsarchitektur ihrerseits wieder als Entities modelliert sind) an neue Verwendungszwecke anpassen lassen.

Die Menge aller Class-Entities eines Netzes wird als „Class-Schicht“ bezeichnet.

Analog der Zuordnung einer Instanz zu einer Klasse in der objektorientierten Programmierung - mittels eines Operators "instanceOf" -, kann in der föderal-Informationsarchitektur einer Instance-Entity eine Class-Entity zugeordnet werden. Wie bei der objektorientierten Programmierung können nur solche Instance-Entities, die strukturell

gleich aufgebaut sind und sich nur durch die Werte ihrer Parameter unterscheiden, der selben übergeordneten Class-Entity zugeordnet werden. Abb. 2 zeigt hierfür ein Beispiel.

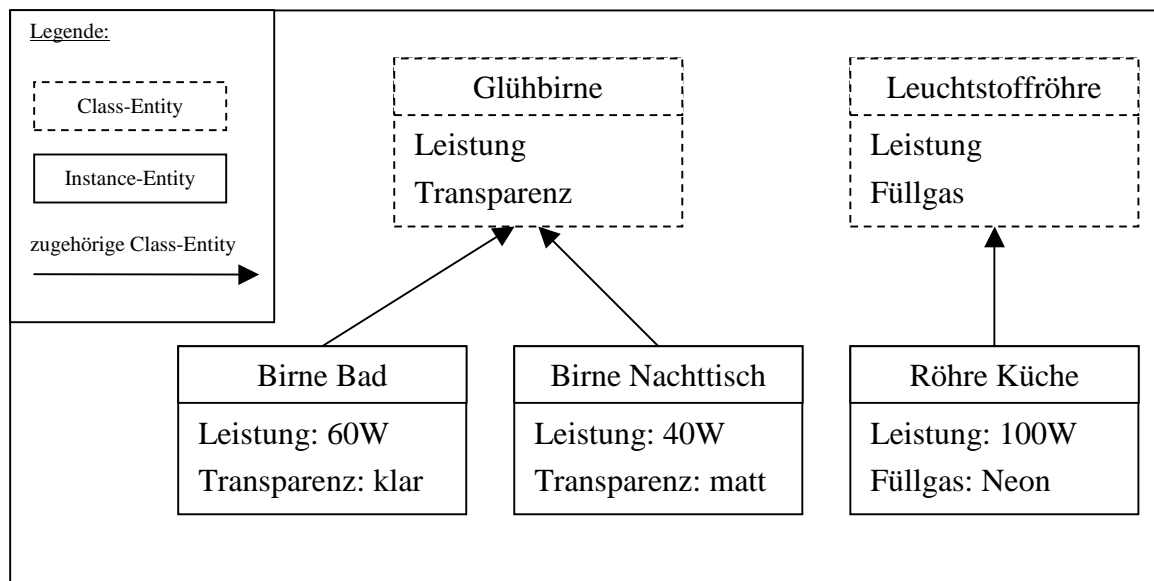


Abb. 2: Beispiel für die Zuordnung von Instance-Entities zu Class-Entities

In Abb. 2 sind zwei Class-Entities zu sehen. „Glühbirnen“ besitzen die Parameter „Leistung“ und „Transparenz“, Leuchtstoffröhren dagegen die Parameter „Leistung“ und „Füllgas“. Die Instance-Entities „Birne Bad“ und „Birne Nachttisch“ gehören beide zur Class-Entity „Glühbirne“. Sie unterscheiden sich nur in den Werten ihrer Parameter.

Die Instance-Entity „Röhre Küche“ dagegen kann nicht der selben Class-Entity zugeordnet werden, wie die beiden Birnen. Sie unterscheidet sich von ihnen nicht nur in den Werten der Parameter, sondern auch durch die Art der vorhandenen Parameter.

Andersherum betrachtet kann man sagen, dass wenn man eine Class-Entity hat, diese vorgibt, wie Instance-Entities aufgebaut sein müssen. Die Class-Entity „Glühbirne“ legt beispielsweise fest, dass entsprechende Instance-Entities einen Parameter „Leistung“ und einen Parameter „Transparenz“ haben müssen.

Um von einer Class-Entity eine neue Instance-Entity abzuleiten, erzeugt man also eine Instance-Entity mit den Parametern, die die Class Entity vorgibt und weist diesen Parametern Werte zu. Dasselbe geschieht in der objektorientierten Programmierung, wenn man eine Klasse instanziiert.

2.2.4 Meta-Entities

Meta-Entities modellieren Gemeinsamkeiten von verschiedenen Class-Entities und stellen somit eine zusätzliche Abstraktionsebene oberhalb der Class-Entities dar, zu der es in der herkömmlichen objektorientierten Programmierung keine Entsprechung gibt. Die Menge aller Meta-Entities eines Netzes wird als „Meta-Schicht“ bezeichnet.

Meta-Entities machen Vorgaben darüber, welche Class-Entities angelegt werden können und welche Beziehungen es zwischen ihnen gibt. Sie geben also vor, wie der Baukasten strukturiert ist. Mit ihrer Hilfe wird ein abstraktes Firmenmodell aufgebaut, das festlegt, in welchen Kategorien Komponenten im jeweiligen Unternehmen eingeteilt werden.

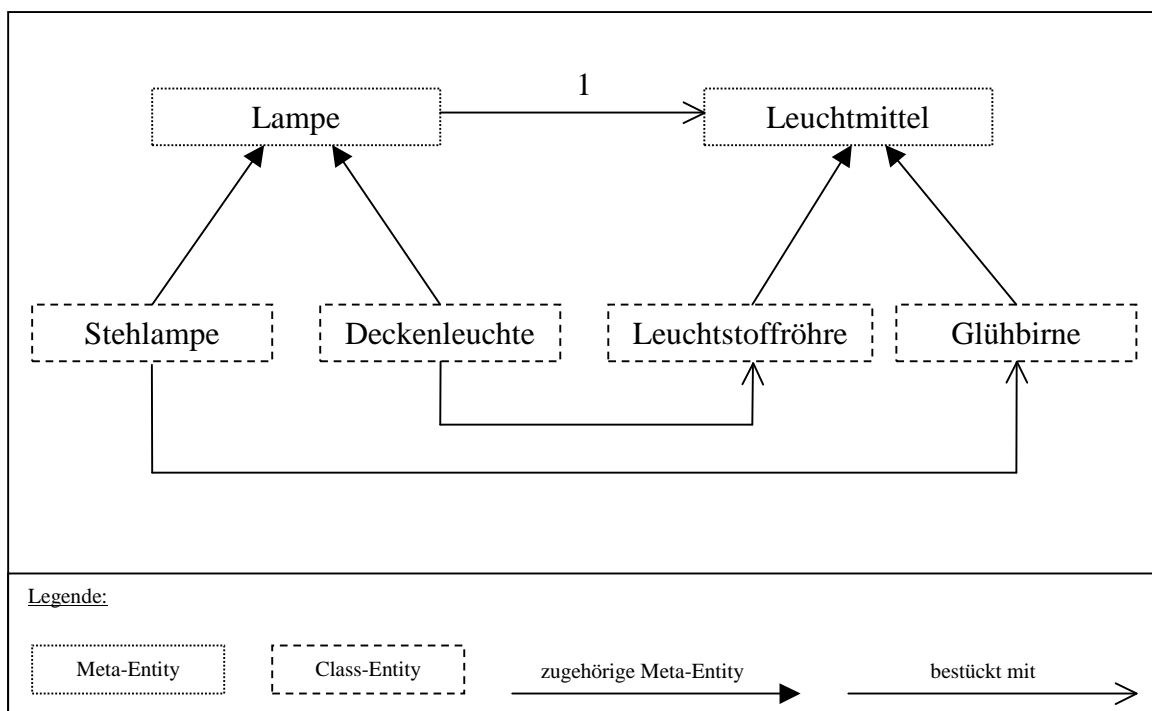


Abb. 3: Beispiel für die Zuordnung von Class-Entities zu Meta-Entities

Abb. 3 zeigt einen Ausschnitt aus einem Beispielnetz, das die Erzeugung von Class-Entities vom Typ „Lampe“ oder „Leuchtmittel“ regelt. Auf der Meta-Ebene wird festgelegt, dass in der Firma Lampen von Leuchtmitteln unterschieden werden und dass jede Lampe mittels einer Assoziation „bestücktMit“ mit genau einem Leuchtmittel verbunden sein muss.

Um von einer Meta-Entity eine neue Class-Entity abzuleiten, erzeugt man eine Class-Entity und stellt die Assoziationen her, die die Meta-Entity vorgibt.

2.2.5 Doppelte Instanziierung

Die Parallelen, die es zwischen der Ableitung einer Instance-Entity von einer Class-Entity und der Ableitung einer Class-Entity von einer Meta-Entity gibt, sind auffällig. In beiden

Fällen ist das Ergebnis eine Entity, die den von der übergeordneten Abstraktionsebene festgelegten Regeln entspricht.

In der objektorientierten Programmierung spricht man beim Übergang von der Klasse zur Instanz, bei dem die Instanz entsprechend den Vorgaben der Klasse aufgebaut wird von „Instanziierung“. Angelehnt an diesen Begriff könnte man bei der föderal- Informationsarchitektur also von einer „doppelten Instanziierung“, bestehend aus dem Übergang von der Meta-Entity zur Class-Entity und dem Übergang von der Class-Entity zur Instance-Entity sprechen.

2.2.6 Units

Ein weiteres Konzept bei der Strukturierung der Informationen mittels der föderal- Informationsarchitektur ist die Zuordnung von Bestandteilen des semantischen Netzes zu einer „Unit“. Zusammengehörige Komponenten, zum Beispiel solche, die Teil einer bestimmte Maschine sind, werden der selben Unit zugeordnet. Units sind hierarchisch gegliedert. Jede Unit kann Unterunits besitzen, wobei der Schachtelungstiefe keine Grenzen gesetzt sind. Ein ähnliches Konzept findet man etwa in der Programmiersprache Java mit der Einteilung von Klassen in Packages.

2.3 Implementierung

Die Software, die die föderal-Informationsarchitektur implementiert, ist in der Programmiersprache Java geschrieben und trägt den Namen „föderal Engineering-Framework“.

2.3.1 Schichtenarchitektur

Die Software ist in mehrere, übereinander liegende Sichten gegliedert, die in Abbildung 4 schematisch dargestellt sind.

Auf der untersten Ebene wird das semantische Netz auf ein Standard-DBMS abgebildet. Dieses sorgt für die persistente Speicherung der Daten und ermöglicht die transaktionssichere Zusammenarbeit mehrerer verteilter Benutzer.

Oberhalb der Datenbankschicht befindet sich, darauf aufbauend, die „semantic net“-Schicht. Sie besteht aus einer Sammlung von Java-Klassen, mit deren Hilfe sich semantische Netze, bestehend aus Knoten und Kanten, aufbauen lassen.

Die nächste Schicht oberhalb der „semantic net“-Schicht ist die „mind“-Schicht. Das von der „semantic net“-Schicht zur Verfügung gestellte Netz wird von der „mind“-Schicht benutzt und erweitert, um die speziellen Eigenschaften der föderal-Informationsarchitektur zu modellieren. So wird in der „mind“-Schicht nicht mehr nur zwischen Knoten und Kanten unterschieden, sondern die Knoten werden in Meta-Entities, Class-Entities und Instance-Entities unterteilt.

Oberhalb der „mind“-Schicht können schließlich Anwendungen aufsetzen, die die Funktionen der föderal-Informationsarchitektur nutzen.

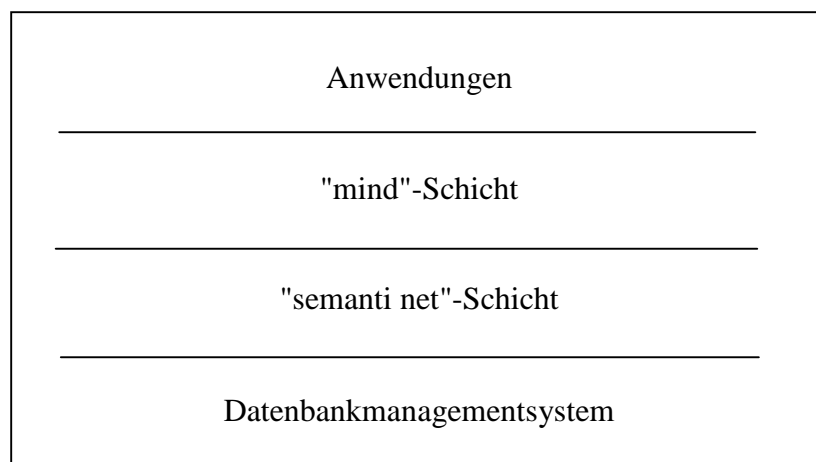


Abb. 4: Schichtstruktur der föderal-Informationsarchitektur

2.3.2 Die „mind“-Schicht

Die "mind"-Schicht ist die Abstraktionsebene, auf der das semantische Netz in dieser Arbeit betrachtet wird. Ihre Beschreibungsmittel sind es, die mit denen von OWL verglichen werden. In diesem Abschnitt werden die verschiedenen Konstrukte beschrieben, aus denen sich die "mind"-Schicht zusammensetzt, denn die mit Ihrer Hilfe darstellbaren Informationen müssen in OWL umgesetzt werden.

Die Java-Klassen zum Zugriff auf die „mind“-Schicht finden sich im Paket `org.foederal.mind` des föderal Engineering-Framework. Dieses Paket definiert in erster Linie eine Reihe von Schnittstellen, die den Zugriff auf Teile des Netzes regeln. Zum Beispiel ist dort die Schnittstelle `IInstanceEntity` zu finden, welche den Zugriff auf Java-Instanzen ermöglicht, die Instance-Entities repräsentieren. Um die Lesbarkeit zu erhöhen werden im Folgenden Ausdrücke wie „Java-Instanzen einer Klasse, die SchnittstelleXY implementiert“ durch „Java-Instanzen von SchnittstelleXY“ abgekürzt.

Die Parallelen, die es bei der Ableitung einer Instance-Entity von einer Class-Entity und dem Instanzieren einer Java-Klasse gibt, legen die Vermutung nahe, dass Class-Entities auf Java-Klassen abgebildet werden und Instance-Entities auf Java-Instanzen. Schon auf Grund der fehlenden Entsprechung zur Meta-Entity in der objektorientierten Programmierung, wäre jedoch eine konsistente Weiterführung dieses Ansatzes im Bereich der Meta-Entities nur schwer vorstellbar. Bei der Implementierung der föderal-Informationsarchitektur wurde also ein anderer Weg gewählt.

Alle Bestandteile des semantischen Netzes werden durch Java-Instanzen repräsentiert. Konkret heißt das zum Beispiel, dass für jede Meta-Entity eine Java-Instanz von `IMetaEntity` in das Netz eingefügt wird, für jede Class-Entity eine Java-Instanz von `IClassEntity` und für jede Instance-Entity ein Java-Instanz von `IInstanceEntity`. Abb. 5 zeigt die wichtigsten Schnittstellen, die beim Zugriff auf das Netz zum Einsatz kommen und welche Vererbungsbeziehungen zwischen ihnen bestehen.

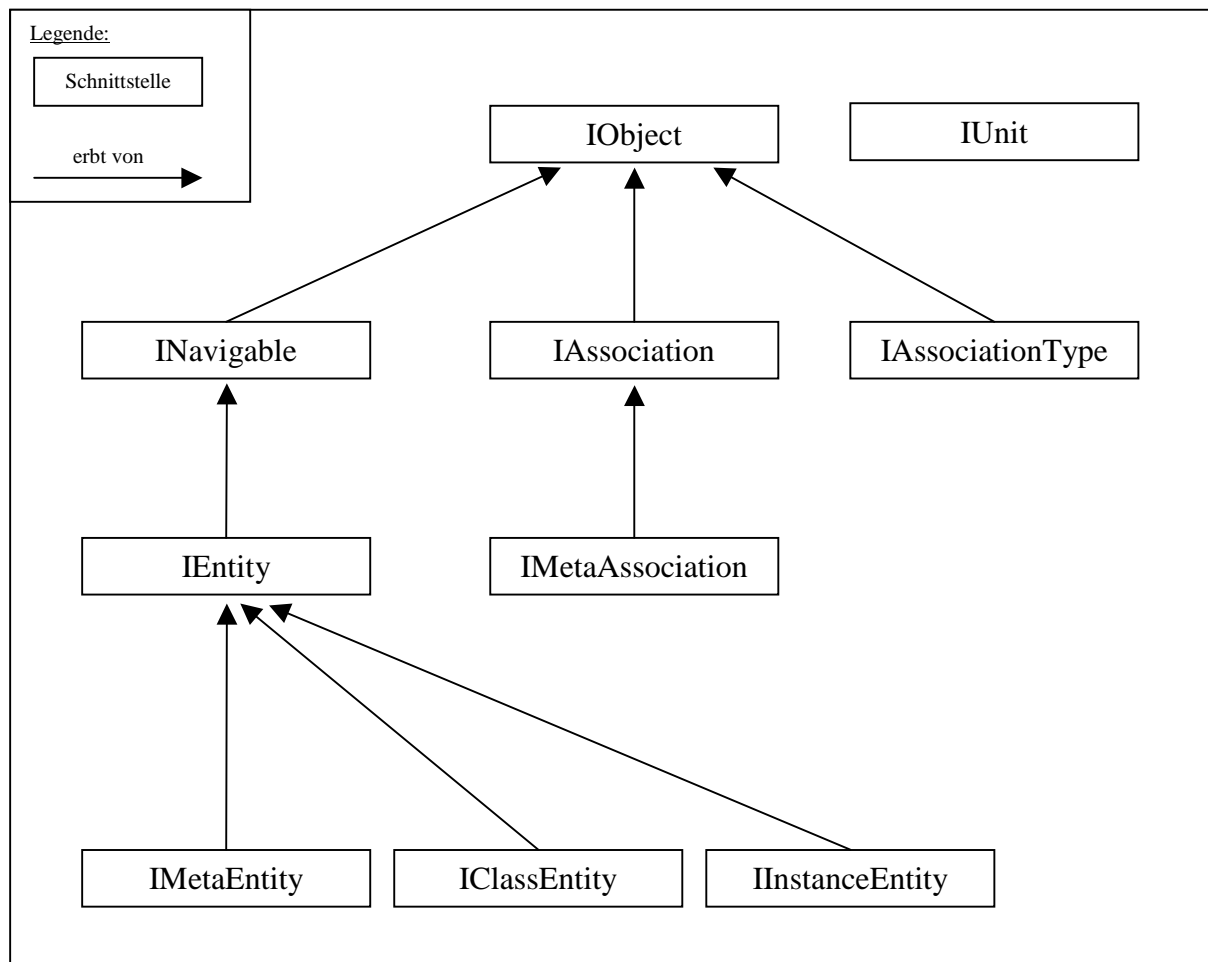


Abb. 5: Vererbungsbeziehungen zwischen Schnittstellen der "mind"-Schicht

Folgendes sind die wichtigsten Funktionen, der einzelnen Schnittstellen:

- IUnit:** In dieser Schnittstelle sind vor allem Methoden definiert, die einen Zugriff auf die zu einer Unit gehörenden Bestandteile des Netzes sowie die über- und untergeordneten Units der Unithierarchie ermöglichen.
- IObject:** Die hier definierten Methoden dienen größtenteils der Interaktion mit graphischen Benutzungsoberflächen und sind im Zusammenhang mit dieser Arbeit nicht von Interesse. Eine Ausnahme bildet die Methode `getOID()`, die eine global eindeutige Identifikation des `IObject` liefert.
- INavigable:** Diese Schnittstelle definiert Methoden, um durch das Netz zu navigieren und um es zu ändern. Sie dienen der technischen Handhabung des Netzes, liefern aber keinerlei inhaltliche Informationen und sind darum in Zusammenhang mit dieser Arbeit nicht von Interesse.
- IEntity:** Dies ist die Basisschnittstelle für alle Knoten des Netzes. Die Methode `getContent()` liefert den Inhalt eines Knotens zurück, also die eigentliche Information, deren Verwaltung die Aufgabe des föderal-Informationsmodells ist. Die Methode `getUnit()` liefert die Unit zurück, zu der der Knoten gehört.
- IMetaEntity:** Diese Schnittstelle dient dem Zugriff auf Java-Instanzen, die Meta-Entities repräsentieren. Sie definiert Methoden zum Auslesen einiger spezieller Attribute, über die nur Meta-Entities verfügen und auf die in dem Kapitel "4.2 Umsetzung der Extraktion" noch näher eingegangen wird. Außerdem stellt sie Methoden bereit, die Listen aller zugehörigen Class- und Instance-Entities zurückgeben.

- IClassEntity:** Diese Schnittstelle dient dem Zugriff auf Java-Instanzen, die Class-Entities repräsentieren. Die Methode `selectMetaEntity()` gibt die Meta-Entity zurück, von der die Class-Entity abgeleitet wurde. Die Methode `selectInstanceEntities()` gibt eine Liste aller zugehörigen Instance-Entities zurück.
- IInstanceEntity:** Diese Schnittstelle dient dem Zugriff auf Java-Instanzen, die Instance-Entities repräsentieren. Die Methoden `selectClassEntity()` liefert die Class-Entity zurück, von der die Instance-Entity abgeleitet wurde. Die Methode `getMetaEntity()` liefert die Meta-Entity zurück, von der die übergeordnete Class-Entity abgeleitet wurde.
- IAssociation:** Diese Schnittstelle dient dem Zugriff auf Java-Instanzen, die eine Assoziation zwischen zwei Entities repräsentieren. Neben einer Methode, mit deren Hilfe auf die beiden Entities zugegriffen werden kann, die durch die Assoziation verbunden werden, stellt sie auch eine Methode zur Verfügung, die den Assoziationstyp (s.u.) zurückgibt.
- IMetaAssociation:** Eine „Meta-Assoziation“ ist eine besondere Art von Assoziation zwischen zwei Meta-Entities. Diese Schnittstelle ermöglicht den Zugriff auf besondere Attribute, die normale Assoziationen nicht haben. So ist es zum Beispiel möglich, einer Meta-Assoziation einen String zuzuweisen, der eine „Rolle“ beschreibt.
- IAssociationType:** Jede Assoziation ist einem Assoziationstyp zugeordnet. Der Assoziationstyp legt fest, in welcher Weise eine Assoziation verwendet werden darf. Zum Beispiel kann er vorschreiben, dass Assoziationen eines Typs nur zwischen Entities bestimmter Schichten (Meta-Schicht, Class-Schicht oder Instance-Schicht) aufgebaut werden dürfen. Auch kann die Anzahl der gleichartigen Assoziationen, die eine Entity haben darf bzw. muss, festgelegt werden. So könnte man etwa für eine Assoziation „hatRad“ festlegen, dass eine Entity „Automobil“ genau vier solcher Assoziationen haben muss.

In Abb. 6 ist das Zusammenspiel von Instanzen der verschiedenen Schnittstellen bei der Verbindung zweier Entities durch eine Assoziation dargestellt. Dabei ist zu beachten, dass die Instanzen von `IEntity` natürlich nicht direkt von dieser Schnittstelle abgeleitet sind, sondern von einer der drei spezifischen Schnittstellen `IMetaEntity`, `IClassEntity` oder `IInstanceEntity`.

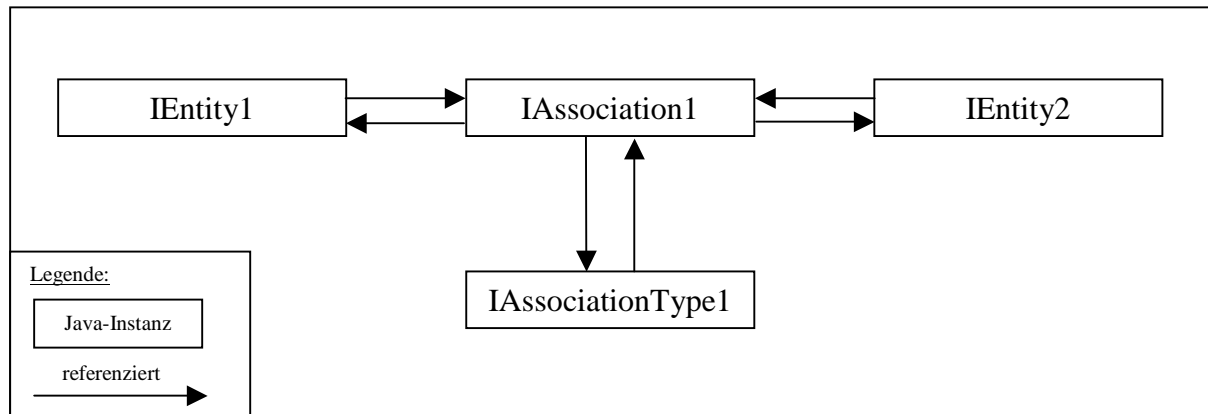


Abb. 6: Zusammenspiel der Schnittstellen bei der Verbindung zweier Entities

Über die bereits vorgestellten Schnittstellen hinaus gibt es noch einige Hilfsklassen und -schnittstellen, die dazu dienen, Daten intern zu strukturieren:

- `INetConstraints`: Ein Teil der Daten von `IAssociationType` ist in Instanzen von `INetConstraints` gekapselt. Jede Instanz von `IAssociationType` umfasst pro Schicht eine Instanz von `INetConstraints`, die angibt, wie Assoziationen des Typs in der jeweiligen Schicht verwendet werden dürfen.
- `MultiplicityIntervall`: Hierbei handelt es sich um eine Datenstruktur, die ein Intervall von Ganzzahlen repräsentiert.
- `Multiplicity`: Hierbei handelt es sich um eine Datenstruktur, die aus ggf. mehreren Instanzen von `MultiplicityIntervall` besteht, die zusammen eine Menge von Ganzzahlen repräsentieren.

UniDirection:

Von UniDirection gibt es genau zwei Instanzen: UniDirection.FORWARD und UniDirection.BACKWARD. Diese dienen zur Identifikation einer Richtung, wenn es darum geht, die beiden von einer Assoziation verbundenen Entities auseinander zu halten. Mit Hilfe der Übergabe von einer dieser beiden Konstanten kann man angeben, auf welche der beiden durch die Assoziation verbundenen Entities man sich bezieht.

Layer:

Ähnlich wie UniDirection für Richtungen, dient Layer dazu IDs für die verschiedenen Schichten zu definieren. Diese können bei Bedarf als Parameter an bestimmte Methoden übergeben werden, um zu spezifizieren, auf welche Schicht man sich bezieht. Es existieren folgende Instanzen:

ILayerConstants.META_LAYER,
ILayerConstants.CLASS_LAYER,
ILayerConstants.INSTANCE_LAYER,
ILayerConstants.ANY_LAYER,
ILayerConstants.NO_LAYER

Damit ist der Ausgangspunkt der Arbeit, die Bestandteile des semantischen Netzes der föderal-Informationsarchitektur beschrieben. Das folgende Kapitel widmet sich nun dem Zielpunkt, der Sprache OWL.

3 OWL

Mit Hilfe von OWL ist es möglich, den bis jetzt im World Wide Web vorherrschenden, nur menschenlesbaren Informationen, maschinenlesbare Versionen zur Seite zu stellen.

Derartige maschinenlesbare Informationen könnten unter anderem für den Betrieb von Suchmaschinen einen erheblichen Fortschritt darstellen. Die Tatsache, dass heutzutage Suchergebnisse noch durch einem reinen Vergleich von Zeichenketten ermittelt werden, führt etwa dazu, dass man bei einer Suche nach "Java" sowohl Seiten über die Programmiersprache als auch über die indonesische Insel erhält. Wären Computer dagegen in der Lage, zu verstehen, welche Bedeutung eines Wortes gemeint ist, könnten sie das Suchergebnis auf diejenigen Seiten eingrenzen, die den Suchbegriff in der gewünschten Bedeutung enthalten. Wenn man es darüber hinaus auch noch schafft, Beziehungen zwischen Begriffen auf maschinenlesbare Art darzustellen, könnte eine Anfrage an eine Suchmaschine zusätzlich Seiten in ihr Ergebnis aufnehmen, die zwar mit dem Suchbegriff in Beziehung stehen, ihn selbst aber nicht enthalten. Eine Suche nach der Insel Java könnte somit zum Beispiel auch Links zu verschiedenen Städten enthalten, die auf dieser Insel liegen.

Die nachfolgenden Abschnitte über OWL enthalten eine Einführung in die Sprache, die es ermöglicht, die Prinzipien und Funktionsweise von OWL im Allgemeinen, sowie die im Rahmen dieser Arbeit verwendeten Konstrukte im Besonderen zu verstehen.

Im Abschnitt "3.1 Grundlagen" werden allgemeine Konzepte und Technologien beschrieben, auf die OWL sich gründet.

Unter "3.2 Beschreibung der wichtigsten Konstrukte" werden, mit wenigen Ausnahmen, die für diese Arbeit nicht relevant sind, die Konstrukte beschrieben, die beim Aufbau einer OWL-Ontologie zur Verfügung stehen.

Der Abschnitt "3.3 Äquivalenz von Klassen und Individuen in OWL Full" erläutert eine spezielle Fähigkeit von OWL Full, die dazu genutzt werden kann, um die "doppelte Instanziierung" der föderal-Informationsarchitektur abzubilden.

Im Abschnitt "3.4 Reasoning" schließlich, wird auf die Möglichkeit von OWL eingegangen, Informationen so zu modellieren, dass Computerprogramme aus einer OWL-Ontologie durch logisches Schließen Informationen herleiten können, die darin nicht explizit enthalten sind.

3.1 Grundlagen

Die Buchstaben in OWL stehen für "Web Ontology Language". Der darin enthaltene Begriff "Ontologie" kommt ursprünglich aus der Philosophie und bezeichnet die Wissenschaft von der Beschreibung der Arten von Dingen und der Beziehungen in denen diese zueinander stehen.

Eine OWL-Ontologie besteht aus Klassen, Eigenschaften (sogenannten "Properties") und Instanzen ("Individuals") mit deren Hilfe sich ein semantisches Netz aufbauen lässt. Klassen und Individuen bilden die Knoten des Netzes, Eigenschaften und deren Instanzen die Kanten. Die formale Semantik [6] von OWL macht es möglich, mit Hilfe von logischen Schlüssen aus einer Ontologie erweiterte Informationen abzuleiten, die darin nicht explizit enthalten sind.

Programme, die auf diese Weise Informationen herleiten, werden "Reasoner" genannt.

Da OWL für die Verwendung im Internet konzipiert ist, ermöglicht die Sprache die Verwendung verschiedener Dokumente als Informationsquellen, die auf verschiedene URLs verteilt sind. Dabei wird von der "offenen Welt"-Annahme ausgegangen. Das heißt, Klassen, die in einer bestimmten Ontologie definiert werden, können auch von anderen Ontologien erweitert werden. Auf diese Weise wird eines der Grundprinzipien von OWL umgesetzt, welches lautet: „Jeder kann Aussagen über alles machen.“

OWL verhindert nicht, dass widersprüchliche Aussagen gemacht werden. Je nach Konstruktion einer Ontologie können Widersprüche aber von einem Reasoner festgestellt werden.

3.1.1 Beziehungen zu anderen Sprachen

OWL ist eine Sprache, die dazu dient, das "SemanticWeb" [7] zu realisieren. Der Begriff "SemanticWeb" steht dabei für die Idee, Informationen im World Wide Web in maschinenlesbarer Form zur Verfügung zu stellen. OWL ist aus der Revision einer anderen Sprache namens DAML+Oil [8] hervorgegangen, die Ihrerseits eine Kombination aus den beiden Vorgängersprachen DAML [9] und Oil [10] ist, die mit dem Ziel entwickelt wurden, das SemanticWeb voran zu bringen.

OWL steht an der Spitze einer Hierarchie mehrerer aufeinander aufbauender Sprachen.

- XML [11] stellt auf der untersten Ebene eine Syntax zur Verfügung, um Informationen in einem Dokument zu strukturieren, macht aber keine Aussagen über die Semantik der Informationen. In XML werden Markierungselemente, sogenannte "Tags" verwendet, um Informationen über die Struktur eines Dokuments auszudrücken. Ein Tag besteht dabei aus einem Text, der in spitze Klammern eingefasst ist. So kann man etwa mit Hilfe eines Tags `<table>` den Beginn einer Tabelle signalisieren.
- XML-Schema [12] dient dazu, Vorgaben über die Struktur von XML-Dokumenten zu machen. Insbesondere lassen sich damit Aussagen über den Datentyp einer Information machen, die an einer bestimmten Stelle eines Dokuments stehen. Hierzu definiert XML-Schema verschiedene Datentypen wie etwa String, Integer oder auch kompliziertere Konstrukte zur Angabe eines Kalenderdatums.
- RDF [13] benutzt eine XML-Syntax, um eine einfache Semantik für „Objekte“ und Beziehungen zwischen diesen zur Verfügung zu stellen. Mit Hilfe von "RDF-Literalen" ermöglicht es eine lexikalische Darstellung von Datenwerten.
- RDF-Schema [14] ermöglicht, RDF-Objekte in Klassen mit bestimmten Eigenschaften einzuteilen und aus diesen Klassen und Eigenschaften Hierarchien von Verallgemeinerungen bzw. Spezialisierungen aufzubauen.
- OWL schließlich stellt zusätzliche Konstrukte zur Verfügung, um Klassen und Eigenschaften zu beschreiben. So gibt es zum Beispiel die Möglichkeit, die Äquivalenz zweier Klassen auszudrücken oder die Kardinalität einer Eigenschaft zu beschränken.

3.1.2 Die drei Versionen von OWL

Von OWL gibt es drei Versionen unterschiedlicher Mächtigkeit. Eine größere Mächtigkeit geht dabei mit einer größeren Komplexität der Berechnungen von Reasonern einher. Benutzer können so eine Sprachversion auswählen, die für Ihre Anforderungen in Sachen Ausdrucksstärke und Berechnungskomplexität am besten geeignet ist. Diese Versionen umfassen dabei verschiedene Teilmengen der Sprache. Ihre Namen lauten OWL Lite, OWL DL und OWL Full.

3.1.2.1 OWL Lite

Bei OWL Lite handelt es sich um eine "leichtgewichtige" Version der Sprache, die zwar gegenüber den beiden anderen Versionen über eine geringere Ausdruckstärke verfügt, dafür aber auch eine geringere Berechnungskomplexität aufweist. Sie richtet sich vor allem an Benutzer, die eine einfache Klassifizierungshierarchie benötigen und Eigenschaften als optional oder benötigt definieren wollen.

In OWL Lite sind nicht alle Konstrukte der Sprache enthalten. Bei den zur Verfügung stehenden Konstrukten müssen zudem teilweise Beschränkungen für deren Benutzung eingehalten werden.

OWL Lite ist eine Teilmenge von OWL DL. Jede gültige OWL Lite Ontologie ist auch eine gültige OWL DL Ontologie. Alle Beschränkungen, die für OWL DL gelten, gelten auch für OWL Lite.

3.1.2.2 OWL DL

Der Name OWL DL rührt von der Entsprechung zur "description logic" [15] her.

In OWL DL sind alle Konstrukte der Sprache enthalten. Bei der Benutzung dieser Konstrukte müssen aber teilweise Beschränkungen eingehalten werden. Es wird die maximale Ausdrucksmöglichkeit zur Verfügung gestellt, die unter der Vorgabe möglich ist, dass die Berechnungen von Reasonern vollständig und entscheidbar sein sollen.

Der Hauptunterschied zu OWL Lite ist, dass OWL DL mehr Möglichkeiten zur Beschränkung von Eigenschaften zur Verfügung stellt. Für den Benutzer sollte sich die Beantwortung der Frage, welche der beiden Versionen er verwendet, danach richten, ob ihm die Möglichkeiten von OWL Lite zur Beschränkung von Eigenschaften genügen.

OWL DL ist eine Teilmenge von OWL Full. Jede gültige OWL DL Ontologie ist eine gültige OWL Full Ontologie.

3.1.2.3 OWL Full

In OWL Full sind alle Konstrukte der Sprache ohne Einschränkung der Benutzung enthalten. Die Version bietet also entsprechend dem Namen den "vollen" Funktionsumfang. Klassen können in OWL Full sowohl als die Menge aller zugehörigen Individuen, als auch selbst als Individuum betrachtet werden. Während OWL DL und OWL Lite verschiedene Arten von Eigenschaften unterscheiden, die vom Typ des Subjekts und des Objekts abhängen, sind in OWL Full alle Arten von Eigenschaften äquivalent.

Für die Entscheidung des Benutzers, ob er OWL Full oder OWL DL verwenden soll, ist in erster Linie die Beantwortung der Frage von Bedeutung, ob er die Möglichkeit von OWL Full, Klassen auch als Individuen zu betrachten benötigt.

Der Nachteil von OWL Full ist, dass keine Garantien über die Berechnungszeit von Reasonern abgegeben werden können.

3.2 Beschreibung der wichtigsten Konstrukte

In diesem Abschnitt werden die in OWL zur Verfügung stehenden Konstrukte und deren Verwendung beschrieben. Dabei gelten die Aussagen grundsätzlich für alle drei Versionen der Sprache. Wenn Konstrukte in OWL DL oder OWL Lite nicht oder nur eingeschränkt verwendet werden dürfen, wird darauf explizit hingewiesen.

Im Abschnitt über Abkürzungen wird beschrieben, wie man mit deren Hilfe OWL-Dokumente besser lesbar machen kann. Im darauf folgenden Abschnitt wird erklärt, was Namespaces sind und wozu sie in OWL verwendet werden. Danach folgen Abschnitte über Klassen, Individuen und Eigenschaften.

3.2.1 Abkürzungen

An vielen Stellen in OWL ist es nötig, eine URL anzugeben. Um den Code lesbarer zu machen, ist es möglich für diese URLs in einem sogenannten "DOCTYPE"-Abschnitt, der den Definitionen der Ontologie vorangestellt wird, Abkürzungen zu definieren. Bei allen folgenden Beispielen wird davon ausgegangen, dass das OWL-Dokument folgenden "DOCTYPE"-Abschnitt enthält:

```
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
]>
```

Die Definition einer Abkürzung besteht aus drei Teilen. !ENTITY ist das Schlüsselwort, dass die Definition einer Abkürzung einleitet. Darauf folgt ein Ausdruck (hier: xsd bzw. owl) der einen anderen, längeren Ausdruck in Zukunft abkürzen soll. Der letzte Teil ist der in Anführungszeichen eingeschlossene längere Ausdruck, der durch den kürzeren abgekürzt werden soll.

Wenn man die in obigem "DOCTYPE"-Abschnitt definierten Abkürzungen im Dokument angeführt von dem Zeichen "&" und gefolgt von dem Zeichen ";" verwendet, werden sie bei der Interpretation zu der angegebenen Zeichenfolge expandiert. Die obigen Zeilen ermöglichen es also, im Dokument

```
&xsd; statt http://www.w3.org/2001/XMLSchema# zu schreiben und
&owl; statt http://www.w3.org/2002/07/owl#.
```

3.2.2 Namespaces

Wenn mehrere Dinge den selben Namen haben können, ist es, wenn man einen solchen Namen benutzt notwendig, das der Zusammenhang klar ist, damit entschieden werden kann, auf welches der gleichnamigen Dinge Bezug genommen wird. So sollte aus einem Text, in dem das Wort "Blatt" vorkommt, etwa hervorgehen, ob es um Bäume oder Schreibblöcke geht. Auch im Bereich der Programmiersprachen tritt dieses Problem auf. Hier wird es häufig dadurch gelöst, dass man einem Namen ein Präfix voranstellt, das einen sogenannten "Namespace" angibt. Ein Namespace stellt also einen Zusammenhang dar, in dem ein Name eindeutig definiert ist.

In OWL werden mit Hilfe von Namespace-Deklarationen Präfixe einem Namespace zugeordnet. Durch die Voranstellung des Präfixes wird ein Name einzigartig.

Bei allen folgenden Beispielen wird davon ausgegangen, dass das OWL-Dokument folgende namespace-Deklarationen enthält:

```

<rdf:RDF
  xmlns:owl = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema#"
>

```

Hierbei wird festgelegt, dass alles, was das Präfix `owl:` trägt sich auf den Namespace bezieht, der unter `http://www.w3.org/2002/07/owl#` definiert ist. Es handelt sich dabei um das OWL-Vokabular.

Entsprechendes gilt für das Präfix `rdf:` und das RDF-Vokabular an der angegebenen Stelle, für das Präfix `rdfs:` und das RDFS-Vokabular sowie das Präfix `xsd:` und die XML-Schema Datentypen.

3.2.3 Einfache Klassen

3.2.3.1 Klassennamen

Es gibt mehrere Wege, in OWL eine Klasse zu definieren. Die einfachste Methode ist die Verwendung des Tags `<owl:Class>` in Verbindung mit dem Attribut `rdf:ID`. Mit seiner Hilfe ist es möglich, benannte Klassen zu definieren:

```

<owl:Class rdf:ID="Mensch"/>
<owl:Class rdf:ID="Tier"/>

```

bzw.

```

<owl:Class rdf:ID="Mensch">
</owl:Class>

<owl:Class rdf:ID="Tier">
</owl:Class>

```

Im obigen Beispiel werden zwei Klassen definiert. Das Attribut `rdf:ID` dient dazu, den Namen der Klasse festzulegen. Innerhalb der Ontologie, in der sie definiert werden, können diese Klassen nun mit dem Zeichen `#` gefolgt von ihrem Namen, also per `#Mensch` oder `#Tier` referenziert werden. Andere Ontologien können die Klassen verwenden, in dem sie der Referenz die URL unter der die Definition zu finden ist voranstellt. Wären die Klassen

also in einer Datei "lebewesen.owl" auf einer Internetseite "http://www.beispiel.de" definiert, könnten andere Ontologien sie mittels

`http://www.beispiel.de/lebewesen#Mensch` bzw.

`http://www.beispiel.de/lebewesen#Tier` referenzieren.

3.2.3.2 Subklassen

OWL ermöglicht Mehrfachvererbung. Ein Klasse kann Subklasse von beliebig vielen anderen Klassen sein.

Um eine Klassenhierarchie aufzubauen, verwendet man das Tag `<rdfs:subClassOf>`.

Will man ausdrücken, dass `Mensch` eine Unterklasse von `Tier` ist und damit die entsprechenden Eigenschaften erbt und `Hund` wiederum eine Subklasse von `Tier`, schreibt man:

```
<owl:Class rdf:ID="Mensch">
  <rdfs:subClassOf rdf:resource="#Tier"/>
</owl:Class>
```

```
<owl:Class rdf:ID="Hund">
  <rdfs:subClassOf rdf:resource="#Tier"/>
</owl:Class>
```

Zwei spezielle Klassen sind `<owl:Thing>` und `<owl:Nothing>`, wobei erstere die allgemeinste Superklasse ist, der alle Instanzen angehören und letztere die leere Klasse, die Subklasse aller Klassen, der keine Individuen angehören.

In OWL Lite können Klassen nur Subklassen von benannten Klassen sein, also von Klassen, die mittels des OWL-Tags `<owl:Class>` in Verbindung mit dem Attribut `rdf:ID` definiert wurden.

3.2.4 Benannte Individuen

3.2.4.1 Definition

Instanzen von Klassen werden in OWL "Individuen" genannt. Ein Individuum kann durch ein Tag definiert werden, das den Namen der instanziierten Klasse trägt. Mit Hilfe des Attributs `rdf:ID` wird der Name des Individuums festgelegt:

```
<Mensch rdf:ID="Moritz" />
<Hund rdf:ID="Waldi" />
```

Alternativ kann man auch das RDF-Tag `<rdf:type>` verwenden, um die Zugehörigkeit zu einer Klasse auszudrücken:

```
<owl:Thing rdf:ID="Moritz" />

<owl:Thing rdf:about="#Moritz">
  <rdf:type rdf:resource="#Mensch" />
</owl:Thing>
```

Die Referenzierung von Individuen erfolgt analog zur Referenzierung von Klassen mit Hilfe des Zeichens # und ggf. vorangestellter URL.

3.2.4.2 Identität

In OWL kann ein Individuum unter mehreren Namen bekannt sein. Nur weil zwei Namen unterschiedlich sind, heißt das nicht zwangsläufig, dass sich dahinter auch unterschiedliche Individuen verbergen. Die Identität zwischen zwei Individuen kann unter Umständen von einem Reasoner hergeleitet werden. Man kann sie aber auch mit Hilfe von `<owl:sameAs>` explizit angeben:

```
<Mensch rdf:ID="Bundeskanzler" />

<Mensch rdf:ID="GerhardSchröder">
  <owl:sameAs rdf:resource="#Bundeskanzler" />
</Mensch>
```

Diese Möglichkeit ist vor allem nützlich, wenn man Informationen, die in zwei verschiedenen OWL-Dokumenten enthalten sind, zusammenführen will.

3.2.4.3 Unterschiedlichkeit zweier Individuen

Entsprechend der Identität zweier Individuen, lässt sich mit `<owl:differentFrom>` auch deren Unterschiedlichkeit ausdrücken:

```
<Mensch rdf:ID="LotharMatthäus">
  <owl:differentFrom
    rdf:resource="http://www.beispiel.de/OWLBibel#Matthäus"/>
</Mensch>
```

3.2.4.4 Unterschiedlichkeit mehrerer Individuen

Dass mehrere Individuen jeweils paarweise unterschiedlich sind, lässt sich mit Hilfe der Tags `<owl:AllDifferent>` und `<owl:distinctMembers>` ausdrücken:

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Grundfarbe rdf:about="#Rot">
    <Grundfarbe rdf:about="#Gelb">
    <Grundfarbe rdf:about="#Blau">
  </owl:distinctMembers>
</owl:AllDifferent>
```

Das Tag `<owl:distinctMembers>` klammert die Liste der paarweise unterschiedlichen Individuen, wobei `rdf:parseType="Collection"` angibt, dass als Liste das in RDF definierte Konstrukt "Collection" verwendet wird. Das Tag `<owl:distinctMembers>` kann nur zusammen mit `rdf:parseType="Collection"` und `<owl:AllDifferent>` eingesetzt werden.

3.2.5 Eigenschaften

Mit Hilfe von Eigenschaften ("Properties") lassen sich in OWL Beziehungen ausdrücken. Allgemein handelt es sich bei Eigenschaften um Prädikate, die ein Subjekt mit einem Objekt verbinden. So ließe sich zum Beispiel eine Prädikat "hatHerrchen" definieren, mit dem man eine Verbindung zwischen einem Hund als Subjekt und einem Menschen als Objekt herstellen kann.

Es wird zwischen mehreren Arten von Eigenschaften unterschieden, die nur in OWL Full äquivalent sind:

- Eine `ObjectProperty` stellt eine Beziehung zwischen zwei Individuen her. Die Eigenschaft "hatHerrchen" wäre zum Beispiel eine `ObjectProperty`:

```
<owl:ObjectProperty rdf:ID="hatHerrchen"/>
```

Dieses Beispiel definiert nur, dass es eine Eigenschaft `hatHerrchen` gibt, die zwei Individuen miteinander verbindet. Die Tatsache, dass üblicherweise Hunde ein Herrchen haben und dieses üblicherweise ein Mensch ist, fließt hier nicht ein.

- Eine `DatatypeProperty` stellt eine Beziehung zwischen einem Individuum und einem XML-Schema Datentyp oder einem RDF-Literal her.

```
<owl:DatatypeProperty rdf:ID="hatAlter"/>
```

Mit welchem Datentyp ein Individuum durch die obige Eigenschaft `hatAlter` verbunden werden kann (hier wäre wohl `Integer` sinnvoll) wird damit allerdings nicht definiert. Wie diese Festlegung vonstatten geht, wird im Abschnitt über `Domain` und `Range` erläutert.

- Eigenschaften vom Typ `AnnotationProperty` oder `OntologyProperty` stellen Metainformationen über Ontologien bzw. deren Teile zur Verfügung und tragen zur eigentlichen Semantik nichts bei:

```
<owl:AnnotationProperty rdf:ID="Autor"/>
```

Mit dieser `AnnotationProperty` könnte man zum Beispiel angeben, wer einen bestimmten Teil einer Ontologie erzeugt hat.

3.2.5.1 Instanziierung

Auf obige Weise definierte Eigenschaften werden instanziiert, indem man bei dem Individuum, das das Subjekt darstellt, ein Tag mit dem Namen der Eigenschaft einfügt, welches auf das Objekt verweist:

```
<Hund rdf:ID="Waldi">
  <hatHerrchen rdf:resource="#Moritz"/>
  <hatAlter rdf:datatype="xsd:integer">8</hatAlter>
  <Autor>Marc Eichler</Autor>
</Hund>
```

Man beachte, dass bei der DatatypeProperty der Datentyp angegeben werden muss. Hierbei wird die oben definierte Abkürzung `&xsd;` verwendet.

3.2.5.2 Domain und Range

In obigem Instanzierungsbeispiel verbindet `hatHerrchen` einen Hund mit einem Menschen. Es gibt aber keine formalen Beschränkungen beim Einsatz der Eigenschaft. Es wäre technisch genauso gut möglich damit einen Kühlschrank und ein Fußballfeld zu verbinden, was jedoch der Alltagserfahrung widersprechen würde, dass im Allgemeinen Kühlschränke keine Herrchen haben und Fußballfelder keine Herrchen von irgendetwas sein können. Um derartige Informationen in eine Ontologie einfließen zu lassen, gibt es in OWL die Möglichkeit, für Eigenschaften festzulegen, welchen Klassen die durch sie verbundenen Individuen angehören. Die hierzu verwendeten Tags lauten `<rdfs:domain>` und `<rdfs:range>`:

```
<owl:ObjectProperty rdf:ID="hatHerrchen">
  <rdfs:domain rdf:resource="#Hund"/>
  <rdfs:range rdf:resource="#Mensch"/>
</owl:ObjectProperty>
```

Das Tag `<rdfs:domain>` beschränkt die Klassenzugehörigkeit der Individuen, die Subjekt der Eigenschaft sein können. Das Tag `<rdfs:range>` beschränkt die Klassenzugehörigkeit der Individuen, die Objekt der Eigenschaft sein können. In obigem Beispiel werden alle Individuen, die Subjekt einer Eigenschaft `hatHerrchen` sind als Hunde definiert, während Individuen, die Objekt der Eigenschaft sind, als Menschen definiert werden.

Diese Beschränkungen ändern jedoch nichts daran, dass `hatHerrchen` nach wie vor beliebige Subjekte und Objekte miteinander verbinden kann. Der Unterschied zu der Version ohne Beschränkung von Domain und Range ist, dass ein Reasoner aus der Beschränkung schließen kann, welchen Klassen Subjekt und Objekt angehören. Beispielsweise kann hergeleitet werden, dass ein Individuum, das Subjekt der Eigenschaft `hatHerrchen` ist, der Klasse `Hund` angehört, auch wenn es nicht explizit als solches deklariert ist. Entsprechend können auch Informationen über die Klassenzugehörigkeit eines Objekts aus der Range abgeleitet werden.

Die Beschränkung von Domain und Range einer Eigenschaft in OWL ist also etwas völlig anderes, als die Typisierung eines Attributs in einer objektorientierten Programmiersprache. In der objektorientierten Programmierung ist die Klassenzugehörigkeit von Instanzen vollständig bekannt und die Typisierung von Attributen dient dazu, gewisse Zuweisungen technisch unmöglich zu machen.

Beschränkungen von Eigenschaften in OWL dienen dazu, Informationen über die Klassenzugehörigkeit von Individuen herzuleiten. In OWL ist die Klassenzugehörigkeit von Individuen nicht vollständig bekannt. Schon weil jedes OWL-Dokument, das sich irgendwo im Internet befindet, Ontologien beliebig erweitern kann, können Informationen über ein Individuum bzw. dessen Klassenzugehörigkeit über das ganze World Wide Web verteilt sein, so dass ständig implizit neue Informationen über Klassenzugehörigkeiten hinzukommen können.

In OWL Lite können nur Klassen, die einen Namen haben, als Domain oder Range dienen.

3.2.5.3 Hierarchien

Wie Klassen mittels des Tags `<rdfs:subClassOf>`, können auch Eigenschaften in einer Spezialisierungshierarchie organisiert werden. Das entsprechende Tag hierfür lautet `<rdfs:subPropertyOf>`.

```
<owl:ObjectProperty rdf:ID="istNachkommeVon">
  <rdfs:domain rdf:resource="#Mensch"/>
  <rdfs:range rdf:resource="#Mensch"/>
</owl:ObjectProperty>
```

```

<owl:ObjectProperty rdf:ID="istTochterVon">
  <rdfs:subPropertyOf rdf:resource="#istNachkommeVon"/>
  <rdfs:domain rdf:resource="#Frau"/>
  <rdfs:range rdf:resource="#Mensch"/>
</owl:ObjectProperty>

```

Mit `istNachkommeVon` kann man Verwandtschaftsverhältnisse zwischen Menschen abbilden. Die Eigenschaft `istTochterVon` ist eine Spezialisierung von `istNachkommeVon`. Hieraus lässt sich schließen, dass jedes Individuum X , das über die Eigenschaft `istTochterVon` mit einem Individuum Y verbunden ist, auch über die Eigenschaft `istNachkommeVon` mit Y verbunden ist, ohne dass diese Verbindung explizit angegeben werden muss.

3.2.5.4 Charakteristiken

Eine Möglichkeit, zusätzliche Aussagen über eine Eigenschaft zu formulieren ist die Verwendung von Charakteristiken. Mit ihrer Hilfe lassen sich die Möglichkeiten, Informationen automatisch herzuleiten verbessern.

Transitive Eigenschaften:

Mit Hilfe des Tags `<rdf:type rdf:resource="&owl;TransitiveProperty"/>` lässt sich eine Eigenschaft als transitiv definieren. Hierbei wird die im Abschnitt "3.2.1 Abkürzungen" definierte Abkürzung `&owl;` verwendet.

Bei einer transitiven Eigenschaft P gilt für alle Individuen x , y und z :
 $P(x,y)$ und $P(y,z)$ impliziert $P(x,z)$.

Die Eigenschaft `istNachkommeVon` könnte zum Beispiel wie folgt als transitiv definiert werden:

```

<owl:ObjectProperty rdf:ID="istNachkommeVon">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Mensch"/>
  <rdfs:range rdf:resource="#Mensch"/>
</owl:ObjectProperty>

```

```
<Mensch rdf:ID="Klaus">
  <istNachkommeVon rdf:resource="#Elke"/>
</Mensch>
```

```
<Mensch rdf:ID="Christine">
  <istNachkommeVon rdf:resource="#Klaus"/>
</Mensch>
```

Auf Grund des transitiven Charakters von `istNachkommeVon` kann ein Reasoner aus diesen Angaben schließen, dass, da Christine ein Nachkomme von Klaus ist und Klaus wiederum ein Nachkomme von Elke, auch Christine ein Nachkomme von Elke ist.

In OWL DL dürfen transitive Eigenschaften nicht gleichzeitig funktionale Eigenschaften (s.u.) sein. Außerdem dürfen in OWL DL transitive Eigenschaften nicht mittels

`<owl:minCardinality>`, `<owl:maxCardinality>` oder `<owl:cardinality>` beschränkt werden (siehe "3.2.6 Beschränkungen").

Symmetrische Eigenschaften:

Mit Hilfe des Tags `<rdf:type rdf:resource="#owl:SymmetricProperty"/>` lässt sich eine Eigenschaft als symmetrisch definieren.

Bei einer symmetrischen Eigenschaft P gilt für alle Individuen x und y :

$P(x,y)$ genau dann wenn $P(y,x)$.

Zum Beispiel lässt sich eine Eigenschaft `istVerheiratetMit` wie folgt als symmetrisch definieren:

```
<owl:ObjectProperty rdf:ID="istVerheiratetMit">
  <rdf:type rdf:resource="#owl:SymmetricProperty"/>
  <rdfs:domain rdf:resource="#Mensch"/>
  <rdfs:range rdf:resource="#Mensch"/>
</owl:ObjectProperty>
```

```
<Mensch rdf:ID="Fabian">
  <istVerheiratetMit rdf:resource="#Simone"/>
</Mensch>
```

Auf Grund des symmetrischen Charakters von `istVerheiratetMit` kann ein Reasoner schließen, dass nicht nur Fabian mit Simone verheiratet ist, sondern auch umgekehrt.

Funktionale Eigenschaften:

Mit Hilfe des Tags `<rdf:type rdf:resource="&owl;FunctionalProperty"/>` lässt sich eine Eigenschaft als funktional definieren.

Bei einer funktionalen Eigenschaft P gilt für alle Individuen x, y und z :

$P(x,y)$ und $P(x,z)$ impliziert $y = z$.

Die folgende Eigenschaft `hatMutter` kann als Beispiel dafür dienen:

```
<owl:ObjectProperty rdf:ID="hatMutter">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Mensch"/>
  <rdfs:range rdf:resource="#Mensch"/>
</owl:ObjectProperty>

<Mensch rdf:ID="Paul">
  <hatMutter rdf:resource="#Anja"/>
  <hatMutter rdf:resource="#Frau Müller"/>
</Mensch>
```

Auf Grund des funktionalen Charakters von `hatMutter` kann ein Reasoner aus diesen Angaben schließen, dass Anja und Frau Müller identisch sind.

In OWL DL dürfen funktionale Eigenschaften nicht gleichzeitig transitiv sein.

Inverse Eigenschaften:

Es ist möglich, eine Eigenschaft als die Inverse einer anderen zu definieren. Hierzu wird das Tag `<owl:inverseOf>` benutzt.

Bei einer Eigenschaft P , die Inverse einer Eigenschaft Q ist, gilt für alle Individuen x und y : $P(x,y)$ genau dann wenn $Q(y,x)$.

```
<owl:ObjectProperty rdf:ID="istNachkommeVon">
  <rdfs:domain rdf:resource="#Mensch"/>
  <rdfs:range rdf:resource="#Mensch"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="istVorfahreVon">
  <owl:inverseOf rdf:resource="#istNachkommeVon"/>
</owl:ObjectProperty>
```

```
<Mensch rdf:ID="Petra">
  <istNachkommeVon rdf:resource="#Daniel"/>
</Mensch>
```

Auf Grund der Tatsache, dass `istVorfahreVon` die Inverse von `istNachkommeVon` ist, kann ein Reasoner aus diesen Angaben schließen, dass Daniel ein Vorfahre von Petra ist.

Invers funktionale Eigenschaften:

`<rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>` ist ein Tag, mit dessen Hilfe sich eine Eigenschaft als invers funktional definieren lässt. Eine Eigenschaft deren Inverse funktional ist, ist selbst invers funktional.

Bei einer invers funktionalen Eigenschaft P gilt für alle Individuen x und z :

$P(y,x)$ und $P(z,x)$ impliziert $y = z$.

Als Beispiel lässt sich eine invers funktionale Eigenschaft `istVaterVon` wie folgt definieren:

```
<owl:ObjectProperty rdf:ID="istVaterVon">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
  <rdfs:domain rdf:resource="#Mensch"/>
  <rdfs:range rdf:resource="#Mensch"/>
</owl:ObjectProperty>
```

Diese Eigenschaft ordnet einem Vater ein Kind zu. Sie ist nicht als funktional definiert, da ein Vater mehrere Kinder haben kann. Die Inverse dieser Eigenschaft würde einem Kind seinen Vater zuweisen. Da jedes Kind nur einen Vater haben kann, was einen funktionalen Charakter bedeutet, macht es Sinn, die Eigenschaft `istVaterVon` als invers funktional zu definieren.

Auf Grund der Unterscheidung von `DatatypeProperties` und `ObjectProperties` in OWL DL können `TransitiveProperty`, `SymmetricProperty`, `InverseFunctionalProperty` und `inverseOf` hier nicht für `DatatypeProperties` verwendet werden.

3.2.6 Beschränkungen

Zusätzlich zu der in Abschnitt "3.2.3 Einfache Klassen" vorgestellten Möglichkeit, Klassen mit einem Namen zu definieren, bietet OWL die Möglichkeit unbenannte Klassen zum

Beispiel über sogenannte Beschränkungen zu definieren. Hierbei kommt das Tag `<owl:Restriction>` zum Einsatz. Beschränkungen definieren Klassen darüber, dass der Wert oder die Kardinalität einer Eigenschaft in einem bestimmten Bereich liegen. So lässt sich zum Beispiel die Klasse `Vegetarier` darüber definieren, dass es sich bei den Werten der Eigenschaft `ernährtSichVon` nur um Pflanzen handeln darf.

Beschränkungen bestehen aus zwei Teilen. Mit Hilfe des Tags `<owl:onProperty>` wird die Eigenschaft angegeben, die zur Definition einer Klasse herangezogen wird. Mit Hilfe eines zweiten Tags wird definiert, welche Voraussetzungen bei der Verwendung der Eigenschaft erfüllt sein müssen, damit eine Zugehörigkeit zur entsprechenden Klasse gegeben ist.

3.2.6.1 allValuesFrom

Mit Hilfe des Tags `<owl:allValuesFrom>` lässt sich eine Klasse darüber definieren, dass alle Objekte einer Eigenschaft einer bestimmten Klasse angehören:

```
<owl:Class rdf:="#Vegetarier">
  <rdfs:subClassOf rdf:resource="#Mensch"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#ernährtSichVon"/>
      <owl:allValuesFrom rdf:resource="#Pflanze"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Die Klasse `Vegetarier` ist eine Subklasse der anonymen Klasse, die von der Beschränkung definiert wird. Anhand der Tatsache, dass `<owl:Restriction>` innerhalb von `<owl:subClassOf>` steht, wird sehr gut deutlich, dass `<owl:Restriction>` eine Klasse definiert und nicht etwa irgendwelche Änderungen an der Definition der Eigenschaft `#ernährtSichVon` vornimmt. Die Eigenschaft wird lediglich dazu benutzt, um eine Klasse zu definieren. An der Eigenschaft selbst ändert sich dadurch nichts. Individuen, die keine `Vegetarier` sind, können weiterhin auch Nicht-Pflanzen als Wert dieser Eigenschaft haben.

In OWL Lite dürfen mittels `<owl:allValuesFrom>` nur benannte Klassen oder Datentypen referenziert werden.

3.2.6.2 someValuesFrom

Die Beschränkung `<owl:someValuesFrom>` ähnelt `<owl:allValuesFrom>`. Sie definiert eine anonyme Klasse darüber, dass Individuen mindestens eine Instanz der angegebenen Eigenschaft haben müssen, deren Objekt einer bestimmten Klasse angehört:

```
<owl:Class rdf:="Fleischesser">
  <rdfs:subClassOf rdf:resource="#Mensch"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#ernährtSichVon"/>
      <owl:someValuesFrom rdf:resource="#Tier"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

In OWL Lite dürfen mittels `<owl:someValuesFrom>` nur benannte Klassen oder Datentypen referenziert werden.

3.2.6.3 hasValue

Mit Hilfe der Beschränkung `<owl:hasValue>` lässt sich eine Klasse darüber definieren, dass eine Eigenschaft einen bestimmten Wert hat, also der Wert nicht nur einer bestimmten Klasse angehört. Im Gegensatz zu `<owl:allValuesFrom>` und `<owl:someValuesFrom>` handelt es sich bei der referenzierten Ressource also um ein Individuum und nicht um eine Klasse:

```
<owl:Class rdf:ID="Stuttgarter">
  <rdfs:subClassOf rdf:resource="#Mensch"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#lebtIn"/>
      <owl:hasValue rdf:resource="#Stuttgart"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

In OWL Lite steht `<owl:hasValue>` nicht zur Verfügung.

3.2.6.4 minCardinality

Die Beschränkung `<owl:minCardinality>` definiert eine Klasse über die Anzahl der mit einem Individuum in Verbindung stehenden anderen Individuen. Angehörige der Klasse müssen mindestens die angegebene Anzahl Werte für die Eigenschaft haben:

```
<owl:Class rdf:ID="MultinationalerKonzern">
  <rdfs:subClassOf rdf:resource="#Konzern"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatNiederlassungInLand"/>
      <owl:minCardinality rdf:datatype=
        "&xsd;nonNegativeInteger">2</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Hier wird festgelegt, dass multinationale Konzerne Niederlassungen in mindestens zwei Ländern haben. Die Zahl zwei, die die Mindestkardinalität angibt, hat dabei den in XSD-Schema definierten Typ "nonNegativeInteger". Bei der Angabe dieses Datentyps wird die weiter oben definierte Abkürzung `&xsd;` verwendet.

Ein Individuum der Klasse sähe zum Beispiel so aus:

```
<Konzern rdf:ID="Müller-Export"/>
  <hatNiederlassungInLand rdf:resource="#Deutschland"/>
  <hatNiederlassungInLand rdf:resource="#Frankreich"/>
</Konzern>
```

In OWL Lite dürfen für `<owl:minCardinality>` nur die Werte 0 und 1 verwendet werden.

3.2.6.5 maxCardinality

Die Beschränkung `<owl:maxCardinality>` definiert eine Klasse wie `<owl:minCardinality>` über die Anzahl der mit einem Individuum über eine bestimmte Eigenschaft in Verbindung stehenden anderen Individuen. In diesem Fall wird jedoch die maximale Kardinalität festgelegt.

In OWL Lite dürfen für `<owl:maxCardinality>` nur die Werte 0 und 1 verwendet werden.

3.2.6.6 cardinality

Mit der Beschränkung `<owl:cardinality>` lässt sich die Kardinalität auf einen bestimmten Wert festlegen. Alternativ zur Verwendung von `<owl:cardinality>` kann man auch `<owl:minCardinality>` und `<owl:maxCardinality>` auf den selben Wert setzen.

In OWL Lite dürfen für `<owl:cardinality>` nur die Werte 0 und 1 verwendet werden.

3.2.7 Komplexe Klassen

Zusätzlich zu den oben erläuterten Möglichkeiten, Klassen mit einem Namen oder durch eine Beschränkung zu definieren, ermöglicht es OWL, Klassen durch Aussagen über die Menge der Individuen, die der Klasse angehören zu definieren. Neben einer expliziten Aufzählung aller Individuen, die der Klasse angehören, stehen hierfür die klassischen Mengenoperationen Schnitt, Vereinigung und Komplement und einige andere Konstrukte zur Verfügung.

3.2.7.1 Aufzählung aller Individuen

Bei der Definition einer Klasse mittels des Tags `<owl:oneOf>` werden alle Individuen, die der Klasse angehören, aufgezählt. Andere Individuen als die aufgezählten gehören der Klasse nicht an:

```

<owl:Class rdf:ID="Jahreszeit">
  <owl:oneOf rdf:parseType="Collection">
    <Jahreszeit rdf:about="#Frühling"/>
    <Jahreszeit rdf:about="#Sommer"/>
    <Jahreszeit rdf:about="#Herbst"/>
    <Jahreszeit rdf:about="#Winter"/>
  </owl:oneOf>
</owl:Class>

```

Hierbei gibt `rdf:parseType="Collection"` an, dass zum Auflisten der Individuen das in RDF definierte Konstrukt "Collection" verwendet wird.

In OWL Lite steht `<owl:oneOf>` nicht zur Verfügung.

3.2.7.2 Schnitt von Klassen

Das Tag `<owl:intersectionOf>` dient dazu, die Menge der Individuen, die einer Klasse angehören, als Schnittmenge von verschiedenen Individuen, die anderen Klassen angehörenden zu definieren:

```

<owl:Class rdf:ID="Salzwasserfisch">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Fisch"/>
    <owl:Class rdf:about="#Meeresbewohner"/>
  </owl:intersectionOf>
</owl:Class>

```

Es können dabei nicht nur benannte Klassen für die Bildung der Schnittmenge verwendet werden. Vielmehr kann jede Art von Klassendefinition an der entsprechenden Stelle eingesetzt werden, zum Beispiel eine Beschränkung:

```

<owl:Class rdf:ID="Salzwasserfisch">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Fisch"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#lebtIn"/>
      <owl:hasValue rdf:resource="#Meer"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

```

In OWL Lite kann das Tag `<owl:intersectionOf>` nur zusammen mit Listen verwendet werden, die eine Länge größer eins haben und nur benannten Klassen und Beschränkungen enthalten.

3.2.7.3 Vereinigung von Klassen

Mit Hilfe des Tags `<owl:unionOf>` kann man die Menge der Individuen, die zu einer Klasse gehören als Vereinigung von verschiedenen Individuen, die zu anderen Klassen gehören definieren. Auch in diesem Fall sind alle Varianten der Klassendefinition erlaubt:

```
<owl:Class rdf:ID="Fisch">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Süßwasserfisch" />
    <owl:Class rdf:about="#Salzwasserfisch" />
  </owl:unionOf>
</owl:Class>
```

In OWL Lite steht das Tag `<owl:unionOf>` nicht zur Verfügung.

3.2.7.4 Komplement von Klassen

Das Tag `<owl:complementOf>` dient dazu, die Menge der Individuen, die einer Klasse angehören als Komplement einer Menge von Individuen, die einer anderen Klasse angehören zu definieren. Wiederum sind alle Varianten der Klassendefinition erlaubt:

```
<owl:Class rdf:ID="Nichtschwimmer">
  <owl:complementOf>
    <owl:Class rdf:about="#Schwimmer" />
  </owl:complementOf>
</owl:Class>
```

Das folgende Beispiel zeigt eine Variante, bei der die Klasse, deren Komplement gebildet wird, selbst über eine Menge definiert ist:

```

<owl:Class rdf:ID="Wildtier">
  <owl:complementOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Haustier"/>
        <owl:Class rdf:about="#Nutztier"/>
      </owl:unionOf>
    </owl:Class>
  </owl:complementOf>
</owl:Class>

```

In OWL Lite steht das Tag `<owl:complementOf>` nicht zur Verfügung.

3.2.7.5 Äquivalente Klassen

Mit Hilfe des Tags `<owl:equivalentClass>` lassen sich Klassen definieren, denen genau dieselben Individuen angehören, wie einer bestimmten anderen Klasse. Die beiden Klassen sind dann äquivalent, was jedoch nicht bedeutet, dass sie völlig identisch sind. Die Klassen selbst können trotzdem weiter auf Grund ihrer Namen unterschieden werden. Das ist sinnvoll, wenn sie eine unterschiedliche Sichtweise auf die zugehörigen Individuen zum Ausdruck bringen, also diese in eine unterschiedliche Kategorisierungen einordnen:

```

<owl:Class rdf:ID="ÖffentlichRechtlicherSender">
  <owl:equivalentClass rdf:resource="#Gebührenempfänger"/>
</owl:Class>

```

In OWL Lite darf `<owl:equivalentClass>` nur in Zusammenhang mit benannten Klassen und mit Beschränkungen verwendet werden.

3.2.7.6 Klassen ohne gemeinsame Individuen

Mit Hilfe des Tags `<owl:disjointWith>` lässt sich ausdrücken, dass es in der Menge der zu einer Klasse gehörenden Individuen keine Individuen geben darf, die auch zu einer bestimmten anderen Klasse gehören. Ein Individuum kann also maximal einer der beiden Klassen angehören:

```

<owl:Class rdf:ID="Diskette">
  <rdfs:subClassOf rdf:resource="#Speichermedium"/>
  <owl:disjointWith rdf:resource="#Festplatte"/>
  <owl:disjointWith rdf:resource="#CD-Rom"/>
</owl:Class>

```

Bei diesem Beispiel ist zu beachten, dass nur etwas über das Verhältnis zwischen `Diskette` und `Festplatte` und das Verhältnis zwischen `Diskette` und `CD-Rom` ausgesagt wird.

Das Verhältnis zwischen `Festplatte` und `CD-Rom` ist hiervon nicht betroffen.

In OWL Lite steht `<owl:disjointWith>` nicht zur Verfügung.

3.2.8 Anonyme Individuen

Im Abschnitt über benannte Individuen wurden Individuen vorgestellt, denen mittels des Attributs `rdf:ID` ein Name zugewiesen wird, über den sie dann mittels des Zeichens `#` referenziert werden können. Es ist jedoch nicht zwingend nötig einem Individuum einen Namen zuzuweisen. Statt dessen können auch anonyme Individuen definiert werden. Das folgende Beispiel enthält ein anonymes Individuum einer hier nicht näher definierten Klasse `Name`:

```

<Mensch rdf:ID="Kunde359">
  <hatZuletztGekauft rdf:resource="#Fahrrad"/>
  <hatName>
    <Name>
      <bestehtAusVorname rdf:datatype="&xsd:string">
        Hans
      </bestehtAusVorname>
      <bestehtAusNachname rdf:datatype="&xsd:string">
        Radler
      </bestehtAusNachname>
    </Name>
  </hatName>
</Mensch>

```

Das anonyme Individuum der Klasse `Name`, welches angibt, dass der Vorname des Kunden `Kunde359` "Hans" ist und der Nachname "Radler", kann nur einmal, an der Stelle, an der es definiert ist, verwendet werden, da es keine Möglichkeit gibt, es von außen zu referenzieren.

3.3 Äquivalenz von Klassen und Individuen in

OWL Full

In im Gegensatz zu OWL DL und OWL Lite gibt es in OWL Full keine strikte Trennung zwischen Klassen, Individuen, Eigenschaften und Datenwerten. Sie alle können auch als Individuen betrachtet werden. Eine Klasse `Brot` könnte zum Beispiel gleichzeitig die Klasse aller tatsächlich gebackenen Brote darstellen und als Individuum in einer Aufzählung der Grundnahrungsmittel dienen. Folgender Code wäre also in OWL Full zulässig:

```
<owl:Class rdf:ID="Brot" />

<Brot rdf:ID="HaraldsBrot"/>

<owl:Class rdf:ID="Grundnahrungsmittel">
  <owl:oneOf rdf:parseType="Collection">
    <Grundnahrungsmittel rdf:about="#Brot" />
    <Grundnahrungsmittel rdf:about="#Wasser" />
  </owl:oneOf>
</owl:Class>
```

Man beachte, dass im Rahmen von `<owl:oneOf>` Individuen aufgezählt werden müssen, die der Klasse angehören. Die Klasse `#Brot` selbst wird also als Individuum betrachtet, das der Klasse `#Grundnahrungsmittel` angehört. Es werden nicht etwa die einzelnen Individuen der Klasse `#Brot` in die Klasse `#Grundnahrungsmittel` aufgenommen, sondern die Klasse selbst.

Die Aufhebung der strikten Trennung hat auch zur Folge, dass ein Unterscheiden zwischen den verschiedenen Arten von Eigenschaften keinen Sinn mehr macht. Da zum Beispiel ein Datenwert auch als Individuum betrachtet werden kann, kann er auch als Wert einer `ObjectProperty` dienen. In OWL Full sind darum alle Arten von Eigenschaften äquivalent. Statt der unterschiedlichen Arten von Eigenschaften kann darum auch `rdf:Property` verwendet werden, wovon alle anderen Arten von Eigenschaften abgeleitet sind.

Eine weitere Konsequenz der Aufhebung der strikten Trennung zwischen Klassen und Individuen ist, dass auch Klassen Eigenschaften haben können:

```
<owl:ObjectProperty rdf:ID="erfundenIn"/>

<owl:Class rdf:ID="Pizza">
  <erfundenIn rdf:resource="#Italien">
</owl:Class>
```

In OWL DL und OWL Lite könnten nur Individuen von `Pizza` eine Eigenschaft, zum Beispiel `erfundenIn`, haben. In OWL Full kann auch die Klasse selbst Eigenschaften haben.

3.4 Reasoning

In OWL können Informationen so dargestellt werden, dass spezielle Computerprogramme, sogenannte "Reasoner", daraus erweiterte Informationen ableiten können, die nicht explizit angegeben sind.

Ein einfaches Beispiel hierfür ist die Subklassen-Beziehung. Wenn `Roboter` eine Subklasse von `Maschine` ist und `R2D2` ein Individuum von `Roboter`, so kann ein Reasoner daraus schließen, dass `R2D2` gleichzeitig auch ein Individuum von `Maschine` ist.

Eine andere Möglichkeit auf die Klassenzugehörigkeit von Individuen zu schließen, ist die Verwendung der Informationen über Domain und Range von Eigenschaften. Wenn die Instanz einer Eigenschaft `wächstAuf` mit Domain `Frucht` ein Individuum `GoldenDelicious` mit einem anderen Individuum, zum Beispiel `Apfelbaum` verbindet, so kann daraus automatisch geschlossen werden, dass `GoldenDelicious` ein Individuum von `Frucht` ist.

Auch aus Beschränkungen der Kardinalität kann auf die Klassenzugehörigkeit geschlossen werden. Ein Individuum, das mittels einer Instanz der Eigenschaft `hatMitspieler` mit 12 anderen verbunden ist, kann nicht der Klasse `Fußballmannschaft` angehören, wenn diese über die genaue Kardinalität 11 für die Eigenschaft `hatMitspieler` definiert ist. Mit Hilfe von Charakteristiken von Eigenschaften können nicht nur Klassenzugehörigkeiten hergeleitet werden, sondern auch Eigenschaften, die nicht explizit angegeben sind. Wenn etwa ein Individuum `Otto` eine symmetrische Eigenschaft `istFreundVon` hat, die es mit

einem Individuum `Doris` verbindet, so kann daraus geschlossen werden, dass auch `Doris` eine Eigenschaft `istFreundVon` hat, die sie mit `Otto` verbindet.

Auf die Identität zweier Individuen kann unter anderem geschlossen werden, wenn sie beide das Objekt von Instanzen der selben funktionalen Eigenschaft mit dem selben Subjekt sind.

Es gibt viele andere Beispiele dafür, wie in OWL automatisch Informationen hergeleitet werden können. Bei der Frage, ob OWL für ein bestimmtes Anwendungsgebiet die richtige Technologie ist, sollte die Frage, ob diese Möglichkeiten benötigt werden eine wichtige Rolle spielen. Im Gegensatz zu herkömmlichen Programmiersprachen, wo ein Fehler bei der Typisierung meistens dazu führt, dass eine Fehlermeldung generiert wird, zieht ein OWL-Reasoner daraus eventuell nur Schlüsse, die inhaltlich nicht zutreffend sind.

4 Extraktion von OWL-Ontologien aus semantischen Netzen der föderal- Informationsarchitektur

Nachdem die vorangegangenen Kapitel sich mit der Einführung in die beteiligten Technologien beschäftigt haben, geht es in diesem Kapitel nun um die eigentliche Extraktion von OWL-Ontologien aus semantischen Netzen der föderal-Informationsarchitektur. Begonnen wird mit einem allgemeinen Vergleich der Beschreibungsmittel von OWL und der föderal-Informationsarchitektur.

Im Abschnitt "4.2 Umsetzung der Extraktion" werden die verschiedenen Fragestellungen und Lösungsansätze diskutiert, die bei der Erstellung einer Software zur Durchführung der Extraktion auftreten. Zunächst wird das verwendete Beispielnetz vorgestellt. Nach der Identifikation relevanter Information werden allgemeine Leitlinien für den Entwurf der OWL-Ontologien erläutert. Es folgt die Darstellung grundlegender Ideen zur Umsetzung bestimmter Konstrukte in OWL und der damit verbundenen Probleme, ehe dann im Detail auf die in den einzelnen Sprachversionen gewählten Lösungen eingegangen wird.

Für die Beschreibung der konkreten Lösungen in den verschiedenen Versionen bildet der Abschnitt über die OWL Lite-Ontologie die Basis. Da sie einfach strukturiert und leicht zu verstehen ist, bietet sie einen guten Einstieg in die Materie. Sie wird in der Folge in OWL DL und weiter in OWL Full modifiziert. Die Abschnitte über die Extraktion der OWL DL- und der OWL Full-Ontologie beschreiben entsprechend jeweils die Modifikationen gegenüber der Vorgängerversion. Auf diese Weise werden die Unterschiede zwischen den verschiedenen Versionen dargestellt.

4.1 Allgemeiner Vergleich der Beschreibungsmittel von OWL und föderal-Informationsarchitektur

Bei einem Vergleich der Beschreibungsmittel der föderal-Informationsarchitektur mit jenen von OWL muss man zunächst festhalten, dass beide für völlig unterschiedliche Verwendungszwecke konzipiert wurden.

OWL ist eine universell einsetzbare Sprache, um allgemein Beziehungen zwischen Dingen auszudrücken. Widersprüchliche Informationen werden nicht ausgeschlossen und es ist klar, dass man potenziell nie alle Informationen über etwas zur Verfügung hat, da man nicht alle Orte im World Wide Web kennen kann, an denen möglicherweise eine Aussage darüber gemacht wird. Ein Hauptzweck von OWL ist es, mit Hilfe von Reasonern Informationen herleiten zu können, die nicht explizit angegeben sind und an die vielleicht auch bei der Konzeption einer Ontologie noch gar nicht gedacht wurde. Jeder soll bestehende Ontologien durch eigene Dokumente erweitern und ändern können.

Die föderal-Informationsarchitektur dagegen erfüllt einen sehr speziellen Zweck. Sie stellt Informationen über Maschinen dar. Sie bewegt sich damit in einem klar umgrenzten, relativ übersichtlichen Einsatzbereich. Es ist klar definiert, welche Informationen relevant sind und es ist sichergestellt, dass all diese Informationen im semantischen Netz enthalten sind. Es ist nicht notwendig aus dem semantischen Netz Informationen abzuleiten, die darin nicht explizit angegeben sind. Es ist gerade nicht erwünscht, dass jede beliebige Person beliebige Änderungen vornehmen kann, sondern es soll nur eine bestimmte Gruppe von Personen Zugriff haben, die nur bestimmte Änderungen durchführen darf. Die Hauptaufgabe der föderal-Informationsarchitektur ist es, die Konsistenz des semantischen Netzes sicher zu stellen, also etwas, wozu OWL explizit nicht in der Lage ist. Trotzdem gibt es Übereinstimmungen und Parallelen zwischen den Beschreibungsmitteln:

Sowohl in der föderal-Informationsarchitektur als auch in OWL werden Informationen auf eine Weise modelliert, die es ermöglicht, sie als semantisches Netz, bestehend aus Knoten und Kanten darzustellen. Die Meta- Class- und Instance-Entities der föderal-Informationsarchitektur lassen sich in OWL auf OWL-Klassen und –Individuen abbilden, die dann in der OWL-Ontologie die selbe Funktion übernehmen, wie in der föderal-Informationsarchitektur die Entities. Sie bilden die Knoten des Netzes. Dabei besteht

zwischen Meta-, Class- und Instance-Entities bei der "doppelten Instanziierung" eine ähnliche Beziehung wie zwischen OWL-Klassen und OWL-Individuen.

In OWL stellen die Eigenschaften die Menge der Kanten dar, in der föderal-Informationenarchitektur die Assoziationen. Beide haben die Aufgabe, Knoten des jeweiligen Netzes miteinander zu verbinden.

Neben den Beschreibungsmitteln für Knoten und Kanten, die in der jeweils anderen Technologie eine direkte Entsprechung haben, gibt es auch solche, die sich zwar ähneln, jedoch nicht exakt die selbe Bedeutung haben. Zum Beispiel lassen sich in der föderal-Informationenarchitektur mit Hilfe von "RateConstraints" Aussagen darüber machen, wie viele Knoten über Assoziationen eines bestimmten Typs verbunden werden können. Ähnliches ist in OWL mit Hilfe von Beschränkungen und `<owl:minCardinality>` bzw. `<owl:maxCardinality>` möglich. Es gibt jedoch unterschiedliche Bedeutungen. Die föderal-Informationenarchitektur stellt sicher, dass die RateConstraints eingehalten werden, damit das Netz konsistent bleibt. OWL stellt dies nicht sicher. Hier werden die Informationen über die Kardinalität dazu benutzt, um Informationen über die Klassenzugehörigkeit der Knoten herzuleiten. Bei Knoten, die die Beschränkung der Kardinalität nicht einhalten, wird lediglich geschlossen, dass sie nicht der entsprechenden Klasse angehören.

Zu einer Reihe von Konstrukten der föderal-Informationenarchitektur gibt es in OWL überhaupt keine Entsprechung. Es handelt sich hierbei zum Beispiel um die Festlegung der Tiefe, die ein Graph, der aus bestimmten Assoziationen aufgebaut ist, haben darf und ob dieser zyklisch sein darf, sowie Festlegungen darüber, wie Änderungen durch Kopieren oder Löschen im semantischen Netz propagiert werden sollen. Solche Informationen können nur in menschenlesbarer Form in eine OWL-Ontologie integriert werden.

4.2 Umsetzung der Extraktion

4.2.1 Beispielnetze

Zur Demonstration der Extraktion stand als Beispiel ein semantisches Netz der föderal-Informationenarchitektur zur Verfügung, das den Aufbau einer Fräsmaschine beschreibt. Dieses Netz bestand aus 100 Meta-Entities, 351 Class-Entities, 2126 Instance-Entities, 65

Assoziationsstypen und 2924 Assoziationen. Die daraus extrahierten Ontologien wiesen eine erhebliche Größe auf. So führte zum Beispiel die Extraktion der OWL Full-Ontologie zu einer ca. 2 MB großen Datei. Da sich derart große Ontologien mit den derzeit verfügbaren Werkzeugen nicht automatisch validieren lassen, wurde ein zweites, kleineres Netz konstruiert welches in Abb 7. dargestellt ist und im Folgenden "Automobilbeispiel" genannt wird.

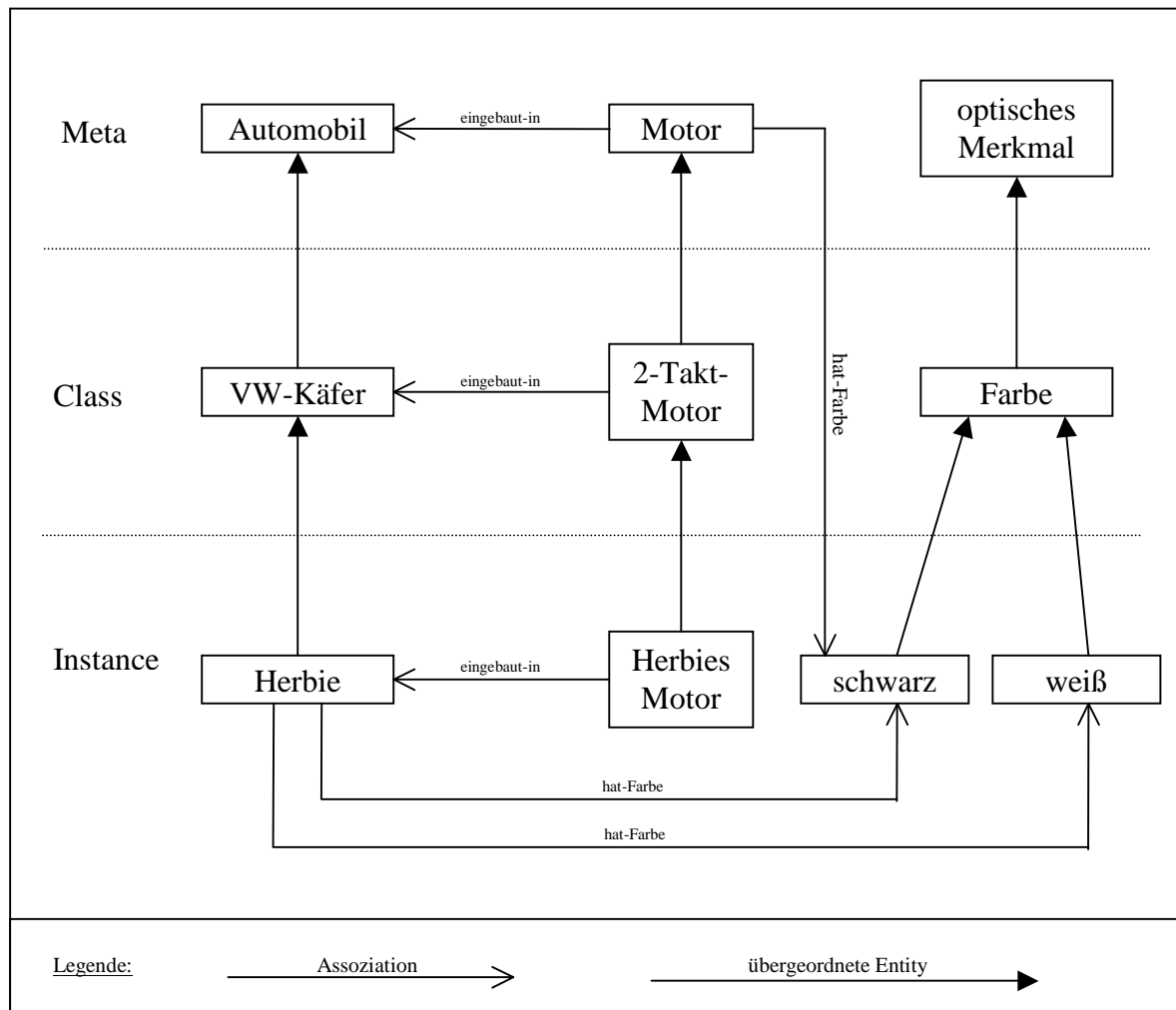


Abb. 7: "Automobilbeispiel"

Das Beispiel macht Aussagen über Beziehungen von Teilen von Automobilen und deren Farben. Auf der Meta-Ebene wird zwischen Automobilen Motoren und optischen Merkmalen entschieden, was eine Grundsätzliche Methode der Kategorisierung darstellt. Alternativ hätte man zum Beispiel bei den Automobilen auch zwischen Personenkraftwagen und Lastkraftwagen unterscheiden und so andere Vorgaben für die Class-Ebene machen können. Auf der Class-Ebene folgen Teile des Baukastens, die durch

das Setzen bestimmter Parameterwerte instanziiert werden können. Beim Motor wären etwa verschiedene PS-Zahlen möglich oder bei der Farbe unterschiedliche Mischungsverhältnisse von verschiedenen Grundfarben. Auf der Instance-Ebene findet man einige fertig parametrisierte Komponenten.

Das Beispiel soll vor allem möglichst viele technische Fälle abdecken, die in einem semantischen Netz der föderal-Informationsarchitektur vorkommen können. Dafür wurden einige Schwächen bei den Inhalten, die es zum Ausdruck bringt in Kauf genommen.

Insbesondere könnte man zum Beispiel die Frage stellen, ob Motoren tatsächlich immer schwarz sind. Es ist aber sehr schwierig, ein treffendes Beispiel für den Fall zu finden, in dem eine Assoziation eine Meta-Entity direkt mit einer Instance-Entity verbindet. Auch in den als Beispiel zur Verfügung stehenden größeren semantischen Netzen war kein Beispiel für eine solche Assoziation enthalten. Da die föderal-Informationsarchitektur die Möglichkeit zur Konstruktion einer derartigen Assoziation aber anbietet, wurde diese Assoziation zu Testzwecken in das Beispiel aufgenommen.

4.2.2 Identifizierung relevanter Informationen

Die Schnittstellen und Java-Klassen der „mind“-Schicht des föderal Engineering-Framework umfassen neben den Methoden zum Zugriff auf die Benutzerdaten auch solche, die allein der technischen Handhabung der Daten dienen, etwa der Kommunikation mit der Benutzerschnittstelle. Hinzu kommen Fälle, in denen mehrere Methoden redundante Zugriffsmöglichkeiten auf die selben Daten bieten. Um aus dem semantischen Netz eine möglichst redundanzfreie OWL-Ontologie zu extrahieren, die zudem keine Daten enthält, die im föderal Engineering-Framework nur der technischen Handhabung des Netzes dienen, musste für jede Methode herausgefunden werden, zu welcher der drei folgenden Kategorien sie gehört:

- 1.) Methoden, die Benutzerdaten zurückliefern und deren Rückgabewerte darum in die OWL-Ontologie eingebaut werden müssen.
- 2.) Methoden, die eine redundante Zugriffsmöglichkeit auf Benutzerdaten darstellen, die bereits als Rückgabewert einer anderen Methode in die OWL-Ontologie eingebaut werden. Der Rückgabewert solcher Methoden wird in der Regel nicht nochmals in die Ontologie eingebaut.
- 3.) Methoden, die keine Benutzerdaten zurückliefern, sondern der technischen Handhabung des Netzes dienen. Der Rückgabewert solcher Methoden wird ebenfalls nicht in die Ontologie eingebaut.

Um aus einem semantischen Netz der föderal Informationsarchitektur eine möglichst redundanzfreie OWL-Ontologie extrahieren zu können, die nur Benutzerdaten enthält, muss man wissen, welche Methoden zur 1. Gruppe gehören, denn nur deren Rückgabewerte sollten in die Ontologie einfließen. Leider lässt sich aus der Dokumentation des föderal Engineering-Framework nicht schließen, zu welcher der drei Gruppen die verschiedenen Methoden gehören. Die Zuordnung erfolgte darum auf Grund einer Analyse des Quellcodes sowie des zur Verfügung gestellten Beispielnetzes.

Es folgt eine Aufstellung der der Gruppe 1 zugeordneten Methoden, gruppiert nach Java-Klassen bzw. Schnittstellen, in denen sie definiert sind.

org.foederal.mind.IObject:

```
OID getOID()
    gibt eine global eindeutige ID des Objekts zurück
String getUIString()
    gibt eine Repräsentation des Objekts als Text zurück
boolean isMutable()
    gibt zurück, ob das Objekt geändert werden darf
```

org.foederal.mind.IEntity:

```
String getContent()
    gibt den Inhalt der Entity, also die eigentliche, zu verwaltende Information zurück
```

Layer getLayer()

gibt eine ID der Schicht zurück, zu der die Entity gehört

IUnit getUnit()

gibt die Unit zurück, zu der die Entity gehört

org.foederal.mind.IMetaEntity:

boolean isAbstract()

gibt den Wert des Attributs isAbstract zurück, das von auf der mind-Schicht aufbauenden Anwendungen verwendet wird und hier keine weitere Bedeutung hat

boolean isAttribute()

gibt den Wert des Attributs isAttribute zurück, das von auf der mind-Schicht aufbauenden Anwendungen verwendet wird und hier keine weitere Bedeutung hat

String getAttributeType()

gibt den Wert des Attributs attributeType zurück, das von auf der mind-Schicht aufbauenden Anwendungen verwendet wird und hier keine weitere Bedeutung hat

List selectClassEntities()

gibt eine Liste mit allen Class-Entities zurück, die von einer MetaEntity abgeleitet wurden

List selectInstanceEntities()

gibt eine Liste mit allen Instance-Entities zurück, die von Class-Entities abgeleitet wurden, die von dieser Meta-Entity abgeleitet wurden

org.foederal.mind.IClassEntity:

List selectInstanceEntities()

gibt eine Liste mit allen Instance-Entities zurück, die von einer Class-Entity abgeleitet wurden

IMetaEntity selectMetaEntity()

gibt die Meta-Entity zurück, von der die Class-Entity abgeleitet wurde

org.foederal.mind.IInstanceEntity:

`IClassEntity selectClassEntity()`

gibt die Class-Entity zurück, von der diese Instance-Entity abgeleitet wurde

`IMetaEntity selectMetaEntity()`

gibt die Meta-Entity zurück, von der die Class-Entity abgeleitet wurde, von der diese Instance-Entity abgeleitet wurde

org.foederal.mind.IAssociation:

`int getIndex()`

gibt eine Zahl zurück, die angibt, an welcher Stelle die Entity am mit FORWARD bezeichneten Ende der Assoziation in einer Auflistung mehrerer Entities auftauchen soll

`IEntity selectEntity(UniDirection)`

gibt die Entity zurück, die sich am als Parameter übergebenen Ende der Assoziation (FORWARD oder BACKWARD) befindet

`IAssociationType selectAssociationType()`

gibt den Typ der Assoziation zurück

org.foederal.mind.IMetaAssociation:

`String getRole(UniDirection)`

gibt den Wert des Attributs `role` für die übergebene Richtung (FORWARD oder BACKWARD) zurück

org.foederal.mind.IAssociationType:

`Layer getAllowedDrainLayer(Layer)`

gibt die Schicht zurück, in der die FORWARD-Entity liegen muss, wenn die BACKWARD-Entity in der als Argument übergebenen Schicht liegt

`String getName()`

gibt den Namen des Assoziationstyps zurück

`INetConstraints getNetConstraints(Layer)`

gibt ein `INetConstraints`-Objekt zurück, das festlegt, wie eine Assoziation verwendet werden darf, deren Knoten am mit `BACKWARD` bezeichneten Ende in der als Parameter übergebenen Schicht liegt

`IUnit getUnit()`

gibt die Unit zurück, zu der der Assoziationstyp gehört

`org.foederal.sn.INetConstraints:`

`int getCopyPropagation(UniDirection)`

gibt eine `int`-Konstante zurück, die angibt ob und ggf. wie der Knoten am als Parameter übergebenen Ende (`FORWARD` oder `BACKWARD`) mitkopiert werden soll, wenn eine Assoziation kopiert wird, für die dieses `INetConstraints`-Objekt gilt

`Multiplicity getDepthConstraint()`

gibt die Tiefe zurück, die ein Graph haben darf, der aus Assoziationen aufgebaut ist, für die dieses `INetConstraints`-Objekt gilt

`boolean getNonCyclicConstraint()`

gibt zurück, ob ein Graph, der aus Assoziationen aufgebaut ist, für die dieses `INetConstraints`-Objekt gilt, Zyklen haben darf

`int getOrderConstraint()`

gibt eine `int`-Konstante zurück, die angibt, in welcher Reihenfolge die Knoten, die von Assoziationen dieses Typs verbunden werden, angezeigt werden sollen

`Multiplicity getRateConstraint(UniDirection)`

gibt zurück, mit wie vielen Assoziationen des Typs, für den dieses `INetConstraints`-Objekt gilt, die Entity am übergebenen Ende (`FORWARD` oder `BACKWARD`) verbunden sein darf

`int getRemovePropagation(UniDirection)`

gibt eine `int`-Konstante zurück, die angibt, ob das Löschen der Entity am übergebenen Ende einer Assoziation (`FORWARD` oder `BACKWARD`) dazu führen soll, dass auch die Entity am anderen Ende gelöscht wird.

org.foederal.mind.IUnit:

String getName()

gibt den Namen der Unit zurück

IUnit getSuperUnit()

gibt die übergeordnete Unit dieser Unit zurück

boolean isMutable()

gibt zurück, ob die Unit geändert werden darf

boolean isReadOnly()

gibt zurück, ob es sich um eine Unit handelt, die nur gelesen werden darf

org.foederal.sn.MultiplicityInterval:

int getLowerBound()

gibt die einschließliche Untergrenze des Intervalls zurück

int getUpperBound()

gibt die einschließliche Obergrenze des Intervalls zurück

org.foederal.sn.Multiplicity:

String toString()

gibt eine String-Repräsentation der zugehörigen Intervalle zurück

org.foederal.mind.Layer:

String toString()

gibt einen String mit dem Namen der Schicht zurück

org.foederal.sn.UniDirection:

String toString()

gibt einen String mit dem Namen der Richtung zurück

4.2.3 Leitlinien beim Design der OWL-Ontologien

Wie immer beim Design von Software, gibt es nicht eindeutig richtige und eindeutig falsche Möglichkeiten, sondern mehrere Alternativen, die alle ihre Vor- und Nachteile haben.

Im Rahmen dieser Arbeit entstanden drei Softwarepakete, von denen eines eine OWL Full-, eines eine OWL DL- und eines eine OWL Lite-Ontologie aus einem semantischen Netz der föderal-Informationsarchitektur extrahiert. Dabei wurde vor allem darauf geachtet, dass die generierten Ontologien soviel Gebrauch wie möglich von den speziellen Möglichkeiten der jeweiligen Sprachversion machen und möglichst viele Informationen so modelliert sind, dass sie mit Hilfe von Reasonern automatisch abgeleitet werden können. Darüber hinaus wurde auch in Bereichen, die nicht von der Sprachversion abhängen die Modellierung variiert, um verschiedene alternative Möglichkeiten aufzuzeigen.

4.2.4 Modellierungsansätze für grundlegende Konstrukte

4.2.4.1 Entities

Entities sind die Knoten in semantischen Netzen der föderal-Informationsarchitektur. In OWL-Ontologien bilden OWL-Klassen und OWL-Individuen die Knoten. Es liegt darum nahe, Entities auf OWL-Klassen oder –Individuen abzubilden.

In OWL DL und OWL Lite können nur Individuen `ObjectProperties` und `DatatypeProperties` haben. Da diese benötigt werden, um die Kanten zu modellieren, werden Entities in OWL DL und OWL Lite auf Individuen abgebildet.

In OWL Full können OWL-Klassen auch gleichzeitig als Individuen betrachtet werden, die dementsprechend `Properties` haben können. Dies eröffnet die Möglichkeit, Entities in OWL-Klassen umzusetzen, die gleichzeitig als Individuen der Klasse der übergeordneten Abstraktionsebene betrachtet werden. Im Automobil-Beispiel könnte es also die OWL-Klassen `<owl:Class rdf:ID="Automobil" />` und `<owl:Class rdf:ID="VW-Käfer" />` geben, wobei der OWL-Klasse VW-Käfer ein Individuum `<VW-Käfer rdf:ID="Herbie" />` angehört und die OWL-Klasse VW-Käfer selbst wieder als Individuum der Klasse `Automobil` angehört. Auf diese Weise könnte das gedankliche Modell der "doppelten Instanziierung" aus der föderal-Informationsarchitektur, das dort mittels Referenzen zwischen Java-Instanzen, die in Beziehung stehende Entities repräsentieren, gelöst ist, in OWL auch technisch auf eine tatsächliche Instanziierung

abgebildet werden. Vor- und Nachteile dieses Ansatzes werden im Abschnitt über die Extraktion der OWL-Full Ontologie näher erläutert.

4.2.4.2 Assoziationen und Assoziationstypen

Assoziationen bilden die Menge der Kanten im semantischen Netz der föderal- Informationsarchitektur. Assoziationstypen (`IAssociationType`) machen Aussagen darüber, wie die Assoziationen benutzt werden dürfen und beinhalten weitere Zusatzinformationen, wie etwa die Zugehörigkeit zu einer bestimmten Unit. In OWL besteht die Menge der Kanten aus den Eigenschaften. Der naheliegende Ansatz, Assoziationstypen auf die Definition von Eigenschaften abzubilden und Assoziationen auf Instanzen von Eigenschaften bringt Probleme mit sich. So wird es in OWL DL und OWL Lite schwierig, eine Stelle zu finden, an der man die Zusatzinformationen der Assoziationstypen unterbringen kann. Definiert man etwa im Automobilbeispiel eine Eigenschaft `<owl:ObjectProperty rdf:ID="eingebautIn" />`, fehlt ein Platz für die Information, zu welcher Unit dieser Assoziationstyp gehört. Die Definition einer zweiten Eigenschaft `<owl:ObjectProperty rdf:ID="inUnit" />` scheidet aus, da Instanzen von `ObjectProperties` nur Individuen miteinander verbinden können, nicht aber andere Eigenschaften mit Individuen. Um dieses Problem zu lösen wurden drei Ansätze entwickelt. In OWL Lite werden Assoziationstypen und Assoziationen auf Individuen abgebildet, in OWL DL werden `AnnotationProperties` verwendet und in OWL Full werden Eigenschaften als Individuen betrachtet. Näheres dazu findet sich in den Abschnitten über die Umsetzung der Extraktion in den einzelnen OWL-Version.

4.2.4.3 Constraints

Constraints machen in der föderal-Informationsarchitektur Angaben darüber, wie bestimmte Assoziationen verwendet werden dürfen.

Mit Hilfe von `RateConstraints` kann etwa angegeben werden, wie viele Knoten durch eine Assoziation eines bestimmten Typs miteinander verbunden werden dürfen. Diese Information lässt sich in OWL Full und OWL DL mit Hilfe von `<owl:minCardinality>` und `<owl:maxCardinality>` darstellen. In OWL Lite sind die Zahlenwerte für diese OWL-Konstrukte auf 0 und 1 beschränkt, was nicht ausreicht, da `RateConstraints` beliebige

Intervalle positiver Ganzzahlen umfassen können. In OWL Lite können Informationen über RateConstraints darum nur in menschenlesbarer Form kodiert werden.

Bei LayerConstraints, die angeben, in welchen Schichten sich die durch eine Assoziation verbundenen Entities befinden müssen, kann man `<owl:allValuesFrom>` benutzen.

Hierbei kann man sich die Tatsache zu Nutze machen, dass die Individuen, die Entities repräsentieren von einer der OWL-Klassen `<owl:Class rdf:ID="InstanceEntity"/>`, `<owl:Class rdf:ID="ClassEntity"/>` und `<owl:Class rdf:ID="MetaEntity"/>` (bzw. `<owl:Class rdf:ID="SuperMetaEntity"/>` in OWL Full) abgeleitet sind. Diese Klassen können als Attribut von `<owl:allValuesFrom>` dienen und so die Beschränkung auf eine Schicht realisieren. Auch hierzu findet sich Näheres in den Abschnitten über die Umsetzung der Extraktion in den einzelnen OWL-Version.

4.2.4.4 Eindeutige Namen

OWL geht nicht davon aus, dass es sich bei Dingen, die einen unterschiedlichen Namen haben, auch zwangsläufig um unterschiedliche Dinge handeln muss. Andersherum ist es aber so, dass bei Individuen, die den selben Namen tragen, davon ausgegangen wird, dass es sich um dasselbe Ding handelt. In Folge dessen muss bei der Extraktion einer OWL-Ontologie darauf geachtet werden, dass unterschiedliche Bestandteile des semantischen Netzes auch unterschiedliche Namen (genauer gesagt `rdf:IDs`) haben, damit nicht Informationen über unterschiedliche aber gleichnamige Bestandteile fälschlicher Weise zusammengeführt werden. So könnte etwa eine Maschine, die aus mehreren von einander unabhängig ein- und ausschaltbaren Komponenten besteht mehrere Entities mit Namen "SchalterAus" haben. Würde man entsprechend mehrere Individuen mit der `rdf:ID` "SchalterAus" in die Ontologie einfügen, würde ein Reasoner annehmen, dass es sich um ein und den selben Schalter handelt und die Informationen der verschiedenen Individuen zusammenführen. Die bei den meisten Komponenten des Netzes vorhandenen Methoden `getName()` oder `getUIString()` können für die `rdf:ID` nicht verwendet werden, da das föderal Engineering-Framework nicht sicherstellt, dass diese Namen einzigartig sind. Statt dessen wird zur eindeutigen Benennung die Methode `IObject.getOID()` verwendet, die zwar einen wenig aussagekräftigen, dafür aber eindeutigen Namen liefert. Alternativ hätte man auch während der Extraktion eine Tabelle mit allen verwendeten Namen aufbauen können,

um identische Namen zu finden und durchzunummerieren. Hierbei ist allerdings zu bedenken, dass bei größeren Netzen dadurch ein größerer Zeit- und Speicherplatz-Aufwand auftreten würde, dem kein systematischer sondern nur ein optischer Nutzen gegenüber steht.

Auch in Zusammenhang mit Eigenschaften stellt sich das Problem, wie Namen konstruiert werden sollen. Sie werden unter anderem dazu benutzt, Rückgabewerte von Methoden in die Ontologie einzufügen.

Beispielsweise wäre ein Ansatz den Rückgabewert der Methode

`IAssociationType.getUnit()` mit Hilfe der folgenden Eigenschaft in die Ontologie einzubauen:

```
<owl:ObjectProperty rdf:ID="inUnit">
  <rdfs:domain rdf:resource="#AssociationType"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>
```

Es gibt aber auch andere Java-Klassen bzw. –Schnittstellen im föderal Engineering-Framework, die über eine Methode `getUnit()` verfügen, wie etwa `IEntity.getUnit()`. Würde man für diese Methode eine zweite Eigenschaft "inUnit" mit der Domain `Entity` definieren, würde auch hier ein Reasoner davon ausgehen, dass es sich um ein und die selbe Eigenschaft handelt und die beiden unterschiedlichen Informationen über die Domain zusammenführen. Da mehrere Angaben zur Domain so interpretiert werden, dass es sich bei der Domain um die Schnittmenge der angegebenen Klassen handelt, würde das im Fall von `inUnit` dazu führen, dass für Individuen, die über `inUnit` mit einer Unit verbunden sind, geschlossen werden könnte dass sie sowohl Assoziationsstyp als auch Entity sind, was nicht der Realität entspricht.

Eine andere Möglichkeit wäre, eine einzige Eigenschaft `inUnit` mit Domain `Object` zu definieren. Dabei könnte aber ein Reasoner aus der Tatsache, dass ein Individuum mittels dieser Eigenschaft mit einer Unit verbunden ist nur noch schließen, dass das Individuum der Klasse `Object` angehört und nicht welcher spezifischen Subklasse von `Object`, wie es bei der Verwendung mehrerer verschiedener Eigenschaften mit spezifischeren Domains zur Abbildung von `getUnit()` der Fall wäre.

Um das Problem zu lösen, wird bei der Extraktion der Ontologien jeder Eigenschaft, die den Rückgabewert einer Methode referenziert, der Name der Klasse, die die Domain der Eigenschaft bildet vorangestellt. Darauf folgt ein Unterstrich und der Name der Methode. Im Fall von `getUnit()` werden also folgende zwei Eigenschaften definiert:

```
<owl:ObjectProperty rdf:ID="AssociationType_InUnit">  
  <rdfs:domain rdf:resource="#AssociationType"/>  
  <rdfs:range rdf:resource="#Unit"/>  
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="Entity_InUnit">  
  <rdfs:domain rdf:resource="#Entity"/>  
  <rdfs:range rdf:resource="#Unit"/>  
</owl:ObjectProperty>
```

AssociationTypes werden dann mittels `AssociationType_InUnit` mit ihrer Unit verbunden und Entities mittels `Entity_InUnit`.

In der Praxis würde es genügen, diese Art der Namensgebung nur bei Methodennamen zu praktizieren, die mehrfach vorkommen. Im Sinne guten Software-Engineerings wurde die Voranstellung der Domain jedoch zum allgemeinen Prinzip erhoben. Auf diese Weise ist sichergestellt, dass das Übersehen doppelter Namen keine negativen Auswirkungen hat und bei möglichen zukünftigen Änderungen an der Schnittstelle des föderal Engineering-Framework muss nicht jedes Mal überprüft werden, ob sich neue doppelte Methodennamen ergeben haben.

4.2.5 Extraktion der OWL Lite-Ontologie

4.2.5.1 Grundsätzliches

Bei der Modellierung der OWL Lite-Ontologie wurde versucht, die Struktur des Netzes aus dem föderal Engineering-Framework möglichst originalgetreu in OWL nachzubauen. Das heißt, alle Java-Klassen bzw. –Schnittstellen aus dem föderal Engineering-Framework wurden auf OWL-Klassen abgebildet und alle Java-Instanzen aus dem föderal Engineering-Framework wurden auf OWL-Individuen abgebildet. Dies hat unter anderem zur Folge, dass auch Assoziationen auf OWL-Individuen abgebildet werden und nicht auf Eigenschaften. Die Eigenschaften werden nur benutzt, um Verbindungen zwischen den Individuen herzustellen. Der Nachteil dieser Vorgehensweise ist, dass nur sehr wenige Informationen im OWL-Netz

mit Hilfe von Reasonern automatisch abgeleitet werden können. Dieser Nachteil hält sich allerdings dadurch in Grenzen, dass es auf Grund der beschränkten Möglichkeiten von OWL Lite ohnehin nur wenige Fälle gibt, in denen man diese Fähigkeit nutzen könnte. Der Vorteil daran ist, dass das entstehende Netz eine einfache Struktur hat und so gerade als Einstieg in die Materie gut geeignet ist. Wenn Modellierungsalternativen, die in den beiden anderen OWL-Versionen implementiert sind, auch in OWL Lite zur Verfügung gestanden hätten wird darauf hingewiesen.

4.2.5.2 Parameter von Methoden

Während es sich bei einer OWL-Ontologie um eine rein, statische Datenstruktur handelt, bestehen die Java-Klassen und –Instanzen, aus denen das föderal Engineering-Framework aufgebaut ist, nicht nur aus den Daten, sondern umfassen auch Methoden und damit Befehle, die ausgeführt werden können und dem System damit eine gewisse Dynamik verleihen. Insbesondere ist im föderal Engineering-Framework, wie in der objektorientierten Programmierung üblich, ein direkter Zugriff auf die Daten, die Attribute der Java-Instanzen, aus denen das semantische Netz besteht, nicht möglich. Statt dessen werden ausschließlich Methoden aufgerufen, die Befehle ausführen und die gewünschten Informationen zurückgeben. Kein Problem erwächst hieraus bei Methoden, die keine Parameter haben. Sie liefern bei gleichem Netz stets den selben Wert zurück. Als Beispiel hierfür soll die Methode `IUnit.isMutable()` dienen. Der von ihr gelieferte Wert lässt sich relativ einfach in eine OWL-Ontologie aufnehmen, indem man zum Beispiel in das OWL-Individuum, das die Unit repräsentiert eine Eigenschaft `isMutable` einbaut und dort den Rückgabewert der Methode (in folgendem Beispiel "true") einträgt:

```
<Unit rdf:ID="rootUnit">  
  <isMutable rdf:datatype="&xsd:boolean">true</isMutable>  
</Unit>
```

Anders sieht es aber aus, wenn eine Methode einen Parameter hat. In diesem Fall gibt es mehrere mögliche Rückgabewerte einer Methode. Die Methode führt intern eine Berechnung durch und liefert einen Rückgabewert je nach übergebenem Parametern zurück. Da es in OWL keine Befehle gibt, mit denen man Berechnungen durchführen könnte, muss der zu einem Parameter gehörende Rückgabewert direkt aus den Daten ablesbar sein. Es ist darum

notwendig für jede Kombination eines Parameters mit dem entsprechenden Rückgabewert einer Methode einen Eintrag in die Ontologie zu machen, der diese zueinander in Beziehung setzt. Im Rahmen dieser Arbeit wird von zwei verschiedenen Möglichkeiten, diese Beziehungen darzustellen gebrauch gemacht, die nun am Beispiel der Methode `IAssociation.selectEntity(UniDirection)` erläutert werden.

Die erste Möglichkeit ist, ein zusätzliches Konstrukt einzuführen, das jeweils einen Parameter und den zugehörigen Rückgabewert umfasst. Dieser Ansatz wurde in der OWL Lite-Ontologie verfolgt. Die hierfür eingeführten Konstrukte wurden "Assertions" genannt. Abb. 8 zeigt schematisch am Beispiel der Assoziation `eingebautIn` zwischen den Instance-Entities `Herbie` und `HerbiesMotor` aus dem Automobilbeispiel wie die verschiedenen OWL-Individuen bei der Modellierung dieser Assoziation zusammenspielen.

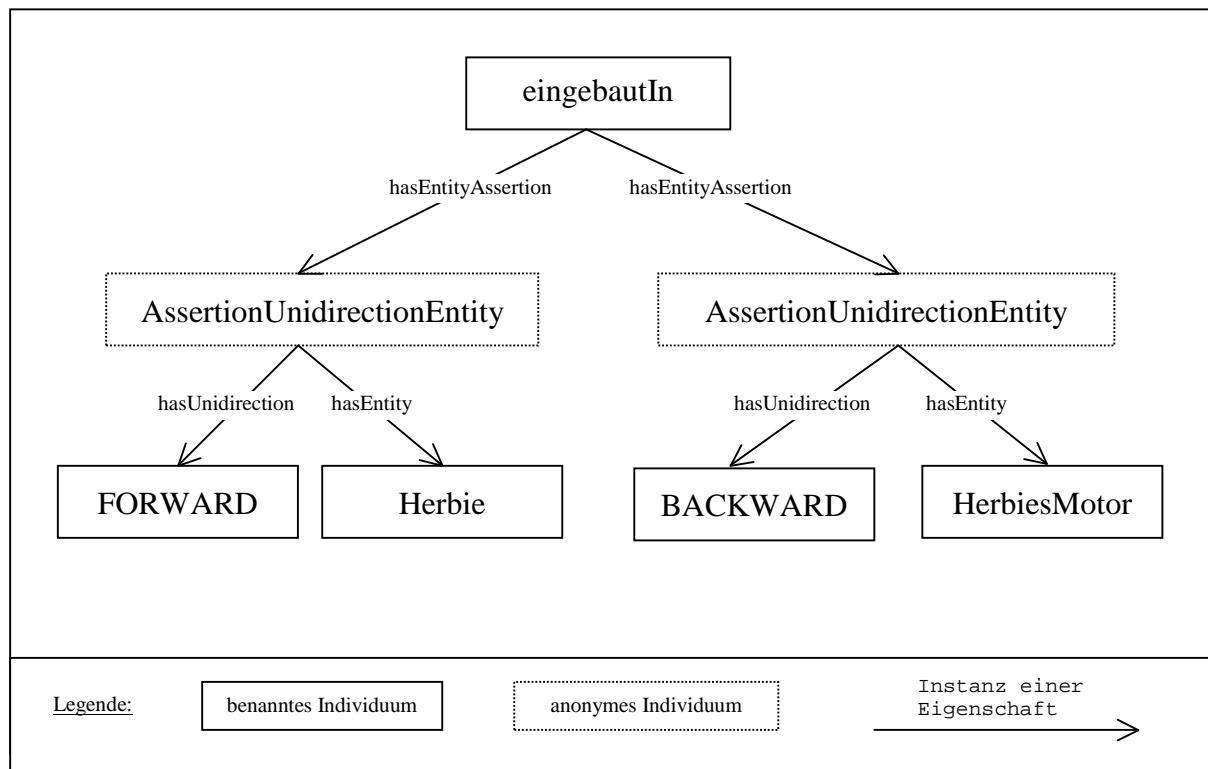


Abb. 8: Modellierung von Methoden mit Parameter mittels Assertions

Ein Individuum der OWL-Klasse `AssertionUnidirectionEntity` stellt jeweils eine Verbindung zwischen den OWL-Repräsentanten eines Parameters vom Typ `UniDirection` und eines Rückgabewerts vom Typ `Entity` her.

Es folgt der OWL-Code zur Umsetzung des Diagramms aus Abb. 8. Dabei wird davon

ausgegangen, dass an anderer Stelle bereits die Individuen `Herbie` und `HerbiesMotor`

definiert sind. Bei der Namensgebung wurden der Anschaulichkeit wegen von den Konventionen der Extraktionssoftware abgewichen.

```
<owl:Class rdf:ID="AssertionUnidirectionEntity">
</owl:Class>

<owl:ObjectProperty rdf:ID="hasEntityAssertion">
  <rdfs:domain rdf:resource="#Association"/>
  <rdfs:range rdf:resource="#AssertionUnidirectionEntity"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasUnidirection">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionEntity"/>
  <rdfs:range rdf:resource="#Unidirection"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasEntity">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionEntity"/>
  <rdfs:range rdf:resource="#Entity"/>
</owl:ObjectProperty>

<Association rdf:ID="eingebautIn">

  <hasEntityAssertion>
    <AssertionUnidirectionEntity>
      <hasUnidirection rdf:resource="#backward"/>
      <hasEntity rdf:resource="#HerbiesMotor"/>
    </AssertionUnidirectionEntity>
  </hasEntityAssertion>

  <hasEntityAssertion>
    <AssertionUnidirectionEntity>
      <hasUnidirection rdf:resource="#forward"/>
      <hasEntity rdf:resource="#Herbie"/>
    </AssertionUnidirectionEntity>
  </hasEntityAssertion>

</Association>
```

Die zweite Möglichkeit, Methoden mit Parametern zu modellieren ist, Eigenschaften zu konstruieren, deren Namen eine Kombination aus dem Namen der Methode und den verschiedenen Parametern ist. Als Wert dieser Eigenschaften wird der jeweilige Rückgabewert der Methode eingetragen.

Für die zwei möglichen Werte vom Typ `UniDirection`, `FORWARD` und `BACKWARD`, sieht das Beispiel wie folgt aus:

```
<Association rdf:ID="eingebautIn">
  <selectEntityForward rdf:resource="#Herbie"/>
  <selectEntityBackward rdf:resource="#HerbiesMotor"/>
</Association>
```

Der Vorteil dieser Variante gegenüber der Verwendung von Assertions ist die größere Kompaktheit. Der Nachteil ist, dass unter Umständen mehrdeutige Namen von Eigenschaften entstehen können, da die Grenzen zwischen dem Methoden- und dem Parameterteil des Namens nicht klar sind. Eine Eigenschaft `selectEntityForward` kann genauso für eine Methode `select`, die einen Parameter mit Namen `EntityForward` entgegen nimmt erzeugt werden, wie auch für eine Methode `selectEntity`, die einen Parameter mit Namen `Forward` entgegen nimmt. Auch eine Trennung der verschiedenen Namensteile durch Sonderzeichen stellt keine vollständige Lösung des Problems dar, da nicht mit Sicherheit ausgeschlossen werden kann, dass das entsprechende Sonderzeichen nicht auch in normalen Namen vorkommt. Derartige Namenskonflikte kommen zwar in der Praxis zur Zeit bei den Elementen des föderal Engineering-Framework nicht vor, da aber nicht sicher gestellt ist, dass das auch in Zukunft so bleibt wurde bei der Extraktion der OWL Lite-Ontologie der Ansatz mit Assertions gewählt.

4.2.5.3 Definitionen von Klassen

Wie bereits erwähnt, wird in der OWL Lite-Ontologie für jede Java-Klasse bzw. – Schnittstelle des föderal Engineering-Framework eine entsprechende OWL-Klasse definiert. Im Einzelnen handelt es sich dabei um folgende Klassen:

```
<owl:Class rdf:ID="Object">
</owl:Class>
```

```
<owl:Class rdf:ID="Entity">
  <rdfs:subClassOf rdf:resource="#Object"/>
</owl:Class>

<owl:Class rdf:ID="MetaEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
</owl:Class>

<owl:Class rdf:ID="ClassEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
</owl:Class>

<owl:Class rdf:ID="InstanceEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
</owl:Class>

<owl:Class rdf:ID="AssociationType">
  <rdfs:subClassOf rdf:resource="#Object"/>
</owl:Class>

<owl:Class rdf:ID="Association">
  <rdfs:subClassOf rdf:resource="#Object"/>
</owl:Class>

<owl:Class rdf:ID="MetaAssociation">
  <rdfs:subClassOf rdf:resource="#Association"/>
</owl:Class>

<owl:Class rdf:ID="Unit">
</owl:Class>

<owl:Class rdf:ID="Layer">
</owl:Class>

<owl:Class rdf:ID="Netconstraints">
</owl:Class>

<owl:Class rdf:ID="Unidirection">
</owl:Class>

<owl:Class rdf:ID="Multiplicity">
</owl:Class>
```

```
<owl:Class rdf:ID="MultiplicityInterval">
</owl:Class>
```

Hinzu kommen die Klassen der Assertions, die die Verbindung zwischen einem an eine Methode übergebenen Parameter und deren Rückgabewert herstellen:

```
<owl:Class rdf:ID="AssertionUnidirectionMultiplicity">
</owl:Class>
```

```
<owl:Class rdf:ID="AssertionUnidirectionInteger">
</owl:Class>
```

```
<owl:Class rdf:ID="AssertionUnidirectionEntity">
</owl:Class>
```

```
<owl:Class rdf:ID="AssertionUnidirectionString">
</owl:Class>
```

```
<owl:Class rdf:ID="AssertionLayerLayer">
</owl:Class>
```

```
<owl:Class rdf:ID="AssertionLayerNetconstraints">
</owl:Class>
```

4.2.5.4 Definitionen von Eigenschaften

Die meisten Eigenschaften dienen dazu, Attribute von Java-Klassen, genauer gesagt parameterlosen Methoden, die auf Attribute von Java-Klassen zugreifen, abzubilden. Diese Eigenschaften sind sich alle sehr ähnlich. Ihre Domain ist die OWL-Entsprechung der Java-Klasse, die die Methode definiert, ihre Range eine OWL-Entsprechung des Rückgabetyps der Methode. Außerdem haben sie funktionalen Charakter. Als Beispiel für diese Art Eigenschaften kann hier `Entity_HasContent` dienen, welche die Methode `IEntity.getContent()` abbildet:

```
<owl:DatatypeProperty rdf:ID="Entity_HasContent">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
```

Im Folgenden sind alle derartigen Eigenschaften und die Methoden, die sie abbilden tabellarisch aufgelistet.

Eigenschaft	abgebildete Methode
Object_HasUIString	<code>IObject.getUIString()</code>
Object_IsMutable	<code>IObject.isMutable()</code>
Entity_HasContent	<code>IEntity.getContent()</code>
Entity_InLayer	<code>IEntity.getLayer()</code>
Entity_InUnit	<code>IEntity.getUnit()</code>
MetaEntity_IsAttribute	<code>IMetaEntity.isAttribute()</code>
MetaEntity_IsEO	<code>IMetaEntity.isEO()</code>
MetaEntity_HasEOName	<code>IMetaEntity.getEOName()</code>
MetaEntity_IsAbstract	<code>IMetaEntity.isAbstract()</code>
MetaEntity_HasAttributetype	<code>IMetaEntity.getAttributeType()</code>
Association_HasIndex	<code>IAssociation.getIndex()</code>
Association_HasType	<code>IAssociation. selectAssociationType()</code>
AssociationType_HasName	<code>IAssociationType.getName()</code>
AssociationType_InUnit	<code>IAssociationType.getUnit()</code>
Netconstraints_HasNoncyclicConstraint	<code>INetConstraints. getNonCyclicConstraint()</code>
Netconstraints_HasDepthConstraint	<code>INetConstraints. getDepthConstraint()</code>
Netconstraints_HasOrderConstraint	<code>INetConstraints. getOrderConstraints()</code>
Unit_IsReadOnly	<code>IUnit.isReadOnly()</code>
Unit_IsMutable	<code>IUnit.isMutable()</code>

Eine weitere Gruppe von Eigenschaften bildet Methoden mit Parametern ab. Ihre Domain ist die OWL-Entsprechung der Java-Klasse, die die Methode definiert, ihre Range ist eine Assertion, die einem Parameter den zugehörigen Rückgabewert zuordnet. Ein Beispiel hierfür ist die Eigenschaft `Association_HasEntityAssertion`. Sie bildet die Methode `IAssociation.selectEntity(UniDirection)` ab:

```

<owl:ObjectProperty
  rdf:ID="Association_HasEntityAssertion">
  <rdfs:domain rdf:resource="#Association"/>
  <rdfs:range rdf:resource="#AssertionUnidirectionEntity"/>
</owl:ObjectProperty>

```

Eine `AssertionUnidirectionEntity` beinhaltet dabei ein Paar, bestehend aus einem Parameter (einer `Unidirection`) und dem entsprechenden Rückgabewert (einer `Entity`). Instanzen dieser Eigenschaft geben an, welche `Entity` sich am durch die `UniDirection` spezifizierten Ende einer Assoziation befindet.

Folgende Eigenschaften dienen dazu, unter Zuhilfenahme einer Assertion Methoden mit Parametern abzubilden:

Eigenschaft	abgebildete Methode
<code>Association_HasEntityAssertion</code>	<code>IAssociation.selectEntity(UniDirection)</code>
<code>MetaAssociation_HasRoleAssertion</code>	<code>IMetaAssociation.getRole(UniDirection)</code>
<code>AssociationType_HasNetconstraintsAssertion</code>	<code>IAssociationType.getNetConstraints(Layer)</code>
<code>AssociationType_HasLayerAssertion</code>	<code>IAssociationType.getAllowedDrainLayer(Layer)</code>
<code>Netconstraints_HasRateAssertion</code>	<code>INetConstraints.getRateConstraint(UniDirection)</code>
<code>Netconstraints_DefinesCopypropagation</code>	<code>INetConstraints.getCopyPropagation(UniDirection)</code>
<code>Netconstraints_DefinesRemovepropagation</code>	<code>INetConstraints.getRemovePropagation(UniDirection)</code>

Einige Eigenschaften werden als Inverse einer anderen definiert. Dies nutzt die Möglichkeit, dass Reasoner aus einer Ontologie Umkehrbeziehungen automatisch ableiten können, ohne dass Instanzen dieser als Inversen definierten Eigenschaften in der Ontologie explizit enthalten sein müssen. Als Beispiel hierfür dient die Eigenschaft `Unit_ContainsEntity` die Verbindungen zwischen einer `Unit` und allen darin enthaltenen `Entities` herstellt. Es wird dabei benutzt, dass `Entities` eine Eigenschaft `Entity_InUnit` haben, die eine Verbindung in umgehrter Richtung herstellt:

```

<owl:ObjectProperty rdf:ID="Unit_ContainsEntity">
  <owl:inverseOf rdf:resource="#Entity_InUnit"/>
</owl:ObjectProperty>

```

Folgende Eigenschaften sind als Inverse definiert und stellen so implizite Umkehrbeziehungen her:

Eigenschaft	inverseOf
Unit_ContainsEntity	Entity_InUnit
Unit_ContainsAssociationtype	AssociationType_InUnit
Unit_HasSubunit	Unit_HasSuperunit
AssociationType_TypeOf	Association_HasType
Entity_AssociatedToSub	Entity_AssociatedToSuper
MetaEntity_AssociatedToClass	ClassEntity_AssociatedToMeta
ClassEntity_AssociatedToInstance	InstanceEntity_AssociatedToClass

Bei der Abbildung der Beziehungen zwischen den Meta-, Class- und Instance-Entities wird die Möglichkeit von OWL, Informationen so darzustellen, dass sie mittels eines Reasoners automatisch hergeleitet werden können, besonders intensiv genutzt. Es werden folgende Eigenschaften verwendet:

```
<owl:ObjectProperty rdf:ID="Entity_AssociatedToSuper">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Entity"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="Entity_AssociatedToSub">
  <owl:inverseOf
    rdf:resource="#Entity_AssociatedToSuper"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="ClassEntity_AssociatedToMeta">
  <rdfs:subPropertyOf
    rdf:resource="#Entity_AssociatedToSuper"/>
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#ClassEntity"/>
  <rdfs:range rdf:resource="#MetaEntity"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="MetaEntity_AssociatedToClass">
  <rdfs:subPropertyOf rdf:resource="#Entity_AssociatedToSub"/>
  <owl:inverseOf
    rdf:resource="#ClassEntity_AssociatedToMeta"/>
</owl:ObjectProperty>
```

```

<owl:ObjectProperty rdf:ID="InstanceEntity_AssociatedToClass">
  <rdfs:subPropertyOf
    rdf:resource="#Entity_AssociatedToSuper" />
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="#InstanceEntity" />
  <rdfs:range rdf:resource="#ClassEntity" />
</owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:ID="ClassEntity_AssociatedToInstance">
  <rdfs:subPropertyOf rdf:resource="#Entity_AssociatedToSub" />
  <owl:inverseOf
    rdf:resource="#InstanceEntity_AssociatedToClass" />
</owl:ObjectProperty>

```

Zur Veranschaulichung des Codes werden die Beziehungen zwischen diesen Eigenschaften graphisch dargestellt. Abbildung 9 zeigt die Vererbungsbeziehungen, Abbildung 10 den Zusammenhang der Eigenschaften und OWL-Klassen mit den verschiedenen Entities.

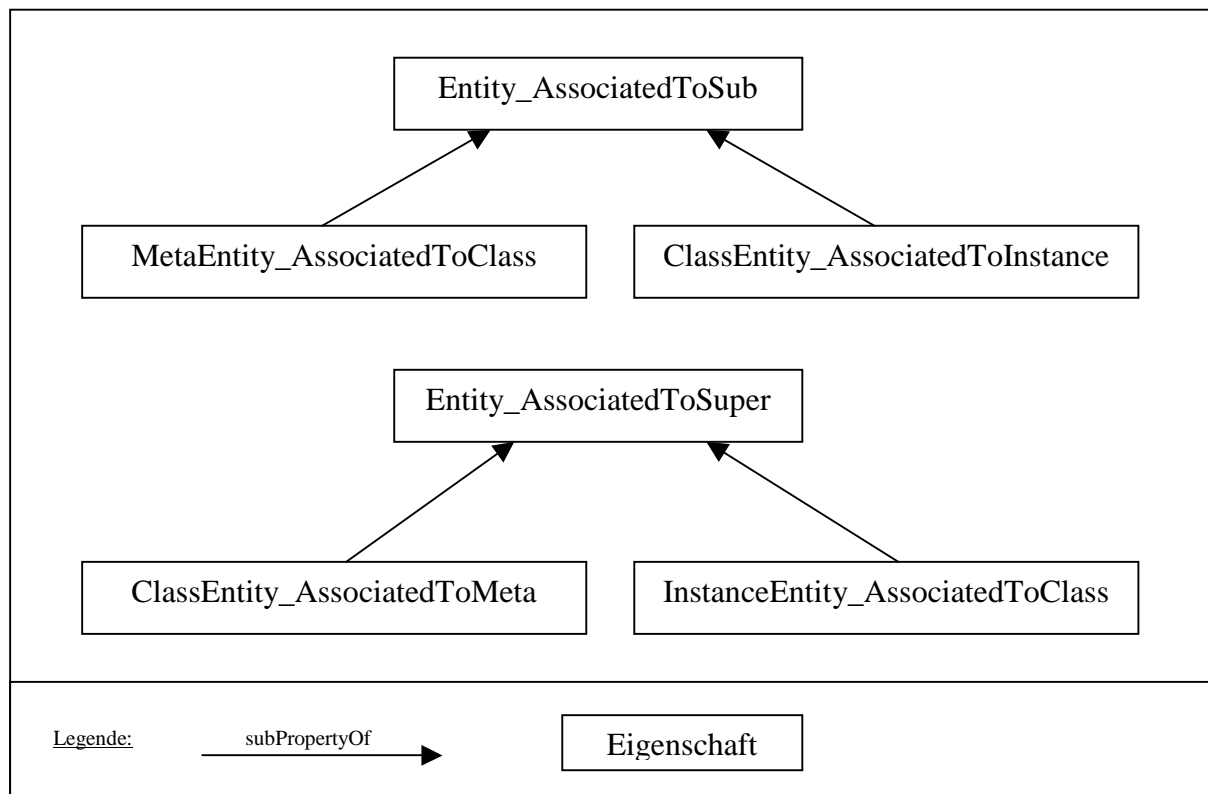


Abb. 9: Vererbungsbeziehungen zwischen Eigenschaften, die Ableitungsbeziehungen von Entities darstellen

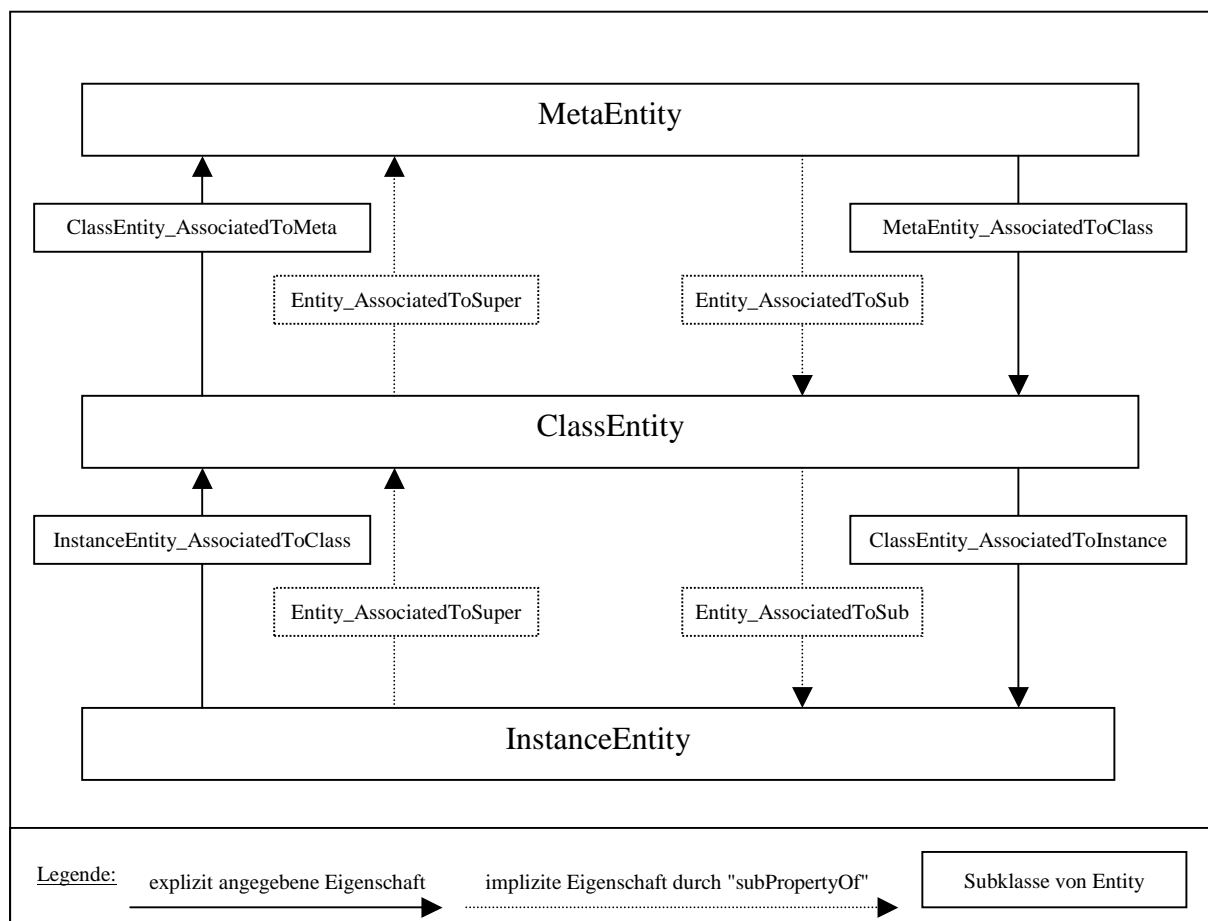


Abb.10 : Zusammenspiel verschiedener Eigenschaften bei der Darstellung von Ableitungsbeziehungen zwischen Entities

Dadurch, dass sowohl `InstanceEntity_AssociatedToClass` als auch `ClassEntity_AssociatedToMeta` von der transitiven Eigenschaft `Entity_AssociatedToSuper` erben, entsteht eine implizite `Entity_AssociatedToSuper`-Kette von `InstanceEntity` zu `MetaEntity`, mit deren Hilfe ein Reasoner zu jeder Instance-Entity die zugehörig Meta-Entity herleiten kann, ohne dass diese Beziehung explizit angegeben wäre. Gleiches passiert in umgekehrter Richtung mittels der Eigenschaft `Entity_AssociatedToSub`.

Für die Abbildung der Schachtelungsbeziehungen zwischen den Units werden die Eigenschaften `Unit_HasSuperunit` und `Unit_HasSubunit` verwendet. Sie sind ebenfalls transitiv, sodass ein Reasoner alle über- und untergeordneten Units einer Unit herleiten kann.

```

<owl:ObjectProperty rdf:ID="Unit_HasSuperunit">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:ID="Unit_HasSubunit">
  <owl:inverseOf rdf:resource="#Unit_HasSuperunit"/>
</owl:ObjectProperty>

```

4.2.5.5 Individuen und Instanzen von Eigenschaften

Zusätzlich zur Definition der OWL-Klassen und Eigenschaften müssen die OWL-Individuen erzeugt werden. Hierzu wird das semantische Netz der föderal Informationsarchitektur durchlaufen und es werden für die Java-Instanzen entsprechende OWL-Individuen inklusive zugehörigen Instanzen von Eigenschaften in die Ontologie eingefügt.

So wird zum Beispiel für die Class-Entity "VW-Käfer" aus dem Automobilbeispiel ein Individuum von `ClassEntity` erzeugt:

```

<ClassEntity rdf:ID="-3472960911693331053">
  <Object_HasUIString rdf:datatype="&xsd:string">
    VW-Käfer</Object_HasUIString>
  <Object_IsMutable
    rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">
    VW-Käfer</Entity_HasContent>
  <Entity_InLayer rdf:resource="#C"/>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <ClassEntity_AssociatedToMeta
    rdf:resource="#1945574459181637797"/>
</ClassEntity>

```

Wie bereits erwähnt, wird als `rdf:ID` der Rückgabewert der Methode `getOID()` verwendet, welcher in diesem Fall `-3472960911693331053` lautet. Die Instanz von `ClassEntity_AssociatedToMeta` verweist auf das Individuum `#1945574459181637797`, wobei es sich dabei um das Individuum handelt, das die Meta-Entity "Automobil" repräsentiert. Das Individuum `C`, auf das `Entity_InLayer` verweist, ist das Individuum, das die Class-Schicht repräsentiert.

Die Assoziation "eingebaut-in", die die Class-Entity "2-Takt-Motor" mit "VW-Käfer" verbindet, sieht wie folgt aus:

```
<Association
  rdf:ID="-4346918702622399155;-1155961494047928391;-
    3472960911693331053">

  <Association_HasIndex
    rdf:datatype="&xsd;integer">0</Association_HasIndex>

  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection
        rdf:resource="#forward"/>
      <AssertionUnidirectionEntity_HasEntity
        rdf:resource="#-3472960911693331053"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>

  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection
        rdf:resource="#backward"/>
      <AssertionUnidirectionEntity_HasEntity
        rdf:resource="#-4346918702622399155"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>

  <Association_HasType rdf:resource="#-1155961494047928391"/>
</Association>
```

Assoziationen haben im Allgemeinen sehr lange Ids. In diesem Fall umfasst sie 61 Zeichen.

Man erkennt die beiden Instanzen der Eigenschaft

Association_HasEntityAssertion, die je ein anonymes Individuum einer

AssertionUnidirectionEntity referenzieren. Die

AssertionUnidirectionEntity wiederum stellen eine Verbindung zwischen einer

Entity und der Seite der Assoziation her, an der diese sich befindet. Eine Instanz von

Association_HasType referenziert ein Individuum #-1155961494047928391,

welches den Assoziationstyp der Assoziation repräsentiert.

Sehr umfangreich werden die Beschreibungen der Assoziationstypen. Sie enthalten jedoch nichts qualitativ Neues gegenüber den Entities und Assoziationen. Sie bestehen ebenfalls aus Instanzen von Eigenschaften, die auf andere benannte oder anonyme Individuen verweisen. Lediglich auf eine Kleinigkeit soll hier eingegangen werden. Sie hängt mit der Darstellung von Multiplicities zusammen. Man betrachtet hierzu folgendes Beispiel, welches zeigt wie die Ganzzahlen aus denen eine Multiplicity besteht in Intervalle eingeteilt werden:

```
<Multiplicity>

  <Multiplicity_IncludesIntervall>
    <MultiplicityIntervall>
      <MultiplicityIntervall_HasLowerBound
        rdf:datatype="&xsd;integer">0
      </MultiplicityIntervall_HasLowerBound>
      <MultiplicityIntervall_HasUpperBound
        rdf:datatype="&xsd;integer">0
      </MultiplicityIntervall_HasUpperBound>
    </MultiplicityIntervall>
  </Multiplicity_IncludesIntervall>

  <Multiplicity_IncludesIntervall>
    <MultiplicityIntervall>
      <MultiplicityIntervall_HasLowerBound
        rdf:datatype="&xsd;integer">1
      </MultiplicityIntervall_HasLowerBound>
      <MultiplicityIntervall_HasUpperBound
        rdf:datatype="&xsd;integer">1
      </MultiplicityIntervall_HasUpperBound>
    </MultiplicityIntervall>
  </Multiplicity_IncludesIntervall>

</Multiplicity>
```

Es stellt sich hier die Frage, warum der Bereich 0..1 in zwei Intervalle aufgeteilt wird, die jeweils nur aus den Zahlen 0 und 1 bestehen, anstatt ein einziges Intervall zu konstruieren, das von 0 bis 1 reicht. Die Antwort darauf ist, dass das föderal Engineering-Framework Intervalle, die nur aus zwei Zahlen bestehen intern in zwei Intervalle von nur einer Zahl umwandelt. Es wäre möglich solche Fälle bei der Extraktion der Ontologie herauszufiltern und in ein

einziges Intervall umzuwandeln. Da es aber semantisch keinen Unterschied macht, wurde darauf verzichtet.

Die restlichen Bestandteile der OWL Lite-Ontologie weisen keine Besonderheiten auf und bedürfen keiner besonderen Erklärung. Die vollständige OWL Lite-Ontologie, die aus dem Automobilbeispiel extrahiert wurde findet sich in Anhang A.

4.2.6 Extraktion der OWL DL-Ontologie

4.2.6.1 Grundsätzliches

Die Beschreibung der Modellierung der OWL DL-Ontologie baut auf der Beschreibung der OWL-Lite-Ontologie auf. Konstrukte der OWL Lite-Ontologie, die auch in der OWL DL-Version verwendet werden, werden nicht erneut beschrieben, sondern die Unterschiede zur OWL Lite-Version.

Der wichtigste Unterschied zwischen der Modellierung der OWL DL-Ontologie und der OWL Lite-Version ist, dass Assoziationstypen und Assoziationen in OWL DL nicht auf OWL-Individuen abgebildet werden. Statt dessen modellieren Eigenschaften die Assoziationstypen und Instanzen von Eigenschaften die Assoziationen. Diese Art der Modellierung stünde zwar auch schon in OWL Lite zur Verfügung, würde aber keinen besonderen Vorteil bringen, da die Möglichkeiten bei der Definition von Beschränkungen zu begrenzt sind. Da in OWL DL in Zusammenhang mit `<owl:minCardinality>` und `<owl:maxCardinality>` auch andere Werte als 0 und 1 zulässig sind, ist es möglich die RangeConstraints eines Assoziationstyps mit Hilfe OWL-Beschränkungen abzubilden. Außerdem lassen sich die LayerConstraints mit Hilfe von `<owl:allValuesFrom>` abbilden, indem man die Range auf eine der OWL-Klassen `MetaEntity`, `ClassEntity` oder `InstanceEntity` beschränkt.

4.2.6.2 Definitionen von Eigenschaften

Zusätzlich zu den bereits aus OWL Lite bekannten Eigenschaften zur Abbildung von Rückgabewerten von Methoden, werden in OWL DL Eigenschaften zur Abbildung von Assoziationstypen definiert. Eine vereinfachte Eigenschaft zur Abbildung des Assoziationstyps "eingebaut-in" aus dem Automobilbeispiel könnte beispielsweise wie folgt aussehen:

```

<owl:ObjectProperty rdf:ID="eingebautIn">
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Entity"/>
</owl:ObjectProperty>

```

Als Teil der Definition der OWL-Klassen `MetaEntity`, `ClassEntity` und `InstanceEntity` können dann die `RateConstraints` und `LayerConstraints` des Assoziationsstyps mit Hilfe von Beschränkungen abgebildet werden. Für die `LayerConstraints` sähe das beispielsweise folgendermaßen aus:

```

<owl:Class rdf:ID="InstanceEntity">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn"/>
      <owl:allValuesFrom rdf:resource="#InstanceEntity"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

Diese Beschränkung drückt aus, dass Individuen von `InstanceEntity` mittels `eingebautIn` nur mit anderen Individuen von `InstanceEntity` verbunden werden.

Folgendes Beispiel zeigt das Prinzip, wie sich die Kardinalität beschränken lässt:

```

<owl:Class rdf:ID="InstanceEntity">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn"/>
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

Diese Beschränkung drückt aus, dass Individuen von `InstanceEntity` mittels maximal einer Instanz von `eingebautIn` mit anderen Individuen verbunden sind. Dieser Ausschnitt aus dem Automobilbeispiel bedeutet einfach gesagt, dass jedes Teil in höchstens ein anderes

Teil eingebaut sein kann. Es wird also die Zahl der Individuen beschränkt, die als Objekt der Eigenschaft eingebaut In fungieren. In der föderal-Informationsarchitektur lässt sich aber nicht nur Kardinalität der Objekte einer Assoziation beschränken sondern auch die der Subjekte. Es lässt sich also auch eine Aussage darüber machen, wie viele Teile in ein bestimmtes größeres Teil eingebaut sein können. In OWL lassen sich keine Aussagen über die Kardinalität des Subjekts einer Eigenschaft formulieren. Um die Kardinalität in beide Richtungen ausdrücken zu können, wird darum für jeden Assoziationstyp nicht nur eine Eigenschaft definiert, sondern gleichzeitig auch ihre Inverse. Bei der inversen Eigenschaft sind die Rollen von Subjekt und Objekt gegenüber der Ausgangseigenschaft vertauscht. Die Inverse kann Aussagen über die Kardinalität ihrer Objekte machen, welche gleichzeitig die Subjekte der Ausgangseigenschaft sind. Um die beiden Eigenschaften, die für einen Assoziationstyp generiert werden unterscheiden zu können, werden die Namen mit den Zusätzen "_forward" und "_backward" versehen.

Da Instanzen von Eigenschaften als Tags mit dem Namen der Eigenschaft in die Ontologien eingefügt werden, muss außerdem darauf geachtet werden, dass der Name einer Eigenschaft auch ein gültiger Name für ein Tag ist. Da die in OWL Lite als Name verwendete Rückgabe der Methode `IAssociationType.getOID()` im Allgemeinen mit einer Zahl beginnt, Tags aber nicht mit Zahlen beginnen dürfen, wird der Rückgabe von `IAssociationType.getOID()` noch die Rückgabe von `IAssociationType.getName()` vorangestellt. Der Name einer Eigenschaft, die einen Assoziationstyp repräsentiert setzt sich also wie folgt zusammen:

Name + OID + Zusatz für die Richtung ("_forward" oder "_backward")

Ein Problem bei der Abbildung von Assoziationstypen und Assoziationen ist, dass es schwierig ist, eine Stelle zu finden, an der man Attribute von Assoziationstypen, die sich nicht mit Hilfe von Beschränkungen abbilden lassen, unterbringen kann. Während Assoziationstypen in der OWL Lite-Ontologie auf Individuen abgebildet sind, welche Eigenschaften haben können, können die Eigenschaften in der OWL DL-Ontologie selbst keine herkömmlichen weiteren Eigenschaften haben. Eine Möglichkeit, die betreffenden Informationen zumindest noch in menschenlesbarer Form in die Ontologie zu integrieren besteht in der Verwendung von `AnnotationProperties`. Da diese

AnnotationProperties nur dazu gedacht sind, Meta-Informationen über die Ontologie zur Verfügung zu stellen und zur eigentlichen Semantik nichts beitragen, wurde die Abbildung auf AnnotationProperties nur exemplarisch für den Rückgabewert der Methode `IAssociationType.getUnit()` durchgeführt:

```
<owl:AnnotationProperty rdf:ID="AssociationType_InUnit"/>

<owl:ObjectProperty
  rdf:ID="hatFarbe8410406129852271022_forward">
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Entity"/>
  <AssociationType_InUnit>UnitMerkmale
</AssociationType_InUnit>
</owl:ObjectProperty>
```

4.2.6.3 Definition von Klassen

Die OWL DL-Ontologie verwendet im Wesentlichen die selben OWL-Klassen wie die OWL Lite-Version.

Die Assertions werden nicht benötigt. Entweder sie fallen im Zuge der Modellierung von Assoziationen als Eigenschaften weg, wie etwa `AssertionLayerLayer` für das `Layer-Constraint`, das jetzt mit Hilfe von `<owl:allValuesFrom>` abgebildet wird oder sie tragen in OWL Lite zur Modellierung von Methoden bei, die in OWL DL als `AnnotationProperties` modelliert werden und fallen auf Grund der Entscheidung nur exemplarisch die `AnnotationProperty AssociationType_InUnit` zu implementieren weg.

Die Definitionen der Subklassen von `Entity` werden um die Beschränkungen der Eigenschaften, die Assoziationstypen modellieren, erweitert. Da für jeden Assoziationstyp in jede `Entity`-Subklasse Beschränkungen eingefügt werden, werden diese Klassendefinitionen mit steigender Zahl von Assoziationstypen dabei sehr schnell sehr groß.

Wie die Beschränkungen in die Klassendefinitionen eingebaut werden, wurde bereits im Abschnitt über die Definition der Eigenschaften aufgezeigt. Hier soll nur noch auf einen besonderen Aspekt eingegangen werden. Es handelt sich dabei um die Abbildung der Java-Klasse `Multiplicity`, die dazu dient einen Zahlbereich, bestehend aus mehreren Intervallen, darzustellen. `Multiplicity` wird zur Angabe der `RateConstraints` verwendet. Das Beispiel zur Darstellung der `RateConstraints` im Abschnitt über die Definition von

Eigenschaften war stark vereinfacht. Auf der nächsten Seite folgt ein tatsächlicher Ausschnitt aus der Ontologie, die aus dem Automobilbeispiel extrahiert wurde.

```

<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource=
              "#hatFarbe8410406129852271022_forward" />
            <owl:minCardinality
              rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource=
              "#hatFarbe8410406129852271022_forward" />
            <owl:maxCardinality
              rdf:datatype="&xsd;nonNegativeInteger">3
            </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource=
              "#hatFarbe8410406129852271022_forward" />
            <owl:minCardinality rdf:datatype=
              "&xsd;nonNegativeInteger">5
            </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource=
              "#hatFarbe8410406129852271022_forward" />
            <owl:maxCardinality rdf:datatype=
              "&xsd;nonNegativeInteger">2147483647
            </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>

```

Es handelt sich bei diesem Beispiel um einen Ausschnitt aus der Klassendefinition von `ClassEntity`. Es wird die Kardinalität der Eigenschaft "hatFarbe" auf die Intervalle 0..3 und 5.. 2147483647 beschränkt. Hierfür werden die Mengenkonstrukte `<owl:intersectionOf>` und `<owl:unionOf>` benutzt. Zunächst wird `<owl:intersectionOf>` benutzt um mit Hilfe von je einer Beschränkung `<owl:minCardinality>` und `<owl:maxCardinality>` ein Intervall zu bilden. Danach wird `<owl:unionOf>` benutzt, um die verschiedenen Intervalle zu einem Wertebereich, bestehend aus mehreren Intervallen zusammenzufassen.

4.2.6.4 Individuen und Instanzen von Eigenschaften

Der Unterschied bei der Definition der Individuen gegenüber der OWL Lite-Version ist, dass die Individuen der OWL-Klasse `Association` wegfallen, da es diese OWL-Klasse nicht mehr gibt. Statt dessen werden Assoziationen in Eigenschaften umgesetzt, von denen Instanzen in die Individuen integriert werden, die Entities modellieren. Das folgende Beispiel zeigt die Definition der Meta-Entity "Motor" aus dem Automobilbeispiel. Man erkennt die Instanzen der Eigenschaften, die die Assoziationen "eingebautIn" und "hatFarbe" modellieren. Sie verweisen auf Individuen #4689577404034775497 und #7068964904715211730 welche die Meta-Entity "Automobil" und die Instance-Entity "schwarz" modellieren.

```

<MetaEntity rdf:ID="-1607458782913708539">
  <Object_HasUIString
    rdf:datatype="&xsd:string">Motor</Object_HasUIString>
  <Object_IsMutable
    rdf:datatype="&xsd:boolean">>true</Object_IsMutable>
  <Entity_HasContent
    rdf:datatype="&xsd:string">Motor</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <eingebautIn2200621423933057199_forward
    rdf:resource="#4689577404034775497"/>
  <hatFarbe8410406129852271022_forward
    rdf:resource="#7068964904715211730"/>
  <MetaEntity_IsAttribute
    rdf:datatype="&xsd:boolean">>false</MetaEntity_IsAttribute>
  <MetaEntity_IsEO
    rdf:datatype="&xsd:boolean">>false</MetaEntity_IsEO>
</MetaEntity>

```

4.2.7 Extraktion der OWL Full-Ontologie

4.2.7.1 Grundsätzliches

Die Grundlage der OWL Full-Ontologie ist die OWL DL-Version. Davon ausgehend werden Änderungen beschrieben, die die besonderen Fähigkeiten von OWL Full ausnutzen.

In OWL Full gibt es keine strikte Trennung zwischen Klassen, Eigenschaften, Individuen und Datenwerten. Für die Extraktion der OWL Full-Ontologie ergeben sich daraus zwei Ansätze für Veränderungen gegenüber OWL DL.

Die Tatsache, dass Klassen auch als Individuen betrachtet werden können, macht es möglich, die "doppelte Instanziierung" der föderal-Informationsarchitektur auf eine tatsächliche Instanziierung abzubilden, indem man Meta-Entities und Class-Entities als OWL-Klassen modelliert und die OWL-Klasse `ClassEntity` dann gleichzeitig als Individuum betrachtet, das der OWL-Klasse `MetaEntity` angehört.

Des Weiteren kann man Eigenschaften als Individuen betrachten. Auf diese Weise wird es möglich, dass Eigenschaften selbst wieder Eigenschaften haben. Bei den Assoziationstypen, die auf Eigenschaften abgebildet werden, ist man somit nicht mehr auf `AnnotationProperties` zur Abbildung ihrer Attribute angewiesen, sondern kann normale `ObjectProperties` bzw. `DatatypeProperties` verwenden, wodurch die Information prinzipiell auch von Reasonern verwertet werden kann.

Im Folgenden werden diese beiden Änderungen erläutert.

4.2.7.2 Doppelte Instanziierung

Abbildung der Entities:

In OWL Full können Klassen auch als Individuen betrachtet werden. Folgender Ausschnitt aus einer an das Automobilbeispiel angelehnten Ontologie zeigt, wie diese Eigenschaft zur Darstellung der Beziehungen zwischen Meta-, Class- und Instance-Entity verwendet werden können. Zur Verbesserung der Anschaulichkeit werden dabei nicht die Werte von `getOID()` als `rdf:IDs` verwendet, sondern die Rückgabewerte von `getContent()`. Die Betrachtung beginnt mit der Darstellung einer Instance-Entity:

```
<owl:Thing rdf:ID="Herbie">
  <rdf:type rdf:resource="#VW-Käfer"/>
</owl:Thing>
```

InstanceEntities werden auch in OWL Full auf Individuen abgebildet. Um die Zugehörigkeit zu einer Klasse zu definieren, wird in diesem Fall kein Tag mit dem Namen der Klasse verwendet, sondern das `rdf`-Tag `rdf:type`. Semantisch sind diese beiden Syntaxvarianten äquivalent. Genauso gut hätte man die Zugehörigkeit von `Herbie` zur OWL-Klasse `VW-Käfer` auch folgendermaßen ausdrücken können:

```
<VW-Käfer rdf:ID="Herbie">
</VW-Käfer>
```

Aus Konsistenzgründen mit der Abbildung von Class- und Meta-Entities, bei denen die Verwendung von `rdf:type` unumgänglich ist, wird bei Instance-Entities ebenfalls `rdf:type` verwendet.

Man beachte, dass `Herbie` nun nicht mehr mittels einer Eigenschaft `AssociatedToClass` mit der zugehörigen Class-Entity verbunden ist, sondern dass es sich bei `Herbie` nunmehr tatsächlich um eine Instanz der Klasse `VW-Käfer` handelt.

Die OWL-Klasse `VW-Käfer`, welche die entsprechende Class-Entity modelliert, sieht folgendermaßen aus:

```

<owl:Class rdf:ID="VW-Käfer">
  <rdfs:subClassOf rdf:resource="#ClassEntity"/>
  <rdf:type rdf:resource="#Automobil"/>
</owl:Class>

```

Man sieht, dass die Class-Entity VW-Käfer im Gegensatz zu den Lite- und DL-Ontologien auf eine OWL-Klasse abgebildet wird. `<rdfs:subClassOf>` betrachtet VW-Käfer als Klasse und bildet eine Subklasse. `<rdf:type>` dagegen betrachtet VW-Käfer als Individuum und stellt eine Instanziierungsbeziehung her.

Instance-Entities, die zur Class-Entity VW-Käfer gehören, werden auf Individuen dieser OWL-Klasse abgebildet. Dies wäre auch schon in OWL Lite und OWL DL möglich gewesen. Da man in OWL Full OWL-Klassen gleichzeitig auch als Individuen betrachten kann, ist es nun aber auch möglich, die OWL-Klasse VW-Käfer gleichzeitig als Individuum zu betrachten, das selbst wieder einer übergeordneten OWL-Klasse Automobil angehört, welche eine Meta-Entity modelliert. Auf diese Weise wird die "doppelte Instanziierung" aus der föderal-Informationsarchitektur in OWL Full tatsächlich auf eine Instanziierung abgebildet.

Als letztes folgt nun noch die Definition Meta-Entity Automobil:

```

<owl:Class rdf:ID="Automobil">
  <rdfs:subClassOf rdf:resource="#MetaEntity"/>
  <rdf:type rdf:resource="#SuperMetaEntity"/>
</owl:Class>

```

Die Definition dieser Meta-Entity ähnelt stark der vorherigen Class-Entity VW-Käfer, was auf Grund der Übereinstimmungen zwischen dem Übergang von der Meta-Entity zur Class-Entity und dem Übergang von der Class-Entity zur Instance-Entity auch nicht weiter verwunderlich ist. Auffällig ist, dass die OWL-Klasse Automobil gleichzeitig als Individuum einer weiteren OWL-Klasse SuperMetaEntity definiert wird. Worum es sich bei dieser OWL-Klasse handelt und wozu sie benötigt wird, wird im nächsten Abschnitt über die Abbildung der Constraints erläutert.

Abbildung der Constraints:

In der OWL DL-Ontologie werden alle Entities auf Individuen abgebildet. Instance-Entities werden auf Individuen der OWL-Klasse `InstanceEntity` abgebildet, Class-Entities auf Individuen von `ClassEntity` und Meta-Entities auf Individuen von `MetaEntity`.

Um die Constraints abzubilden, werden dann in die Definition dieser OWL-Klassen Beschränkungen eingefügt. Um zum Beispiel auszudrücken, dass Instance-Entities mittels Assoziationen vom Typ "eingebautIn" nur mit anderen Instance-Entities verbunden werden dürfen, fügt man in die Definition der OWL-Klasse `InstanceEntity` folgende Beschränkung ein (Zur besseren Lesbarkeit wurde von den Namenskonventionen abgewichen.):

```
<owl:Class rdf:ID="InstanceEntity">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn" />
      <owl:allValuesFrom rdf:resource="#InstanceEntity" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

In OWL Full sind die Individuen, die Instance-Entities abbilden keine Instanzen einer OWL-Klasse `InstanceEntity` sondern der OWL-Klasse, die die jeweilige übergeordnete `ClassEntity` modelliert. Diese OWL-Klassen, die Class-Entities modellieren, sind ihrerseits Subklassen der OWL-Klasse `ClassEntity`. Das bedeutet, dass Individuen, die Instance-Entities modellieren Individuen von `ClassEntity` sind und nicht von `InstanceEntity`. Demzufolge müssen Beschränkungen, die Constraints der Instance-Schicht modellieren, nicht in der OWL-Klasse `InstanceEntity` sondern in der OWL-Klasse `ClassEntity` definiert werden.

Derselbe Effekt tritt auch eine Abstraktionsebene höher bei den Constraints der Class-Schicht auf. Die als Individuen betrachteten OWL-Klassen, die Class-Entities abbilden sind selbst Individuen von `MetaEntity`, weshalb Beschränkungen, die Constraints der Class-Schicht modellieren, in `MetaEntity` definiert werden müssen.

Analog zum "Hochrutschen" der Beschränkungen zur Abbildung der Constraints der anderen Schichten, müssen nun auch die Beschränkungen für die Abbildung der Constraints der Meta-Schicht in die Definition der Entity der übergeordneten Abstraktionsebene verlegt werden. In

der föderal-Informationsarchitektur gibt es jedoch keine Abstraktionseben oberhalb der Meta-Schicht. Um einen Platz für die Beschränkungen der Meta-Schicht zu haben, wurde in der OWL Full-Ontologie darum die OWL-Klasse `SuperMetaEntity` eingeführt. Diese Klasse dient einzig und allein dazu, technisch die Beschränkungen für die Meta-Schicht aufzunehmen. Alle OWL-Klassen, die Meta-Entities modellieren, werden, als Individuen betrachtet, mittels `rdf:type` als `SuperMetaEntity` zugehörig definiert, sodass die dort definierten Beschränkungen für sie gelten.

4.2.7.3 Attribute von Assoziationstypen

Da in OWL Full auch Eigenschaften als Individuen betrachtet werden können, ist es möglich, dass Eigenschaften selbst wieder Eigenschaften haben können. Somit lassen sich die Attribute von Assoziationstypen, die auf Eigenschaften abgebildet werden, mit Hilfe von `ObjectProperties` und `DatatypeProperties` darstellen. Das Prinzip wird leicht anhand der Unit-Zugehörigkeit von Assoziatinstypen klar:

```
<owl:ObjectProperty rdf:ID="AssociationType_InUnit">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssociationType"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>

<owl:ObjectProperty
  rdf:ID="eingebautIn-7175480290805356222_forward">
  <rdfs:subPropertyOf rdf:resource="#AssociationType"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Entity"/>
  <AssociationType_InUnit rdf:resource="#UnitFahrzeuge"/>
</owl:ObjectProperty>
```

5 Fazit

Im Rahmen dieser Arbeit hat sich gezeigt, dass semantische Netze der föderal-
Informationsarchitektur sich prinzipiell mit Hilfe von OWL beschreiben lassen.

Unterschiede ergeben sich aus dem Zweck, dem beide Technologien dienen. In semantischen
Netzen der föderal-Informationsarchitektur sind alle relevanten Informationen explizit
modelliert. Der Hauptzweck ist, die Konsistenz des Netzes sicherzustellen. Ein Hauptzweck
von OWL ist dagegen, Informationen, die nicht explizit modelliert sind, automatisch herleiten
zu können.

Für viele Konstrukte, die in der föderal-Informationsarchitektur verwendet werden, lassen
sich in OWL Entsprechungen finden. Dies gilt vor allem für die Grundbestandteile des
Netzes, für Knoten und Kanten, die sich in OWL auf Klassen und Individuen sowie
Eigenschaften und deren Instanzen abbilden lassen.

Darüber hinaus gibt es zum Beispiel im Fall der RateConstraints der föderal-
Informationsarchitektur und der Beschränkung von Kardinalitäten von Eigenschaften in OWL
ähnliche Konzepte, die aber nicht die selbe Bedeutung haben. So stellen etwa die
RateConstraints aus der föderal-Informationsarchitektur sicher, dass sich die Zahl bestimmter
Assoziationen in einem gewissen Bereich bewegt, wohingegen eine
Kardinalitätsbeschränkung in OWL diese nicht sicher stellt, sondern dazu führt, dass bei
einem Verlassen des vorgegebenen Bereichs möglicherweise falsche Schlüsse über die
Klassenzugehörigkeit eines Individuums gezogen werden.

Für andere Konstrukte, wie etwa das DepthConstraint, das in der föderal-
Informationsarchitektur festlegt, welche Tiefe ein aus Assoziationen eines bestimmten Typs
aufgebauter Baum haben darf, gibt es in OWL gar keine Entsprechung. Derartige
Informationen können nur in menschenlesbarer Form in OWL-Ontologien integriert werden.

Die folgende Tabelle stellt die wesentlichen Unterschiede zwischen den verschiedenen OWL-
Versionen bei der Extraktion schematisch dar.

	OWL Lite	OWL DL	OWL Full
Umsetzung von Entities	mittels Individuen	mittels Individuen	mittels Klassen, die gleichzeitig als Individuen betrachtet werden
Umsetzung von Assoziationstypen	mittels Individuen	mittels Eigenschaften	mittels Eigenschaften
Umsetzung von Assoziationen	mittels Individuen	mittels Instanzen von Eigenschaften	mittels Instanzen von Eigenschaften
Umsetzung von RangeConstraints	menschenlesbar mittels Assertion	mittels <owl:minCardinality> und <owl:maxCardinality>	mittels <owl:minCardinality> und <owl:maxCardinality>
Umsetzung von LayerConstraints	menschenlesbar mittels Assertion	mittels <owl:allValuesFrom>	mittels <owl:allValuesFrom>
Umsetzung von CopyPropagation	menschenlesbar mittels Assertion	menschenlesbar mittels AnnotationProperty	menschenlesbar mittels ObjectProperty
Umsetzung von RemovePropagation	menschenlesbar mittels Assertion	menschenlesbar mittels AnnotationProperty	menschenlesbar mittels ObjectProperty
Umsetzung von OrderConstraint	menschenlesbar mittels Assertion	menschenlesbar mittels AnnotationProperty	menschenlesbar mittels DataTypeProperty
Umsetzung von NonCyclicConstraint	menschenlesbar mittels Assertion	menschenlesbar mittels AnnotationProperty	menschenlesbar mittels DataTypeProperty
Umsetzung von DepthConstraint	menschenlesbar mittels Assertion	menschenlesbar mittels AnnotationProperty	menschenlesbar mittels DataTypeProperty
Größe des OWL-Dokuments bei Extraktion des Automobilbeispiels	57 kB	25 kB	39 kB

6 Abbildungsverzeichnis

Abb. 1: Beispielausschnitt eines semantischen Netzes der föderal-Informationsarchitektur ..	10
Abb. 2: Beispiel für die Zuordnung von Instance-Entities zu Class-Entities.....	11
Abb. 3: Beispiel für die Zuordnung von Class-Entities zu Meta-Entities.....	12
Abb. 4: Schichtstruktur der föderal-Informationsarchitektur.....	14
Abb. 5: Vererbungsbeziehungen zwischen Schnittstellen der "mind"-Schicht	15
Abb. 6: Zusammenspiel der Schnittstellen bei der Verbindung zweier Entities.....	18
Abb. 7: "Automobilbeispiel"	51
Abb. 8: Modellierung von Methoden mit Parameter mittels Assertions.....	64
Abb. 9: Vererbungsbeziehungen zwischen Eigenschaften, die Ableitungsbeziehungen von Entities darstellen	72
Abb.10 : Zusammenspiel verschiedener Eigenschaften bei der Darstellung von Ableitungsbeziehungen zwischen Entities	73

7 Literaturverzeichnis

- [1] McGuinness, Deborah L.; van Harmelen, Frank (Hrsg.): OWL Web Ontology Language Overview (29.12.2003) <http://www.w3.org/TR/2003/CR-owl-features-20030818/>
- [2] Smith, Michael K.; Welty, Chris; McGuinness, Deborah L. (Hrsg.): OWL Web Ontology Language Guide (29.12.2003) <http://www.w3.org/TR/2003/CR-owl-guide-20030818/>
- [3] Bechhofer, Sean; van Harmelen, Frank; Hendler, Jim; Horrocks, Ian; McGuinness, Deborah L.; Patel-Schneider, Peter F.; Stein, Lynn Andrea: OWL Web Ontology Language Reference (29.12.2003) <http://www.w3.org/TR/2003/CR-owl-ref-20030818/>
- [4] Homepage der föderal-Informationsarchitektur (11.7.2004) <http://www.foederal.de>
- [5] ISO/IEC Informationstechnik – Begriffe – Teil 15: ISO2382-15 (1998)
- [6] Winskel, Glynn: The Formal Semantics of Programming Languages. Cambridge, London: MIT Press 1993
- [7] The Semantic Web Community Portal (11.7.2004) <http://www.semanticweb.org>
- [8] Connolly, Dan; van Harmelen, Frank; Horrocks, Ian; McGuinness, Deborah L.; Patel-Schneider, Peter F.; Stein, Lynn Andrea: DAML+OIL (March 2001) Reference Description (11.7.2004) <http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218>
- [9] The DARPA Agent Markup Language Homepage (11.7.2004) <http://www.daml.org/>
- [10] Oil Homepage (11.7.2004) <http://oil.semanticweb.org/>
- [11] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C.M.; Maler, Eve; Yergeau, François; Cowan, John (Hrsg.): Extensible Markup Language (XML) 1.1 (11.7.2004) <http://www.w3.org/TR/2004/REC-xml11-20040204/>
- [12] Fallside, David C. (Hrsg.): XML Schema Part 0: Primer (11.7.2004) <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>
- [13] Klyne, Graham; Carroll, Jeremy J. (Hrsg.): Resource Description Framework (RDF): Concepts and Abstract Syntax (29.12.2003) <http://www.w3.org/TR/2003/PR-rdf-concepts-20031215/>
- [14] Brickley, Dan; Guha, R.V. (Hrsg.): RDF Vocabulary Description Language 1.0: RDF Schema (29.12.2003) <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>
- [15] Baader, Franz; Calvanese, Diego; McGuinness, Deborah; Nardi, Daniele; Patel-Schneider, Peter (Hrsg.): The Description Logic Handbook. Cambridge: University Press 2003

8 Anhang

8.A Aus dem Automobilbeispiel extrahierte OWL Lite-Ontologie

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
]>

<rdf:RDF
  xml:base = "http://www.MarcEichler.de/owl/AutomobilLite#"
  xmlns: = "http://www.MarcEichler.de/owl/AutomobilLite#"
  xmlns:owl = "&owl;"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "&xsd;"
>

<owl:Class rdf:ID="Unidirection">
</owl:Class>

<owl:Class rdf:ID="AssertionUnidirectionMultiplicity">
</owl:Class>

<owl:ObjectProperty rdf:ID="AssertionUnidirectionMultiplicity_HasUnidirection">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionMultiplicity"/>
  <rdfs:range rdf:resource="#Unidirection"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssertionUnidirectionMultiplicity_HasMultiplicity">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionMultiplicity"/>
  <rdfs:range rdf:resource="#Multiplicity"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="AssertionUnidirectionInteger">
</owl:Class>

<owl:ObjectProperty rdf:ID="AssertionUnidirectionInteger_HasUnidirection">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionInteger"/>
  <rdfs:range rdf:resource="#Unidirection"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="AssertionUnidirectionInteger_HasInteger">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionInteger"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="AssertionUnidirectionEntity">
</owl:Class>

<owl:ObjectProperty rdf:ID="AssertionUnidirectionEntity_HasUnidirection">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionEntity"/>
  <rdfs:range rdf:resource="#Unidirection"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssertionUnidirectionEntity_HasEntity">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionEntity"/>
  <rdfs:range rdf:resource="#Entity"/>
</owl:ObjectProperty>
```

```

<owl:Class rdf:ID="AssertionUnidirectionString">
</owl:Class>

<owl:ObjectProperty rdf:ID="AssertionUnidirectionString_HasUnidirection">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionString"/>
  <rdfs:range rdf:resource="#Unidirection"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="AssertionUnidirectionString_HasString">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionUnidirectionString"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="AssertionLayerLayer">
</owl:Class>

<owl:ObjectProperty rdf:ID="AssertionLayerLayer_HasSource">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionLayerLayer"/>
  <rdfs:range rdf:resource="#Layer"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssertionLayerLayer_HasDrain">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionLayerLayer"/>
  <rdfs:range rdf:resource="#Layer"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="AssertionLayerNetconstraints">
</owl:Class>

<owl:ObjectProperty rdf:ID="AssertionLayerNetconstraints_HasLayer">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionLayerNetconstraints"/>
  <rdfs:range rdf:resource="#Layer"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssertionLayerNetconstraints_HasNetconstraints">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssertionLayerNetconstraints"/>
  <rdfs:range rdf:resource="#Netconstraints"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="MultiplicityIntervall">
</owl:Class>

<owl:DatatypeProperty rdf:ID="MultiplicityIntervall_HasLowerBound">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#MultiplicityIntervall"/>
  <rdfs:range rdf:resource="&xsd:integer"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MultiplicityIntervall_HasUpperBound">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#MultiplicityIntervall"/>
  <rdfs:range rdf:resource="&xsd:integer"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Multiplicity">
</owl:Class>

<owl:ObjectProperty rdf:ID="Multiplicity_IncludesIntervall">
  <rdfs:domain rdf:resource="#Multiplicity"/>
  <rdfs:range rdf:resource="#MultiplicityIntervall"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Unit">
</owl:Class>

<owl:DatatypeProperty rdf:ID="Unit_IsReadOnly">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>

```

```

    <rdfs:domain rdf:resource="#Unit"/>
    <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="Unit_IsMutable">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Unit_HasSuperunit">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Unit_HasSubunit">
  <owl:inverseOf rdf:resource="#Unit_HasSuperunit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Unit_ContainsEntity">
  <owl:inverseOf rdf:resource="#Entity_InUnit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Unit_ContainsAssociationtype">
  <owl:inverseOf rdf:resource="#AssociationType_InUnit"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Layer">
</owl:Class>

<owl:Class rdf:ID="Object">
</owl:Class>

<owl:DatatypeProperty rdf:ID="Object_HasUIString">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Object"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="Object_IsMutable">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Object"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Entity">
  <rdfs:subClassOf rdf:resource="#Object"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="Entity_HasContent">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Entity_InLayer">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Layer"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Entity_InUnit">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Entity_AssociatedToSuper">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Entity"/>
</owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:ID="Entity_AssociatedToSub">
  <owl:inverseOf rdf:resource="#Entity_AssociatedToSuper"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="MetaEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsAttribute">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#MetaEntity"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsEO">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#MetaEntity"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_HasEOName">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#MetaEntity"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsAbstract">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#MetaEntity"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_HasAttributetype">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#MetaEntity"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="MetaEntity_AssociatedToClass">
  <rdfs:subPropertyOf rdf:resource="#Entity_AssociatedToSub"/>
  <owl:inverseOf rdf:resource="#ClassEntity_AssociatedToMeta"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="ClassEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="ClassEntity_AssociatedToMeta">
  <rdfs:subPropertyOf rdf:resource="#Entity_AssociatedToSuper"/>
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#ClassEntity"/>
  <rdfs:range rdf:resource="#MetaEntity"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="ClassEntity_AssociatedToInstance">
  <rdfs:subPropertyOf rdf:resource="#Entity_AssociatedToSub"/>
  <owl:inverseOf rdf:resource="#InstanceEntity_AssociatedToClass"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="InstanceEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="InstanceEntity_AssociatedToClass">
  <rdfs:subPropertyOf rdf:resource="#Entity_AssociatedToSuper"/>
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#InstanceEntity"/>
  <rdfs:range rdf:resource="#ClassEntity"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Association">
  <rdfs:subClassOf rdf:resource="#Object"/>
</owl:Class>

```

```

<owl:ObjectProperty rdf:ID="Association_HasEntityAssertion">
  <rdfs:domain rdf:resource="#Association"/>
  <rdfs:range rdf:resource="#AssertionUnidirectionEntity"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="Association_HasIndex">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Association"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Association_HasType">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Association"/>
  <rdfs:range rdf:resource="#AssociationType"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="MetaAssociation">
  <rdfs:subClassOf rdf:resource="#Association"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="MetaAssociation_HasRoleAssertion">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#MetaAssociation"/>
  <rdfs:range rdf:resource="#AssertionUnidirectionString"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="AssociationType">
  <rdfs:subClassOf rdf:resource="#Object"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="AssociationType_HasName">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssociationType"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="AssociationType_HasNetconstraintsAssertion">
  <rdfs:domain rdf:resource="#AssociationType"/>
  <rdfs:range rdf:resource="#AssertionLayerNetconstraints"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssociationType_HasLayerAssertion">
  <rdfs:domain rdf:resource="#AssociationType"/>
  <rdfs:range rdf:resource="#AssertionLayerLayer"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssociationType_InUnit">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AssociationType"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssociationType_TypeOf">
  <owl:inverseOf rdf:resource="#Association_HasType"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Netconstraints">
</owl:Class>

<owl:DatatypeProperty rdf:ID="Netconstraints_HasNoncyclicConstraint">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Netconstraints"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Netconstraints_HasDepthConstraint">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Netconstraints"/>
  <rdfs:range rdf:resource="#Multiplicity"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="Netconstraints_HasOrderConstraint">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>

```

```

    <rdfs:domain rdf:resource="#Netconstraints"/>
    <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Netconstraints_HasRateAssertion">
  <rdfs:domain rdf:resource="#Netconstraints"/>
  <rdfs:range rdf:resource="#AssertionUnidirectionMultiplicity"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Netconstraints_DefinesCopypropagation">
  <rdfs:domain rdf:resource="#Netconstraints"/>
  <rdfs:range rdf:resource="#AssertionUnidirectionInteger"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Netconstraints_DefinesRemovepropagation">
  <rdfs:domain rdf:resource="#Netconstraints"/>
  <rdfs:range rdf:resource="#AssertionUnidirectionInteger"/>
</owl:ObjectProperty>

<Unidirection rdf:ID="forward">
</Unidirection>

<Unidirection rdf:ID="backward">
</Unidirection>

<Layer rdf:ID="I">
</Layer>

<Layer rdf:ID="C">
</Layer>

<Layer rdf:ID="M">
</Layer>

<Layer rdf:ID="any_layer">
</Layer>

<Layer rdf:ID="no_layer">
</Layer>

<Association rdf:ID="-7323715489535824728;3530475817318219628;-3928075398917443468">
  <Association_HasIndex rdf:datatype="&xsd;integer">0</Association_HasIndex>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#forward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-3928075398917443468"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#backward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-7323715489535824728"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasType rdf:resource="#3530475817318219628"/>
</Association>

<Association rdf:ID="-4063242352671119412;3530475817318219628;-5406144385051450058">
  <Association_HasIndex rdf:datatype="&xsd;integer">0</Association_HasIndex>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#forward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-5406144385051450058"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#backward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-4063242352671119412"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasType rdf:resource="#3530475817318219628"/>
</Association>

```

```

<Association rdf:ID="-1633451223456955100;3530475817318219628;6450843075075210605">
  <Association_HasIndex rdf:datatype="&xsd;integer">0</Association_HasIndex>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#forward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#6450843075075210605"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#backward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-1633451223456955100"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasType rdf:resource="#3530475817318219628"/>
</Association>

<AssociationType rdf:ID="3530475817318219628">
  <Object_HasUIString rdf:datatype="&xsd:string">-6570387664937708759%eingebautIn$
</Object_HasUIString>
<Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
<AssociationType_HasName rdf:datatype="&xsd:string">eingebautIn</AssociationType_HasName>
<AssociationType_HasNetconstraintsAssertion>
  <AssertionLayerNetconstraints>
    <AssertionLayerNetconstraints_HasLayer rdf:resource="#I"/>
    <AssertionLayerNetconstraints_HasNetconstraints>
      <Netconstraints>
        <Netconstraints_HasNoncyclicConstraint rdf:datatype="&xsd:boolean">>false
        </Netconstraints_HasNoncyclicConstraint>
        <Netconstraints_HasDepthConstraint>
          <Multiplicity>
            <Multiplicity_IncludesIntervall>
              <MultiplicityIntervall>
                <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
                </MultiplicityIntervall_HasLowerBound>
                <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">2147483647
                </MultiplicityIntervall_HasUpperBound>
              </MultiplicityIntervall>
            </Multiplicity_IncludesIntervall>
          </Multiplicity>
        </Netconstraints_HasDepthConstraint>
        <Netconstraints_HasOrderConstraint rdf:datatype="&xsd;integer">41
        </Netconstraints_HasOrderConstraint>
        <Netconstraints_HasRateAssertion>
          <AssertionUnidirectionMultiplicity>
            <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#forward"/>
            <AssertionUnidirectionMultiplicity_HasMultiplicity>
              <Multiplicity>
                <Multiplicity_IncludesIntervall>
                  <MultiplicityIntervall>
                    <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
                    </MultiplicityIntervall_HasLowerBound>
                    <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">0
                    </MultiplicityIntervall_HasUpperBound>
                  </MultiplicityIntervall>
                </Multiplicity_IncludesIntervall>
              </Multiplicity>
            </AssertionUnidirectionMultiplicity_HasMultiplicity>
          </AssertionUnidirectionMultiplicity>
        </Netconstraints_HasRateAssertion>
        <Netconstraints_HasRateAssertion>
          <AssertionUnidirectionMultiplicity>
            <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#backward"/>
            <AssertionUnidirectionMultiplicity_HasMultiplicity>
              <Multiplicity>
                <Multiplicity_IncludesIntervall>

```

```

        <MultiplicityIntervall>
          <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
        </MultiplicityIntervall_HasLowerBound>
          <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">
            2147483647</MultiplicityIntervall_HasUpperBound>
        </MultiplicityIntervall>
      </Multiplicity_IncludesIntervall>
    </Multiplicity>
  </AssertionUnidirectionMultiplicity_HasMultiplicity>
</AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_DefinesCopypropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesCopypropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesRemovepropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
<Netconstraints_DefinesRemovepropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
</Netconstraints>
</AssertionLayerNetconstraints_HasNetconstraints>
</AssertionLayerNetconstraints>
</AssociationType_HasNetconstraintsAssertion>
<AssociationType_HasNetconstraintsAssertion>
  <AssertionLayerNetconstraints>
    <AssertionLayerNetconstraints_HasLayer rdf:resource="#C"/>
    <AssertionLayerNetconstraints_HasNetconstraints>
      <Netconstraints>
        <Netconstraints_HasNoncyclicConstraint rdf:datatype="&xsd;boolean">>false
      </Netconstraints_HasNoncyclicConstraint>
        <Netconstraints_HasDepthConstraint>
          <Multiplicity>
            <Multiplicity_IncludesIntervall>
              <MultiplicityIntervall>
                <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
              </MultiplicityIntervall_HasLowerBound>
                <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">2147483647
              </MultiplicityIntervall_HasUpperBound>
            </MultiplicityIntervall>
          </Multiplicity_IncludesIntervall>
        </Multiplicity>
      </Netconstraints_HasDepthConstraint>
        <Netconstraints_HasOrderConstraint rdf:datatype="&xsd;integer">41
      </Netconstraints_HasOrderConstraint>
        <Netconstraints_HasRateAssertion>
          <AssertionUnidirectionMultiplicity>
            <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#forward"/>
            <AssertionUnidirectionMultiplicity_HasMultiplicity>
              <Multiplicity>
                <Multiplicity_IncludesIntervall>
                  <MultiplicityIntervall>
                    <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
                  </MultiplicityIntervall_HasLowerBound>
                    <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">0
                  </MultiplicityIntervall_HasUpperBound>
                </MultiplicityIntervall>
              </Multiplicity>
            </AssertionUnidirectionMultiplicity_HasMultiplicity>
          </AssertionUnidirectionMultiplicity>
        </Netconstraints_HasRateAssertion>
      </Netconstraints>
    </AssertionLayerNetconstraints_HasNetconstraints>
  </AssertionLayerNetconstraints>
</AssociationType_HasNetconstraintsAssertion>

```

```

        </MultiplicityIntervall_HasUpperBound>
    </MultiplicityIntervall>
</Multiplicity_IncludesIntervall>
<Multiplicity_IncludesIntervall>
    <MultiplicityIntervall>
        <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">1
    </MultiplicityIntervall_HasLowerBound>
        <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">1
    </MultiplicityIntervall_HasUpperBound>
    </MultiplicityIntervall>
</Multiplicity_IncludesIntervall>
</Multiplicity>
</AssertionUnidirectionMultiplicity_HasMultiplicity>
</AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_HasRateAssertion>
    <AssertionUnidirectionMultiplicity>
        <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#backward"/>
        <AssertionUnidirectionMultiplicity_HasMultiplicity>
            <Multiplicity>
                <Multiplicity_IncludesIntervall>
                    <MultiplicityIntervall>
                        <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
                    </MultiplicityIntervall_HasLowerBound>
                        <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">
                            2147483647</MultiplicityIntervall_HasUpperBound>
                    </MultiplicityIntervall>
                </Multiplicity_IncludesIntervall>
            </Multiplicity>
        </AssertionUnidirectionMultiplicity_HasMultiplicity>
    </AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_DefinesCopypropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
    </AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesCopypropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
    </AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesRemovepropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
    </AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
<Netconstraints_DefinesRemovepropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
    </AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
</Netconstraints>
</AssertionLayerNetconstraints_HasNetconstraints>
</AssertionLayerNetconstraints>
</AssociationType_HasNetconstraintsAssertion>
<AssociationType_HasNetconstraintsAssertion>
    <AssertionLayerNetconstraints>
        <AssertionLayerNetconstraints_HasLayer rdf:resource="#M"/>
        <AssertionLayerNetconstraints_HasNetconstraints>
            <Netconstraints>
                <Netconstraints_HasNoncyclicConstraint rdf:datatype="&xsd;boolean">false
            </Netconstraints_HasNoncyclicConstraint>
                <Netconstraints_HasDepthConstraint>
                    <Multiplicity>
                        <Multiplicity_IncludesIntervall>

```

```

    <MultiplicityIntervall>
      <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
    </MultiplicityIntervall_HasLowerBound>
      <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">2147483647
    </MultiplicityIntervall_HasUpperBound>
    </MultiplicityIntervall>
  </Multiplicity_IncludesIntervall>
</Multiplicity>
</Netconstraints_HasDepthConstraint>
<Netconstraints_HasOrderConstraint rdf:datatype="&xsd;integer">41
</Netconstraints_HasOrderConstraint>
<Netconstraints_HasRateAssertion>
  <AssertionUnidirectionMultiplicity>
    <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#forward"/>
    <AssertionUnidirectionMultiplicity_HasMultiplicity>
      <Multiplicity>
        <Multiplicity_IncludesIntervall>
          <MultiplicityIntervall>
            <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
          </MultiplicityIntervall_HasLowerBound>
            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">0
          </MultiplicityIntervall_HasUpperBound>
          </MultiplicityIntervall>
        </Multiplicity_IncludesIntervall>
        <Multiplicity_IncludesIntervall>
          <MultiplicityIntervall>
            <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">1
          </MultiplicityIntervall_HasLowerBound>
            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">1
          </MultiplicityIntervall_HasUpperBound>
          </MultiplicityIntervall>
        </Multiplicity_IncludesIntervall>
      </Multiplicity>
    </AssertionUnidirectionMultiplicity_HasMultiplicity>
  </AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_HasRateAssertion>
  <AssertionUnidirectionMultiplicity>
    <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#backward"/>
    <AssertionUnidirectionMultiplicity_HasMultiplicity>
      <Multiplicity>
        <Multiplicity_IncludesIntervall>
          <MultiplicityIntervall>
            <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
          </MultiplicityIntervall_HasLowerBound>
            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">
              2147483647</MultiplicityIntervall_HasUpperBound>
          </MultiplicityIntervall>
        </Multiplicity_IncludesIntervall>
      </Multiplicity>
    </AssertionUnidirectionMultiplicity_HasMultiplicity>
  </AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_DefinesCopypropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesCopypropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesRemovepropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>

```

```

<Netconstraints_DefinesRemovepropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
</Netconstraints>
</AssertionLayerNetconstraints_HasNetconstraints>
</AssertionLayerNetconstraints>
</AssociationType_HasNetconstraintsAssertion>
<AssociationType_HasLayerAssertion>
  <AssertionLayerLayer>
    <AssertionLayerLayer_HasSource rdf:resource="#I"/>
    <AssertionLayerLayer_HasDrain rdf:resource="#I"/>
  </AssertionLayerLayer>
</AssociationType_HasLayerAssertion>
<AssociationType_HasLayerAssertion>
  <AssertionLayerLayer>
    <AssertionLayerLayer_HasSource rdf:resource="#C"/>
    <AssertionLayerLayer_HasDrain rdf:resource="#C"/>
  </AssertionLayerLayer>
</AssociationType_HasLayerAssertion>
<AssociationType_HasLayerAssertion>
  <AssertionLayerLayer>
    <AssertionLayerLayer_HasSource rdf:resource="#M"/>
    <AssertionLayerLayer_HasDrain rdf:resource="#M"/>
  </AssertionLayerLayer>
</AssociationType_HasLayerAssertion>
<AssociationType_InUnit rdf:resource="#UnitFahrzeuge"/>
</AssociationType>

<InstanceEntity rdf:ID="-7323715489535824728">
  <Object_HasUIString rdf:datatype="&xsd:string">MotorXY</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">MotorXY</Entity_HasContent>
  <Entity_InLayer rdf:resource="#I"/>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <InstanceEntity_AssociatedToClass rdf:resource="#-4063242352671119412"/>
</InstanceEntity>

<InstanceEntity rdf:ID="-3928075398917443468">
  <Object_HasUIString rdf:datatype="&xsd:string">Herbie</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Herbie</Entity_HasContent>
  <Entity_InLayer rdf:resource="#I"/>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <InstanceEntity_AssociatedToClass rdf:resource="#-5406144385051450058"/>
</InstanceEntity>

<ClassEntity rdf:ID="-5406144385051450058">
  <Object_HasUIString rdf:datatype="&xsd:string">VW-K fer</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">VW-K fer</Entity_HasContent>
  <Entity_InLayer rdf:resource="#C"/>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <ClassEntity_AssociatedToMeta rdf:resource="#6450843075075210605"/>
</ClassEntity>

<ClassEntity rdf:ID="-4063242352671119412">
  <Object_HasUIString rdf:datatype="&xsd:string">K fer-Motor</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">K fer-Motor</Entity_HasContent>
  <Entity_InLayer rdf:resource="#C"/>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <ClassEntity_AssociatedToMeta rdf:resource="#-1633451223456955100"/>
</ClassEntity>

<MetaEntity rdf:ID="6450843075075210605">
  <Object_HasUIString rdf:datatype="&xsd:string">Automobil</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Automobil</Entity_HasContent>
  <Entity_InLayer rdf:resource="#M"/>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>

```

```

    <MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">false</MetaEntity_IsAttribute>
    <MetaEntity_IsEO rdf:datatype="&xsd:boolean">false</MetaEntity_IsEO>
</MetaEntity>

<MetaEntity rdf:ID="-1633451223456955100">
  <Object_HasUIString rdf:datatype="&xsd:string">Motor</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Motor</Entity_HasContent>
  <Entity_InLayer rdf:resource="#M"/>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">false</MetaEntity_IsAttribute>
  <MetaEntity_IsEO rdf:datatype="&xsd:boolean">false</MetaEntity_IsEO>
</MetaEntity>

<Unit rdf:ID="UnitFahrzeuge">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">true</Unit_IsMutable>
  <Unit_HasSuperunit rdf:resource="#rootUnit"/>
</Unit>

<Association rdf:ID="-3928075398917443468;-1072328004570732921;4873894939970285764">
  <Association_HasIndex rdf:datatype="&xsd:integer">0</Association_HasIndex>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#forward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#4873894939970285764"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#backward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-3928075398917443468"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasType rdf:resource="#-1072328004570732921"/>
</Association>

<Association rdf:ID="-3928075398917443468;-1072328004570732921;-7539133666533682201">
  <Association_HasIndex rdf:datatype="&xsd:integer">1</Association_HasIndex>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#forward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-7539133666533682201"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#backward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-3928075398917443468"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasType rdf:resource="#-1072328004570732921"/>
</Association>

<Association rdf:ID="-1633451223456955100;-1072328004570732921;4873894939970285764">
  <Association_HasIndex rdf:datatype="&xsd:integer">0</Association_HasIndex>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#forward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#4873894939970285764"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasEntityAssertion>
    <AssertionUnidirectionEntity>
      <AssertionUnidirectionEntity_HasUnidirection rdf:resource="#backward"/>
      <AssertionUnidirectionEntity_HasEntity rdf:resource="#-1633451223456955100"/>
    </AssertionUnidirectionEntity>
  </Association_HasEntityAssertion>
  <Association_HasType rdf:resource="#-1072328004570732921"/>
</Association>

<AssociationType rdf:ID="-1072328004570732921">
  <Object_HasUIString rdf:datatype="&xsd:string">6524238825941876821%hatFarbe$
</Object_HasUIString>

```

```

<Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
<AssociationType_HasName rdf:datatype="&xsd:string">hatFarbe</AssociationType_HasName>
<AssociationType_HasNetconstraintsAssertion>
  <AssertionLayerNetconstraints>
    <AssertionLayerNetconstraints_HasLayer rdf:resource="#I"/>
    <AssertionLayerNetconstraints_HasNetconstraints>
      <Netconstraints>
        <Netconstraints_HasNoncyclicConstraint rdf:datatype="&xsd:boolean">>false
        </Netconstraints_HasNoncyclicConstraint>
        <Netconstraints_HasDepthConstraint>
          <Multiplicity>
            <Multiplicity_IncludesIntervall>
              <MultiplicityIntervall>
                <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd:integer">0
                </MultiplicityIntervall_HasLowerBound>
                <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd:integer">
                  2147483647</MultiplicityIntervall_HasUpperBound>
              </MultiplicityIntervall>
            </Multiplicity_IncludesIntervall>
          </Multiplicity>
        </Netconstraints_HasDepthConstraint>
        <Netconstraints_HasOrderConstraint rdf:datatype="&xsd:integer">41
        </Netconstraints_HasOrderConstraint>
        <Netconstraints_HasRateAssertion>
          <AssertionUnidirectionMultiplicity>
            <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#forward"/>
            <AssertionUnidirectionMultiplicity_HasMultiplicity>
              <Multiplicity>
                <Multiplicity_IncludesIntervall>
                  <MultiplicityIntervall>
                    <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd:integer">0
                    </MultiplicityIntervall_HasLowerBound>
                    <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd:integer">
                      2147483647</MultiplicityIntervall_HasUpperBound>
                  </MultiplicityIntervall>
                </Multiplicity_IncludesIntervall>
              </Multiplicity>
            </AssertionUnidirectionMultiplicity_HasMultiplicity>
          </AssertionUnidirectionMultiplicity>
        </Netconstraints_HasRateAssertion>
        <Netconstraints_HasRateAssertion>
          <AssertionUnidirectionMultiplicity>
            <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#backward"/>
            <AssertionUnidirectionMultiplicity_HasMultiplicity>
              <Multiplicity>
                <Multiplicity_IncludesIntervall>
                  <MultiplicityIntervall>
                    <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd:integer">0
                    </MultiplicityIntervall_HasLowerBound>
                    <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd:integer">
                      2147483647</MultiplicityIntervall_HasUpperBound>
                  </MultiplicityIntervall>
                </Multiplicity_IncludesIntervall>
              </Multiplicity>
            </AssertionUnidirectionMultiplicity_HasMultiplicity>
          </AssertionUnidirectionMultiplicity>
        </Netconstraints_HasRateAssertion>
        <Netconstraints_DefinesCopypropagation>
          <AssertionUnidirectionInteger>
            <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
            <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd:integer">50
            </AssertionUnidirectionInteger_HasInteger>
          </AssertionUnidirectionInteger>
        </Netconstraints_DefinesCopypropagation>
        <Netconstraints_DefinesCopypropagation>
          <AssertionUnidirectionInteger>
            <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
            <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd:integer">50
            </AssertionUnidirectionInteger_HasInteger>
          </AssertionUnidirectionInteger>
        </Netconstraints_DefinesCopypropagation>
        <Netconstraints_DefinesRemovepropagation>
          <AssertionUnidirectionInteger>
            <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>

```

```

    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
<Netconstraints_DefinesRemovepropagation>
  <AssertionUnidirectionInteger>
    <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
    <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
  </AssertionUnidirectionInteger_HasInteger>
  </AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
</Netconstraints>
</AssertionLayerNetconstraints_HasNetconstraints>
</AssertionLayerNetconstraints>
</AssociationType_HasNetconstraintsAssertion>
<AssociationType_HasNetconstraintsAssertion>
  <AssertionLayerNetconstraints>
    <AssertionLayerNetconstraints_HasLayer rdf:resource="#C"/>
    <AssertionLayerNetconstraints_HasNetconstraints>
      <Netconstraints>
        <Netconstraints_HasNoncyclicConstraint rdf:datatype="&xsd;boolean">>false
      </Netconstraints_HasNoncyclicConstraint>
      <Netconstraints_HasDepthConstraint>
        <Multiplicity>
          <Multiplicity_IncludesIntervall>
            <MultiplicityIntervall>
              <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
            </MultiplicityIntervall_HasLowerBound>
            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">2147483647
            </MultiplicityIntervall_HasUpperBound>
          </MultiplicityIntervall>
        </Multiplicity_IncludesIntervall>
      </Multiplicity>
    </Netconstraints_HasDepthConstraint>
    <Netconstraints_HasOrderConstraint rdf:datatype="&xsd;integer">41
  </Netconstraints_HasOrderConstraint>
  <Netconstraints_HasRateAssertion>
    <AssertionUnidirectionMultiplicity>
      <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#forward"/>
      <AssertionUnidirectionMultiplicity_HasMultiplicity>
        <Multiplicity>
          <Multiplicity_IncludesIntervall>
            <MultiplicityIntervall>
              <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
            </MultiplicityIntervall_HasLowerBound>
            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">3
            </MultiplicityIntervall_HasUpperBound>
          </MultiplicityIntervall>
        </Multiplicity_IncludesIntervall>
        <Multiplicity_IncludesIntervall>
          <MultiplicityIntervall>
            <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">5
            </MultiplicityIntervall_HasLowerBound>
            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">
              2147483647</MultiplicityIntervall_HasUpperBound>
          </MultiplicityIntervall>
        </Multiplicity_IncludesIntervall>
      </Multiplicity>
    </AssertionUnidirectionMultiplicity_HasMultiplicity>
  </AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_HasRateAssertion>
  <AssertionUnidirectionMultiplicity>
    <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#backward"/>
    <AssertionUnidirectionMultiplicity_HasMultiplicity>
      <Multiplicity>
        <Multiplicity_IncludesIntervall>
          <MultiplicityIntervall>
            <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
            </MultiplicityIntervall_HasLowerBound>
            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">
              2147483647</MultiplicityIntervall_HasUpperBound>
          </MultiplicityIntervall>
        </Multiplicity_IncludesIntervall>
      </Multiplicity>
    </AssertionUnidirectionMultiplicity_HasMultiplicity>
  </AssertionUnidirectionMultiplicity>

```

```

        </Multiplicity>
    </AssertionUnidirectionMultiplicity_HasMultiplicity>
</AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_DefinesCopypropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
    </AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesCopypropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
    </AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesRemovepropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
    </AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
<Netconstraints_DefinesRemovepropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
    </AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
</Netconstraints>
</AssertionLayerNetconstraints_HasNetconstraints>
</AssertionLayerNetconstraints>
</AssociationType_HasNetconstraintsAssertion>
<AssociationType_HasNetconstraintsAssertion>
    <AssertionLayerNetconstraints>
        <AssertionLayerNetconstraints_HasLayer rdf:resource="#M"/>
        <AssertionLayerNetconstraints_HasNetconstraints>
            <Netconstraints>
                <Netconstraints_HasNoncyclicConstraint rdf:datatype="&xsd;boolean">>false
            </Netconstraints_HasNoncyclicConstraint>
            <Netconstraints_HasDepthConstraint>
                <Multiplicity>
                    <Multiplicity_IncludesIntervall>
                        <MultiplicityIntervall>
                            <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
                        </MultiplicityIntervall_HasLowerBound>
                            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">2147483647
                        </MultiplicityIntervall_HasUpperBound>
                    </MultiplicityIntervall>
                </Multiplicity_IncludesIntervall>
            </Multiplicity>
        </Netconstraints_HasDepthConstraint>
        <Netconstraints_HasOrderConstraint rdf:datatype="&xsd;integer">41
    </Netconstraints_HasOrderConstraint>
    <Netconstraints_HasRateAssertion>
        <AssertionUnidirectionMultiplicity>
            <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#forward"/>
            <AssertionUnidirectionMultiplicity_HasMultiplicity>
                <Multiplicity>
                    <Multiplicity_IncludesIntervall>
                        <MultiplicityIntervall>
                            <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
                        </MultiplicityIntervall_HasLowerBound>
                            <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">2
                        </MultiplicityIntervall_HasUpperBound>
                    </MultiplicityIntervall>
                </Multiplicity_IncludesIntervall>
            </Multiplicity_IncludesIntervall>
        </Multiplicity_IncludesIntervall>
    </AssertionUnidirectionMultiplicity_HasMultiplicity>
        <MultiplicityIntervall>
            <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">4
        </MultiplicityIntervall_HasLowerBound>
    </MultiplicityIntervall>
    </AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>

```

```

        <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">
            2147483647</MultiplicityIntervall_HasUpperBound>
    </MultiplicityIntervall>
</Multiplicity_IncludesIntervall>
</Multiplicity>
</AssertionUnidirectionMultiplicity_HasMultiplicity>
</AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_HasRateAssertion>
    <AssertionUnidirectionMultiplicity>
        <AssertionUnidirectionMultiplicity_HasUnidirection rdf:resource="#backward"/>
    <AssertionUnidirectionMultiplicity_HasMultiplicity>
        <Multiplicity>
            <Multiplicity_IncludesIntervall>
                <MultiplicityIntervall>
                    <MultiplicityIntervall_HasLowerBound rdf:datatype="&xsd;integer">0
                </MultiplicityIntervall_HasLowerBound>
                    <MultiplicityIntervall_HasUpperBound rdf:datatype="&xsd;integer">
                        2147483647</MultiplicityIntervall_HasUpperBound>
                </MultiplicityIntervall>
            </Multiplicity_IncludesIntervall>
        </Multiplicity>
    </AssertionUnidirectionMultiplicity_HasMultiplicity>
</AssertionUnidirectionMultiplicity>
</Netconstraints_HasRateAssertion>
<Netconstraints_DefinesCopypropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesCopypropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesCopypropagation>
<Netconstraints_DefinesRemovepropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#forward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
<Netconstraints_DefinesRemovepropagation>
    <AssertionUnidirectionInteger>
        <AssertionUnidirectionInteger_HasUnidirection rdf:resource="#backward"/>
        <AssertionUnidirectionInteger_HasInteger rdf:datatype="&xsd;integer">50
    </AssertionUnidirectionInteger_HasInteger>
</AssertionUnidirectionInteger>
</Netconstraints_DefinesRemovepropagation>
</Netconstraints>
</AssertionLayerNetconstraints_HasNetconstraints>
</AssertionLayerNetconstraints>
</AssociationType_HasNetconstraintsAssertion>
<AssociationType_HasLayerAssertion>
    <AssertionLayerLayer>
        <AssertionLayerLayer_HasSource rdf:resource="#I"/>
        <AssertionLayerLayer_HasDrain rdf:resource="#I"/>
    </AssertionLayerLayer>
</AssociationType_HasLayerAssertion>
<AssociationType_HasLayerAssertion>
    <AssertionLayerLayer>
        <AssertionLayerLayer_HasSource rdf:resource="#C"/>
        <AssertionLayerLayer_HasDrain rdf:resource="#I"/>
    </AssertionLayerLayer>
</AssociationType_HasLayerAssertion>
<AssociationType_HasLayerAssertion>
    <AssertionLayerLayer>
        <AssertionLayerLayer_HasSource rdf:resource="#M"/>
        <AssertionLayerLayer_HasDrain rdf:resource="#I"/>
    </AssertionLayerLayer>
</AssociationType_HasLayerAssertion>

```

```

    </AssociationType_HasLayerAssertion>
    <AssociationType_InUnit rdf:resource="#UnitMerkmale"/>
</AssociationType>

<InstanceEntity rdf:ID="-7539133666533682201">
  <Object_HasUIString rdf:datatype="&xsd:string">WeiÃŸ</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">WeiÃŸ</Entity_HasContent>
  <Entity_InLayer rdf:resource="#I"/>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
  <InstanceEntity_AssociatedToClass rdf:resource="#-6572988048988707652"/>
</InstanceEntity>

<InstanceEntity rdf:ID="4873894939970285764">
  <Object_HasUIString rdf:datatype="&xsd:string">Schwarz</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Schwarz</Entity_HasContent>
  <Entity_InLayer rdf:resource="#I"/>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
  <InstanceEntity_AssociatedToClass rdf:resource="#-6572988048988707652"/>
</InstanceEntity>

<ClassEntity rdf:ID="-6572988048988707652">
  <Object_HasUIString rdf:datatype="&xsd:string">Farbe</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Farbe</Entity_HasContent>
  <Entity_InLayer rdf:resource="#C"/>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
  <ClassEntity_AssociatedToMeta rdf:resource="#-66685964784945408"/>
</ClassEntity>

<MetaEntity rdf:ID="-66685964784945408">
  <Object_HasUIString rdf:datatype="&xsd:string">OptischesMerkmal</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">OptischesMerkmal</Entity_HasContent>
  <Entity_InLayer rdf:resource="#M"/>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
  <MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">false</MetaEntity_IsAttribute>
  <MetaEntity_IsEO rdf:datatype="&xsd:boolean">false</MetaEntity_IsEO>
</MetaEntity>

<Unit rdf:ID="UnitMerkmale">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">true</Unit_IsMutable>
  <Unit_HasSuperunit rdf:resource="#rootUnit"/>
</Unit>

<Unit rdf:ID="rootUnit">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">false</Unit_IsMutable>
</Unit>

</rdf:RDF>

```

8.B Aus dem Automobilbeispiel extrahierte OWL DL-Ontologie

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
]>

<rdf:RDF
  xml:base = "http://www.MarcEichler.de/owl/AutomobilDL#"
  xmlns: = "http://www.MarcEichler.de/owl/AutomobilDL#"
  xmlns:owl = "&owl;"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "&xsd;"
>

<owl:Class rdf:ID="Unit">
</owl:Class>

<owl:DatatypeProperty rdf:ID="Unit_IsReadOnly">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="Unit_IsMutable">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Unit_HasSuperunit">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Unit_HasSubunit">
  <owl:inverseOf rdf:resource="#Unit_HasSuperunit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Unit_ContainsEntity">
  <owl:inverseOf rdf:resource="#Entity_InUnit"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Object">
</owl:Class>

<owl:DatatypeProperty rdf:ID="Object_HasUIString">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Object"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="Object_IsMutable">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Object"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Entity">
  <rdfs:subClassOf rdf:resource="#Object"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="Entity_HasContent">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
```

```

<owl:ObjectProperty rdf:ID="Entity_InUnit">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Entity_AssociatedToSuper">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Entity"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Entity_AssociatedToSub">
  <owl:inverseOf rdf:resource="#Entity_AssociatedToSuper"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="MetaEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward"/>
      <owl:allValuesFrom rdf:resource="#MetaEntity"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward"/>
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward"/>
              <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward"/>
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward"/>
              <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_backward"/>
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_backward"/>
              <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>

```

```

        </owl:intersectionOf>
    </owl:Class>
    </owl:unionOf>
</owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
        <owl:allValuesFrom rdf:resource="#InstanceEntity"/>
    </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
                        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
                        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
                        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">4
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
                        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:unionOf>
    </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbel218494115086901730_backward"/>
                        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbel218494115086901730_backward"/>
                        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:unionOf>
    </owl:Class>
</rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsAttribute">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#MetaEntity"/>
    <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsEO">

```

```

    <rdf:type rdf:resource="&owl;FunctionalProperty" />
    <rdfs:domain rdf:resource="#MetaEntity" />
    <rdfs:range rdf:resource="&xsd;boolean" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_HasEOName">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#MetaEntity" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsAbstract">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#MetaEntity" />
  <rdfs:range rdf:resource="&xsd;boolean" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_HasAttributetype">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#MetaEntity" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="MetaEntity_AssociatedToClass">
  <rdfs:subPropertyOf rdf:resource="#Entity_AssociatedToSub" />
  <owl:inverseOf rdf:resource="#ClassEntity_AssociatedToMeta" />
</owl:ObjectProperty>

<owl:Class rdf:ID="ClassEntity">
  <rdfs:subClassOf rdf:resource="#Entity" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward" />
      <owl:allValuesFrom rdf:resource="#ClassEntity" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward" />
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:minCardinality>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward" />
              <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:maxCardinality>
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward" />
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl:minCardinality>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_forward" />
              <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl:maxCardinality>
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class>

```

```

    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
        <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_backward"/>
        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#eingebautIn1489042623048229543_backward"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</owl:unionOf>
</owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
    <owl:allValuesFrom rdf:resource="#InstanceEntity"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">3
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">5
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbel218494115086901730_forward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbel218494115086901730_backward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbel218494115086901730_backward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</owl:unionOf>

```



```

        </owl:Class>
    </owl:unionOf>
</owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource="#hatFarbe1218494115086901730_forward"/>
        <owl:allValuesFrom rdf:resource="#InstanceEntity"/>
    </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbe1218494115086901730_forward"/>
                        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbe1218494115086901730_forward"/>
                        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:unionOf>
    </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbe1218494115086901730_backward"/>
                        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hatFarbe1218494115086901730_backward"/>
                        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:unionOf>
    </owl:Class>
</rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="InstanceEntity_AssociatedToClass">
    <rdfs:subPropertyOf rdf:resource="#Entity_AssociatedToSuper"/>
    <rdfs:type rdf:resource="&owl;TransitiveProperty"/>
    <rdfs:domain rdf:resource="#InstanceEntity"/>
    <rdfs:range rdf:resource="#ClassEntity"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="eingebautIn1489042623048229543_forward">
    <rdfs:domain rdf:resource="#Entity"/>
    <rdfs:range rdf:resource="#Entity"/>
    <AssociationType_InUnit>UnitFahrzeuge</AssociationType_InUnit>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="eingebautIn1489042623048229543_backward">
    <owl:inverseOf rdf:resource="#eingebautIn1489042623048229543_forward"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hatFarbe1218494115086901730_forward">
    <rdfs:domain rdf:resource="#Entity"/>
    <rdfs:range rdf:resource="#Entity"/>
    <AssociationType_InUnit>UnitMerkmale</AssociationType_InUnit>
</owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:ID="hatFarbel218494115086901730_backward">
  <owl:inverseOf rdf:resource="#hatFarbel218494115086901730_forward"/>
</owl:ObjectProperty>

<owl:AnnotationProperty rdf:ID="AssociationType_InUnit"/>

<InstanceEntity rdf:ID="-5405652485843849646">
  <Object_HasUIString rdf:datatype="&xsd:string">MotorXY</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">MotorXY</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <eingebautIn1489042623048229543_forward rdf:resource="#-7382333189627944410"/>
  <InstanceEntity_AssociatedToClass rdf:resource="#-1564706898822228111"/>
</InstanceEntity>

<InstanceEntity rdf:ID="-7382333189627944410">
  <Object_HasUIString rdf:datatype="&xsd:string">Herbie</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Herbie</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <hatFarbel218494115086901730_forward rdf:resource="#1673286411370268811"/>
  <hatFarbel218494115086901730_forward rdf:resource="#6707573048513016829"/>
  <InstanceEntity_AssociatedToClass rdf:resource="#8824974531218705464"/>
</InstanceEntity>

<ClassEntity rdf:ID="8824974531218705464">
  <Object_HasUIString rdf:datatype="&xsd:string">VW-KÄofer</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">VW-KÄofer</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <ClassEntity_AssociatedToMeta rdf:resource="#325133859829489799"/>
</ClassEntity>

<ClassEntity rdf:ID="-1564706898822228111">
  <Object_HasUIString rdf:datatype="&xsd:string">KÄofer-Motor</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">KÄofer-Motor</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <eingebautIn1489042623048229543_forward rdf:resource="#8824974531218705464"/>
  <ClassEntity_AssociatedToMeta rdf:resource="#8001047373428329198"/>
</ClassEntity>

<MetaEntity rdf:ID="325133859829489799">
  <Object_HasUIString rdf:datatype="&xsd:string">Automobil</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Automobil</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">>false</MetaEntity_IsAttribute>
  <MetaEntity_IsEO rdf:datatype="&xsd:boolean">>false</MetaEntity_IsEO>
</MetaEntity>

<MetaEntity rdf:ID="8001047373428329198">
  <Object_HasUIString rdf:datatype="&xsd:string">Motor</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Motor</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <hatFarbel218494115086901730_forward rdf:resource="#1673286411370268811"/>
  <eingebautIn1489042623048229543_forward rdf:resource="#325133859829489799"/>
  <MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">>false</MetaEntity_IsAttribute>
  <MetaEntity_IsEO rdf:datatype="&xsd:boolean">>false</MetaEntity_IsEO>
</MetaEntity>

<Unit rdf:ID="UnitFahrzeuge">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">>false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">true</Unit_IsMutable>
  <Unit_HasSuperunit rdf:resource="#rootUnit"/>
</Unit>

<InstanceEntity rdf:ID="6707573048513016829">
  <Object_HasUIString rdf:datatype="&xsd:string">WeiÄÿ</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">WeiÄÿ</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>

```

```

    <InstanceEntity_AssociatedToClass rdf:resource="#-6959927080016735339"/>
</InstanceEntity>

<InstanceEntity rdf:ID="1673286411370268811">
  <Object_HasUIString rdf:datatype="&xsd:string">Schwarz</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Schwarz</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
  <InstanceEntity_AssociatedToClass rdf:resource="#-6959927080016735339"/>
</InstanceEntity>

<ClassEntity rdf:ID="-6959927080016735339">
  <Object_HasUIString rdf:datatype="&xsd:string">Farbe</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Farbe</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
  <ClassEntity_AssociatedToMeta rdf:resource="#-2874225587809405728"/>
</ClassEntity>

<MetaEntity rdf:ID="-2874225587809405728">
  <Object_HasUIString rdf:datatype="&xsd:string">OptischesMerkmal</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">OptischesMerkmal</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
  <MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">false</MetaEntity_IsAttribute>
  <MetaEntity_IsEO rdf:datatype="&xsd:boolean">false</MetaEntity_IsEO>
</MetaEntity>

<Unit rdf:ID="UnitMerkmale">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">true</Unit_IsMutable>
  <Unit_HasSuperunit rdf:resource="#rootUnit"/>
</Unit>

<Unit rdf:ID="rootUnit">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">false</Unit_IsMutable>
</Unit>

</rdf:RDF>

```

8.C Aus dem Automobilbeispiel extrahierte OWL Full-Ontologie

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
]>

<rdf:RDF
  xml:base = "http://www.MarcEichler.de/owl/AutomobilFull#"
  xmlns: = "http://www.MarcEichler.de/owl/AutomobilFull#"
  xmlns:owl = "&owl;"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "&xsd;"
>

<owl:Class rdf:ID="Unit">
</owl:Class>

<owl:DatatypeProperty rdf:ID="Unit_IsReadOnly">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="Unit_IsMutable">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Unit_HasSuperunit">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Unit"/>
  <rdfs:range rdf:resource="#Unit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Unit_HasSubunit">
  <owl:inverseOf rdf:resource="#Unit_HasSuperunit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Unit_ContainsEntity">
  <owl:inverseOf rdf:resource="#Entity_InUnit"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="Unit_ContainsAssociationType">
  <owl:inverseOf rdf:resource="#AssociationType_InUnit"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Object">
</owl:Class>

<owl:DatatypeProperty rdf:ID="Object_HasUIString">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Object"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="Object_IsMutable">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Object"/>
  <rdfs:range rdf:resource="&xsd:boolean"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Entity">
  <rdfs:subClassOf rdf:resource="#Object"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="Entity_HasContent">
```

```

    <rdf:type rdf:resource="&owl;FunctionalProperty" />
    <rdfs:domain rdf:resource="#Entity" />
    <rdfs:range rdf:resource="&xsd:string" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="Entity_InUnit">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Entity" />
  <rdfs:range rdf:resource="#Unit" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssociationType">
  <rdfs:domain rdf:resource="#Entity" />
  <rdfs:range rdf:resource="#Entity" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="AssociationType_InUnit">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#AssociationType" />
  <rdfs:range rdf:resource="#Unit" />
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="AssociationType_CopyPropagation">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#AssociationType" />
  <rdfs:range rdf:resource="&xsd:integer" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="AssociationType_DepthConstraint">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#AssociationType" />
  <rdfs:range rdf:resource="#Multiplicity" />
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="AssociationType_NonCyclicConstraint">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#AssociationType" />
  <rdfs:range rdf:resource="&xsd:boolean" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="AssociationType_OrderConstraint">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#AssociationType" />
  <rdfs:range rdf:resource="&xsd:integer" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="AssociationType_RemovePropagation">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#AssociationType" />
  <rdfs:range rdf:resource="&xsd:integer" />
</owl:DatatypeProperty>

<owl:Class rdf:ID="SuperMetaEntity">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward" />
      <owl:allValuesFrom rdf:resource="#SuperMetaEntity" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward" />
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward" />
              <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>

```

```

    </owl:intersectionOf>
  </owl:Class>
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
</owl:unionOf>
</owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_backward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_backward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
    <owl:allValuesFrom rdf:resource="#ClassEntity"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:Class>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
          <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">4
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
          <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
        </owl:Restriction>
      </owl:Restriction>
    </owl:Class>
  </owl:Class>
</rdfs:subClassOf>

```

```

        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_backward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
          </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_backward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
          </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsAttribute">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#SuperMetaEntity"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsEO">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#SuperMetaEntity"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_HasEOName">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#SuperMetaEntity"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_IsAbstract">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#SuperMetaEntity"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MetaEntity_HasAttributetype">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#SuperMetaEntity"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="MetaEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
      <owl:allValuesFrom rdf:resource="#MetaEntity"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:minCardinality>
          </owl:intersectionOf>
        </owl:Class>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>

```

```

        </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">0
      </owl:maxCardinality>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
    </owl:minCardinality>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
    </owl:maxCardinality>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
</owl:unionOf>
</owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_backward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
          </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_backward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
          </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
    <owl:allValuesFrom rdf:resource="#ClassEntity"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
          </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">3
          </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:Class>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
          <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">5
        </owl:minCardinality>
      </owl:intersectionOf>
    </owl:Class>
  </owl:unionOf>
</owl:Class>

```

```

        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
          <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
          </owl:maxCardinality>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:unionOf>
</owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_backward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_backward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
            </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="ClassEntity">
  <rdfs:subClassOf rdf:resource="#Entity"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
      <owl:allValuesFrom rdf:resource="#ClassEntity"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
              </owl:minCardinality>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
              <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">0
              </owl:maxCardinality>
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
              </owl:minCardinality>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_forward"/>
              <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
              </owl:maxCardinality>
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>

```

```

</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_backward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#eingebautIn-6217510651617369709_backward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
            </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
    <owl:allValuesFrom rdf:resource="#ClassEntity"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_forward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
            </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_backward"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0
            </owl:minCardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hatFarbe-2073502395880478378_backward"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2147483647
            </owl:maxCardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="eingebautIn-6217510651617369709_forward">
  <rdfs:subPropertyOf rdf:resource="#AssociationType"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Entity"/>
  <AssociationType_InUnit rdf:resource="#UnitFahrzeuge"/>

```

```

<Object_HasUIString rdf:datatype="&xsd:string">-8129356600157484165%eingebautIn$
</Object_HasUIString>
<Object_IsMutable rdf:datatype="&xsd:boolean">>true</Object_IsMutable>
<AssociationType_DepthConstraint_Meta>
  <Multiplicity>
    <Multiplicity_IncludesIntervall>
      <MultiplicityInterval>
        <MultiplicityInterval_HasLowerBound rdf:datatype="&xsd:integer">0
        </MultiplicityInterval_HasLowerBound>
        <MultiplicityInterval_HasUpperBound rdf:datatype="&xsd:integer">2147483647
        </MultiplicityInterval_HasUpperBound>
      </MultiplicityInterval>
    </Multiplicity_IncludesIntervall>
  </Multiplicity>
</AssociationType_DepthConstraint_Meta>
<AssociationType_NonCyclicConstraint_Meta rdf:datatype="&xsd:boolean">>false
</AssociationType_NonCyclicConstraint_Meta>
<AssociationType_OrderConstraint_Meta rdf:datatype="&xsd:integer">41
</AssociationType_OrderConstraint_Meta>
<AssociationType_RemovePropagation_Meta rdf:datatype="&xsd:integer">50
</AssociationType_RemovePropagation_Meta>
<AssociationType_CopyPropagation_Meta rdf:datatype="&xsd:integer">50
</AssociationType_CopyPropagation_Meta>
<AssociationType_DepthConstraint_Class>
  <Multiplicity>
    <Multiplicity_IncludesIntervall>
      <MultiplicityInterval>
        <MultiplicityInterval_HasLowerBound rdf:datatype="&xsd:integer">0
        </MultiplicityInterval_HasLowerBound>
        <MultiplicityInterval_HasUpperBound rdf:datatype="&xsd:integer">2147483647
        </MultiplicityInterval_HasUpperBound>
      </MultiplicityInterval>
    </Multiplicity_IncludesIntervall>
  </Multiplicity>
</AssociationType_DepthConstraint_Class>
<AssociationType_NonCyclicConstraint_Class rdf:datatype="&xsd:boolean">>false
</AssociationType_NonCyclicConstraint_Class>
<AssociationType_OrderConstraint_Class rdf:datatype="&xsd:integer">41
</AssociationType_OrderConstraint_Class>
<AssociationType_RemovePropagation_Class rdf:datatype="&xsd:integer">50
</AssociationType_RemovePropagation_Class>
<AssociationType_CopyPropagation_Class rdf:datatype="&xsd:integer">50
</AssociationType_CopyPropagation_Class>
<AssociationType_DepthConstraint_Instance>
  <Multiplicity>
    <Multiplicity_IncludesIntervall>
      <MultiplicityInterval>
        <MultiplicityInterval_HasLowerBound rdf:datatype="&xsd:integer">0
        </MultiplicityInterval_HasLowerBound>
        <MultiplicityInterval_HasUpperBound rdf:datatype="&xsd:integer">2147483647
        </MultiplicityInterval_HasUpperBound>
      </MultiplicityInterval>
    </Multiplicity_IncludesIntervall>
  </Multiplicity>
</AssociationType_DepthConstraint_Instance>
<AssociationType_NonCyclicConstraint_Instance rdf:datatype="&xsd:boolean">>false
</AssociationType_NonCyclicConstraint_Instance>
<AssociationType_OrderConstraint_Instance rdf:datatype="&xsd:integer">41
</AssociationType_OrderConstraint_Instance>
<AssociationType_RemovePropagation_Instance rdf:datatype="&xsd:integer">50
</AssociationType_RemovePropagation_Instance>
<AssociationType_CopyPropagation_Instance rdf:datatype="&xsd:integer">50
</AssociationType_CopyPropagation_Instance>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="eingebautIn-6217510651617369709_backward">
  <rdfs:subPropertyOf rdf:resource="#AssociationType"/>
  <owl:inverseOf rdf:resource="#eingebautIn-6217510651617369709_forward"/>
  <AssociationType_InUnit rdf:resource="#UnitFahrzeuge"/>
  <Object_HasUIString rdf:datatype="&xsd:string">
    -8129356600157484165%eingebautIn$</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">>true</Object_IsMutable>
  <AssociationType_DepthConstraint_Meta>
    <Multiplicity>

```

```

    <Multiplicity_IncludesIntervall>
      <MultiplicityInterval>
        <MultiplicityInterval_HasLowerBound
          rdf:datatype="&xsd;integer">0</MultiplicityInterval_HasLowerBound>
        <MultiplicityInterval_HasUpperBound
          rdf:datatype="&xsd;integer">2147483647</MultiplicityInterval_HasUpperBound>
        </MultiplicityInterval>
      </Multiplicity_IncludesIntervall>
    </Multiplicity>
  </AssociationType_DepthConstraint_Meta>
  <AssociationType_NonCyclicConstraint_Meta
    rdf:datatype="&xsd;boolean">>false</AssociationType_NonCyclicConstraint_Meta>
  <AssociationType_OrderConstraint_Meta
    rdf:datatype="&xsd;integer">41</AssociationType_OrderConstraint_Meta>
  <AssociationType_RemovePropagation_Meta
    rdf:datatype="&xsd;integer">50</AssociationType_RemovePropagation_Meta>
  <AssociationType_CopyPropagation_Meta
    rdf:datatype="&xsd;integer">50</AssociationType_CopyPropagation_Meta>
  <AssociationType_DepthConstraint_Class>
    <Multiplicity>
      <Multiplicity_IncludesIntervall>
        <MultiplicityInterval>
          <MultiplicityInterval_HasLowerBound
            rdf:datatype="&xsd;integer">0</MultiplicityInterval_HasLowerBound>
          <MultiplicityInterval_HasUpperBound
            rdf:datatype="&xsd;integer">2147483647</MultiplicityInterval_HasUpperBound>
          </MultiplicityInterval>
        </Multiplicity_IncludesIntervall>
      </Multiplicity>
    </AssociationType_DepthConstraint_Class>
  <AssociationType_NonCyclicConstraint_Class
    rdf:datatype="&xsd;boolean">>false</AssociationType_NonCyclicConstraint_Class>
  <AssociationType_OrderConstraint_Class
    rdf:datatype="&xsd;integer">41</AssociationType_OrderConstraint_Class>
  <AssociationType_RemovePropagation_Class
    rdf:datatype="&xsd;integer">50</AssociationType_RemovePropagation_Class>
  <AssociationType_CopyPropagation_Class
    rdf:datatype="&xsd;integer">50</AssociationType_CopyPropagation_Class>
  <AssociationType_DepthConstraint_Instance>
    <Multiplicity>
      <Multiplicity_IncludesIntervall>
        <MultiplicityInterval>
          <MultiplicityInterval_HasLowerBound
            rdf:datatype="&xsd;integer">0</MultiplicityInterval_HasLowerBound>
          <MultiplicityInterval_HasUpperBound
            rdf:datatype="&xsd;integer">2147483647</MultiplicityInterval_HasUpperBound>
          </MultiplicityInterval>
        </Multiplicity_IncludesIntervall>
      </Multiplicity>
    </AssociationType_DepthConstraint_Instance>
  <AssociationType_NonCyclicConstraint_Instance
    rdf:datatype="&xsd;boolean">>false</AssociationType_NonCyclicConstraint_Instance>
  <AssociationType_OrderConstraint_Instance
    rdf:datatype="&xsd;integer">41</AssociationType_OrderConstraint_Instance>
  <AssociationType_RemovePropagation_Instance
    rdf:datatype="&xsd;integer">50</AssociationType_RemovePropagation_Instance>
  <AssociationType_CopyPropagation_Instance
    rdf:datatype="&xsd;integer">50</AssociationType_CopyPropagation_Instance>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hatFarbe-2073502395880478378_forward">
  <rdfs:subPropertyOf rdf:resource="#AssociationType" />
  <rdfs:domain rdf:resource="#Entity" />
  <rdfs:range rdf:resource="#Entity" />
  <AssociationType_InUnit rdf:resource="#UnitMerkmale" />
  <Object_HasUIString
    rdf:datatype="&xsd:string">7835537084971084134%hatFarbe$</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd;boolean">>true</Object_IsMutable>
  <AssociationType_DepthConstraint_Meta>
    <Multiplicity>
      <Multiplicity_IncludesIntervall>
        <MultiplicityInterval>
          <MultiplicityInterval_HasLowerBound
            rdf:datatype="&xsd;integer">0</MultiplicityInterval_HasLowerBound>

```

```

    <MultiplicityInterval_HasUpperBound
      rdf:datatype="&xsd;integer">2147483647</MultiplicityInterval_HasUpperBound>
    </MultiplicityInterval>
  </Multiplicity_IncludesIntervall>
</Multiplicity>
</AssociationType_DepthConstraint_Meta>
<AssociationType_NonCyclicConstraint_Meta
  rdf:datatype="&xsd;boolean">>false</AssociationType_NonCyclicConstraint_Meta>
<AssociationType_OrderConstraint_Meta
  rdf:datatype="&xsd;integer">41</AssociationType_OrderConstraint_Meta>
<AssociationType_RemovePropagation_Meta
  rdf:datatype="&xsd;integer">50</AssociationType_RemovePropagation_Meta>
<AssociationType_CopyPropagation_Meta
  rdf:datatype="&xsd;integer">50</AssociationType_CopyPropagation_Meta>
<AssociationType_DepthConstraint_Class>
  <Multiplicity>
    <Multiplicity_IncludesIntervall>
      <MultiplicityInterval>
        <MultiplicityInterval_HasLowerBound
          rdf:datatype="&xsd;integer">0</MultiplicityInterval_HasLowerBound>
        <MultiplicityInterval_HasUpperBound
          rdf:datatype="&xsd;integer">2147483647</MultiplicityInterval_HasUpperBound>
        </MultiplicityInterval>
      </Multiplicity_IncludesIntervall>
    </Multiplicity>
  </AssociationType_DepthConstraint_Class>
<AssociationType_NonCyclicConstraint_Class
  rdf:datatype="&xsd;boolean">>false</AssociationType_NonCyclicConstraint_Class>
<AssociationType_OrderConstraint_Class
  rdf:datatype="&xsd;integer">41</AssociationType_OrderConstraint_Class>
<AssociationType_RemovePropagation_Class
  rdf:datatype="&xsd;integer">50</AssociationType_RemovePropagation_Class>
<AssociationType_CopyPropagation_Class
  rdf:datatype="&xsd;integer">50</AssociationType_CopyPropagation_Class>
<AssociationType_DepthConstraint_Instance>
  <Multiplicity>
    <Multiplicity_IncludesIntervall>
      <MultiplicityInterval>
        <MultiplicityInterval_HasLowerBound
          rdf:datatype="&xsd;integer">0</MultiplicityInterval_HasLowerBound>
        <MultiplicityInterval_HasUpperBound
          rdf:datatype="&xsd;integer">2147483647</MultiplicityInterval_HasUpperBound>
        </MultiplicityInterval>
      </Multiplicity_IncludesIntervall>
    </Multiplicity>
  </AssociationType_DepthConstraint_Instance>
<AssociationType_NonCyclicConstraint_Instance
  rdf:datatype="&xsd;boolean">>false</AssociationType_NonCyclicConstraint_Instance>
<AssociationType_OrderConstraint_Instance
  rdf:datatype="&xsd;integer">41</AssociationType_OrderConstraint_Instance>
<AssociationType_RemovePropagation_Instance
  rdf:datatype="&xsd;integer">50</AssociationType_RemovePropagation_Instance>
<AssociationType_CopyPropagation_Instance
  rdf:datatype="&xsd;integer">50</AssociationType_CopyPropagation_Instance>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hatFarbe-2073502395880478378_backward">
  <rdfs:subPropertyOf rdf:resource="#AssociationType" />
  <owl:inverseOf rdf:resource="#hatFarbe-2073502395880478378_forward" />
  <AssociationType_InUnit rdf:resource="#UnitMerkmale" />
  <Object_HasUIString
    rdf:datatype="&xsd:string">7835537084971084134%hatFarbe$</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd;boolean">>true</Object_IsMutable>
  <AssociationType_DepthConstraint_Meta>
    <Multiplicity>
      <Multiplicity_IncludesIntervall>
        <MultiplicityInterval>
          <MultiplicityInterval_HasLowerBound
            rdf:datatype="&xsd;integer">0</MultiplicityInterval_HasLowerBound>
          <MultiplicityInterval_HasUpperBound
            rdf:datatype="&xsd;integer">2147483647</MultiplicityInterval_HasUpperBound>
          </MultiplicityInterval>
        </Multiplicity_IncludesIntervall>
      </Multiplicity>
    </AssociationType_DepthConstraint_Meta>
  </owl:ObjectProperty>

```

```

</AssociationType_DepthConstraint_Meta>
<AssociationType_NonCyclicConstraint_Meta
  rdf:datatype="&xsd:boolean">false</AssociationType_NonCyclicConstraint_Meta>
<AssociationType_OrderConstraint_Meta
  rdf:datatype="&xsd:integer">41</AssociationType_OrderConstraint_Meta>
<AssociationType_RemovePropagation_Meta
  rdf:datatype="&xsd:integer">50</AssociationType_RemovePropagation_Meta>
<AssociationType_CopyPropagation_Meta
  rdf:datatype="&xsd:integer">50</AssociationType_CopyPropagation_Meta>
<AssociationType_DepthConstraint_Class>
  <Multiplicity>
    <Multiplicity_IncludesIntervall>
      <MultiplicityInterval>
        <MultiplicityInterval_HasLowerBound
          rdf:datatype="&xsd:integer">0</MultiplicityInterval_HasLowerBound>
        <MultiplicityInterval_HasUpperBound
          rdf:datatype="&xsd:integer">2147483647</MultiplicityInterval_HasUpperBound>
        </MultiplicityInterval>
      </Multiplicity_IncludesIntervall>
    </Multiplicity>
  </AssociationType_DepthConstraint_Class>
<AssociationType_NonCyclicConstraint_Class
  rdf:datatype="&xsd:boolean">false</AssociationType_NonCyclicConstraint_Class>
<AssociationType_OrderConstraint_Class
  rdf:datatype="&xsd:integer">41</AssociationType_OrderConstraint_Class>
<AssociationType_RemovePropagation_Class
  rdf:datatype="&xsd:integer">50</AssociationType_RemovePropagation_Class>
<AssociationType_CopyPropagation_Class
  rdf:datatype="&xsd:integer">50</AssociationType_CopyPropagation_Class>
<AssociationType_DepthConstraint_Instance>
  <Multiplicity>
    <Multiplicity_IncludesIntervall>
      <MultiplicityInterval>
        <MultiplicityInterval_HasLowerBound
          rdf:datatype="&xsd:integer">0</MultiplicityInterval_HasLowerBound>
        <MultiplicityInterval_HasUpperBound
          rdf:datatype="&xsd:integer">2147483647</MultiplicityInterval_HasUpperBound>
        </MultiplicityInterval>
      </Multiplicity_IncludesIntervall>
    </Multiplicity>
  </AssociationType_DepthConstraint_Instance>
<AssociationType_NonCyclicConstraint_Instance
rdf:datatype="&xsd:boolean">false</AssociationType_NonCyclicConstraint_Instance>
  <AssociationType_OrderConstraint_Instance
rdf:datatype="&xsd:integer">41</AssociationType_OrderConstraint_Instance>
  <AssociationType_RemovePropagation_Instance
rdf:datatype="&xsd:integer">50</AssociationType_RemovePropagation_Instance>
  <AssociationType_CopyPropagation_Instance
rdf:datatype="&xsd:integer">50</AssociationType_CopyPropagation_Instance>
</owl:ObjectProperty>

<owl:Thing rdf:ID="-3813715327638606293">
  <rdf:type rdf:resource="#-781092503947426308"/>
  <Object_HasUIString rdf:datatype="&xsd:string">MotorXY</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">MotorXY</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <eingebautIn-6217510651617369709_forward rdf:resource="#-7003075268791637919"/>
</owl:Thing>

<owl:Thing rdf:ID="-7003075268791637919">
  <rdf:type rdf:resource="#6383504144065764264"/>
  <Object_HasUIString rdf:datatype="&xsd:string">Herbie</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Herbie</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <hatFarbe-2073502395880478378_forward rdf:resource="#5274738304922376338"/>
  <hatFarbe-2073502395880478378_forward rdf:resource="#5572773085498366100"/>
</owl:Thing>

<owl:Class rdf:ID="-781092503947426308">
  <rdfs:subClassOf rdf:resource="#ClassEntity"/>
  <rdf:type rdf:resource="#6829377704062624121"/>
  <Object_HasUIString rdf:datatype="&xsd:string">K ffer-Motor</Object_HasUIString>

```

```

<Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
<Entity_HasContent rdf:datatype="&xsd:string">KÄfer-Motor</Entity_HasContent>
<Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
<eingebautIn-6217510651617369709_forward rdf:resource="#6383504144065764264"/>
</owl:Class>

<owl:Class rdf:ID="6383504144065764264">
  <rdfs:subClassOf rdf:resource="#ClassEntity"/>
  <rdf:type rdf:resource="#3058552971172345172"/>
  <Object_HasUIString rdf:datatype="&xsd:string">VW-KÄfer</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">VW-KÄfer</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
</owl:Class>

<owl:Class rdf:ID="3058552971172345172">
  <rdfs:subClassOf rdf:resource="#MetaEntity"/>
  <rdf:type rdf:resource="#SuperMetaEntity"/>
  <Object_HasUIString rdf:datatype="&xsd:string">Automobil</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Automobil</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">false</MetaEntity_IsAttribute>
  <MetaEntity_IsEO rdf:datatype="&xsd:boolean">false</MetaEntity_IsEO>
</owl:Class>

<owl:Class rdf:ID="6829377704062624121">
  <rdfs:subClassOf rdf:resource="#MetaEntity"/>
  <rdf:type rdf:resource="#SuperMetaEntity"/>
  <Object_HasUIString rdf:datatype="&xsd:string">Motor</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Motor</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitFahrzeuge"/>
  <eingebautIn-6217510651617369709_forward rdf:resource="#3058552971172345172"/>
  <hatFarbe-2073502395880478378_forward rdf:resource="#5274738304922376338"/>
  <MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">false</MetaEntity_IsAttribute>
  <MetaEntity_IsEO rdf:datatype="&xsd:boolean">false</MetaEntity_IsEO>
</owl:Class>

<Unit rdf:ID="UnitFahrzeuge">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">true</Unit_IsMutable>
  <Unit_HasSuperunit rdf:resource="#rootUnit"/>
</Unit>

<owl:Thing rdf:ID="5572773085498366100">
  <rdf:type rdf:resource="#4514609767254383333"/>
  <Object_HasUIString rdf:datatype="&xsd:string">WeiÃŸ</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">WeiÃŸ</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
</owl:Thing>

<owl:Thing rdf:ID="5274738304922376338">
  <rdf:type rdf:resource="#4514609767254383333"/>
  <Object_HasUIString rdf:datatype="&xsd:string">Schwarz</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Schwarz</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
</owl:Thing>

<owl:Class rdf:ID="4514609767254383333">
  <rdfs:subClassOf rdf:resource="#ClassEntity"/>
  <rdf:type rdf:resource="#-3880204268215563835"/>
  <Object_HasUIString rdf:datatype="&xsd:string">Farbe</Object_HasUIString>
  <Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
  <Entity_HasContent rdf:datatype="&xsd:string">Farbe</Entity_HasContent>
  <Entity_InUnit rdf:resource="#UnitMerkmale"/>
</owl:Class>

<owl:Class rdf:ID="-3880204268215563835">
  <rdfs:subClassOf rdf:resource="#MetaEntity"/>
  <rdf:type rdf:resource="#SuperMetaEntity"/>
  <Object_HasUIString rdf:datatype="&xsd:string">OptischesMerkmal</Object_HasUIString>

```

```

<Object_IsMutable rdf:datatype="&xsd:boolean">true</Object_IsMutable>
<Entity_HasContent rdf:datatype="&xsd:string">OptischesMerkmal</Entity_HasContent>
<Entity_InUnit rdf:resource="#UnitMerkmale"/>
<MetaEntity_IsAttribute rdf:datatype="&xsd:boolean">>false</MetaEntity_IsAttribute>
<MetaEntity_IsEO rdf:datatype="&xsd:boolean">>false</MetaEntity_IsEO>
</owl:Class>

<Unit rdf:ID="UnitMerkmale">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">>false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">true</Unit_IsMutable>
  <Unit_HasSuperunit rdf:resource="#rootUnit"/>
</Unit>

<Unit rdf:ID="rootUnit">
  <Unit_IsReadOnly rdf:datatype="&xsd:boolean">>false</Unit_IsReadOnly>
  <Unit_IsMutable rdf:datatype="&xsd:boolean">>false</Unit_IsMutable>
</Unit>

</rdf:RDF>

```

Ich versichere, dass ich diese Arbeit selbständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.