F A C H H O C H S C H U L E W E D E L

S E M I N A R A R B E I T

in der Fachrichtung

Medieninformatik

Die Beschreibungssprache OWL: Zweck, Aufbau und Beispiel

Eingereicht von:	Christian Köhn Neuer Sielpool 2 21129 Hamburg Tel. 040 / 742 13 715 mi2219@fh-wedel.de
Matr.-Nr.:	mi2219
Erarbeitet im:	7. Verwaltungssemester
Abgegeben am:	20. Dezember 2006
Referent:	Prof. Dr. Sebastian Iwanowski Fachhochschule Wedel Feldstrasse 143 22880 Wedel

Inhaltsverzeichnis

1. Einleitung

Diese Ausarbeitung beschäftigt sich mit der Web Ontology Language (OWL), welche vom World Wide Web Consortium (W3C) als ein Standard für das Semantic Web definiert wurde. Hierbei wird einleitend kurz auf das Semantic Web eingegangen, um die grundlegende Motivation dieses Standards zu erläutern.

In den folgenden Abschnitten werden Ontologien im Allgemeinen vorgestellt und anschließend genauer auf die Web Ontology Language (OWL) eingegangen. Die wichtigsten Sprachkonstrukte von OWL werden anhand eines durchgängigen Beispiels diskutiert, um einen besseren Gesamtüberblick zu erhalten.

1.1 Motivation

Das Semantic Web ist eine Erweiterung des aktuellen Internets und ordnet den vorhandenen Informationen eine konkrete Bedeutung zu. Hierbei geht es vor allem darum, dass Menschen sowie Maschinen das Gleiche unter einem Begriff verstehen. Es soll ein gemeinsames Verständnis der Welt geschaffen werden, um es Maschinen zu erleichtern Informationen zu finden, zusammenzufassen und zu verarbeiten. Das World Wide Web Consortium (W3C) hat für dieses Ziel eine Reihe von Arbeitsgruppen geschaffen, die sich um Teilaspekte dieser Vision kümmern.

Ein Teil hiervon war die Arbeitsgruppe der Web Ontology Language (OWL). Sie beschäftigte sich mit der formalen Repräsentation von Wissen im Semantic Web. Aufbauend auf RDF und RDFS definiert OWL ein Vokabular, mit dessen Hilfe sich die Bedeutung der im Internet vertretenen Begriffe beschreiben lässt. Aus dieser formalen Definition können durch geeignete Tools Fakten abgeleitet werden, die nicht explizit in der Ontologie vorhanden sind. Das W3C hat diese im Folgenden vorgestellte Spezifikation am 10. Februar 2004 als Standard verabschiedet.

1.2 Ontologie

Eine Ontologie enthält Wissen über einen speziellen Bereich. Dieses Wissen ist in formalen Definitionen abgelegt, die Beschreibungen der Konzepte sowie den untereinander auftretenden Beziehungen enthalten. Eine Ontologie stellt somit ein Netzwerk von Informationen mit logischen Relationen dar. In der Literatur finden sich folgende Definitionen einer Ontologie:

> *„Ontologie ist ein aus der Philosophie entliehener Ausdruck, der sich auf die Wissenschaft bezieht, die die unterschiedlichen Arten von Entitäten in der Welt und ihre Beziehungen untereinander beschreibt.“* - w3c OWL Web Ontology Language Guide

> *„Eine Ontologie ist eine explizite formale Spezifikation einer gemeinsamen Konzeptualisierung.“* - T. R. Gruber: A translation approach to portable ontologies. In: Knowledge Acquisition, Band 5, Nummer 2, Seite 199-220, 1993

Der Nutzen von Ontologien liegt in den formalen Definitionen begründet. Hierdurch erhalten Menschen, sowie Maschinen, ein gemeinsames Verständnis von Informationen. Es spielt also keine Rolle mehr, ob nun Menschen mit anderen Menschen, Software-Agenten mit anderen Software-Agenten oder Menschen mit Software-Agenten kommunizieren. Jeder Kommunikationspartner weiß, wovon der andere spricht.

Eine wichtige Eigenschaft von Ontologien ist das logische Folgern (Inferenz). Anhand der formalen Definitionen lassen sich Rückschlüsse aus den vorhanden Daten ziehen, Widersprüche erkennen und fehlendes Wissen selbstständig aus dem Vorhandenen ergänzen. Eine schlussfolgernde Software (Reasoner) könnte aus einer Ontologie die sich mit Fahrzeugen, Bussen, Personen, Fahrern und Busfahrern beschäftigt, folgendes schließen:

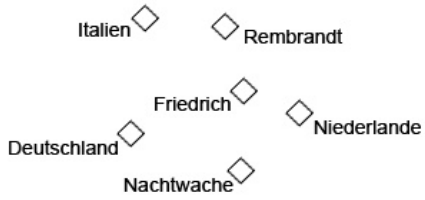
- Busfahrer sind Personen und fahren einen Bus
- Ein Bus ist ein Fahrzeug
- Ein Busfahrer fährt ein Fahrzeug, also muss er ein Fahrer sein

Ohne konkrete formale Definitionen der Begriffe wirken diese Schlussfolgerungen natürlich aus der Luft gegriffen, sollen jedoch einen ersten kleinen Einstieg vermitteln, was Reasoner anhand von Ontologien ermitteln können.

Im folgenden Abschnitt werden nun Instanzen, Eigenschaften und Konzepte, die Bestandteile einer Ontologie sind, vorgestellt. Hierdurch soll eine Basis für die später folgende Beschreibung der Web Ontology Language geschaffen werden.

1.2.1 Instanzen

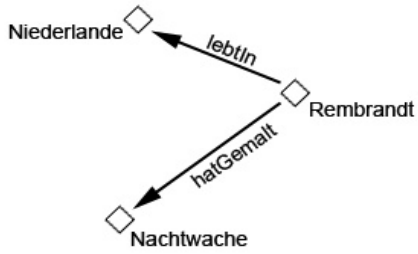
Instanzen stellen Objekte innerhalb des definierten Wissensbereichs einer Ontologie dar. Sie werden anhand der später vorgestellten Konzepte erzeugt und können mit Eigenschaften versehen werden. Instanzen werden auch als Individuen bezeichnet.



Anhand der oberen Abbildung wissen wir bisher nur, dass die Individuen *Italien*, *Deutschland*, *Niederlande*, *Rembrandt*, *Friedrich* und *Nachtwache* in unserer Ontologie enthalten sind. Weitere Aussagen können wir noch nicht machen.

1.2.2 Eigenschaften

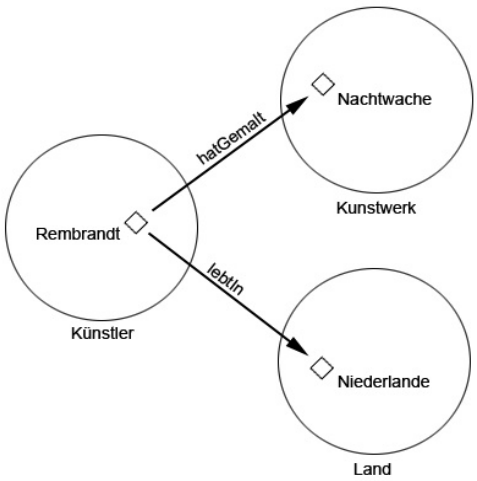
Eigenschaften verbinden Individuen miteinander. Sie beschreiben die speziellen Beziehungen eines Individuums mit dessen Umwelt. Eigenschaften stellen hierbei immer eine binäre Relation dar.



Die obere Abbildung zeigt, dass das Individuum *Rembrandt* die Eigenschaften *hatGemalt* und *lebtIn* besitzt. Rembrandt ist so mit den Individuen *Niederlande* und *Nachtwache* verbunden. Zu welchen Konzepten diese Individuen gehören, wird im folgenden Abschnitt deutlich.

1.2.3 Konzepte

Konzepte beschreiben gemeinsame Eigenschaften von Objekten. Sie werden auch als Klassen bezeichnet und können mit dem Klassenkonzept in objektorientierten Programmiersprachen verglichen werden. Wie in diesen Sprachen auch, können Konzepte in Taxonomien, also in einfache Hierarchien, mit Ober- und Unterklassen dargestellt werden.



Anhand der Abbildung sehen wir das *Rembrandt* zum Konzept *Künstler*, *Nachtwache* zum Konzept *Kunstwerke* und *Niederlande* zum Konzept *Land* gehört.

2. Web Ontology Language (OWL)

2.1 Entstehungsgeschichte

Die Web Ontology Language wurde aufbauend auf dem Ressource Description Framework (RDF) entwickelt. Die Arbeitsgruppe die sich hiermit beschäftigen sollte, wurde im Jahr 2001 vom W3C ins Leben gerufen. Diese Notwendigkeit bestand, da sich mit Hilfe des RDF Vokabulars zwar Klassen, Eigenschaften und einfache Einschränkungen definieren ließen, eine Unterstützung von Datentypen, Aufzählungen, Mengenoperationen etc. aber fehlte.

Schon bevor diese Arbeitsgruppe gegründet wurde, gab es eine Reihe von konkurrierenden Ontologiesprachen. Hierzu gehörte zum einen die Sprache DAML-ONT, die von mehreren amerikanischen Universitäten und Unternehmen entwickelt wurde. Der direkte Konkurrent hierzu war die Sprache OIL (Ontology Inference Layer), welche von europäischen Universitäten entwickelt wurde. Um jedoch eine gemeinsame Basis für die OWL Entwicklung zu schaffen, einigten sich beide Parteien darauf, ihre Ontologiesprachen zu verschmelzen (DAML+OIL) und als Ausgangspunkt für OWL zu benutzen. Am 10. Februar 2004 wurde die Entwicklung abgeschlossen und OWL als Standard des W3C verabschiedet.

Die Abkürzung OWL anstatt WOL entstand aus einem Vorschlag eines Entwicklers heraus, der sich mit den Vorteilen der vertauschten Abkürzung auseinander setzte. Seine Argumente waren die folgenden:

- einfachere Aussprache und einfach zu verstehen
- bessere Möglichkeiten Logos zu entwickeln
- Eulen repräsentieren die Klugheit, Weisheit

Seine Argumente wurden erhört und so einigte man sich auf die einzige Inkonsistenz der Web Ontology Language, in der sich sonst alles um Konsistenz dreht.

2.2 Ausprägungen

Die Web Ontology Language stellt drei Untersprachen mit zunehmender Ausdrucksstärke bereit. Diese Unterteilung wurde gemacht, um den Bedürfnissen unterschiedlicher Benutzergruppen entgegen zu kommen.

OWL Lite

OWL Lite ist der einfachste Vertreter der drei Untersprachen und richtet sich vorwiegend an Benutzer die eine einfache Klassifikationshierarchie und einfache Restriktionsmöglichkeiten benötigen. OWL Lite ist hierbei eine Erweiterung einer eingeschränkten Sicht auf RDF. Diese Einfachheit garantiert, dass Menschen OWL Lite Dokumente leicht verstehen können. Angaben zur Kardinalität einer Eigenschaft können hier zum Beispiel nur mit 0 oder 1 gemacht werden.

OWL DL

OWL DL ist eine eingeschränkte Version von OWL Full. Sie bietet maximale Ausdrucksstärke und komplexe Restriktionen bei gleichzeitiger Zusicherung, dass Daten in endlicher Zeit ausgewertet werden können. OWL DL ist wie Lite auch, eine Erweiterung einer eingeschränkten Sicht auf RDF. Das Kürzel DL steht in diesem Zusammenhang für Description Logic, welche eine Teilmenge der Prädikatenlogik erster Stufe darstellt.

OWL Full

OWL Full ist die ausdrucksstärkste Version der OWL Sprachen. Sie ist eine echte Erweiterung von RDF, garantiert im Gegensatz zu Lite und DL jedoch nicht mehr, dass Daten wirklich ausgewertet werden können. Da die Beziehungen in OWL Full Dokumenten derart komplex sein können, ist es unwahrscheinlich, dass eine Software jedes Feature unterstützen wird.

2.3 Aufbau eines Dokuments

2.3.1 Namensraum

Die Angaben der verwendeten Namensräume leiten jedes OWL-Dokument ein. Hier definiert der Entwickler der Ontologie, auf welches Vokabular er in seinen Definitionen zurückgreift, um eine Fehlinterpretation ausschließen zu können.

```
<rdf:RDF
    xmlns     ="http://www.ckoehn.de/seminar#"
    xml:base  ="http://www.ckoehn.de/seminar#"
    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#">

    <Ontologie Header>

    <Ontologie>
</rdf:RDF>
```

Das Wurzelelement einer jeden Ontologie bildet das aus dem RDF Namensraum stammende `rdf:RDF` Element. Innerhalb dieses Elements werden alle weiteren Definitionen, wie Metainformationen über die Ontologie und natürlich die Ontologie selber, ablegen. Als Attribute erhält dieses Element Angabe über die verwendeten Namensräume.

Die ersten beiden Definitionen bilden hierbei den verwendeten Standard-namensraum. Sie geben an, worauf sich Elemente ohne Angabe eines Präfixes beziehen. In diesem Fall auf die URI *http://www.ckoehn.de/seminar#*. Die folgenden vier Deklarationen sagen aus, dass sich die Präfixe `owl:`, `rdf:`, `rdfs:` und `xsd:` jeweils auf die Spezifikationen des W3C beziehen.

Die hier außerdem noch aufgeführten Elemente `<Ontologie Header>` und `<Ontologie>` stellen Platzhalter für später vorgestellte Angaben innerhalb eines OWL-Dokuments dar.

2.3.2 Ontologie Header

Im Ontologie Header können Metainformationen über die definierte Ontologie abgelegt werden. Hier kann der Entwickler Kommentare verfassen, Angaben zu verschiedenen Versionen seines Dokuments machen, sowie Aussagen darüber, welche Fremdontologien importiert werden.

```
<owl:Ontology rdf:about="">
  <rdfs:comment>Meine Seminar-Ontologie</rdfs:comment>

  <rdfs:label>Kuenstler Ontologie</rdfs:label>

  <owl:priorVersion
      rdf:resource="http://www.ckoehn.de/seminar.01#"/>

  <owl:imports
      rdf:resource="http://www.ckoehn.de/eineAndere#"/>
</owl:Ontology>
```

Eingeleitet wird der Ontologie Header durch das `owl:Ontology` Element. Das Attribut `rdf:about` erlaubt es in diesem Fall einen Namen für die Ontologie zu vergeben. Ist die Angabe leer, wird die URI des zuvor definierten Standard-namensraums verwendet.

Der Inhalt des `owl:Ontology` Elements befasst sich mit der Angabe von Kommentaren (`rdfs:comment`), dem Festlegen des Namens (`rdfs:label`), mit der Auszeichnung von früheren Versionen (`owl:priorVersion`) und mit importieren Fremdontologien (`owl:imports`). Das `rdf:resource` Attribut nimmt hierbei jeweils die URI der früheren Version, sowie der importieren Ontologie auf.

Besonders hervorzuheben sei hierbei die Angabe über den Import andere Ontologien. Hiermit ist es dem Benutzer möglich, neue Ontologien mit verhältnismäßig wenig Aufwand zu erstellen. Es lassen sich so komplette Fremdontologien als Ausgangspunkt für die Eigene benutzen. Durch diese Angabe werden alle Aussagen, die in der importierten Ontologie getroffen werden, ebenso wie die dort importierten Ontologien, in die Eigene inkludiert. Durch diesen Mechanismus muss nicht jeder Entwickler einer Ontologie das Rad neu erfinden, sondern kann sich so auf schon etablierte und geprüfte Ontologien stützen.

2.4 Sprachkonstrukte

Nachdem nun die verwendeten Namensräume und die Metainformationen der Ontologie definiert worden sind, folgt die eigentliche Spezifikation der Ontologie. Hierbei wird anhand eines durchgängigen Beispiels erklärt, wie mit Hilfe des Web Ontology Language Vokabulars Klassen, Eigenschaften sowie Merkmale und Einschränkungen von Eigenschaften definiert werden können. Unsere Ontologie, die wir im Folgenden aufbauen wollen, beschäftigt sich mit Künstlern, Kunstwerken und den Beziehungen die zwischen diesen Objekten herrschen.

2.4.1 Klassen

Die Grundlage einer Ontologie bilden Klassen. Sie definieren gemeinsame Eigenschaften von Dingen (vgl. 1.2.3 Konzepte). In OWL werden sie mit Hilfe des Präfixes `owl:Class` eingeleitet. Alle so definierten Klassen sind implizit Unterklasse von `owl:Thing`. Sie stellt die vordefinierte Superklasse der Web Ontology Language dar und ist somit die allgemeinste Klasse. Alle Individuen einer OWL Ontologie sind Mitglied dieser Klasse. Die Mitglieder einer Klasse werden auch als die Extension einer Klasse bezeichnet.

OWL stellt außerdem die vordefinierte leere Klasse `owl:Nothing` zur Verfügung. Sie ist die speziellste Klasse, besitzt keine Individuen und ist Unterklasse aller OWL Klassen.

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

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

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

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

Der obere Codeabschnitt definiert die Klassen *Kunstwerk*, *Kuenstler*, *Gebaeude* und *Technik*. Der Name wird hierbei durch die Verwendung des Attributs `rdf:ID` festgelegt. Um in späteren Definitionen die Klasse *Kunstwerk* referenzieren zu können, wird die Notation *#Kunstwerk* verwendet. Dieses gilt jedoch nur für Referenzen im gleichen Dokument. Um Klassen aus entfernten Ontologien zu referenzieren, wird die URI des Dokuments mit nachgestelltem Bezeichner

verwendet. Dies könnte zum Beispiel so aussehen: *http://www.ckoehn.de/eineAndereOntologie#EinAnderesObjekt.*

Diese Referenzen tauchen in OWL Dokumenten in zwei Kontexten auf. Zum einen im Attribut `rdf:resource` und zum anderen im Attribut `rdf:about`. `rdf:resource` ist hierbei für die einfache Referenz auf eine bestimmte Ressource (Klasse, Eigenschaft etc.) vorgesehen auf die sich bezogen wird, während mit Hilfe von `rdf:about` eine Ressource angegeben wird, deren Definition erweitert werden soll. So lassen sich zum Beispiel Klassen aus fremden Ontologien mit eigenen Definitionen versehen, ohne die fremde Ontologie in das eigene Dokument kopieren zu müssen und dann die Modifikation vorzunehmen.

Um nun eine hierarchische Struktur in unserer Beispielontologie zu implementieren, werden wir eine Reihe von Unterklassen anlegen. Hierbei greift OWL auf auf die RDFS Vokabel `rdfs:subClassOf` zurück, mit der eine Klasse als eine Unterklasse einer anderen Klasse definiert werden kann.

```
<owl:Class rdf:ID="Bild">
    <rdfs:subClassOf rdf:resource="#Kunstwerk"/>
</owl:Class>

<owl:Class rdf:ID="Skulptur">
    <rdfs:subClassOf rdf:resource="#Kunstwerk"/>
</owl:Class>

...
```

Im oberen Codebeispiel sehen wir die Definition der Klassen *Bild* und *Skulptur*, die beide Unterklassen von *Kunstwerk* sind. Die Definitionen der Klassen *Maler* und *Bildhauer* und die Definitionen der Klassen *Galerie* und *Museum* wurden hier weggelassen. Die nun entstandene Klassenhierarchie ist der *Abbildung 1 im Anhang* zu entnehmen.

2.4.2 Individuen

Individuen stellen Mitglieder von Klassen dar. Sie werden als die speziellen Ausprägungen eines Konzeptes bezeichnet. Im folgenden Codeabschnitt werden die Maler *Rembrandt* und *Friedrich*, die Bilder *FrauImFenster*, *DerSommer* und *DieNachtwache* in der Ontologie definiert.

```
<Maler rdf:ID="Rembrandt" />
<Maler rdf:ID="Friedrich" />

<Bild rdf:ID="FrauImFenster" />
<Bild rdf:ID="DerSommer" />
<Bild rdf:ID="DieNachtwache" />
```

Die Definition von Instanzen wird durch den Bezeichner der Klasse, in der das Individuum Mitglied ist, eingeleitet und erhält durch das schon bekannte `rdf:ID` Attribut einen Namen. Hierbei ist wichtig zu wissen, dass Instanzen mit verschiedenen Namen in OWL per se nicht zwangsläufig auch unterschiedlich sind. Zwei Individuen könnten unterschiedlich sein, sie könnten aber auch gleich sein. Das Gleiche gilt ebenso für Klassen und Eigenschaften. Wie sich Gleich- und Ungleichheit von Klassen, Eigenschaften und Instanzen ausdrücken lässt, wird in *2.4.6 Gleich- und Ungleichheit* behandelt.

Die zweite Möglichkeit, die Instanz *Rembrandt* zu erzeugen und als Mitglied der Klasse *Maler* zu definieren, ist im folgenden Abschnitt demonstriert.

```
<owl:Thing rdf:ID="Rembrandt">
    <rdf:type rdf:resource="#Maler"/>
</owl:Thing>
```

Hierbei wird zuerst eine Instanz *Rembrandt* als `owl:Thing` definiert und mit Hilfe von `rdf:type` als Mitglied der Klasse *Maler* registriert.

2.4.3 Eigenschaften

Eigenschaften stellen binäre Relationen dar und verbinden Instanzen mit anderen Instanzen oder Instanzen mit Datentypen. Hierbei wird der erste Typ in OWL als *Object property* und der Zweite als *Datatype property* bezeichnet. Zulässige Datentypen sind alle in der XML-Schema Definition vom W3C festgelegten oder eine RDF Literal Angabe. Zulässige XML-Schema Datentypen sind zum Beispiel: `xsd:string` und `xsd:integer`. Die zugehörige URI hierfür lautet: *http://www.w3.org/2001/XMLSchema.*

Nachfolgend wird eine *Object property* definiert, die Künstler mit Kunstwerken verbindet, und eine *Datatype property* die es erlaubt, einer Instanzen einen Namen zuzuweisen.

```
<owl:ObjectProperty rdf:ID="erzeugt">
    <rdfs:range rdf:resource="#Kunstwerk"/>
    <rdfs:domain rdf:resource="#Kuenstler"/>
</owl:ObjectProperty>
```

Objekteigenschaften werden mit Hilfe des `owl:ObjectProperty` Elements definiert. Durch das schon bekannte Attribut `rdf:ID` wird der Eigenschaft ein Bezeichner zugeordnet. Zusätzlich können Entwickler Angaben zur Grundmenge (`rdfs:domain`), sowie zum Wertebereich (`rdfs:range`) der Eigenschaft machen. Durch das Element `rdfs:domain` wird die Grundmenge festgelegt. Das heißt, dass diese Eigenschaft Mitgliedern der dort angegebenen Ressource zugewiesen werden soll. `rdfs:range` reglementiert die Werte, die diese Eigenschaft annehmen kann. Die Eigenschaft *erzeugt* kann nun dazu verwendet werden festzulegen, dass ein Künstler ein bestimmtes Kunstwerk erzeugt hat.

Die zweite Art von Eigenschaften, die *Datatype property,* wird ähnlich definiert. Eingeleitet durch das Element `owl:DatatypeProperty` kann auch hier der Wertebereich, sowie die Grundmenge eingeschränkt werden. Der Unterschied ist jedoch, dass sich der Wertebereich aus XML-Schema Datentypen und RDF Literal zusammensetzt.

```
<owl:DatatypeProperty rdf:ID="name">
    <rdfs:range
        rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

Ebenso wie Klassen, lassen sich auch Eigenschaften in Hierarchien anordnen. Dies wird über das RDFS Element `rdfs:subPropertyOf` ausgedrückt. Wir wollen dieses Element nutzen, um die Eigenschaften *malt* und *schlaegt* als Untereigenschaften von *erzeugt* zu definieren.

```
<owl:ObjectProperty rdf:ID="malt">
    <rdfs:subPropertyOf rdf:resource="#erzeugt"/>
    <rdfs:domain rdf:resource="#Maler"/>
    <rdfs:range rdf:resource="#Bild"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="schlaegt">
    <rdfs:subPropertyOf rdf:resource="#erzeugt"/>
    <rdfs:domain rdf:resource="#Bildhauer"/>
    <rdfs:range rdf:resource="#Skulptur"/>
</owl:ObjectProperty>
```

Diese Definitionen schränken jeweils die Grundmenge (`rdfs:domain`) und den Wertebereich (`rdfs:range`) der Eigenschaft *erzeugt* weiter ein. *malt* verbindet Instanzen der Klasse *Maler* mit Instanzen der Klasse *Bild* und schlaegt verbindet *Bildhauer* mit *Skulptur*. Mit dem Attribut `rdfs:resource` wird hierbei festgelegt welches die Obereigenschaft darstellt.

Die Definition der Instanz *Rembrandt* könnte nun durch folgenden Ausschnitt ersetzt werden:

```
<Maler rdf:ID="Rembrandt">
   <malt rdf:resource="#DieNachtwache" />

   <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      Rembrandt Harmenszoon van Rijn
   </name>
<Maler>
```

2.4.4 Merkmale von Eigenschaften

Um Eigenschaften noch genauer spezifizieren zu können, stellt OWL fünf so genannte *Property Characteristica* bereit. Sie erlauben ein erweitertes Folgern aus Eigenschaften. Der folgende Abschnitt stellt alle fünf Merkmale kurz vor, wobei `owl:inverseOf` Verwendung in der Beispielontologie findet.

Transitive Eigenschaften

Eigenschaften können mit `owl:TransitiveProperty` als transitiv definiert werden. Dies bedeutet, dass wenn die Individuen a und b und die Individuen b und c über die gleiche transitive Eigenschaft verbunden sind, geschlussfolgert werden kann, dass auch a und c über diese Eigenschaft in Verbindung stehen. Beispiele für eine solche Eigenschaft wären *hatVorfahre* oder *lokalisiertIn*.

```
<owl:TransitiveProperty rdf:ID="hatVorfahre">
    ...
</owl:TransitiveProperty>
```

Symmetrische Eigenschaften

Symmetrische Eigenschaften sind so definiert, dass wenn die Individuen a und b verbunden sind, daraus geschlossen werden kann dass auch b und a über die gleiche Eigenschaft verbunden sind. Ein Beispiel hierfür wäre eine Eigenschaft *hatNachbar*.

```
<owl:SymmetricProperty rdf:ID="hatNachbar">
    ...
</owl:SymmetricProperty>
```

Funktionale Eigenschaften

Über eine funktionale Eigenschaft kann ein Individuum mit höchstens einem anderen Individuum verbunden werden. *hatMutter* wäre ein Beispiel hierfür. Ein Individuum kann höchstens eine Mutter besitzen.

```
<owl:ObjectProperty rdf:ID="hatMutter">
    <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>
```

Anhand des Codebeispiels ist außerdem eine alternative Auszeichnung der Eigenschaftsmerkmale ersichtlich. Innerhalb des `owl:ObjectProperty` Elementes wird die Eigenschaft als Mitglied der Ressource `&owl;FunctionalProperty` registriert. Diese Schreibweise kann für transitive, symmetrische, funktionale sowie invers-funktionale Eigenschaften benutzt werden.

Inverse Eigenschaften

Mit `owl:inverseOf` lässt sich ausdrücken, dass eine Eigenschaft das Gegenteil einer anderen Eigenschaft ist. Wenn eine Instanz i1 über die Eigenschaft a mit der Instanz i2 verbunden ist und b die inverse Eigenschaft zu a darstellt, dann kann hieraus gefolgert werden, dass i2 über die Eigenschaft b mit i1 verbunden ist.

In der Beispielontologie werden wir diese Merkmal benutzen, um die Eigenschaften *erzeugtVon*, *geschlagenVon* und *gemaltVon* einzuführen. Sie stellen die inversen Eigenschaften zu den vorher definierten *erzeugt*, *schlaegt* und *malt* dar.

```
<owl:ObjectProperty rdf:ID="erzeugtVon">
    <owl:inverseOf rdf:resource="#erzeugt"/>
    <rdfs:domain rdf:resource="#Kunstwerk"/>
    <rdfs:range rdf:resource="#Kuenstler"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="geschlagenVon">
    <rdfs:subPropertyOf rdf:resource="#erzeugtVon"/>
    <owl:inverseOf rdf:resource="#schlaegt"/>
    <rdfs:domain rdf:resource="#Skulptur"/>
    <rdfs:range rdf:resource="#Bildhauer"/>
</owl:ObjectProperty>

...
```

Die Beziehungen, die nun zwischen den Klassen *Kuenstler* und *Kunstwerk*, *Maler* und *Bild*, sowie Bildhauer und Skulptur herrschen, kann der *Abbildung 2 im Anhang* entnommen werden.

Invers-funktionale Eigenschaften

Eine invers-funktionale Eigenschaft in OWL sagt aus, dass die inverse Eigenschaft funktional ist. Als funktionale Eigenschaft haben wir zuvor *hatMutter* definiert, da jedes Individuum nur eine Mutter haben kann. Die invers funktionale Eigenschaft hierzu wäre *istMutterVon*. Sollten jetzt die Individuen a und b und die Individuen c

und b über diese Eigenschaft verbunden sein, kann hieraus geschlossen werden, dass a und c das selbe Individuum sein müssen. Gebildet wird eine invers-funktionale Eigenschaft über das OWL Element `owl:InversFunctionalProperty` oder wie im Abschnitt *Funktionale Eigenschaften* gesehen, über das `rdf:type` Element.

2.4.5 Einschränkungen von Eigenschaften

Einschränkungen sind in OWL vorgesehen, um den Wertebereich und die Kardinalität einer Eigenschaft für spezifische Ressourcen festzulegen. Hierbei werden die Definitionen innerhalb eines `rdfs:subClassOf` und `owl:Restriction` Elements notiert.

Wertebereich

Um den Wertebereich einer Eigenschaft in spezifischen Kontexten einschränken zu können, werden in OWL die Elemente `owl:allValuesFrom` und `owl:someValuesFrom` benutzt. Bisher konnten wir die Grundmenge und den Wertebereich nur global einschränken. Hierzu wurden die Angaben von `rdfs:domain` und `rdfs:range` innerhalb der Objekt- oder Datentypeigenschaft benutzt (vgl. *2.4.3 Eigenschaften*). Jetzt wird es jedoch möglich, schon bei einer Klassendefinition festzulegen, dass Eigenschaften von Instanzen dieser Klasse nur Werte aus einem bestimmten Wertebereich annehmen dürfen.

```
<owl:Class rdf:ID="Kuenstler">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#erzeugt"/>
        <owl:allValuesFrom rdf:resource="#Kunstwerk"/>
      </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

`owl:onProperty` gibt hierbei an, welche Eigenschaft eingeschränkt werden soll, während mit `owl:allValuesFrom` ausgesagt wird, dass nur Kunstwerke als mögliche Werte in frage kommen. Es wird aber nicht gefordert, dass ein Künstler eine solche Eigenschaft besitzen muss.

Durch Ersetzen von `owl:allValuesFrom` durch `owl:someValuesFrom` wird gefordert, dass mindestens einer der Werte ein Kunstwerk sein muss. Um dieser Forderung nachzukommen, müssen alle Individuen der Klasse *Kuenstler* eine Eigenschaft *erzeugt* aufweisen. Hierbei könnten jetzt auch Werte auftreten, die nicht Mitglied der Klasse *Kunstwerk* sind, solange mindestens ein Kunstwerk angegeben wurde.

Kardinalität

Bisher konnten in unsere Beispielontologie noch keine Aussagen darüber gemacht werden, wie oft eine Instanz eine bestimmte Eigenschaft besitzen darf. Mit Hilfe von `owl:cardinality`, `owl:minCardinality` und `owl:maxCardinality` kann jetzt die exakte Anzahl, sowie eine Ober- und Untergrenze definiert werden.

```
<owl:Class rdf:ID="Kuenstler">
...
<rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource="#erzeugt"/>
        <owl:minCardinality
            rdf:datatype="&xsd;nonNegativeInteger">1
        </owl:minCardinality>
    </owl:Restriction>
</rdfs:subClassOf>
...
</owl:Class>
```

Wir wollen `owl:minCardinality` dazu benutzen, um auszusagen, dass ein Künstler mindestens eine Eigenschaft *erzeugt* aufweist. Die Verwendung der anderen beiden Sprachkonstrukte geschieht analog zum oberen Beispiel.

hasValue Einschränkung

`owl:hasValue` kann benutzt werden, um Klassen zu konstruieren die auf der Existenz eines Eigenschaftswerts basieren. Wenn eine Instanz die angegebene Eigenschaft besitzt, ist sie automatisch Mitglied diese Klasse. Mit Hilfe dieses Konstruktes wollen wir die Beispielontologie um die Klasse *RembrandtsBilder* erweitern.

```
<owl:Class rdf:ID="RembrandtsBilder">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#erzeugtVon"/>
            <owl:hasValue rdf:resource="#Rembrandt" />
        </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
```

Der Codeausschnitt demonstriert, wie die Klasse *RembrandtsBilder,* aufbauend auf der Eigenschaft *erzeugtVon* mit dem Wert *Rembrandt,* definiert wird. Alle Instanzen der Ontologie, die eine Eigenschaft *erzeugtVon* aufweisen, deren Wert *Rembrandt* ist, sind nun Mitglied dieser Klasse.

Als Änderung zu den vorhergehenden Beispielen wird hier für die Klassendefinition eine anonyme Klasse verwendet, und diese mit Hilfe von `owl:equivalentClass` als äquivalent zu *RembrandtsBilder* definiert. Genaueres zur Gleichheit findet sich im folgenden Abschnitt.

2.4.6 Gleich- und Ungleichheit

Bei der Entwicklung einer neuen Ontologie sollte sich ein Entwickler umschauen und schon vorhandene Fremdontologien nutzen, um seine Eigene zu modellieren. Hierbei gibt es oft die Notwendigkeit zu sagen, dass zwei Individuen, Klassen oder Eigenschaften das Gleiche oder etwas Unterschiedliches beschreiben.

Der folgende Abschnitt soll einen kurzen Überblick darüber verschaffen, wie sich dies mit Hilfe der OWL Syntax ausdrücken lässt.

Klassen und Eigenschaften

- `owl:equivalentClass`
- `owl:disjointWith`
- `owl:equivalentProperty`

Die Verwendung von `owl:equivalentClass` und `owl:equivalentProperty` ist dafür vorgesehen, um über Klassen oder Eigenschaften auszusagen, dass sie exakt das Gleiche beschreiben. Eine Anwendung hiervon haben wir bereits bei der hasValue Einschränkung (vgl. 2.4.5 Einschränkungen von Eigenschaften) gesehen. Durch die Angabe von `owl:disjointWith` lassen sich Klassen als unterschiedlich

definieren. Eine Instanz kann so nicht gleichzeitig Mitglied der markierten Klassen sein. In unserer Beispielontologie könnte dies für *Gebaeude*, *Kunstwerk* und *Kuenstler* verwendet werden.

Individuen

- `owl:sameAs`
- `owl:differentFrom`
- `owl:AllDifferentFrom`

Mit der Angabe von `owl:sameAs`, bei der Definition eines Individuums, lässt sich aussagen, dass dieses äquivalent zu einem anderen schon definierten Individuum ist. Mit dieser Angabe versehen, würden zum Beispiel die Individuen *Gebiet* und *Region* das Gleiche beschreiben. Die gegenteilige Wirkung ließe sich mit `owl:differentFrom` erzielen. Es markiert zwei Individuen gezielt als unterschiedlich. Hierbei muss diese Angabe jedoch für jedes Instanzpaar einmal angegeben werden. Um mehrere Individuen, die alle als unterschiedlich markiert werden sollen, auflisten zu können, eignet sich `owl:AllDifferentFrom`.

2.4.7 Komplexe Klassen

In OWL ist es möglich durch die Benutzung von Mengenoperationen oder Aufzählungen neue Klassen zu konstruieren.

Mengenoperationen

- `owl:intersectionOf`
- `owl:unionOf`
- `owl:complementOf`

Durch die Verwendung der oberen drei Elemente, lassen sich neue Klassen als Schnitt-, Vereinigungs- oder Komplementärmenge zweier andere Klassen definieren. `owl:oneOf` erlaubt es, durch Aufzählung, die Klassenmitglieder explizit festzulegen. Eine Erweiterung der Klasse um weitere Mitglieder ist ausgeschlossen. Vorstellbar wäre es hier zum Beispiel, die Klasse Wochentage durch Aufzählung der einzelnen Tage zu definieren.

3. Anhang

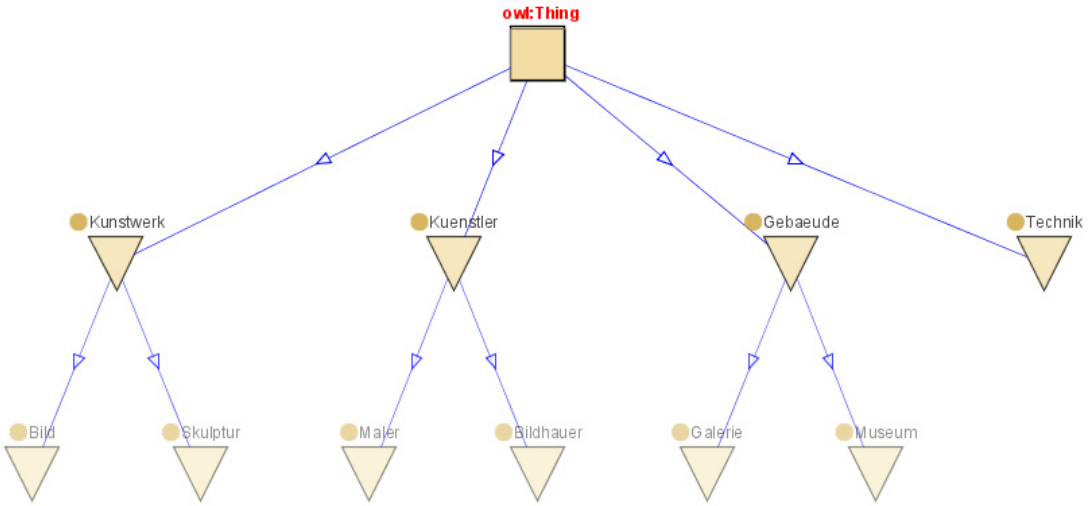


Abbildung 1: Klassenhierarchie

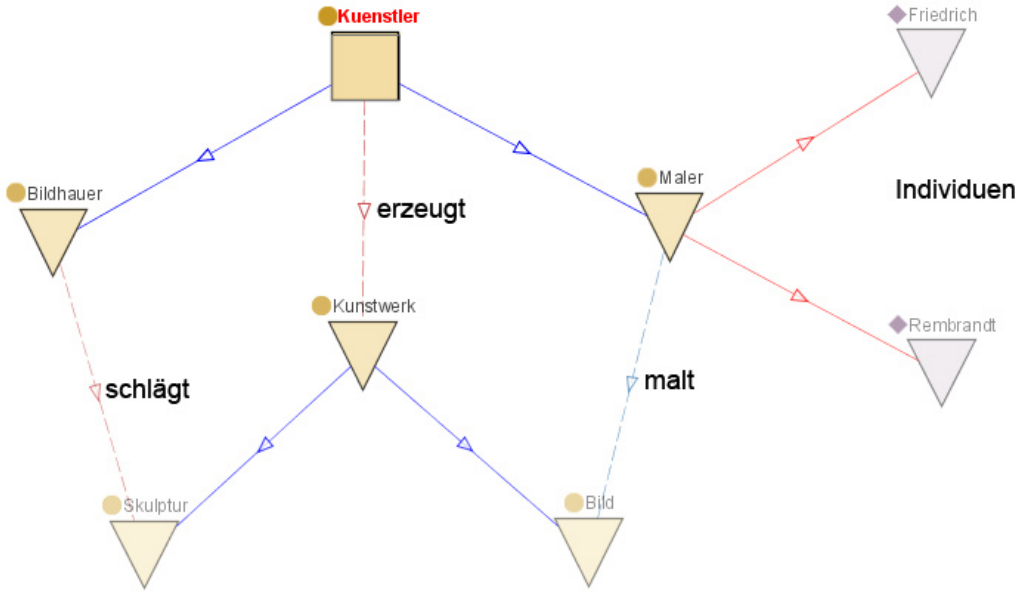


Abbildung 2: Definition der Eigenschaften

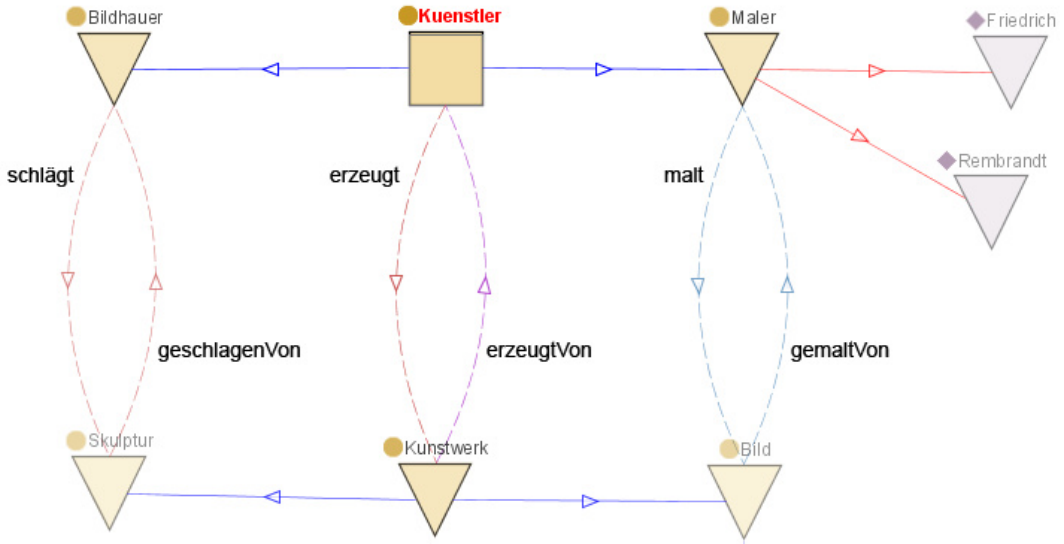


Abbildung 3: Definition der inversen Eigenschaften

4. Quellen

W3C Spezifikation der Web Ontology Language:

http://www.w3.org/2004/OWL/

http://www.w3.org/TR/owl-features/

http://www.w3.org/TR/owl-guide/

http://www.w3.org/TR/owl-ref/

http://www.w3.org/TR/webont-req/

Andere Quellen:

http://de.wikipedia.org/wiki/Ontologie

http://de.wikipedia.org/wiki/Web_Ontology_Language

http://en.wikipedia.org/wiki/Web_Ontology_Language

http://www.co-ode.org/resources/tutorials/intro/ontologies-intro-v04.pdf

http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

http://lists.w3.org/Archives/Public/www-webont-wg/2001Dec/0169.html

http://www.ics.forth.gr/isl/swprimer/presentations/Chapter4.ppt

http://www.ics.forth.gr/isl/swprimer/index2.php?selected=1&opened=1

http://www.w3schools.com/rdf/rdf_owl.asp