OWL - Web Ontology Language

Hansi Keijonen

Seminaariraportti
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 7. maaliskuuta 2013

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Hansi Keijonen

Työn nimi — Arbetets titel — Title

OWL - Web Ontology Language

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaariraportti	7. maaliskuuta 2013	19

Tiivistelmä — Referat — Abstract

Tiivistelmä.

Avainsanat — Nyckelord — Keywords
avainsana 1, avainsana 2, avainsana 3

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

use case

1 Johdanto

Semanttinen web on visio tulevaisuuden webistä, jossa informaatiolle annetaan eksplisiittinen merkitys mahdollistaen näin koneiden kyky prosessoida ja yhdistellä webissä olevaa tietoa [6]. Tim Berners-Lee, James Hendler ja Ora Lassila toteavat artikkelissa "Semantic web", että

> "Semanttinen web ei ole erillinen web vaan laajennos tämänhetkiseen webiin, jossa informaatiolle on annettu hyvin muotoiltu merkitys mahdollistaen koneiden ja ihmisten paremman yhteistyön. TULEEX VIITE MIHIN?"

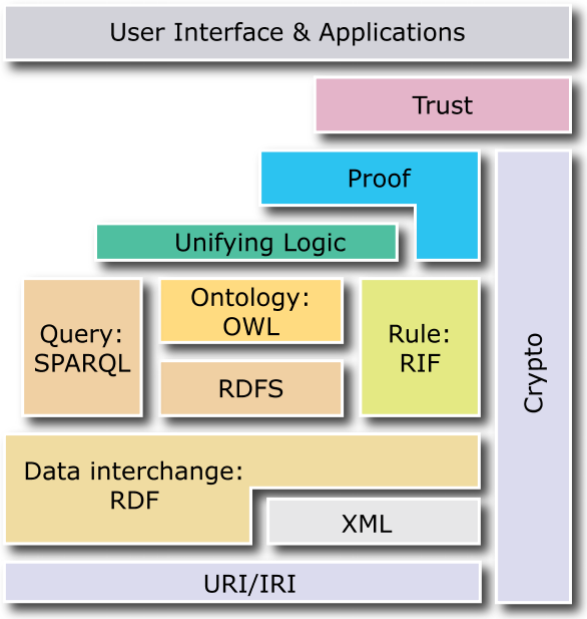
Suurin osa tämän päivän webin sisällöstä on tarkoitettu ihmisten luettavaksi sekä tulkittavaksi. Kone pystyy tulkitsemaan esim. html-tiedoston ja esittämään dokumentin siinä määritellyllä tavalla mutta se ei *ymmärrä* dokumentin sisällön merkitystä, semantiikkaa [2].Tämä rajoittaa esimerkiksi haut internetissä olevista dokumenteista yksinkertaiseksi hakusanojen etsimiseksi. Sen sijaan jos hakukoneet ymmärtäisivät asioiden merkityksen ja niiden välillä vallitsevat yhteydet, olisivat hakutulokset tarkempia ja sisältäisivät mahdollisesti laajennettuja hakuja alkuperäisen asian ympäriltä [4]. Semanttisella webillä on mahdollisuuksia myös verkkokaupankäynnissä, jossa myyjäagentit ja ostaja-agentit voivat kommunikoida keskenään tuotetietojen pohjalta luotujen ontologioiden avulla [4]. Myös eri toimijoiden tuottamien web-palveluiden koostamisessa semanttisen webin teknologioilla on keskeinen rooli: palveluiden tuottajat voivat kuvata tarjoamansa palvelun jolloin palveluita kokoavat sovellukset voivat niitä hyödyntää tehokkaasti [4]. Maailmanlaajuinen tietoverkko tulisi siis muuttaa dokumenttien verkosta tiedon verkoksi [2].

Semantiikkaa voidaan webissä ilmaista ontologioilla. Tietojenkäsittelytieteessä ontologialla tarkoitetaan dokumenttia, jossa kerrrotaan asioiden välisistä yhteyksistä [2]. Ontologioiden luomiseksi pitää olla menetelmiä käsitteiden määritykseen, käsitteiden ominaisuuksien määritykseen sekä käsitteiden välisten suhteiden määritykseen [2]. World Wide Web Consortium W3C on määritellyt joukon standardeja (suosituksia) kielille ja sanastoille, joilla merkitysten määrittäminen voidaan toteuttaa. Tässä artikkelissa selvitetään pääasiassa OWL:n periaatteita. Lyhenne OWL tulee sanoista Web Ontology Language ja sille on W3C:n suositus standardiksi vuodelta 2004. Uudempi suositus on OWL2:lle vuodelta 2012.

Jotta olisi mahdollista ymmärtää OWL:n toimintaperiaate on käytävä soveltuvin osin läpi myös teknologiat, joihin se perustuu. Tässä artikkelissa esitellään RDF ja RDF Schema, joiden konstruktioihin OWL vahvasti nojaa. Lopuksi selvitetään lyhyesti OWL:n laajennoksen OWL2:n tuomat lisäominaisuudet.

2 Teknologiat ja kielet jotka mahdollistavat OWL:n

W3C tarjoaa semanttisen webin toteuttamiseen suositukset teknologioista ja kielistä. Kuvassa 1 on semanttisen webin teknologia- ja konseptipino. Osa teknologioista on jo todellisuutta ja käytössä, osa vasta ideatasolla. Jokainen kerros hyödyntää alemman kerroksen toteuttamia palveluita. Seuraavissa kappaleissa käydään läpi kaavion teknologioita ja kieliä alhaalta ylöspäin kohti OWL:ää. Jokaisesta teknologiasta ja kielestä käsitellään tarkemmin ne konstruktiot, jotka ovat olennaisia ja käytössä myös OWL:ssä.



Kuva 1: Semanttisen webin toteutukseen tarvittavat teknologiat, kielet ja konseptit. Lähde: http://commons.wikimedia.org/wiki/File:Semantic_Web_Stack.png

2.1 URI ja XML

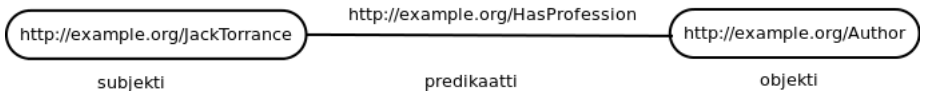
Semanttisessa webissä määritettyjä luokkia, ilmentymiä, ominaisuuksia ja ominaisuuksien arvoja kutsutaan resursseiksi [2]. Jotta sekaannusta jo määritettyjen resurssien sekä uusien määritysten kanssa ei syntyisi, identifioidaan kaikki resurssit yksilöllisesti URI(Unified Resource Identifier):n avulla. URI:n avulla voidaan viitata mihin tahansa resurssiin webissä. Useimmiten URIna toimii perinteinen URL(Unified Resourse Locator)-osoite [2]. IRI (Internationalized Resource Identifier) on ainoastaan merkistölaajennos URI:in.

Semanttisen webin kuvaukset toteutetaan useimmiten XML-kielellä. XML-kieli on notaatio monimutkaisen rakenteisen tiedon esittämiseen ja tarjoaa

näin standardoidun mallin tiedon vaihtamiseen prosessoijien välillä. Tärkeä XML:n ominaisuus on nimiavaruudet, jotka mahdollistavat ja helpottavat resurssien identifiointia URI:en avulla. Nimiavaruuksien käyttöä selvitetään tarkemmin OWL-ontologioiden yhteydessä.

2.2 Tiedon esittäminen RDF-triploilla

Resource Description Framework RDF on kieli webissä olevien resurssien kuvaamiseen [5]. RDF perustuu resurssien identifiointiin URI:lla ja näiden kuvaamiseen ominaisuuksilla ja ominaisuuksien arvoilla. Tämä mahdollistaa yksinkertaisten lausumien esittämisen verkkoina, joissa resurssit ja ominaisuuksien arvot ovat soluja ja ominaisuudet verkon kaaria [5].



Kuva 2: RDF-tripla joka kuvaa yksinkertaisen lausuman. Jokainen triplan solmu ja kaari on identifioitu URI:lla.

Kuvassa 2 on kuvattu yksinkertainen *RDF-tripla*. Triplan subjekti, predikaatti ja objekti kertovat, että "Jack Torrance on ammatiltaan kirjailija". Subjekti on siis asia, jota kuvataan, predikaatti on ominaisuus, jolla kuvataan ja objekti on ominaisuuden arvo [5]. Jokainen solu ja kaari on esimerkissä identifioitu URI:lla. Objekti voi olla myös literaali, jolloin sitä ei identifioida erikseen, mutta formaatti voidaan määritellään esimerkiksi XML Scheman datatyyppien avulla.

Yleisin tapa esittää tripla on XML-notaatio. Myös muut tavat ovat mahdollisia, kuten esimerkiksi *JSON* [1] ja *turtle* [2]. Alla on esitetty kuvassa 2 esitetyn triplan XML-notaatio:

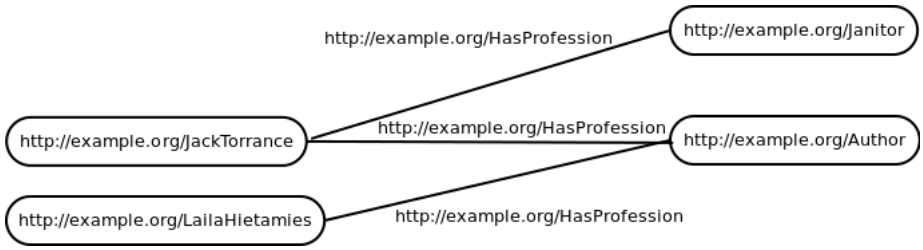
```
<?xml version="1.0"?>

<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://www.esimerkki.com/">

    <rdf:Description rdf:about="http://esim.fi/JackTorrance">
        <ex:onAmmatiltaan rdf:resource="http://esim.fi/Kirjailija"/>
    </rdf:Description>
</rdf:RDF>
```

Esimerkistä näkee, kuinka subjekti (Jack Torrance)ja objekti (Kirjailija) ovat identifioitu eksplisiittisellä URI:lla. Predikaattiin (onAmmatiltaan) viitataan

[1] JSON on nönnöö
[2] turtle on nönnöö

myös URI:lla, mutta nimiavaruuden kautta. Esimerkistä ilmenee RDF:n kolme peruskonstruktiota: *rdf:Description* ilmoittaa, että kyseessä on kuvaus, *rdf:about* viittaa subjektiin, *rdf:resource* viittaa resurssiin, tässä tapauksessa ominaisuuden arvoon.

****tää seuraava on ihan kiva läppä, mutta onko paikka tässä? mahdollisesti ennen xml-esimerkkiä?********** Jack Torrance on toiselta ammatiltaan talonmies ja myös Laila Hietamies on ammatiltaan kirjailija. Kun myös nämä triplat lisätään kuvan 2 kaavioon, alkaa pieni mutta informatiivinen verkko syntyä (kuva 3). Tästä verkosta voisimme tehdä jo hakuja, kuten "listaa henkilöt, jotka ovat kirjailijoita".



Kuva 3: RDF-triplat muodostavat verkon.

RDF tarjoaa melko alkeellisen tavan esittää lausumia, jotka muodostavat haut mahdollistavan verkon. RDF toteuttaa myös joukon muita ominaisuuksia, kuten *säiliöitä* (container) tiedon säilömiseen sekä *kokoomatietorakenteita* (collections) asioiden listaamiseen [5]. Näitä käsitellään myöhemmin niiltä osin kuin ne ovat relevantteja OWL-ontologioiden muodostamisessa.

2.3 Yksinkertaiset ontologiat RDF Schemalla

Vaikka RDF:llä voidaan kuvata resursseja ominaisuuksien avulla ja määrittää näin resurssien välisiä suhteita, se ei tarjoa keinoja määrittää itse ominaisuuksia tai ominaisuuksien välisiä suhteita [3]. Tämän mahdollistaa RDF:n *sanastonkuvauslaajennos* RDF Schema. RDF Schemalla on mahdollista määrittää resurssien joukkoja luokiksi (class) sekä näiden (luokkien ja yksilöiden) välisiä suhteita ominaisuuksiksi (property) [3]. RDF Schema on RDF:n semanttinen laajennos joka ei kuitenkaan tarjoa sanastoa asioiden ymmärtämiseen, vaan se tarjoaa työkaluja sanastojen luomiseen [3].

RDF Scheman yksi peruselementti on siis *luokka*, jolla voi olla myös *aliluokkia*. Luokan jäsenet ovat *yksilöitä*, luokan *ilmentymiä*. Toinen peruselementti on *ominaisuus*. Ominaisuus yhdistää luokan ilmentymiä toisiinsa, luo suhteen niiden välille. Ominaisuudella voi olla myös *aliominaisuuksia*. Näiden elementtien avulla voidaan määrittää melko yksinkertaisia luokkien ja suhteiden hierarkkisten järjestelmien kuvauksia, (kevyt)ontologioita.

Seuraavissa kappaleissa esitellään niitä RDF Scheman peruskonstruktioita, joita myös OWL käyttää toteutuksessaan lähes sellaisenaan. *RDFS*

Schema määrittelee myös suurehkon joukon muita konstruktioita, mutta niiden käsittely on OWL:n esittelyn kannalta tässä tarpeetonta. Huomionarvoista on, että RDF Schema -dokumentit ovat RDF-dokumentteja, jotka sisältävät kummankin kielen primitiivejä. **Kaikissa tämän paperin koodiesimerkeissä on esitetty ainoastaan esittelyssä oleva konstruktio. Nimiavaruusmäärittelyt, otsikkotiedot jne. puuttuvat esimerkeistä yksinkertaisuuden vuoksi.**

2.3.1 Luokka ja yksilö

Resurssit voidaan jaotella luokkiin. Luokkien jäseniä nimitetään luokan ilmentymiksi tai yksilöiksi[3]. Luokat ovat myös itse resursseja. Useimmiten luokat identifioidaan RDF:n URI:lla ja niitä voidaan kuvata RDF:n ominaisuuksilla (property). Luokka määritellään rdfs:Class elementissä. rdf:type -elementillä ilmaistaan, että resurssi kuuluu määrättyyn luokkaan [3]. Esimerkiksi kustannusosakeyhtiö "WSOY"on luokan "Kustantamo"jäsen:

```
<rdfs:Class rdf:ID="WSOY" />
    <rdf:type rdf:resource="#Kustantamo"/>
</rdfs:Class>
```

2.3.2 Aliluokka

rdfs:subClassOf -elementillä voidaan ilmaista, että kaikki jonkin luokan jäsenet ovat myös jonkin toisen luokan jäseniä [3]. Esimerkiksi luokka "Kustantamo"on luokan "Liikeyritys"aliluokka:

```
<rdfs:Class rdf:ID="Kustantamo" />
    <rdf:subClassOf rdf:resource="#Liikeyritys"/>
</rdfs:Class>
```

Kaikki yksilöt, jotka kuuluvat luokkaan Kustantamo kuuluvat myös luokkaan Liikeyritys.

2.3.3 Ominaisuus ja periytetty aliominaisuus

Resursien välisiä suhteita kuvataan *ominaisuuksilla*. Esimerkiksi "Matilla on veli Teppo". Voidaan ajatella, että resurssien "Matti"ja "Teppo"välinen suhde on ominaisuus "Veljeys". Ominaisuus "Veljeys"voi olla ominaisuuden "Perhesuhde"aliluokka tai paremminkin aliominaisuus. Tällaisia konstruktioita kuvataan RDF Schemassa termillä subPropertyOf:

```
<rdf:Property rdf:about="Veljeys">
    <rdfs:subPropertyOf rdf:resource="#Perhesuhde" />
</rdf:Property>
```

```
<rdf:Description rdf:about="#Matti">
    <Veljeys rdf:resource="#Teppo" />
</rdf:Description>
```

Esimerkissä on määritetty ensiksi ominaisuus "Veljeys", joka on ominaisuden "Perhesuhde"aliominaisuus. Seuraavaksi kuvataan yksilöä "Matti"ominaisuudella "Veljeys", joka saa arvokseen resurssin "Teppo". subPropertyOf ilmaisee määritelmällisesti, että resurssit joiden välistä suhdetta kuvataan jollain suhteella, voidaan kuvata myös tämän suhteen ylisuhteella, josta ko. suhde on periytynyt [3]. Matin ja Tepon suhdetta voidaan kuvailla ominaisuudella "Veljeys", mutta myös ominaisuudella "Perhesuhde".

2.3.4 Rajoitukset ominaisuuden sovellusalueessa ja arvojoukossa

RDF Schemassa on mahdollista määrittää ominaisuuksille rajoituksia sen suhteen, että minkä luokkien jäsenten välistä suhdetta ominaisuus voi kuvata. Ensinnäkin voidaan rajoittaa ominaisuuden sovellusaluetta (domain). Esimerkiksi ominaisuuden "Työskentelee"sovellusalue voidaan rajata koskemaan ainoastaan luokkaa "Työntekijä". Samoin ominaisuuden saamat arvot (range) voidaan rajata olemaan ainoastaan luokan "Yritys"tai sen aliluokkien ilmentymiä:

```
<rdf:Property rdf:about="Tyoskentelee">
    <rdfs:domain rdf:resource="#Tyontekija" />
    <rdfs:range rdf:resource="#Yritys" />
</rdf:Property>
```

Sovellusalueen rajaus määritetään rdfs:domain -elementillä ja arvojoukon rajaus rdfs:range -elementillä.

3 Kehittyneitä ontologioita OWL:llä

Kehittyneempi tapa ilmaista ontologioita on OWL-ontologiat. Lyhenne OWL tulee sanoista Web Ontology Language. Vaikka määritelmässä on sana language, kieli, on OWL-ontologiat ymmärettävä ennemminkin sanastoina, joita on kuvattu RDF-kielellä. Eräs tapa hahmottaa RDF-triplojen ja OWL-ontologioiden välinen ero on verrata niitä perinteiseen relaatiotietokantaan. RDF-triplat on tapa tallettaa tietoa olioiden ominaisuuksista samalla tavalla kuin relaatiotietokannan taulun riveillä tallennetaan rakenteista tietoa tietokantaolioista. Jokaista riviä relaatiotietokannassa yksilöi yksilöivä avain kun taas RDF-triploissa avaimen virkaa hoitaa URI. Relaatiotietokannoissa tietokantaolion attribuuttien suhteita ilmaistaan taulurakenteilla ja tietokantaolioiden suhteita toisiin tietokantaolioihin ilmaistaan viitteillä taulujen välillä. Vastaavasti OWL-ontologiat kuvaavat ja jäsentävät samalla tavalla RDF-triploilla ilmaistua tietoa: ontologialla voidaan kuvailla monipuolisesti

luokkia ja niiden ilmentymiä sekä ilmentymien suhteita toisiin ominaisuuksien avulla. RDF on kuin kieli jolla ilmaistaan lausumia asioista kun taas OWL on sanasto, jonka avulla lausumien merkitykset ymmärretään.

OWL tarjoaa ontologioiden määrittelyyn [1]:

- *hyvin määritellyn kieliopin* , jotta ontologiat olisivat koneluettavissa
- *hyvin määritellyn semantiikan*, jolla voi ilmaista merkityksiä tarkasti ja konsistentisti
- *tuen koneelliselle päättelylle*, jotta esim. ontologioiden eheys voidaan tarkastaa automaattisesti
- *riittävästi ilmaisuvoimaa* ilmaisemaan kaikki tarvittavat merkitykset
- *miellyttävän ilmaisutavan*, jotta työskentely olisi sujuvaa

Ideaalisesti OWL on RDF:n ja RDF Scheman laajennos [1]. OWL käyttää RDF:n luokkia ja suhteita lisäten niihin omia laajennoksiaan. RDF Schemassa on joitain hyvin vahvoja konstruktioita, kuten rfd:Class (kaikkien luokkien yliluokka) sekä rdf:Property (kaikkien suhteiden yliluokka). Näiden primitiivien ilmaisuvoima yhdistettynä OWL:n tarjoamaan laajennoksiin on ristiriidassa sen tavoitteen kanssa, että ontologiat olisivat koneellisesti pääteltävissä. Tämä tasapainotila mielessäpitäen on määritelty kolme OWL:n alikieltä sen perusteella, painotetaanko ilmaisuvoimaa vai koneellista päättelyä [1].

3.1 OWL:n kolme alikieltä

W3C:n Web Ontology Working Group on määritellyt OWL:lle kolme alikieltä, joiden on takoitus toteuttaa eri aspektit (ilmaisuvoima, koneellinen päättely), joita ontologioiden kuvaamiskieleltä vaaditaan [6]. Alikielet ovat ilmaisuvoiman mukaisesti kasvavassa järjestyksessä:

- *OWL Lite* tarjoaa ainoastaan yksinkertaisen luokitteluhierarkian ja yksinkertaiset rajoitteet[6]. Kardinaalisuusrajoitteet ovat ainoastaan muotoa 0 ta 1. Työkalutuen tarjoaminen on OWL Litelle helpompaa kuin ilmaisuvoimaisemmille versioille [6].
- *OWL DL* on tarkoitettu niille käyttäjille, jotka haluavat mahdollisimman hyvän ilmaisukyvyn siten, että ontologia on koneellisesti pääteltävissä (kaikki päätelmät tehtävissä järjellisessä ajassa) [6]. OWL DL tarjoaa kaikki kielen konstruktiot, mutta niitä voi käyttää tietyin rajoituksin, esimerkiksi luokka voi olla monen luokan aliluokka mutta ei voi olla samalla luokan ilmentymä. DL tulee sanoista Description Logics, deskriptiivinen logiikka, joka on tieteenala, joka tutkii logiikkaa ja on OWL:n perusta [6].

- *OWL Full* on tarkoitettu niille käyttäjille, jotka haluavat maksimaalisen ilmaisukyvyn välittämättä siitä, onko ontologiat enää koneellisesti pääteltävissä [6]. Toisin kuin OWL DL:ssä, luokka voi olla kokoelma yksilöitä (instansseja) samalla kuin luokka itsessään on jonkin luokan yksilö. OWL Full mahdollistaa jo olemassa olevien ontologioiden laajentamisen. On epätodennäköistä, että mikään ohjelmisto pystyy täydellisesti päättelemään OWL Full ontologioita [6].

3.2 OWL-ontologian rakenne

3.2.1 Nimiavaruudet

OWL-dokumentissa tulee määritellä nimiavaruudet (namespace). Nimiavaruuksien avulla voidaan ratkaista mm. samannimisten elementtien aiheuttamia tulkintaongelmia sekä kertoa lukijalle (koneelle tai ihmiselle) konteksti, jonka mukaan elementtien tageja tulee tulkita. OWL-ontologiassa nimiavaruudet määritellään rdf:RDF -kahvojen sisään. *Kaikki OWL-ontologiat ovat RDF-dokumentteja* [8]. Alla olevassa esimerkissä on eräs mahdollinen nimiavaruusmäärittely .

```
<rdf:RDF
    xmlns:esim ="http://www.esimerkki.com/esim#" //vittuun?
    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#">
```

Esimerkin nimiavaruusmäärittelyissä on määritelty nimiavaruus niille tageille, jotka käyttävät etuliitettä esim:. Nimiavaruudet on määritelty myös owl:-, rdf:- ja rdfs:-etuliiteille kertomaan, että näillä etuliitteillä varustetut tagit edustavat OWL:n, RDF:n ja RDF Schemann termistöä. OWL-ontologiassa käytetään myös XMLSchema-datatyypeistä (xsd:), joten myös niiden nimiavaruus tulee määrittää.

3.2.2 Otsikkotiedot

Owl-ontologian otsikkotiedoissa voidaan kertoa yleisiä asioita kuten versiotietoa, kommentteja ja ontologian nimi. Tärkeä ominaisuus on mahdollisuus tuoda toisen tahon määrittelemiä ontologioita itse määriteltävän ontologian käyttöön [8]. Otsikkotiedot määritellään owl:Ontology-elementiksi:

```
<owl:Ontology>
   <rdfs:label>Esimerkkiontologia</rdfs:label>
   <rdfs:comment>Esimerkin voimaa</rdfs:comment>
   <owl:priorVersion>
       rdf:resource="http://www.esimerkki.com/vanhempi#"
```

```
   </owl:priorVersion>
   <owl:imports>rdf:resource="http://purl.org/dc/elements/1.1"</owl:imports>
</owl:Ontology>
```

Esimerkkiotsikossa perustietojen kertomisen lisäksi tuodaan ontologian käyttöön Dublin Core - sanasto [3].

3.2.3 Yksinkertaiset luokat ja aliluokat

OWL-ontologiassa kaikki ilmentymät ovat luokan owl:Thing jäseniä ja kaikki käyttäjän määrittämät luokat owl:Thingin aliluokkia [8]. Myös owl:Nothing on määritelty. Luokkamäärittelyt tapahtuvat hyvin samaan tapaan kuin RDF Schemassa, ainoastaan tagin nimi on owl:Class. Aliluokkanotaatio on RDF Schemasta:

```
<owl:Class  rdf:ID="Auto">
<owl:Class  rdf:ID="Ajoneuvo">

<owl:Class rdf:ID="Auto">
    <rdfs:subClassOf rdf:resource="Ajoneuvo"/>
</owl:Class>
```

Esimerkissä on määritetty kaksi luokkaa "Auto"ja "Ajoneuvo"sekä määritelty edellinen jälkimmäisen aliluokaksi.

OWL:ssä on myös mahdollista määritellä luokka siten, että luokan jäsenet voivat kuulua ko. luokkaan mutta eivät missään tapauksessa implisiittisesti määritettyyn toiseen luokkaan [8]:

```
<owl:Class ref:about="Ajoneuvo">
    <owl:disjointWith rdf:resource="Elain"
</owl>
```

Luokan "Ajoneuvo"ja sen mahdollisten aliluokkien jäsenet eivät voi siis kuulua samaan aikaan luokkaan "Kissa"tai sen mahdollisiin aliluokkiin. Voidaan siis päätellä, että ilmentymä "Lada"ei voi olla luokan "Kissa"jäsen jos "Kissa"on määritetty luokan "Elain"aliluokaksi.

3.2.4 Luokan yksilöt

Luokan jäseniä kutsutaan yksilöiksi, eli luokan ilmentymiksi [8]. Ilmentymän kuuluminen johonkin luokkaan ilmaistaan samalla tavalla kuin RDF Schemassa rdf:type konstruktiolla. Ensiksi määritellään kuitenkin instanssi owl:Thing-elementissä:

[3] Dublin Core on nnnnn

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

<owl:Thing ref:about="Lada">
    <rdf:type= rdf:resource="Autonvalmistaja"/>
</owl:Thing>
```

Esimerkissä määritettiin "Autonvalmistaja-luokan ilmentymä "Lada". Esimerkissä viitataan oletetusti jo aiemmin määritettyyn "Autonvalmistaja-luokkaan.

3.2.5 Luokkaominaisuus ja datatyyppiominaisuus

Ominaisuudet jaetaan OWL:ssä luokkaominaisuuksiksi (Object Property) ja datatyyppiominaisuuksiksi (Data Type Property) sen perusteella liittääkö ominaisuus yhteen kaksi luokan ilmentymää vai luokan ilmentymän ja RDF-literaalin tai XML Schema -datatyypin [8].

3.2.6 Luokkaominaisuus

OWL:ssä luokkaominaisuudet määritellään kuten mitkä tahansa luokat, mutta käyttäen ObjectProperty -konstruktiota. Määrittelyssä kerrotaan myös minkä luokan ilmentymiin ominaisuus on sovellettavissa sekä mitä arvoja (luokkia) ominaisuus voi saada. Nämä rajoitteet ilmaistaan RDF Schemasta tutuilla rdfs:domain- ja rdfs:range-elementeillä:

```
<owl:ObjectProperty rdf:about="Merkki"/>

<owl:ObjectProperty rdf:ID="Merkki">
    <rdfs:domain rdf:resource="Auto"/>
    <rdfs:range rdf:resource="Autonvalmistaja"/>
</owl:ObjectProperty>
```

Esimerkissä on määritelty ominaisuus "Merkki", jolla voi kuvata ainoastaan luokan "Auto"ilmentymiä ja joka voi ainoastaan saada arvokseen ainoastaan luokan "Autonvalmistaja"ilmentymiä.

3.2.7 Datatyyppiominaisuus

Datatyyppiominaisuus liittää yhteen luokan ilmentymän ja datan arvon: RDF-literaalin tai XML Schema -datatyypin. Myös datatyyppiominaisuuksien määrittelyssä kerrotaan minkä luokan ilmentymiin ominaisuus on sovellettavissa sekä minkä tyyppisiä arvoja ominaisuus voi saada [8]. Rajoitteet tehdään samalla tavalla kuin luokkaominaisuuden määrittelyssä:

```
<owl:DataTypeProperty rdf:about="Valmistusvuosi"/>
```

```
<owl:DataTypeProperty rdf:ID="Valmistusvuosi">
    <rdfs:domain rdf:resource="Auto"/>
    <rdfs:range  rdf:resource="&xsd;gYear"/>
<owl:dataTypeProperty>
```

Esimerkissä määritetään ominaisuus "Valmistusvuosi", jolla voidaan kuvata luokan "Auto"ilmentymiä ja joka voi saada arvokseen XML Schema -standardissa määritetyn gYear-tyypin arvon. XML-Schema -standardi määrittelee kymmeniä eri datatyyppejä XML-dokumenteissa käytettäväksi. Kun käytetään määritettyjä datatyyppejä, XML-parseri pystyy tarkistamaan, onko annetut arvot sallittuja ja esitystapa oikea.

3.2.8 Aliominaisuudet

Myös ominaisuuksille voidaan määritellä aliominaisuuksia samaan tapaan kuin luokille voidaan määrittää aliluokkia. Aliominaisuus toteutetaan subPropertyOf-konstruktiolla, joka on toteutettu jo RDF Schemassa [8]. Alla olevassa esimerkissä lisätään ominaisuuteen "Valmistusvuosi"määritys, että se on ominaisuuden "AutonKuvailija"aliominaisuus:

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

<owl:DataTypeProperty rdf:ID="Valmistusvuosi">
    <rdfs:subPropertyOf rdf:resource="AutonKuvailija">
    . . .
<owl:dataTypeProperty>
```

3.2.9 Kardinaalisuusrajoitteet

OWL antaa mahdollisuuden määrätä kardinaalisuusrajoitteita ominaisuuksien arvoille kun ominaisuutta sovelletaan tietyssä kontekstissa [8]. Voidaan esimerkiksi määrätä, että luokka "Ajoneuvo"on joukko asioita, joilla on aina vähintään kaksi arvoa "Renkaat-ominaisuudessa. Toisin sanoen, määritellään anonyymi aliluokka luokalle "Ajoneuvo"[8]. Alla oleva esimerkki selventää asiaa:

```
<owl:Class  rdf:ID="Ajoneuvo">
    <rdfs:subClassOf>
       <owl:Restriction>
           <owl:onProperty rdf:resource="Renkaat"
           <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2
           </owl:minCardinality>
       </owl:Restriction>
    </owl:subClassOf>
<owl:Class/>
```

Anonyymi aliluokka määrätään normaalilla subClassOf-konstruktiolla. Aliluokalle määritetään rajoite owl:Restriction-tagilla ja kardinaalisuusrajoite tässä tapauksessa minCardinality-elementillä. Kardinaalisuus määritetään koskemaan ominaisuutta "Renkaat"onProperty-elementillä. Kardinaalisuusrajoitteita voi määrätä myös maxCardinality-elementillä, joka määrää suhteen yläkardinaliteetin, sekä Cardinality-elementillä, joka määrää tarkan arvon mikä on luokan esiintymän ja ominaisuuden suhde. Voitaisiin esimerkiksi määrittää luokka "Moottoripyörä"siten, että seillä voisi olla tasan kaksi "Renkaat"ominaisuutta.

3.2.10 Ominaisuuden transitiivisuus, symmetrisyys, funktionaalisuus ja inversio

Ominaisuuden voi tarvittaessa määrittää transitiiviseksi, symmetriseksi tai funktionaaliseksi [8]. Näin tekemällä voidaan monipuolistaa ontologioista tehtäviä implisiittisiä päättelyitä. Transitiivinen ominaisuus voidaan määrittää seuraavasti ominaisuudelle P ja ilmentymille x,y ja z [8]:

P(x,y) ja P(y,z) -> P(x,z)

Esimerkiksi omnaisuus rdfs:subClassOf on transitiivinen. Itse määritetyn ominaisuuden voi määrittää transitiiviseksi määrittämällä ominaisuus luokan "TransitiveProperty"jäseneksi:

```
. . .
<rdf:type rdf:resource="&owl;TransitiveProperty"
. . .
```

ESIMERKKI?

Symmetrinen ominaisuus voidaan määrittää ominaisuudelle P ja ilmentymille x,y ja z [8]:

P(x,y) joss P(y,x)

Vastaavasti oman ominaisuuden voi määrittää symmetriseksi määrittämällä se luokan "SymmetricProperty"jäseneksi:

```
. . .
<rdf:type rdf:resource="&owl;SymetricProperty"
. . .
```

Esimerkiksi ominaisuus "Sisar"voitaisin määritellä symmetriseksi. Jos x on y:n sisar niin implisiittisesti voidaan päätellä, että y on x:n sisar.

Funktionaalinen ominaisuus voiadaan määrittää ominaisuudelle P, ilmentymille x,y ja z [8]:

P(x,y) ja P(x,z) -> y = z

Edellisten esimerkkien tapaan ominaisuuden voi määrittää funktionaaliseksi:

```
. . .
<rdf:type rdf:resource="&owl;FunctionalProperty"
. . .
```

Funktionaalinen ominaisuus voisi olla esimerkiksi opiskelijanumero. Jokainen opiskelijaluokan ilmentymä, joka olisi myös saman oppilaitoksen jäsen, voitaisiin identifioida opiskelijanumero-ominaisuuden perusteella, koska on olemassa ainoastaan yksi ominaisuuden ilmentymä, jolla on tietty opiskelijanumero.

INVERSIO JA FUNKTIONAALINEN INVERSIO???

3.2.11 Arvorajoitteet luokkaominaisuuksissa

OWL mahdollistaa rajoittaa ilmentymiä, joita voidaan antaa ominaisuuden arvoiksi luokkaominaisuuksia määritettäessä [8]. Tällä tavoin voidaan rajata aliluokkia näiden rajoitusten avulla. Esimerkiksi on mahdollista määrätä, että luokan "Auto"aliluokka siten, että ominaisuuden "Omistaja"arvona voi olla vain luokan "Henkilo"jäsen:

```
<owl:Class rdf:ID="Auto">
    <rdfs:subClassOf>
       <owl:Restriction>
           <owl:onProperty rdf:resource="Omistaja"/>
           <owl:allValuesFrom rdf:resource="Henkilo"/>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Edellisessä esimerkissä määritettiin, että auton omistaja on aina "Henkilo-luokan jäsen. OWL mahdollistaa myös hieman kevyemmän rajoitteen, jossa määrätään, että ominaisuuden jonkun arvon tulee olla määrätyn luokan jäsen [8]. Voidaan määrittää, että autojen omistajista ainakin yksi on luokan "Henkilo"jasen:

```
. . .
           <owl:onProperty rdf:resource="Omistaja"/>
           <owl:someValuesFrom rdf:resource="Henkilo"/>
. . .
```

Rajoite toteutetaan owl:someValuesFrom-tagilla.

On myös mahdollista määrätä ominaisuuden arvoksi täsmällisesti joku olemassa oleva resurssi [8]. Tällöin aliluokkamäärittelyyn lisätään owl:hasValue -elementti:

```
. . .
           <owl:onProperty rdf:resource="Omistaja"/>
           <owl:hasValue rdf:"Hansi"/>
. . .
```

3.2.12 Kompleksiset luokat

Kompleksiset luokat on erittäin vahva ja ilmaisuvoimainen OWL:n konstruktio [8]. Konstruktio tarjoaa mahdollisuuden määrittää luokkalausekkeita, joissa käytetään joukko-opista tuttuja operaattoreita *yhdistetä*, *leikkausta* ja *komplementtia*. Esimerkiksi voimme määrittää luokan "Lada-auto", joka on luokan "Auto"ja sellaisten asioiden, jotka ovat jollain tapaa "Lada", leikkaus:

```
<owl:Class rdf:ID="Lada-auto">
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Auto"/>
        <owl:Class rdf:about="Lada"/>
    </owl:intersectionOf>
</owl:Class>
```

Esimerkissä leikkaus-operaattori on ilmaistu owl:intersectionOf -elementillä. rdf:parseType="Collection"ilmoittaa parserille, että tämän elementin sisältämät elementit tulee tulkita lueteltuna kokoelmana.

Vastaavasti voidaan määrittää yhdisteluokka käyttämällä elementtiä owl:unionOf [8]. Esimerkkinä voidaan määrittää luokka "Maansiirtokoneet", jonka jäsenet ovat ainakin yhden luetellun luokan jäsen:

```
<owl:Class rdf:ID="Maansiirtokone"/>
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Kauhakuormaaja"/>
        <owl:Class rdf:about="Dumpperi"/>
    </owl:intersectionOf>
</owl:Class>
```

Esimerkissä luokkaan "Maansiirtokone"kuuluu ilmentymät, jotka ovat joko kauhakuormaajia, dumppereita tai peräti molempia.

Komplementtiluokka määritellään elementillä owl:complementOf [8]. Komplementtiluokan jäseniä määrittää se, että ne eivät missään tapauksessa ole komplementiksi ilmoitetun luokan jäseniä. Komplementtiluokalla on helppo esimerkiksi määrittää luokka, "Manuaalivaihteisto", joka määritelmällisesti on luokan "Automaattivaihteisto"komplementti:

```
<owl:Class ref:about="Automaattivaihteisto"/>

<owl:Class ref:about="Manuaalivaihteisto">
    <owl::complementOf rdf:resource="Automaattivaihteisto">
</owl:Class>
```

Esimerkkimäärityksessä luokkaan "Manuaalivaihteisto"kuuluvat vain ne ilmentymät, jotka eivät kuulu luokkaan "Automaattivaihteisto".

3.2.13 Ontologioiden yhdistäminen

Jotta ontologioiden luominen olisi mielekästä, tulee olla kyky yhdistää ontologioita toisiinsa jotta saadaan aikaan uusia laajempia ontologioita [8]. Ontologian otsikkotiedoissa voidaan import-lasuseella tuoda jonkun toisen jo olemassa olevan ontologian määritykset oman ontologian käyttöön. OWL tarjoaa myös elementtejä, joilla voidaan käyttää hyväksi jo olemassa olevia määrityksiä omien määrityksien tekemisessä [8].

owl:equivalentClass -elementillä voidaan oma luokkamääritys määrätä vastaamaan jotain olemassaolevaa määritystä:

```
<owl:Class rdf:ID="Auto">
    <owl:equivalentClass rdf:resource="&kulkuneuvot;Auto"/>
</owl:Class>
```

Esimerkissä siis oletetaan, että otsikkotiedoissa on import-määritys ontologialle "kulkuneuvot". Oma määritys luokalle "Auto"siis asetetaan vastaamaan tuodun ontologian määritystä luokalle "Auto".

OWL mahdollistaa myös määrittää yksittäisen ilmentymän vastaamaan toista ilmentymää [8]. Tämä tehdään sameAs-konstruktiolla:

```
<owl:Class rdf:ID="HansinSuosikkiAuto">
    <owl:sameAs rdf:resource="Lada1200A"/>
</owl:Class>
```

Tällä konstruktiolla ei ole suurtakaan käyttöä. Ainoa, mitä tässä ilmaistaan on, että Hansin suosikkiauto on Ladan klassikkomalli. Käänteinen ominaisuus sameAs-konstruktiolle on differentFrom [8]. Sen käyttö on samaan tapaan suoraviivaista eikä se trivialiteettien lisäksi määritä mitään oleellista.

Käyttökelpoisempi konsruktio on *AllDifferent*, jolla voidaan määrittää kokoelma ilmentymiä olemaan erillisiä toistensa suhteen [8]:

```
<owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
        <Henkilo rdf:about="Matti"/>
        <Henkilo rdf:about="Teppo"/>
        <Henkilo rdf:about="Seppo"/>
    </owl:distinctMembers>
</owl:AllDifferent>
```

Esimerkki kertoo, että jokainen Henkilon ilmentymä viittaa eri ilmentymään.

3.3 OWL 2

OWL 2 on laajennos OWL-kieleen, jolle on annettu W3C-suositus vuonna 2009 [7]. OWL 2:n ominaisuudet ovat suurilta osin samat kuin OWL 1:ssä, osa ainoastaan eri nimellä esitettynä . Myös OWL 2:n ytimen muodostaa

XML/RDF ja mikä tärkeintä, OWL 2 on taaksepäin yhteensopiva OWL 1:n kanssa. Jokainen OWL 1 -ontologia on siis kelvollinen OWL 2 -ontologia.

OWL 2 lisää pääasiassa syntaktista sokeria OWL 1:n päälle. Se tarjoaa kuitenkin myös muutamia uusia toiminnallisuuksia [7]:

- *avaimet*, joka mahdollistaa sen, että ominaisuuksia (tai ominaisuuksien joukkoa) voidaan käyttää avaimena tunnistamaan luokan ilmentymää.
- *ominaisuusketjut*, joka mahdollistaa esimerkiksi määrittää ominaisuuden "Isovanhempi"kahden "Vanhempi-ominaisuuden ketjuksi
- *uusia datatyyppejä*, jotka mahdollistavat esimerkiki ominaisuuksille annettavien numeeristen arvojen rajoittamisen jollekin välille.
- *kvalifioituja kardinaalisuusrajoitteita*
- *asymetriset, refleksiiviset ja poissulkevat ominaisuudet.* Asymmetrisyys on vastakohta OWL 1:n symmetrisyydelle, reflektiivinen ominaisuus viittaa aina itseensä ja poissulkevat ominaisuudet eivät voi olla voimassa samaan aikaan.
- *parannellut kommentointimahdollisuudet*

Lähteet

[1] Antoniou, Grigoris ja Harmelen, Frank Van: *Web Ontology Language: OWL*. Handbook on ontologies, 2(September):91–110, 2009. http://www.springerlink.com/index/10.1007/978-3-540-92673-3.

[2] Berners-Lee, Tim, Hendler, James ja Lassila, Ora: *The Semantic Web*. Scientific American, 284(5):34–43, 2001. http://www.nature.com/doifinder/10.1038/scientificamerican0501-34.

[3] Brickley, Dan ja Guha, R V: *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation, 2009(10 February 2004), 2004. http://www.w3.org/TR/rdf-schema/.

[4] Horrocks, I: *From SHIQ and RDF to OWL: the making of a Web Ontology Language*. Web Semantics Science Services and Agents on the World Wide Web, 1(1):7–26, 2003. http://linkinghub.elsevier.com/retrieve/pii/S1570826803000027.

[5] Manola, Frank ja Miller, Eric: *RDF Primer*. W3C Recommendation, 10(February):1–107, 2004. http://www.w3.org/TR/rdf-primer/.

[6] McGuinness, Deborah L ja Van Harmelen, Frank: *OWL Web Ontology Language Overview*. W3C recommendation, 10(February):1–22, 2004. http://www.w3.org/TR/owl-features/.

[7] Ontology, Web, Document, Language, Recommendation, Latest, Group, O W L Working ja Reserved, All Rights: *OWL 2 Web Ontology Language Document Overview*. October, 2(October):1–12, 2009. http://www.w3.org/TR/owl2-overview/.

[8] Smith, Michael K, Welty, Chris ja McGuinness, Deborah L: *OWL Web Ontology Language Guide*. W3C Recommendation, 10(February):1–46, 2004. http://www.w3.org/TR/owl-guide/.