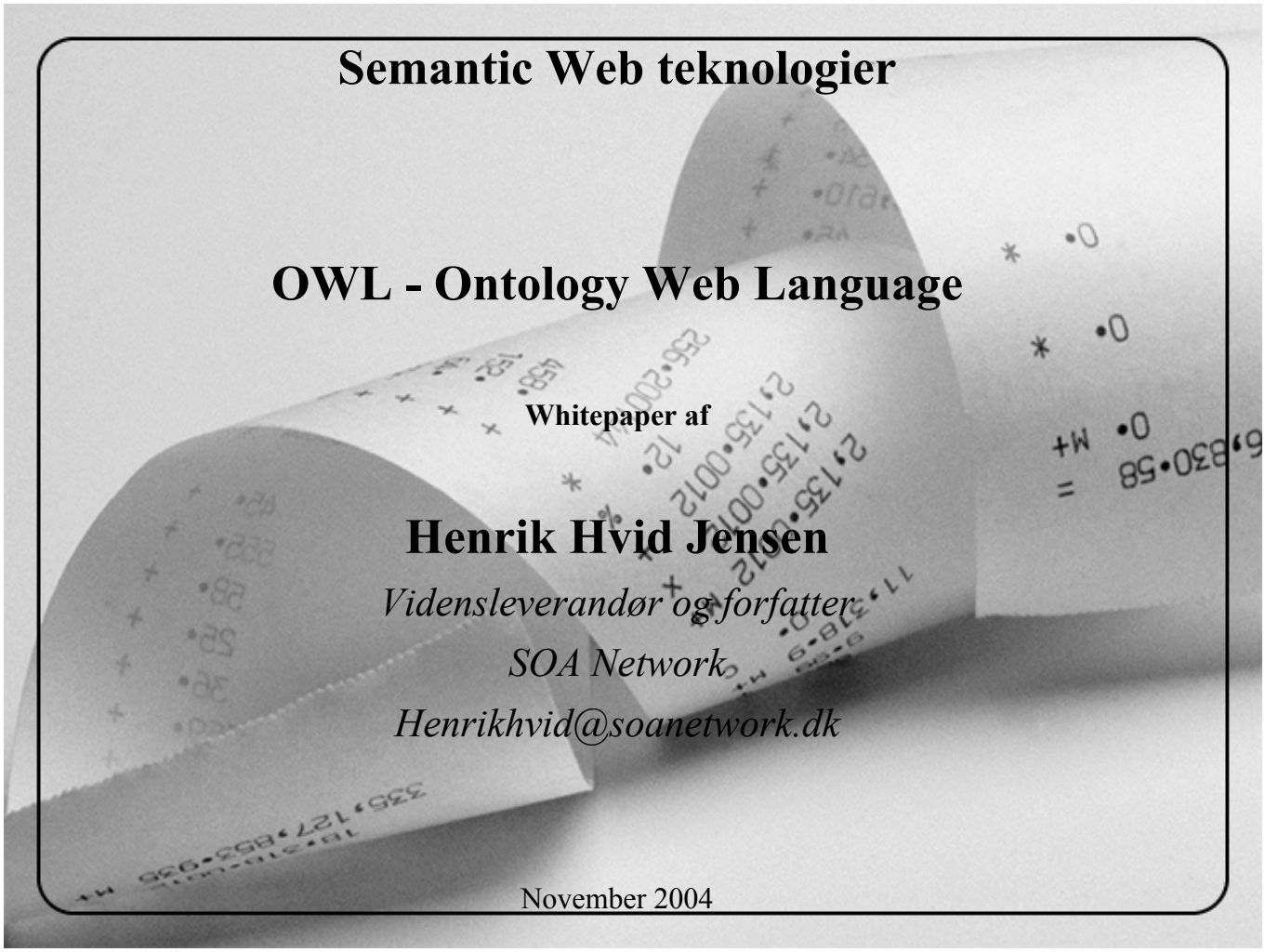


Semantic Web teknologier

OWL - Ontology Web Language

Whitepaper af

Henrik Hvid Jensen

Vidensleverandør og forfatter

SOA Network

Henrikhvid@soanetwork.dk

November 2004

"Når man fortæller en anden person noget, kan denne kombinere den nye viden med gammel og fortælle en noget nyt. Når man fortæller en computer noget i XML, kan det måske fortælle en noget nyt, men det er kun på grund af noget software, det bruger og som ikke er del af XML specifikationen. Dette software kunne være implementeret forskelligt på forskellige computere, samtidig med at de stadig overholder XML specifikationen. Man vil derfor måske få forskellige svar fra disse systemer. Når man fortæller en computer noget nyt i OWL, kan den give en ny information, baseret alene på OWL standarden."

"En mængde af OWL sætninger, tillader selv at konkludere andre OWL sætninger, hvorimod en samling af XML sætninger, ikke tillader XML selv at konkludere andre XML sætninger. For at udnytte XML til at generere ny data, er det nødvendigt at inkludere viden i noget kode et eller andet sted. I stedet for at angive det eksplicit som i OWL"

Indholdsfortegnelse

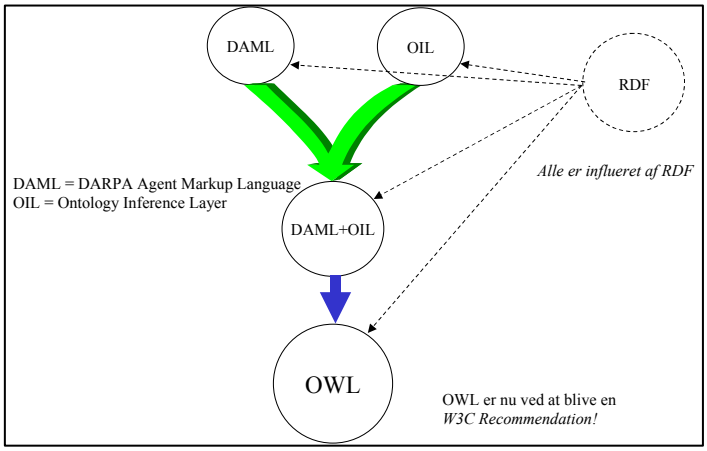
Figurer

Eksempler

Kapitel 1 Ontology Web Language - OWL

RDF leverer de basale sæt af funktioner til modellering af information. Det er simpelt at bruge og kan betragtes som et slags assembler sprog, ovenpå hvilket næsten enhver anden informations modellering metode kan placeres [IDAMLI]. Men dets simpelthed gør, at det også mangler nogle faciliteter såsom data typer, en konsistent måde at udtrykke lister og lignende.

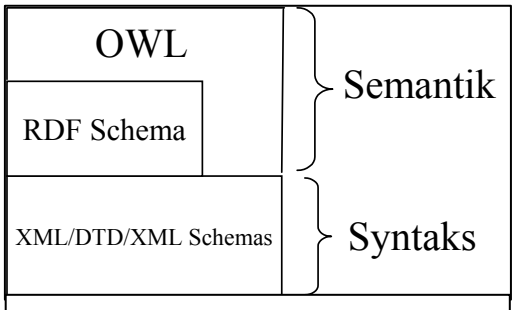
For at imødegå dette udformede DARPA (afdeling af US forsvarsministerium) et ontologi sprog kaldet DAML. Det blev snart samlet med et europæisk initiativ kaldet Ontology Inference Layer (OIL) og resulterede i DAML-OIL, et sprog til at udtrykke langt mere sofistikerede klassifikationer og egenskaber end RDFS



Figur 1-1 Udviklingen af OWL

Siden er W3C kommet på banen og har nedsat "The W3C Web Ontology Working Group" (WebONT) med det formål at definere "*A Web ontology language, that builds on current Web languages that allow the specification of classes and subclasses, properties and subproperties (such as RDFS), but which extends these constructs to allow more complex relationships between entities including: means to limit the properties of classes with respect to number and type, means to infer that items with various properties are members of a particular class, a well-defined model of property inheritance, and similar semantic extensions to the base languages*" [WEBONTCH].

WebONT har den 4. april 2003[1] frigivet fem "Last Call Working Draft" specifikationer for OWL Web Ontology Language version 1.0 (OWL). OWL bygger i høj grad på DAML-OIL specifikationerne (jvf. Figur



Figur 1-2 OWL, RDF Schema og XML

[1] Den 18. august 2003 udgav W3C Candidate Recommendation, evt. ændringer i forhold til tidligere version, vil ikke blive inkluderet i denne rapport.

1-1).

Formålet med OWL er ligesom RDF Schema at levere en XML vokabular til at definere klasser, deres egenskaber og relationerne mellem klasserne for at muliggøre semantiske definitioner, der kan tilgås fra maskiner. OWL er meget rigere end RDF Schema og giver mulighed for at kunne udtrykke relationerne meget mere nuanceret (se Figur 1-3). Alle elementer og attributter, som leveres af RDF og RDF Schema, kan bruges, når man danner et OWL dokument.

- *Lokale begrænsninger af egenskaber:* rdfs:range definerer range af en egenskab (f.eks. spiser) for alle klasser. Men i RDF Schema kan man ikke erklære range restriktioner, som kun gælder for enkelte klasser. For eksempel kan man ikke angive at køer kun spiser planter, mens andre dyr også kan spise kød.
- *Adskillelse af klasser:* Det kan være gavnligt, at angive at klasser er adskilte. For eksempel er mand og kvinde adskilte. Men i RDF Schema kan man kun angive subclass relationer, f.eks. at kvinde er en subclass af mennesker
- *Boolesk kombination af klasser:* Det kan være nyttigt at bygge nye klasser ved at kombinere andre klasser ved brug af forenings-, fælles- og komplementærmængder. For eksempel kunne det være nyttigt at kunne angive, at klassen mennesker er fællesmængden af klasserne kvinde og mand. Dette tillader RDF Schema ikke.
- *Mængde restriktioner:* Det kan være nyttigt at lægge restriktioner på hvor mange forskellige værdier en egenskab må have. For eksempel kan man sige at en person har præcis to forældre. Dette tillader RDF Schema ikke.
- *Specielle karakteristika af en egenskab:* Det kan være nyttigt at kunne angive at en egenskab er transitiv (såsom "større end"), unik (såsom "er mor til") eller er det modsatte af en anden egenskab (såsom "spiser" og "bliver spist af")

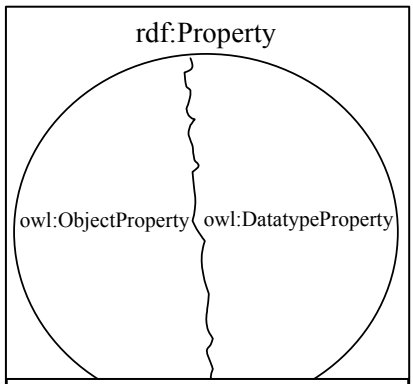
Figur 1-3 Eksempler på mangler ved RDFS.

Der er tre forskellige niveauer i alle ontologi sprog. Det første niveau er defineret af selve ontologisproget, såsom OWL. Semantikken af sproget er givet af sprogets aksiomer (grundsætninger). Ved at bruge sprog elementer fra det første niveau, kan man udtrykke bruger definerede ontologi klasser, underklasser, egenskaber ved dem osv., dette udgør det andet niveau. Det tredje niveau indeholder instanser af ontologien, såsom enkelte instanser som tilhører klassen defineret på andet niveau.

1.1 Brug af OWL til at definere egenskaber.

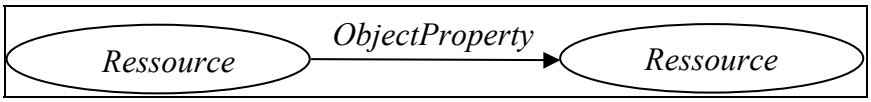
I afsnit **Fejl! Henvisningskilde ikke fundet.** blev der gjort rede for hvorledes RDF Schema tilbyder tre måder til karakterisering af en egenskab *range*, *domain* og *subPropertyOf*. Dette afsnit vil beskrive nogle yderligere muligheder som OWL tilbyder.

RDF schemas rdf:Property var brugt til både at relatere en ressource til en anden og til at relatere en ressource til en rdfs:Literal eller datatype. OWL har opdelt disse i 2 klasser af egenskaber og de har derfor hver deres klasse:



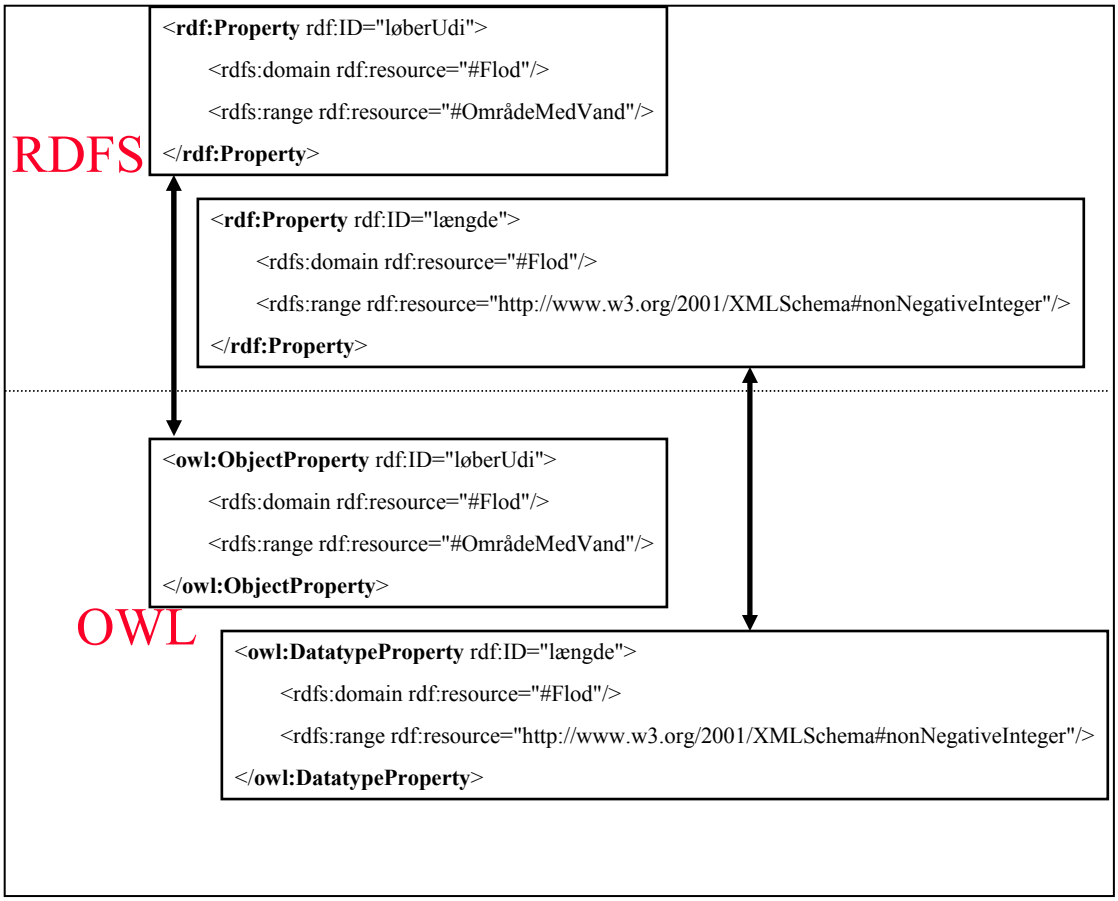
Figur 1-4 Opdeling af rdf:Property

Owl:ObjectProperty bruges til at relaterer en ressource til en anden ressource



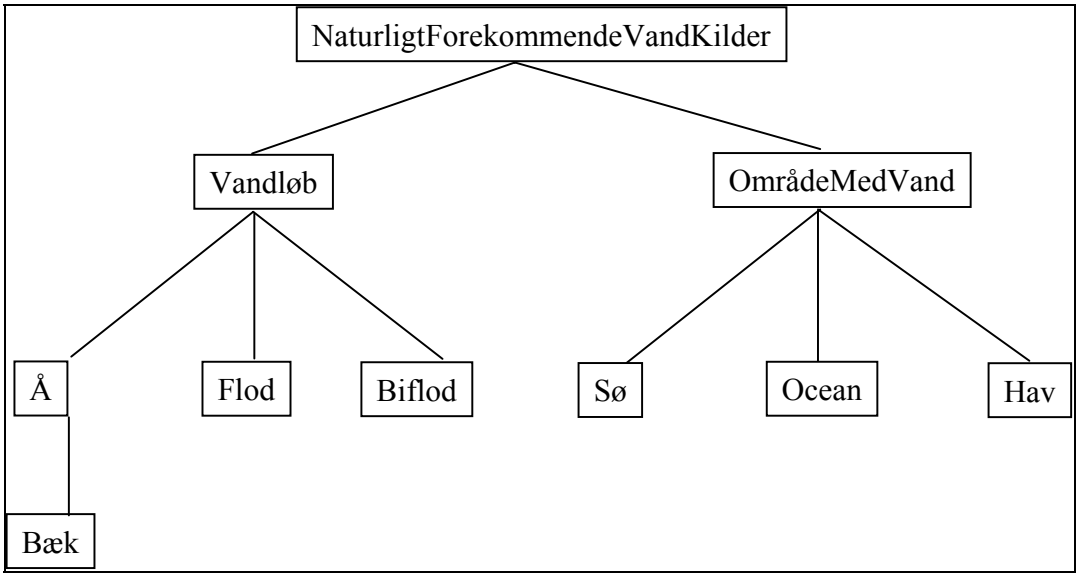
Owl:DatatypeProperty bruges til at relaterer en ressource til en rdfs:Literal eller en indbygget XML Schema datatype.

Figur 1-5 illustrerer hvorledes egenskaber defineres i OWL i forhold til RDFS.



Figur 1-5 Definition af egenskaber i RDFS og OWL

For at illustrere funktionaliteten af OWL vil nedenstående taksonomi (Figur 1-6) bruges gennem hele afsnittet (kilde [OWL]).



Figur 1-6 Vandkilde Taksonomi

1.1.1. Symmetrisk egenskab (Symmetric Property)

En symmetrisk egenskab betyder at hvis vandkilde A er *forbundetMed* vandkilde B så er vandkilde B forbundetMed vandkilde A.

(pA , *rdf:type* , *owl:SymmetricProperty*) fortolkes som "hvis parret (x,y) er en instans af pA, så er parret (y,x) også en instans af pA".

Figur 1-7 owl:SymmetricProperty fortolket

Hvis vi antager at *forbundetMed* er defineret i en OWL ontologi til at være en symmetrisk egenskab, så kan man skrive

```
<?xml version="1.0"?>
<Flod rdf:ID="Gudenå"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
<forbundetMed>
<Sø rdf:about="http://www.danmark.dk/sø#Julsø"/>  </forbundetMed></Flod>
```

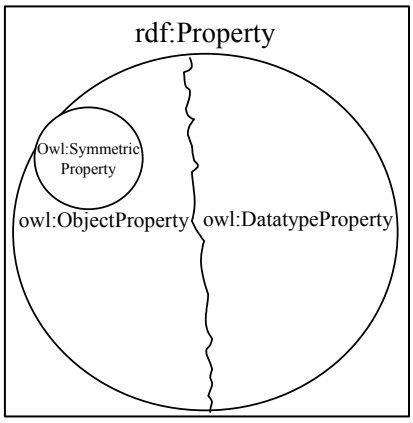
Eksempel 1-1 Brug af forbundetMed

Da forbundetMed er defineret til at være symmetrisk (se Eksempel 1-2), kan man udlede, at Julsø er forbundet med Gudenåen.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende">
<owl:ObjectProperty rdf:ID="forbundetMed">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/><rdfs:domain
rdf:resource="#NaturligtForekommendeVandKilde"/>
<rdfs:range rdf:resource="#NaturligtForekommendeVandKilde "/>
</owl:ObjectProperty>
   ...
</rdf:RDF>
```

Eksempel 1-2 forbundetMed defineres til at være symmetrisk Det kan læses som at forbundetMed er en ObjectProperty. Specielt er det en Symmetrisk ObjectProperty.

Range af Owl:SymmetricProperty kan kun være en ressource, dvs. det kan ikke være en litteral eller datatype, hvorfor Owl:SymmetricProperty er en subclass til Owl:ObjectProperty.



Figur 1-8 Owl:SymmetricProperty er en subClass til owl:ObjectProperty

1.1.2. Transitive egenskaber (Transitive Property)

En transitiv egenskab betyder at hvis A er indeholdtI B og B er indeholdtI C så er A også indeholdtI C. indeholdtI er her den transitive egenskab.

(pA , *rdf:type* , *owl:TransitiveProperty*) fortolkes som "hvis parrene (x,y) og (y,z) er instanser af pA så vil parret (x,z) også være en instans af pA".

Figur 1-9 owl:TransitiveProperty fortolket

```
<?xml version="1.0"?>
<Hav rdf:ID="Øst Kinesiske Hav"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
<indeholdtI>
        <Hav rdf:about="http://www.kina.dk #Kinesiske Hav"/>
</indeholdtI></Hav>
```

Eksempel 1-3 Definition af Øst Kinesiske Hav

```
<?xml version="1.0"?>
<Hav rdf:about="http://www.kina.dk #Kinesiske Hav"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
<indeholdtI>
        <Ocean rdf:about="http://www.verden.dk#Stillehavet"/>
 </indeholdtI></Hav>
```

Eksempel 1-4 Definition af Kinesiske Hav

Hvis man har hentet de to dokumenter angivet i Eksempel 1-3 og Eksempel 1-4, kan man, da indeholdtI er defineret som transitiv (

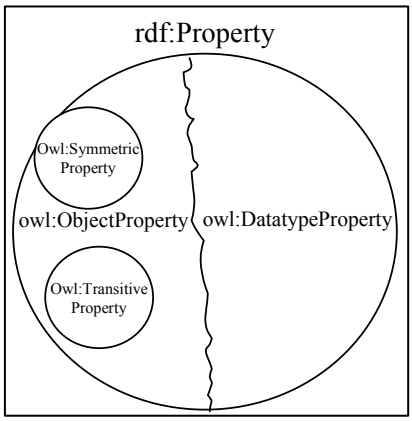
Eksempel 1-5), udlede at da det Østkinesiske Hav er indeholdtI det Kinesiske Hav og det Kinesiske Hav er indeholdtI Stillehavet, er det Østkinesiske Hav også indeholdtI Stillehavet.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende">
<owl:ObjectProperty rdf:ID="indeholdtI">
```

```
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
        <rdfs:domain rdf:resource="#Hav"/>
        <rdfs:range rdf:resource="#OmrådeMedVand"/>
</owl:ObjectProperty>
   ...
</rdf:RDF>
```

Eksempel 1-5 Definition af indeholdtI

Range af Owl:TransitiveProperty kan kun være en ressource, dvs. det kan ikke være en litteral eller datatype, hvorfor Owl:TransitiveProperty er en subclass til Owl:ObjectProperty.



Figur 1-10 Owl:TransitiveProperty er en subClass til owl:ObjectProperty

1.1.3. Funktionelle egenskaber (Functional Property)

En funktionel egenskab betyder, at der for hver instans er højst en værdi for egenskaben.

(pA , *rdf:type* , *owl:FunctionalProperty*) fortolkes som "egenskaben pA kan højst have en (unik) værdi y for hver instans x".

Figur 1-11 owl:FunctionalProperty fortolket

```
<?xml version="1.0"?>
<Flod rdf:about="http://www.danmark.dk/floder#Gudenå"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
<løberUdI rdf:resource="http://www.danmark.dk/#Kattegat"/>
</Flod>
```

Eksempel 1-6 Gudenåen løber ud i Kattegat<?xml version="1.0"?>

```
<Flod rdf:about="http://www.danmark.dk/floder#Gudenå"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
        <løberUdI rdf:resource="http://www.gyldendal_atlas.dk#28-57-0N-11-20E"/>
</Flod>
```

Eksempel 1-7 Gudenåen floden løber ud i 28-57-0N-11-20E

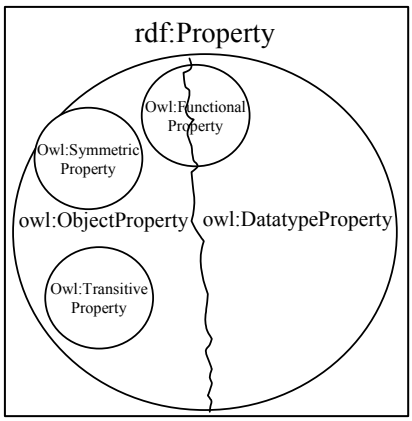
Hvis man har hentet dokumenterne beskrevet i Eksempel 1-6 og Eksempel 1-7 og da løberUdI er defineret til at være funktionel (Eksempel 1-8, kan vi udlede, at det folkene bag www.danmark.dk kalder "Kattegat" er lig 28-57-0N-11-20E i Gyldendals atlas

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:ObjectProperty
rdf:ID="løberUdI">
```

```
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:domain rdf:resource="#Flod"/>
            <rdfs:range rdf:resource="#OmrådeMedVand"/>
          </owl:ObjectProperty>
   ...
</rdf:RDF>
```

Eksempel 1-8 Definerer løberUdI som funktionel

Range af Owl:FunctionalProperty kan både være en ressource, en litteral eller en datatype, hvorfor Owl:TransitiveProperty er en subclass til rdf:Property.

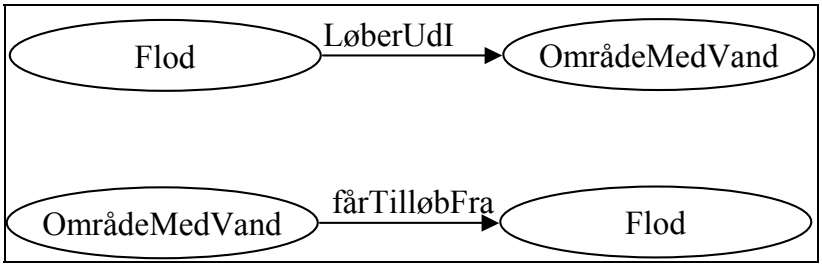


Figur 1-12 Owl:FunctionalProperty er en subclass til rdf:Property

1.1.4. Omvendt egenskab (Inverse Property)

I en omvendt egenskab kan man udlede at hvis egenskab E1 relaterer ressource R1 med ressource R2, så vil den omvendte egenskab E2 relaterer ressource R2 til R1.

Hvis vi kigger på Eksempel 1-6, hvor vi angav at Gudenåen løberUdI Kattegat og vi har angivet at fårTilløbFra og løberUdI, er defineret til at være omvendte egenskaber (Eksempel 1-9), kan man udlede at Kattegat fårTilløbFra Gudenåen. En mere generel definition er illustreret i Figur 1-13.



Figur 1-13 Generel sammenhæng mellem løberUdI og fårTilløbFra

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:ObjectProperty
rdf:ID="løberUdI">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/><rdfs:domain
rdf:resource="#Flod"/>
<rdfs:range rdf:resource="#OmrådeMedVand"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="fårTilløbFra">
<owl:inverseOf rdf:resource="#løberUdI"/><rdfs:domain rdf:resource="#OmrådeMedVand"/>
<rdfs:range rdf:resource="#Flod"/>
</owl:ObjectProperty>
   ...
</rdf:RDF>
```

Eksempel 1-9 løberUdI og fårTilløbFra defineres som omvendte egenskaber

Det er værd at bemærke i Eksempel 1-9 at domain og range bliver byttet om, når den omvendte egenskab defineres.

1.1.5. Omvendt Funktionel egenskab (Inverse Functional Property)

I en Omvendt Funktionel egenskab er domænet unikt for en given range værdi.

(pA , *rdf:type* , *owl:InverseFunctionalProperty*) fortolkes som "en range værdi y kan kun have værdien pA for en enkelt instans af x".

Figur 1-14 owl:InverseFunctionalProperty fortolket

Hvis vi har de to dokumenter angivet i Eksempel 1-10 og Eksempel 1-11 og fårTilløbFra er defineret som Eksempel 1-12, kan vi udlede at Kattegat og "28-57-0N-11-20E" er identiske.

```
<?xml version="1.0"?>
<Hav  rdf:ID="Kattegat"xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
<fårTilløbFra>
        <Flod rdf:about="http://www.danmark.dk/floder#Gudenå"/></fårTilløbFra></Hav>
```

Eksempel 1-10 Kattegat fårTilløbFra Gudenå<?xml version="1.0"?>

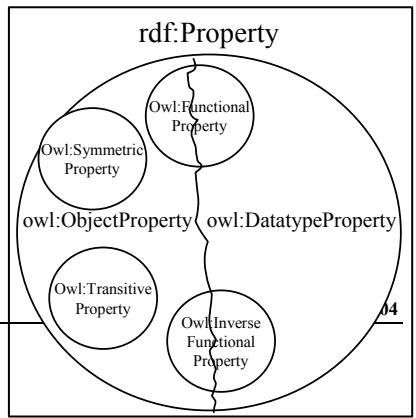
```
<Hav  rdf:ID="atlas:28-57-0N-11-20E"xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
xmlns:atlas="http://www.gyldendal_atlas.dk/nummer#
<fårTilløbFra>
        <Flod rdf:about="http://www.danmark.dk/floder#Gudenå"/>
</fårTilløbFra></Hav>
```

Eksempel 1-11 "28-57-0N-11-20E" får tilløb fra Gudenå

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:ObjectProperty
rdf:ID="løberUdI">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/><rdfs:domain
rdf:resource="#Flod"/>
<rdfs:range rdf:resource="#OmrådeMedVand"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="fårTilløbFra">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
<owl:inverseOf rdf:resource="#løberUdI"/><rdfs:domain
rdf:resource="#OmrådeMedVand"/><rdfs:range rdf:resource="#Flod"/>
</owl:ObjectProperty>
   ...
</rdf:RDF>
```

Eksempel 1-12 Definition af fårTilløbFra som en omvendt funktionel egenskab

Da range af en owl:InverseFunctionalProperty både kan være en ressource, en litteral eller en datatype, er owl:InverseFunctionalProperty en subclass af rdf:Property.



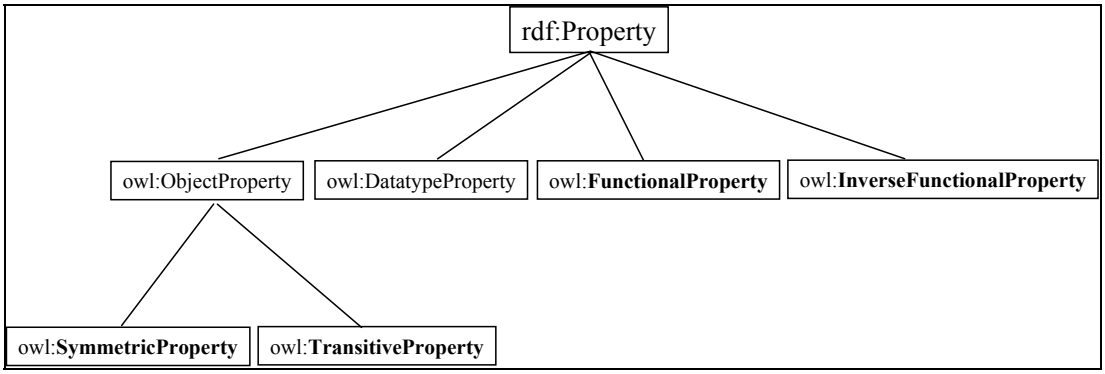
Figur 1-15 owl:InverseFunctionalProperty er en subclass af rdf:Property

1.1.6. Opsummering på definition af egenskaber

Med OWL er der forskellige måder at karakterisere egenskaber, den kan være defineret som:

- En symmetrisk egenskab
- En transitiv egenskab
- En funktionel egenskab
- Det omvendte til en anden egenskab
- En Omvendt Funktionel egenskab

De enkelte egenskabsklasser er inddelt i et hierarki som angivet i Figur 1-16

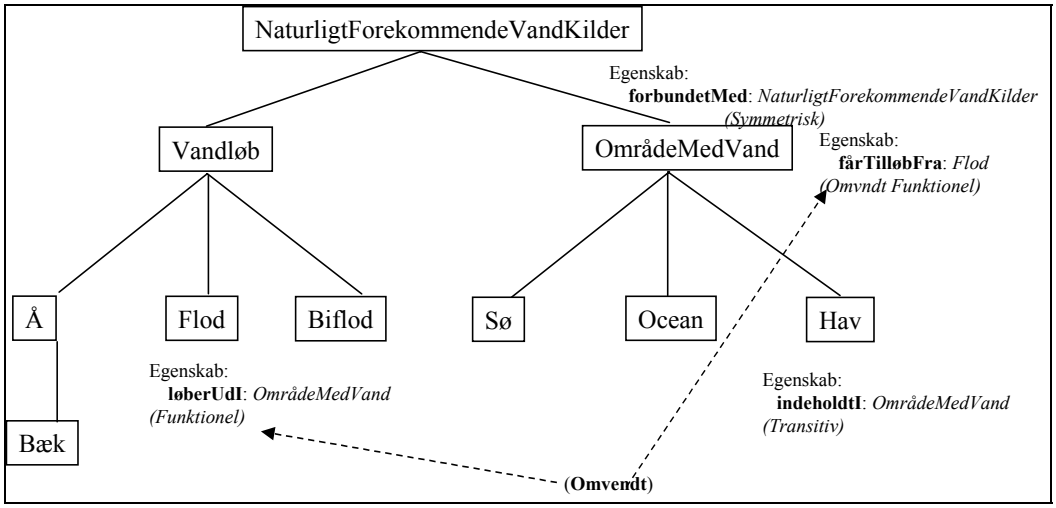


Figur 1-16 Hierarki over egenskabsklasser

Det er værd at bemærke, at owl:inverseOf ikke er vist i ovenstående hierarki, fordi det er en egenskab ikke en klasse.

Det betyder, at owl:SymmetricProperty og owl:TransitiveProperty kun kan bruges til at relatere ressourcer til ressourcer, mens owl:FunctionalProperty og owl:InverseFunctionalProperty kan bruges til at relatere ressourcer til ressourcer eller ressourcer til en RDF Schema Literal eller en XML Schema Datatype.

Figur 1-17 viser en samlet oversigt over vandkilde taksonomien, med de nye definitioner.



Figur 1-17 Vandkilde taksonomi med de definerede egenskaber

Denne taksonomi giver os mulighed for at udlede følgende fra Eksempel 1-13.

- Kattegat har tilløb fra Gudenåen (Da harTilløbFra er det omvendte af løberUdI)
- Julsø er forbundet med Gudenåen (Da forbundetMed er symmetrisk)
- Kattegat er et OmrådeMedVand (Da range af løberUdI er OmrådeMedVand)
- At Julsø er en NaturligtForekommendeVandKilde (Da range af forbundetMed er NaturligtForekommendeVandKilde)

```
<?xml version="1.0"?>
<Flod rdf:ID="Gudenå"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
        <løberUdI rdf:resource=" http://www.danmark.dk/hav#Kattegat"/>
        <forbundetMed rdf:resource="http://www.danmark.dk/sø#Julsø"/>
</River>
```

Eksempel 1-13 Samlet brug af vandkildetaksonomien

1.2 owl:Class

OWL klasser tillader mange flere udtryk end RDF Schema klasser, hvorfor OWL har dannet dets egen klasse owl:Class se Figur 1-18. owl:Class er en subClassOf rdfs:Class.

```
<rdfs:Class rdf:ID="Flod">
    <rdfs:subClassOf rdf:resource="#Vandløb"/>
</rdfs:Class>
```

RDFS

```
<owl:Class rdf:ID="Flod">
    <rdfs:subClassOf rdf:resource="#Vandløb"/>
</owl:Class>
```

OWL

Figur 1-18 Definition af klasser i RDFS og OWL

Hver klasse, der er defineret af en ontologi, er en særskilt mængde. Owl:Class er mængden af alle mulige særskilte mængder. Hver defineret klasse er af type owl:Class.

1.3 Begrænsninger baseret på indholdet.

OWL giver mulighed for at man kan lave yderligere begrænsninger baseret på den sammenhæng, det bliver brugt i. De følgende afsnit vil indeholde en gennemgang af nogle af disse muligheder.

1.3.1. Brug af allValuesFrom

(rA, *owl:allValuesFrom*, objectID) fortolkes som: *"rA er mængden af særskilte x således at hver p-værdi y (hvis der er nogle) tilhører objectID, hvor p er egenskaben og objectID kan være en klasse eller datatype"*.

Figur 1-19 owl:allValuesFrom fortolket

Hvis vi f.eks. var i Frankrig, var det nødvendigt at udvide vores taksonomi med begrebet *flueve*, som betegner floder, der har udløb i et hav. Hvilket betyder at i sammenhæng med *flueve* klassen, skal *løberUdI* egenskaben begrænses til *hav* klassen. Dette gøres som angivet i Eksempel 1-14.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:Class rdf:ID="Flueve">
<rdfs:subClassOf rdf:resource="#Flod"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#løberUdi"/>
<owl:allValuesFrom rdf:resource="#Hav"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
</rdf:RDF>
```

Eksempel 1-14 Brug af owl:allValuesFrom

Det kan læses som, at *Flueve klassen er en subClassOf Flod. Den har en egenskab løberUdI. Alle værdier for løberUdI må være af Hav*. Det er værd at bemærke, at da *Flueve* er en underklasse til *Flod*, arver den alle dens egenskaber, uden at det specificeres eksplicit. Det betyder at hvis vi har et dokument som i Eksempel 1-15, kan vi udlede at Kattegat er et hav.

```
<?xml version="1.0"?>
<Flueve rdf:about="http://www.danmark.dk/floder#Gudenå"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
<løberUdI rdf:resource="http://www.danmark.dk/#Kattegat"/>
</Flueve>
```

Eksempel 1-15 Brug af Flueve

1.3.2. Brug af someValuesFrom og allValuesFrom

For at være en flod kræves det, at mindst en værdi af forbundetMed skal være OmrådeMedVand. Da alle klasser arver forbundetMed egenskaberne (jf. Figur 1-17), betyder det, at alt kan være forbundetMed alt andet. En flod kan være forbundet med mange ting en bæk, bifloder osv. Men en ting det skal være forbundetMed, er OmrådeMedVand (sø, hav eller ocean). Det betyder at i sammenhæng med flod

klassen, skal forbundetMed egenskaben have mindst en værdi, der er OmrådeMedVand. Det defineres som vist i Eksempel 1-16.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:Class rdf:ID="Flod">
<rdfs:subClassOf rdf:resource="#Vandløb"/>
<rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#forbundetMed"/>
          <owl:someValuesFrom rdf:resource="#OmrådeMedVand"/>
        </owl:Restriction>
</rdfs:subClassOf></owl:Class>
   ...
</rdf:RDF>
```

Eksempel 1-16 Brug af owl:someValuesFrom

Det kan læses som *Flod klassen er en subClassOf Vandløb. Det har en egenskab forbundetMed. Mindst en af værdierne for forbundetMed, skal være fra et OmrådeMedVand og der skal være mindst en.* Det betyder at mindst en af Kattegat, Julsø og Langå skal være OmrådeMedVand

```
<?xml version="1.0"?>
<Flueve rdf:about="http://www.danmark.dk/floder#Gudenå"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
<forbundetMed rdf:resource="http://www.danmark.dk/#Kattegat"/>
<forbundetMed rdf:resource="http://www.danmark.dk/#Julsø"/>
<forbundetMed rdf:resource="http://www.danmark.dk/#Langå"/>
</Flueve>
```

Eksempel 1-17 Gudenå forbundetMed

Hvis man i stedet havde valgt at bruge owl:allValuesFrom som i Eksempel 1-18, havde det betydet, at alle værdier skulle være instanser af typen hav, men i modsætning til owl:someValuesFrom kan der med owl:allValuesFrom godt være nul instanser.

```
<owl:onProperty rdf:resource="#løberUdi"/>
<owl:allValuesFrom rdf:resource="#Hav"/>
```

Eksempel 1-18 Brug af owl:allValuesFrom

1.3.3. Brug af owl:hasValue

Hvis man antager, at til OmrådeMedVand er der defineret en type egenskab kaldet FerskVandEllerSaltVand, vil denne egenskab nedarves til Ocean. Da alle oceaner er saltvand, vil det være gavnligt, at kunne angive at for alle oceaner er FerskVandEllerSaltVand lig SaltVand. Dette gøres som specificeret i Eksempel 1-19.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><FerskVandEllerSaltVand
rdf:ID="SaltVand"/><owl:Class rdf:ID="Ocean">
  <rdfs:subClassOf rdf:resource="#OmrådeMedVand"/>
  <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#type"/>
          <owl:hasValue rdf:resource="#SaltVand"/>
        </owl:Restriction>
  </rdfs:subClassOf></owl:Class>
    ...
</rdf:RDF>
```

Eksempel 1-19 Brug af owl:hasValue til at definere "type" egenskaben til at have værdien SaltVand, når den bruges i Ocean.

Det kan læses som, *Ocean* klassen er en *subClassOf OmrådeMedVand* og alle *Oceaner* har en *type* egenskab, hvis værdi er *SaltVand.*

```
<?xml version="1.0"?>
<Ocean rdf:ID="Atlanterhavet"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns=" http://www.mim.dk/vand/NaturligtForekomende #">
        <type rdf:resource=" http://www.mim.dk/vand/NaturligtForekomende #SaltVand"/>
</Ocean>
```

Eksempel 1-20 Atlanterhavet med type egenskaben "SaltVand"Det er ikke nødvendigt at placere type egenskaben i et Ocean instans dokument, type kan udledes fra en owl:hasValue. Dvs. at Ontologien indikerer, at hvis det er et Ocean, så er dets type SaltVand.

1.3.4. Brug af owl:cardinality

owl:Cardinality kan bruges til at definere mængden af forekomster af en egenskab, baseret på i hvilken sammenhæng (klasse) den bruges.

Når vi definerer OmrådeMedVand klassen, vil det være gavnligt at kunne angive, at der kan være højst en maxDybde for et OmrådeMedVand. Dette gøres som i Eksempel 1-21.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:Class
rdf:ID="OmrådeMedVand">
<rdfs:subClassOf rdf:resource="#NaturligtForekommendeVandKilde"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#maxDybde"/><owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
    ...
</rdf:RDF>
```

Eksempel 1-21 Brug af owl:cardinality til at lægge restriktioner på maxDybde. Den kan læses som, OmrådeMedVand klassen er en underklasse af NaturligtForekomendeVandKilder. Det har en egenskab maxDybde. Der kan kun være en maxDybde for en OmrådeMedVand.

```
<?xml version="1.0"?>
<Ocean rdf:ID="Stillehavet"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns=" http://www.mim.dk/vand/NaturligtForekomende#">
        <maxDybde

rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2300</maxDybde></Ocean>
```

Eksempel 1-22 Angivelse af maxDybde for Stillehavet.

Brug af owl:cardinality sætter ikke nogle krav om forekomster af en egenskab i et instans dokument. Hvis man f.eks. kigger på følgende sætninger

1. I en instans dokument kan der kun være en maxDybde egenskab for en OmrådeMedVand
2. En OmrådeMedVand har kun en maxDybde

Den første ville være noget man bør angive i et XML Schema, den anden fortæller noget om informationen. Den sætter ingen begrænsninger på antallet af maxDybde, det kræves bare, at de alle er identiske, da der højst kan være en maxDybde per ressource.

Hvis man har defineret en navn egenskab for NaturligtForekommendeVandKilde, vil denne nedarves til de andre klasser. Det kunne dog være anvendeligt, hvis man i definitionen af Å kunne angive, at det måske ikke har noget navn. Dette gøres ved brug af owl:minCardinality (Eksempel 1-23

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:Class rdf:ID="Å">
<rdfs:subClassOf rdf:resource="#VandLøb"/>
<rdfs:subClassOf>
<owl:Restriction>
        <owl:onProperty rdf:resource="#navn"/><owl:minCardinality
            rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">0
        </owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
   ...
</rdf:RDF>
```

Eksempel 1-23 Brug af owl:minCardinality

Hvis man også ønsker at sætte en øvre grænse bruges owl:maxCardinality (Eksempel 1-24)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:Class rdf:ID="Å">
<rdfs:subClassOf rdf:resource="#VandLøb"/>
```

```
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#navn"/>
        <owl:minCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">0
        </owl:minCardinality>
        <owl:maxCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">10
        </owl:maxCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
  ...
</rdf:RDF>
```

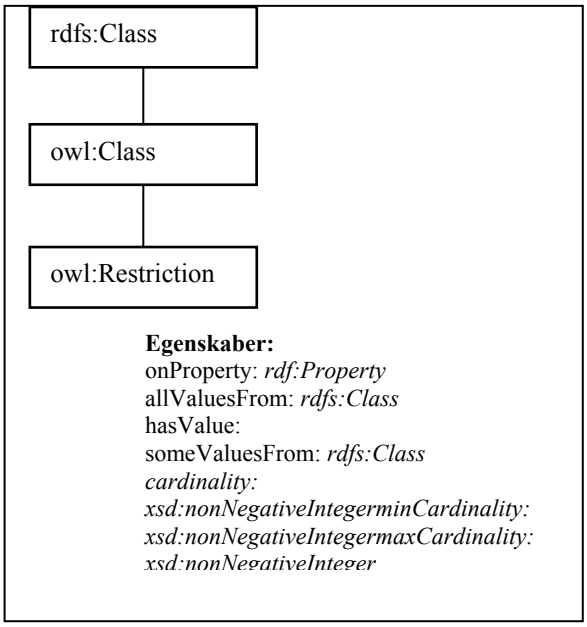
Eksempel 1-24 Sætte et interval ved brug af owl:minCardinality og owl:maxCardinality

En restriktion, der indeholder en mængde restriktion, beskriver en klasse som har mindst N, højst N eller præcis N distinkte range værdier for den egenskab den omhandler.

1.3.5. Opsummering på forskellige måder en klasse kan begrænse en egenskab

De foregående afsnit har vist, at der er forskellige måder en klasse kan begrænse en global egenskab. En egenskab kan begrænses således at:

- Alle værdier skal tilhøre en specifik klasse (brug allValuesFrom)
- Mindst en værdi skal komme fra en specifik klasse (brug someValueFrom)
- Det har en specifik værdi (brug hasValue)
- Ønsker man at angive mængden af en egenskab, kan det angives ved brug af
    - cardinality
    - maxCardinality
    - minCardinality



Figur 1-20 Egenskaber ved Restriction klassen

1.3.6. Brug af owl:equivalent

Hvis man ønsker at angive at to egenskaber er ækvivalente, bruges owl:equivalentProperty. Eksempel 1-25 definerer f.eks. at navn egenskabet er ækvivalent til Title egenskabet i Dublin Core.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:DatatypeProperty rdf:ID="navn">
 <owl:equivalentProperty rdf:resource="http://pur1.org/metadata/dublin-core#Title"/>
<rdfs:domain rdf:resource="#NaturligtForekommendeVandKilde"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
   ...
</rdf:RDF>
```

Eksempel 1-25 Brug af owl:equivalentProperty til at angive at navn er ækvivalent til Title egenskaben i Dublin Core

1.4 Brug af OWL til at definere klasser

OWL giver mulighed for at konstruere klasser ved brug af forskellige operatorer, de efterfølgende afsnit vil gennemgå følgende:

- intersectionOf
- unionOf
- complementOf
- oneOf
- equivalentClass
- disjointWith

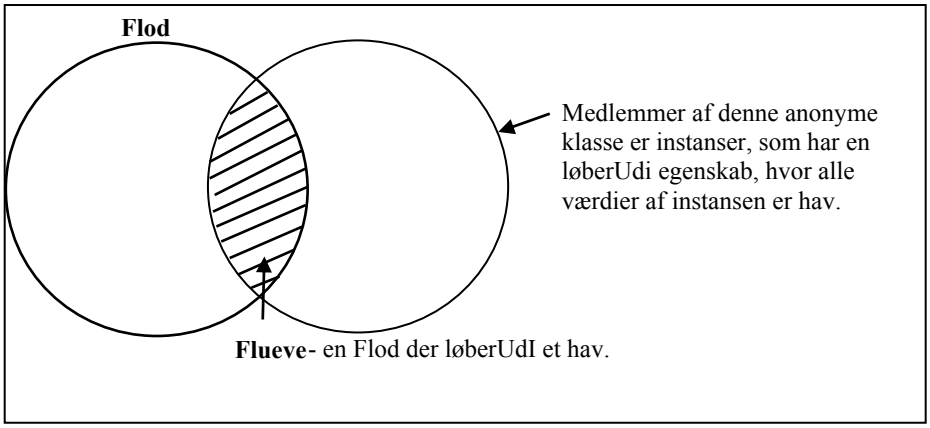
1.4.1. owl:intersectionOf

Hvis vi ønsker at definere *Flueve* ved brug af intersectionOf, kan det gøres som i Eksempel 1-26.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:Class rdf:ID="Flueve">
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Flod"/>
<owl:Restriction>
<owl:onProperty rdf:resource="#løberUdI"/>
<owl:allValuesFrom rdf:resource="#Hav"/>
</owl:Restriction>
</owl:intersectionOf></owl:Class>
</rdf:RDF>
```

Eksempel 1-26 Definition af *Flueve* ved brug af owl:intersectionOf

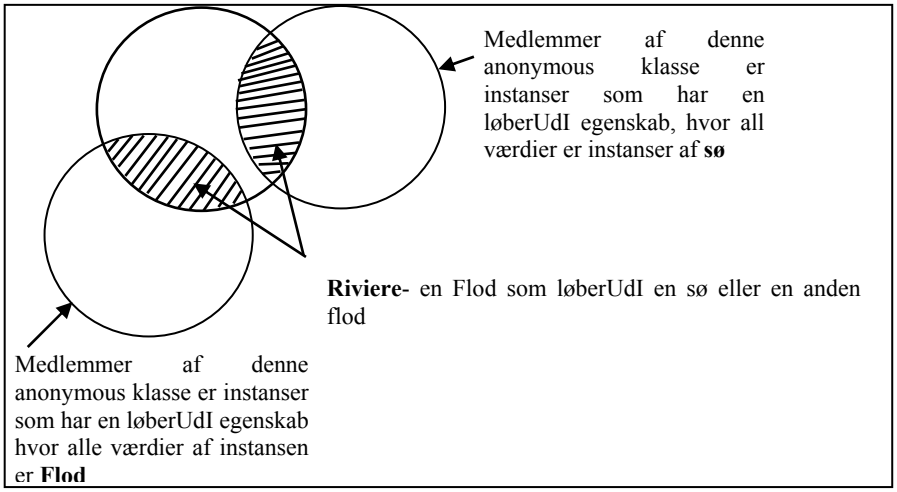
Det kan læses som "*Flueve klassen er fællesmængden af flod klassen og en anonymous klasse, som indeholder en egenskab løberUdI og alle værdier af instansen er hav"* (se også Figur 1-21).



Figur 1-21 Illustration af Flueve definitionen

1.4.2. owl:unionOf

Det franske ord Riviere betyder ”*en flod der løberUdI en sø eller en anden flod” (*Figur 1-22*)*. Det vil derfor være nødvendigt at definere Riviere ved brug af både intersectionOf og unionOf (Eksempel 1-27).



Figur 1-22 Illustration af defintion af Riviere med intersectionOf og unionOf

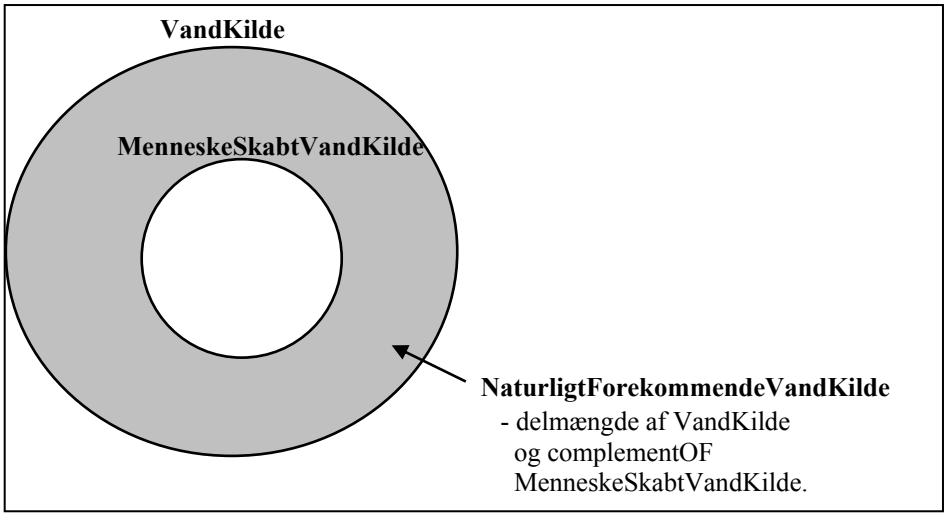
De 2 anonyme klasser er adskilte fordi løberUdI er en funktionel egenskab.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:Class rdf:ID="Riviere">
  <owl:intersectionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#Flod"/>
  <owl:Class>     <owl:unionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="#løberUdI"/>
      <owl:allValuesFrom rdf:resource="#Sø"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#løberUdI"/>
      <owl:allValuesFrom rdf:resource="#Flod"/>
    </owl:Restriction>     </owl:unionOf>
  </owl:Class>   </owl:intersectionOf></owl:Class>
</rdf:RDF>
```

Eksempel 1-27 Defintion af Riviere med intersectionOf og unionOf

1.4.3. owl:complementOf

Hvis vi har alle vandkilder, vil nogle være lavet af mennesker og nogle vil være naturligt forekommende. Dvs. at NaturligtForekommendeVandKilde og MenneskeSkabtVandKilde vil være komplementære, som illustreret i Figur 1-23.



Figur 1-23 NaturligtForekommendeVandKilde er complementOf MenneskeSkabtVandKilde

Det defineres som angivet i Eksempel 1-28.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende"><owl:Class
rdf:ID="NaturligtForekommendeVandKilde">
  <owl:intersectionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#VandKilde"/>
  <owl:Class>     <owl:complementOf rdf:resource="#MenneskeSkabtVandKilde"/>
  </owl:Class>  </owl:intersectionOf></owl:Class>
   ...
</rdf:RDF>
```

Eksempel 1-28 NaturligtForekommendeVandKilde er complementOf MenneskeSkabtVandKilde

1.4.4. owl:oneOf

OWL giver mulighed for at konstruere en klasse ved at opliste dets instanser. I Eksempel 1-29 angives de floder, der er beskyttet under Kyoto traktaten.

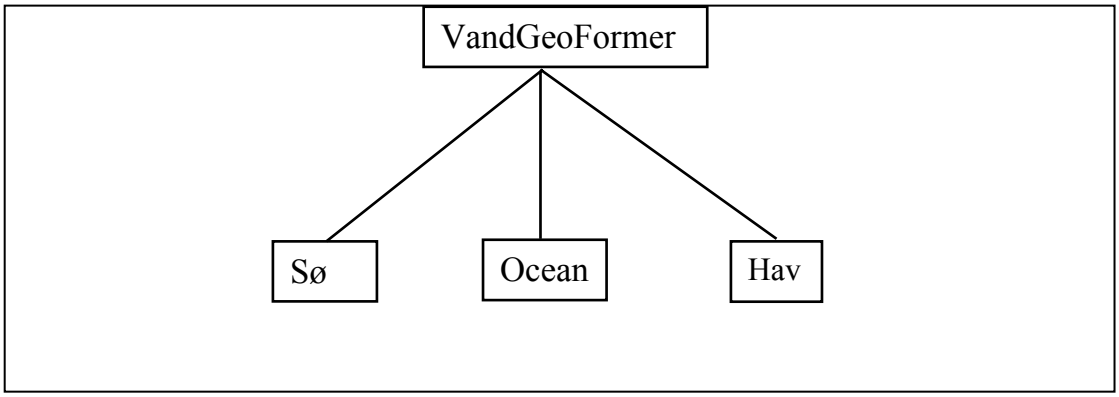
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:geo="http://www.mim.dk/vand/NaturligtForekomende #"xml:base="
http://www.mim.dk/vand/NaturligtForekomende ">
<owl:Class rdf:ID="Kyoto-Beskyttede-Floder ">
<rdfs:subClassOf rdf:resource="#Flod"/>
  <owl:oneOf rdf:parseType="Collection"><geo:Flod
rdf:about="http://www.china.org/geography/rivers#Yangtze"/>
          <geo:Flod rdf:about="http://www.us.org/rivers#Mississippi"/>
          <geo:Flod rdf:about="http://www.africa.org/rivers#Nile"/>
          <geo:Flod rdf:about="http://www.s-america.org/rivers#Amazon"/>
```

```
        …
  </owl:oneOf></owl:Class>
   ...
</rdf:RDF>
```

Eksempel 1-29 Definerer de Kyoto beskyttede floder ved hjælp af owl:oneOf

1.4.5. owl:equivalentClass

owl:equivalentClass bruges til at angive at en klasse er ækvivalent til en anden klasse. Hvis nu et andet OWL dokument definerede en klasse kaldet VandGeoFormer som i Figur 1-24, ville det være ønskeligt at kunne angive, at VandGeoFormer var ækvivalent med OmrådeMedVand. Dette gøres som i Eksempel 1-30.



Figur 1-24 Illustration af VandGeoFormer

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base=" http://www.mim.dk/vand/NaturligtForekomende ">
<owl:Class rdf:ID="OmrådeMedVand">
  <rdfs:subClassOf rdf:resource="#NaturligtForekommendVandKilde"/>
  <owl:equivalentClass rdf:resource="http://www.anden.dk#VandGeoFormer"/></owl:Class>
   ...
</rdf:RDF>
```

Eksempel 1-30 Angivelse af at OmrådeMedVand er ækvivalent til VandGeoFormer

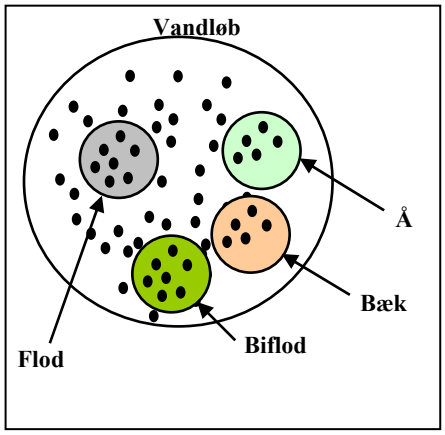
1.4.6. owl:disjointWith

Hvis vi ønsker at angive at to klasser er disjunkte (adskilte) bruges owl:disjointWith. En flod kan for eksempel ikke også være en å, en bæk eller en biflod. Det angives som i Eksempel 1-31.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.mim.dk/vand/NaturligtForekomende ">
<owl:Class rdf:ID="Flod">
  <rdfs:subClassOf rdf:resource="#Vandløb"/>
  <owl:disjointWith rdf:resource="#Å"/>  <owl:disjointWith rdf:resource="#Bæk"/>
<owl:disjointWith rdf:resource="#Biflod"/></owl:Class>
   ...
</rdf:RDF>
```

Eksempel 1-31 Brug af owl:disjointWith

Vi har nu sikret os, at vi ikke kan have en instans, som angiver at Gudenåen er en å. Det er værd at bemærke, at disjointWith er en symmetrisk egenskab, hvorfor vi også har defineret, at en å ikke kan være en flod. Men vi har ikke angivet at å, bæk eller biflod er disjunkte. Det kunne gøres som i Eksempel 1-32.



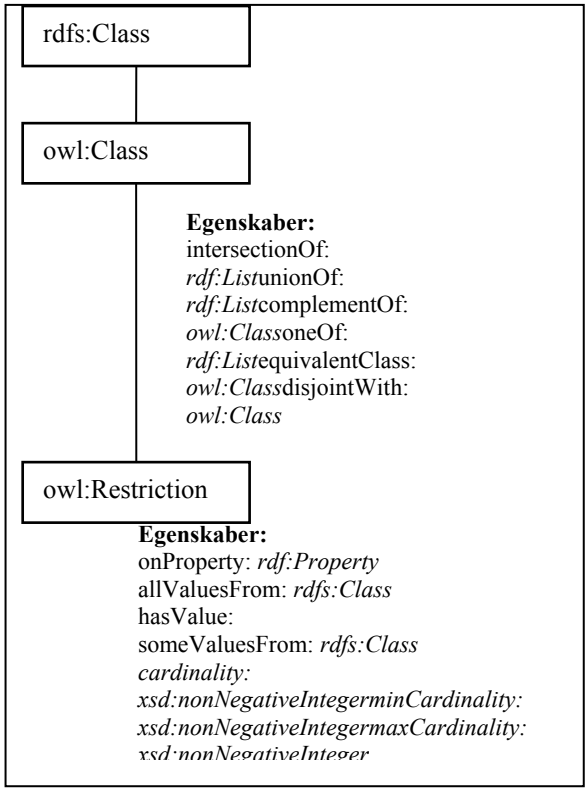

```
<owl:Class rdf:ID="Flod">
  <rdfs:subClassOf rdf:resource="#Vandløb"/>
  <owl:disjointWith rdf:resource="#Å"/>
  <owl:disjointWith rdf:resource="#Bæk"/>
  <owl:disjointWith rdf:resource="#Biflod"/>
</owl:Class>

<owl:Class rdf:ID="Å">
  <rdfs:subClassOf rdf:resource="#Vandløb"/>
  <owl:disjointWith rdf:resource="#Bæk"/>
  <owl:disjointWith rdf:resource="#Biflod"/>
</owl:Class>

<owl:Class rdf:ID="Bæk">
  <rdfs:subClassOf rdf:resource="#Vandløb"/>
  <owl:disjointWith rdf:resource="#Biflod"/>
</owl:Class>
```

Eksempel 1-32 Angive at Flod, Vandløb, Bæk og Biflod er disjunkte

1.4.7. Opsummering på klasse egenskaber

De foregående afsnit har gennemgået de enkelte klasseegenskaber:

- Hvis man ønsker at definere en klasse, som er en delmængde af en anden klasse bruges intersectionOf
- Hvis man ønsker at definere en klasse, som er fællesmængden af andre klasser bruges unionOf
- Hvis man ønsker at angive at klasser er komplementære anvendes complementOf
- Hvis man ønsker at angive en liste, som en klasse skal tilhøre, anvendes oneOf
- Hvis man ønsker, at angive at klasser er ækvivalent, anvendes equivalentClass
- Hvis man ønsker, at angive at klasser er disjunkte, anvendes disjointWith.



Figur 1-25 Opsummering på klasse egenskaber

1.5 OWL udtryk der kan indføjes i instanser.

OWL giver også mulighed for, at man kan indføje nogle udtryk i instanserne. Vi vil i de følgende afsnit kigge på

- owl:sameIndividualAs
- owl:differentFrom
- owl:AllDifferent
- owl:Thing

1.5.1. owl:sameIndividualAs

I afsnit 1.1.3 blev det gennemgået, hvordan man kunne udlede at Kattegat var lig 28-57-0N-11-20E, fordi løberUdI var defineret som en Funktionel egenskab. Hvis man ønsker at angive dette eksplicit, kan det gøres ved brug af owl:sameIndiviualAs som i Eksempel 1-33.

```
<?xml version=”1.0”?>
<Hav rdf:ID=”Kattegat”
xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:owl=”http://www.w3.org/2002/07/owl#”
xmlns=” http://www.mim.dk/vand/NaturligtForekomende#”>
<owl:sameIndividualAs rdf: :resource=http://www.gyldendal_atlas.dk#28-57-0N-11-20E/>     ...
</Hav>
```

Eksempel 1-33 Brug af owl:sameIndividualAs

1.5.2. owl:differentFrom

Det kan også være gavnligt, tydeligt at angive at to instanser er forskellige. Dette gøres ved brug af owl:differentFrom (Eksempel 1-34)

```
<?xml version=”1.0”?>
<Hav rdf:ID=”Vesterhavet”
xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:owl=”http://www.w3.org/2002/07/owl#”
xmlns=” http://www.mim.dk/vand/NaturligtForekomende#”>
<owl:differentFrom rdf:resource=”http://www.danmark.dk#Kattegat”/>     ...
</Hav>
```

Eksempel 1-34 Brug af owl:differentFrom

Hvis man ikke havde defineret Kattegat og Vesterhavet til at være forskellige, ville nedenstående angive at Kattegat var lig Vesterhavet. Men det vil nu indikere, at der er en fejl i instansen.

```
<?xml version=”1.0”?>
<Flod rdf:ID=”Gudenå”
  xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
  xmlns=”http://www.mim.dk/vand/NaturligtForekomende#”>
  <løberUdI rdf:resource=”http://www.danmark.dk/hav#Kattegat”/>
  <løberUdI rdf:resource=”http://www.danmark.dk/hav#Vesterhavet”/>
</Flod>
```

Figur 1-26 Modsætning i instansen

1.5.3. owl:AllDifferent

I modsætning til typiske databasesystemer, antager OWL ikke, at 2 udtryk er forskellige, bare fordi to instanser har et forskelligt navn (eller ID). løberUdI er tidligere angivet som en owl:FunctionalProperty, så hvis man angiver som i Eksempel 1-35, vil systemet ikke komme med en fejl, men det kan antage at http://www.denmark.org/sea#North_Sea er lig http://www.danmark.dk/hav#Vesterhavet.

```
<?xml version="1.0"?>
<Å rdf:ID="Ribe å"
…xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
…xmlns="http://www.mim.dk/vand/NaturligtForekomende#">
…<løberUdI rdf:resource="http://www.denmark.org/sea#North Sea"/>
…<løberUdI rdf:resource="http://www.danmark.dk/hav#Vesterhavet"/>
</Å>
```

Eksempel 1-35 Vesterhavet lig North Sea

Hvis man ønsker at angive, at en samling af instanser er forskellige, kan det angives med owl:AllDifferent som i Eksempel 1-36

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:geo=" http://www.mim.dk/vand/NaturligtForekomende #"xml:base="
http://www.mim.dk/vand/NaturligtForekomende">
<owl:AllDifferent>
…<owl:distinctMembers rdf:parseType="Collection">……<geo:Hav
rdf:about="http://www.denmark.org /geography/sea#North_Sea"/>
……<geo:Hav rdf:about="http://www.danmark.dk #Kattegat"/>
……<geo:Hav rdf:about="http://www.sjælland.dk/hav#Øresund"/>
……<geo:Hav rdf:about="http://www.scandinavia.org#Baltic_Sea"/>
…</owl:distinctMembers>
</owl:AllDifferent>  ...
</rdf:RDF>
```

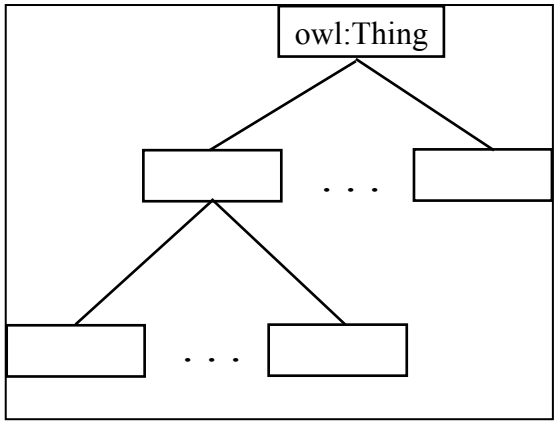
Eksempel 1-36 Brug af owl:AllDifferent

Selvom man godt kan bruge owl:AllDifferent i et instans dokument, vil det typisk bruges i et ontologidokument.

1.5.4. owl:Thing

owl:Thing er en foruddefineret OWL klasse. Alle instanser er medlemmer af owl_Thing. Den er rod klassen for alle andre klasser.

Der eksisterer også en foruddefineret klasse kaldet owl:Nothing, som repræsenterer den tomme mængde.



Figur 1-27 owl:Thing er rod klasse for alle andre klasse

1.6 owl:Ontology

egenskaber

OWL definerer også nogle egenskaber til owl:Ontology. Nogle er illustreret nedenstående.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:geo="http://www.mim.dk/vand/NaturligtForekomende#"xml:base="http://www.mim.dk/vand/NaturligtForekomende ">
<owl:Ontology rdf:about="http://www.mim.dk/vand/NaturligtForekomende">
<owl:priorVersion rdf:resource="=" http://www.mim.dk/vand/NaturligtForekomende/april-03#"/>
<owl:versionInfo>naturligt-forekommende.owl v 2.0</owl:versionInfo>
<owl:imports rdf:resource="http://www.andre.org/andre.owl"/>
</owl:Ontology>  ...
</rdf:RDF>
```

Eksempel 1-37 Ontology header

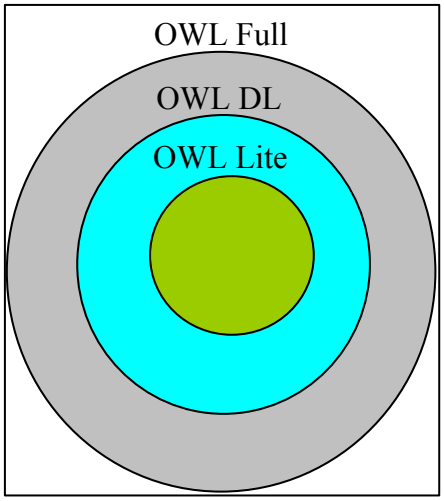
owl:Ontology

Egenskaber:
imports:
versionInfo:
priorVersion: *Ontology*
incompatibleWith: *Ontology*
backwardCompatibleWith: *Ontology*

Figur 1-28 owl:Ontology egenskaber

1.7 Forskellige versioner af OWL

Da ikke alle har behov for alle de muligheder, som OWL tilbyder, er der udformet tre versioner af OWL.



I valget mellem hvilken version man skal bruge, må man prioritere mellem:

- Har man behov for den fulde version
- Hvor let det er at bygge værktøjer til det
- Har man behov for hurtige og komplette svar

Nedenstående uddrag fra [OWLREF] viser formålet med de enkelte dele.

"[...] OWL Lite was designed for easy implementation and to provide users with a functional subset that will get them started in the use of OWL. OWL DL (where DL stands for "Description Logic") was designed to support the existing Description Logic business segment and to provide a language subset that has desirable computational properties for reasoning systems. The complete OWL language (called OWL Full to distinguish it from the subsets) relaxes some of the constraints on OWL DL so as to make available features which may be of use to many database and

knowledge representation systems, but which violate the constraints of Description Logic reasoners."

1.8 Yderligere om OWL

Et OWL dokument er et XML dokument, hvilket betyder at man kan udnytte alle de værktøjer der eksisterer til XML, når man bruger OWL dokumentet. (XSLT, XML APIer (SAX, DOM), XQuery, osv). Der eksisterer dog også et antal OWL API'er man kan benytte.

Selvom XML Schema tilbyder en måde hvorpå man kan konstruere bruger definerede datatyper (f.eks. at datatypen voksenAlder er alle heltal større end 18 eller at datatype af alle ting der starter med et tal) kan sådanne udledte datatyper ikke anvendes i OWL. Der er mange af de indbyggede XML Schema datatyper som ikke kan bruges. OWL reference dokumentet indeholder en liste over alle de XML Schema datatyper der kan bruges, disse inkludere de oftest brugte typer såsom string, integer, boolean, time og dato [OWLREF].

1.9 Eksempel på brug af OWL

Dette afsnit indeholder et relevant eksempler på brug af OWL, eksemplet er baseret på [UOWL]. Eksemplet skal illustrere hvorledes man ved brug af OWL kan opnå væsentlig større fleksibilitet. Eksemplet bruger en kameraontologi angivet i **Fejl! Henvisningskilde ikke fundet.**. Nedenstående er de relevante dele behandlet.

For forståelse af eksemplet skal det fremhæves at (Figur 1-29):

- SLR er af typen Camera
- F-stop er synonymt med aperture
- Focal-length er synonymt med lens size

```
<owl:Class rdf:ID="SLR">

<owl:equivalentClass>
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Camera"/>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#viewFinder"/>
      </owl:onProperty>
        <owl:hasValue rdf:resource="#ThroughTheLens" rdf:type="#Window"/>
    </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</owl:equivalentClass>
</owl:Class>

<owl:DatatypeProperty rdf:ID="f-stop">

  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Lens"/>
  <owl:equivalentProperty>
    <owl:DatatypeProperty rdf:about="#aperture"/>
  </owl:equivalentProperty>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID=”focal-length”>
  <rdfs:range rdf:resource=”http://www.w3.org/2001/XMLSchema#string”/>
  <rdfs:domain rdf:resource=”#Lens”/>
  <owl:equivalentProperty rdf:resource=”#size”/>
</owl:DatatypeProperty>
```

Figur 1-29 Uddrag fra camera.owl: SLR er at typen Camera, , f-stop er synonymt med aperture og Focal-length er synonymt med lens size

Strategien er at alle parter involveret i udveksling af data, kan sammensætte en vilkårlig fysisk repræsentation, forudsat at den fysiske repræsentation overholder de fundamentale datarelationer defineret i det logiske design.

Det betyder, at en person kan danne et kamera dokument som angivet i Eksempel 1-38 og en anden kunne danne et kamera dokument som i Eksempel 1-39.

```
<SLR rdf:ID=”Olympus-OM-10”>
<viewFinder>twin mirror</viewFinder>
<optics>
  <Lens>
    <focal-length>75-300mm zoom</focal-length>
    <f-stop>4.0-4.5</f-stop>
  </Lens>
</optics>
<shutter-speed>1/2000 sec. to 10 sec.</shutter-speed>
</SLR>
```

Eksempel 1-38 Eksempel på et kamera dokument, Bruger terminologien (tag) SLR, f-stop, focal-lenght

```
<Camera rdf:ID=”Olympus-OM-10”>
  <viewFinder>twin mirror</viewFinder>
  <optics>
    <Lens>
      <size>300mm zoom</size>
      <aperture>4.5</aperture>
    </Lens>
  </optics>
  <shutter-speed>1/2000 sec. to 10 sec.</shutter-speed>
</Camera>
```

Eksempel 1-39 Andet eksempel på kamera dokument. Bruger terminologien (tag)Camera, aperture, (lens) size

Ved brug af det fælles logiske design (OWL Ontologien), kan de to personer umiddelbart samarbejde.

Hvis As applikation modtager et instans dokument (i RDF/XML) fra en handels partner B, og As applikation er kodet til at forstå terminologien:

Camera, aperture, (lens) size

Handels partneren udnytter XMLs fleksibilitet og har valgt at bruge terminologien

SLR, f-stop, focal-lenght

Når As applikation parser XML dokumentet, det har modtaget fra B, finder det en klasse, den ikke forstår (<SLR> taget). Den konsulterer derfor kamera ontologien

"Hvad kender du til SLR?"

Ontologien svarer

"SLR er a typen camera"

Denne viden er linket for As applikation, til at forstå relationen mellem noget den ikke kender (SLR) og noget den kender (Camera).

Applikationen fortsætter med at parse og kommer til <f-stop>, igen må den konsulterer kamera ontologien:

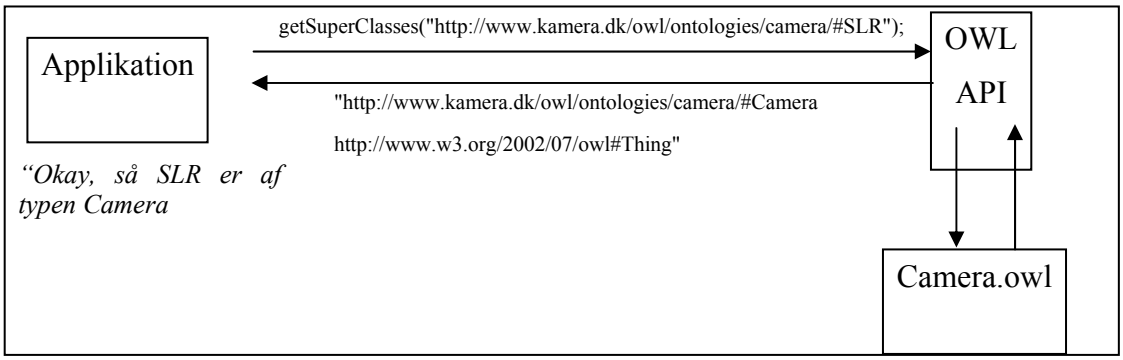
"Hvad kender du til f-stop?"

og ontologien svarer

"f-stop er synonymt med aperture."

Det viser igen, hvordan viden oplagret i ontologien fjerner terminologi hullet, mellem hvad As applikation ikke kender og det som Bs applikation kender. Det giver interoperabilitet på trods af forskel i terminologien. Der kan være mange fysiske repræsentationer i forskellige former. Fleksibiliteten af XML er blevet væsentligt forøget.

Når en applikation behandler et fysisk udtryk, skal det konsultere en ontologi for terminologi det ikke kender til (Figur 1-30).



Figur 1-30 Applikation der tilgår en ontologi

Som tidligere omtalt giver dette en næsten lineær semantisk integration (man skal kun integrerer til en ontologi), i stedet for en n^2 integration hvor alle n partnere skal integrere til hinanden i forholdet en-til-en. Hver part kan danne fysiske udtryk på den måde, som bedst passer til deres behov og ønsker. Dette i modsætning til de tidligere diskuterede markup language (ML), hvor man præcis skulle overholde udtrykkende i disse ML.

For at opnå interoperabilitet er følgende ingredienser nødvendige:

- En OWL ontologi
- Fysisk udtryk (En instans af OWL dokumentet)
- En OWL parser
- Et OWL forespørgelse værktøj

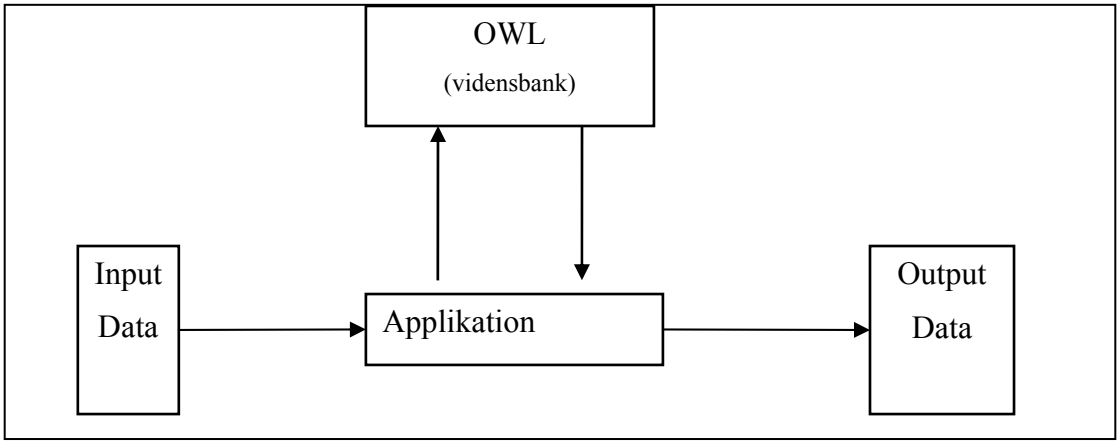
1.10 Opsummering på OWL

En ontologi er et XML dokument beskrevet i OWL som en samling af RDF triples, hver sætning med formen (grundled, udsagnsled, genstandsled), hvor grundleddet og genstandsleddet er ontologi objekter eller værdier og udsagnsleddet er egenskabsrelationer defineret af ontologien.

En OWL ontologi er en sekvens af aksiomer (grundsætninger) og fakta. Klasser, egenskaber og instanser defineret af ontologien har en identifikation, som er en URI reference. OWL datatyper betegner en mængde af værdier, klasser betegner en mængde af instanser, egenskaber relaterer instanser til data værdier (datatype egenskaber) eller til andre instanser (objekt egenskaber)

OWL datatyper er XML Schema Datatyper.

Man kan betragte OWL som en vidensbank, som ens applikation kan udnytte helt på linie med databaser og lignende.



Figur 1-31 OWL som vidensbank

Når man fortæller en anden person noget, kan denne kombinere den nye viden med gammel og fortælle en noget nyt. Når man fortæller en computer noget i XML, kan det måske fortælle en noget nyt, men det er kun på grund af noget software det bruger og som ikke er del af XML specifikationen. Dette software kunne være implementeret forskelligt på forskellige computere, samtidig med at de stadig overholder XML specifikationen. Man vil derfor måske få forskellige svar fra disse systemer. Når man fortæller en computer noget nyt i OWL, kan den give en ny information, baseret alene på OWL standarden. OWL giver computerne en lille ekstra smule autonomi (selvstyre), som gør dem i stand til at udføre lidt mere meningsfuldt arbejde for brugerne.

En mængde af OWL sætninger, tillader selv at konkludere andre OWL sætninger, hvorimod en samling af XML sætninger, ikke tillader XML selv at konkludere andre XML sætninger. For at udnytte XML til at generere ny data, er det nødvendigt at inkludere viden i noget kode et eller andet sted. I stedet for at angive det eksplicit som i OWL. Det understreger en af grundene til at bruge en ontologi, det tillader en næsten lineær semantisk integration frem for n^2 integration. Hver applikation/database relaterer til "lingua franca[2]" af ontologien, fremfor til hinanden.

[2] Lingua Franca: Et medie til kommunikation mellem personer med forskellige sprog

Hvis vi for eksempel har angivet at (*mor* subpropertyOf *forældreTil)* og at (Hanne morTil Anders). Kan man stille spørgsmålet "Hvem er Anders forældre" og få svaret (Hanne forældreTil Anders), uden at det er angivet nogle steder, det kan kun udledes af en OWL applikation.

OWL indeholder den basale infrastruktur, som tillader maskiner at drage de samme slags simple slutninger, som mennesker gør.

Figur 1-32 indeholder en samlet oversigt over OWL.

Symmetric: if P(x,y) then P(y,x)
inverseOf: if P1(x,y) then P2(y,x)
Transitive: if P(x,y) and P(y,z) then P(x,z)
Functional: if P(x,y) and P(x,z) then y=z
InverseFunctional: if P(x,y) and P(z,y) then x=z
allValuesFrom: P(x,y) has y=allValuesFrom©
someValuesFrom: P(x,y) has y=someValuesFrom©
hasValue: P(x,y) and y=hasValue(I)
cardinality: cardinality(P) = *n*
minCardinality: minCardinality(P) = *n*
maxCardinality: maxCardinality(P) = *n*
equivalentProperty: P1 = P2
intersectionOf: C = intersectionOf(C1, C2, …)
unionOf: C = unionOf(C1, C2, …)
complementOf: C = complementOf(C1)
oneOf: C = oneOf(I1, I2, …)
equivalentClass: C1 = C2
disjointWith: C1 != C2
sameIndividualAs: I1 = I2
differentFrom: I1 != I2
AllDifferent: I1 != I2, I1 != I3, I2 != I3, …
Thing: C1, C2, …, I1, I2, …, P1, P2, …

Kilde [SUMOWL]

Figur 1-32 Samlet overblik over OWL

Et OWL dokument indeholder information om:

- **Klasse hierarkiet,** et OWL dokument definerer class/subslass relationer
- **Synonym,** et OWL dokument identificerer ækvivalent klasser og ækvivalente egenskaber
- **Associationer mellem klasser,** et OWL dokument sammenholder en eller flere klasser til en eller flere klasser via en egenskab (f.eks. domæne/range information)
- **Metadata om egenskaber,** et OWL dokument indeholder en mængde metadata om egenskaberne
- **Definition af klasser,** et OWL dokument specificerer sammensætningen af klasser.

Hvor vigtige ontologier bliver for virksomheden, kan også ses af følgende Gartner Group citat:

"By 2005, lightweight ontologies will be part of 75 percent of application integration projects. The relative scarcity of skills in semantic modeling and the unification of information models may be the greatest challenge. Beyond initial development, the need for ongoing information-management processes at the enterprise level will severely tax most enterprises"

Figur 1-33 Gartner Groups forventninger til brug af Ontologier

Med definitionen af OWL eksisterer der nu et ontologisprog, der kan gøre Semantic Web vision virkelig. Om det er den rette løsning til at opnå en kritisk masse, vil kun tiden vise, men fundamentet er i hvert fald solidt.