hyväksymispäivä arvosana

arvostelija

Semanttinen web: ontologioiden esittäminen ja oppiminen

Lilli Nevanlinna

Helsinki 4.11.2010

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Johdanto

Semanttinen web on laajennus olemassa olevaan webiin. Sen tarkoitus on tehdä tiedonhausta tehokkaampaa ja ihmisystävällisempää tarjoamalla hakutuloksiin heti merkityksellistä ja hyvin strukturoitua tietoa. Tämä edellyttää että tietokoneella on keinot tulkita tiedon sisältöä semanttisesti ja siten valita ja yhdistellä semanttisesti järkevää tietoa.

Nykymuodossaan webin sisältö koostuu pääasiassa html-dokumenteista, joissa tieto on ihmisen luettavassa muodossa, mutta koneelle vaikeasti tulkittavaa. Näin ollen hakutoiminnallisuus perustuu hakusanoihin, ja hakutulokset ovat "tyhmiä", mekaanisesti hakusanojen perusteella noudettuja. Ihminen osaa tehdä itselleen tarpeellisen synteesin hakutulosten tiedoista lukemalla relevantilta tuntuvat dokumentit läpi. Kone ei tätä synteesiä osaa tehdä, sillä dokumenttien tietosisältö on piilotettuna luonnolliseen kieleen. Semanttinen web yrittää korjata tätä ongelmaa tarjoamalla teknologioita, joiden avulla dokumenttien tietosisältö olisi, paitsi ihmisen tulkittavana luonnollisena kielenä, myös hyvin struktoroituna datana formaatissa, jota koneen on helppo tulkita.

Semanttinen web vaatii siis toimiakseen, että se informaatio, mikä nyt on html-dokumenteissa luonnollisena kielenä, ja vieläpä sadoilla eri kielillä, olisi tarjolla myös raakana datana koneen tulkittavaksi. Tämän lisäksi tarvitaan sääntöjä, joiden avulla samoihin asioihin liittyvää dataa voidaan yhdistellä ja päätellä uutta tietoa. Tässä on keskeisessä roolissa ontologiat, joiden avulla pyritään mallintamaan mahdollisimman tarkasti jonkin aihealueen käsitteet ja niiden yhteydet.

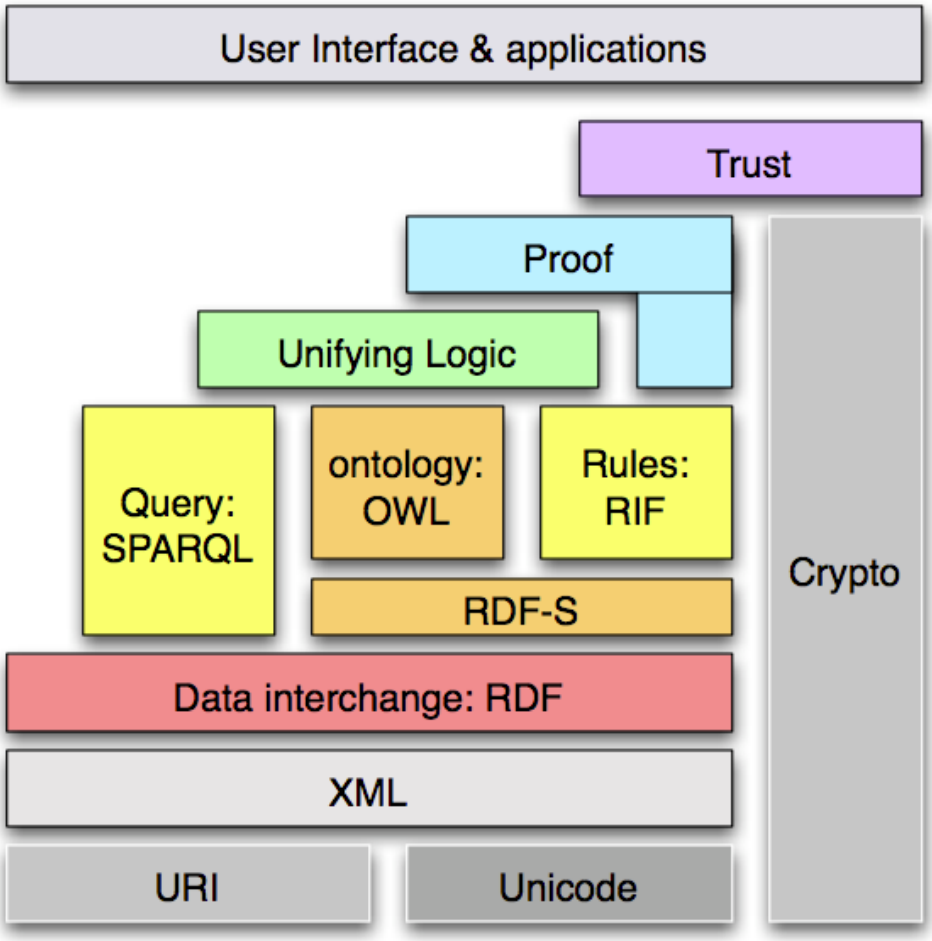
Tässä artikkelissa esitellään lyhyesti teknologiat, jotka World Wide Web Consortium on kehittänyt toteuttamaan semanttisen webin visiota. Lisäksi keskitytään tarkemmin ontologioihin ja niiden muodostamiseen.

2 Semanttisen webin perusteknologiat

Seuraavassa on kuvattu lyhyesti teknologiat, joiden avulla voidaan esittää ja tulkita dataa semanttisen webin vaatimalla tavalla. Teknologiat ovat World Wide Web Consortiumin (W3C) kehittämiä ja julkaisemia.

Teknologiat muodostavat pinon eri abstraktiotasoista. Pohjimmaisena on puhdas syntaksi ilman merkitystä ja logiikkaa. Sen päällä olevat tasot tuovat kukin uu-

den abstraktiotason ja lisää sanastoa käyttään. Semanttisen webin teknologioissa yksittäiset asiat tunnistetaan usein niiden URIn perusteella. Esimerkikisi henkilön tunnisteena voi toimia kotisivu. URI on siis semanttisen webin pohjimmainen rakennuspalikka.



Kuva 1: Semanttisen webin teknologioiden pino [BRA06]

2.1 XML ja XML Schema

Semanttisen webin perussyntaksina on XML, eXtensible Markup Language. XML muodostuu nimetyistä elementeistä ja niiden attribuuteista. Nämä ovat järjestyneet hierarkiseti. XML on laajasti käytetty ja hyvin joustava ja mukautuva kieli, mutta sillä on mahdollista esittää data tiukan formaalisti ja strukturoidusti. Se sopii erinomaisesti datan, ja erityisesti metadatan esittämiseen ja kuljettamiseen.

XML Schema tarjoaa rajoitteita ja sääntöjä XML:n esitystapaan. Se kuvaa sovelluk-

selle, minkälaista sanastoa käytetään, mikä on elementtien hierarkia dokumentissa ja paljonko ja missä järjestyksessä elementtejä voi esiityä. Näin ollen sovellus joka tuntee oikean skeeman, osaa tulkita skeeman mukaista XML-tiedostoa.

2.2 RDF ja RDF Schema

Resource Description Framework, RDF on tapa kuvata metadataa, sen ominaisuuksia ja suhteita. Näin ollen se on ikäänkuin meta-metadataa. RDF:n perus ilmaus on "RDF triple", kolmikko, joka koostuu subjektista, ominaisuudesta, ja objektista (*property, subject, object*). Tällä ilmaistaan että jollakin on jokin ominaisuus, esimerkiksi *artikkelilla* on *kirjoittaja*, joka on *Lilli*. Tämä esitetään kolmella sanalla sulkeissa, tässä tapauksessa $(kirjoittaja, artikkeli, Lilli)$, tässä järjestyksessä.

RDF Schema (RDFS), tuo työkaluja tehdä loogisia johtopäätöksiä ja yhdistellä dataa. RDFS lisää RDF:n resursseille datatyypit. Perus datatyyppejä on esimerkiksi *Resource*, *Class* ja *Property*. Kaikki asiat ovat resursseja, luokatkin (Class) ovat resursseja. Luokat ovat myäs kokoelmia mahdollisia resursseja, esimerkiksi Kirja voisi olla luokka. RDFS tarjoaa mahdollisuuden määritellä mitä ominaisuuksia kullakin luokalla on ja päätellä näiden perustella uutta tietoa.

Muita RDFS:n peruskäsitteitä ovat *domain* ja *range*. Domain kertoo mihin luokkaan ominaisuus pätee, ja range kertoo minkälaisen arvon se voi saada. Tätä valaisee seuraava esimerkki: Kirja, jonka nimi on "Lasipalatsi"ja URI vaikkapa <http://.../isbn/000651409X>. Nimelle pätee seuraavaa:

```
<http://.../isbn/000651409X> :title "Lasipalatsi"

:title
    rdf:type    rdf:Property;
    rdfs:domain :Fiction;
    rdfs:range  rdfs:Literal.
```

Tästä voidaan päätellä seuraavaa:

```
<http://.../isbn/000651409X> rdf:type :Fiction
```

Monissa tapauksissa RDFS riittää, sillä se on itsessään melko ilmaisuvoimainen: siinä löytyy datatyypit, ominaisuudet, ja luokkahierarkia. Kaikkiin tapauksiin RDF ja RDFS ei kuitenkaan riitä. Jos tarvitaan vielä kompleksimpaa logiikkaa ja toimenpiteitä elementeille, tarvitaan OWLia ja ontologioita.

2.3 Web Ontology Language OWL

OWL on kieli, jolla kuvataan ontologioita. Siinä on mahdollista määritellä loogisia päättelysääntöjä, ekvivalenssiluokkia, joukko-operaatioita, omia luokkia etc. Päättelysääntöjen avulla voidaan johtaa uutta tietoa olemassa olevasta valmiista tiedosta.

Ontologia on formaali kuvaus kokoelmasta tietotyyppejä ja niihin liittyvistä säännöistä, rajoitteista ja taksonomioista. Esimerkiksi *perhe* voisi olla ontologia. Ontologiaan kuuluu luokkia, tässä tapauksessa esimerkiksi *isä, mies, äiti, nainen, tytär, poika, lemmekki, kissa*. Luokilla voi olla ominaisuuksia, esimerkiksi *has_parent, has_mother, has_pet*. Ominaisuuksissa voi olla rajoitteita, esimerkiksi *has_mother* ei voi olla määritelty mikäli *has_parent* ei ole. Luokka voi olla toisen luokan aliluokka. Esimerkiksi *isä* voi olla *vanhemman* ja *miehen* aliluokka.

Ontologioita on mahdollista ilmaista kahdella eri standardi syntaksilla. XML syntaksi on tarkoitettu tiedon vaihtoon ja esittämiseen sovellusten välillä, ja sen avulla OWL on yhteensopiva RDF:n kanssa. Abstrakti syntaksi (Abstract Syntax) sopii paremmin ihmisen luettavaksi ja kirjoitettavaksi.

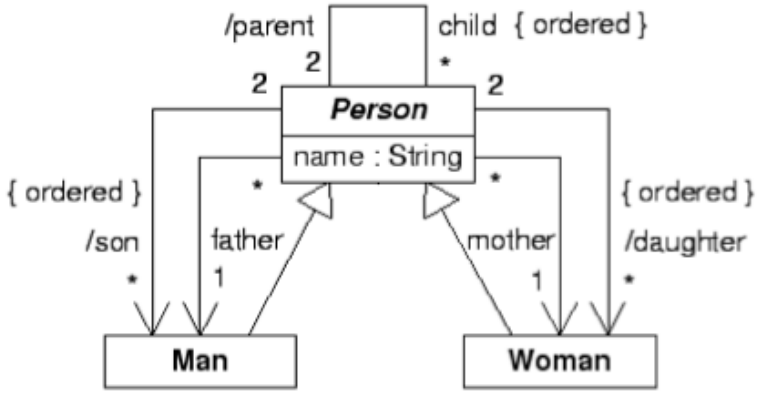
OWLista on olemassa kolme versiota. **OWL Full** laajentaa RDFS:ää ja on tämän kanssa yhteensopiva. Sen on näistä kolmesta laajin ja ilmaisuvoimaisin, mutta toisaalta hallitsemattomin. On epätodennäkäistä, että mikään sovellus pystyy kattamaan täydellisesti koko kielen tarjoaman päättelyn. **OWL DL** on rajoitetumpi versio: kaikkien päättelyiden taataan tulevan valmiiksi rajallisessa ajassa. **OWL Lite** on näistä suppein, siinä esimerkiksi relaation lukumäärä voi olla vain 0 tai 1 (cardinality). Kaikki nämä ovat ylhäältä alaspäin yhteensopivia, toisin sanoen jokainen OWL Lite ontologia on validi OWL DL ja OWL Full ontologia, ja jokainen OWL DL ontologia on validi OWL Full ontologia.

OWLin eri versiot on kuvattu tarkemmin dokumentissa [W3C04]. Valaisevia esimerkkejä päättelystä OWLissa voi lukea dokumentista [MAN03].

3 Ontologioiden rakentaminen ja oppiminen

3.1 Yleistä ontologioiden rakentamisesta

Ontologioiden rakentaminen alusta asti käsin on erittäin työlästä. Usein tämä ei ole tarpeellista, sillä voidaan löytää jo rakennettuja ontologioita, joita voidaan ainakin osittain hyödyntää ja uudelleenkäyttää oman ontologian rakentamiseen [DOA02],



Kuva 2: Pieni esimerkkiontologia esitettynä UML:llä

[YOR02]. Joka tapauksessa ontologian rakentaminen vaatii paljon suunnittelua ja yhteistyötä suunnittelukohteen alan eksperttien kanssa. Lisäksi ei ole triviaalia sovittaa uudelleen käytettäviä ontologioita toimimaan oman ontologian kanssa niin, että ei synny turhia päällekkäisyyksiä, ja että molempien ontologioiden semanttiset yhteydet ymmärretään. Tähän ontologia-mappaukseen voidaan soveltaa koneoppimistekniikoita [DOA02].

Ontologian suunnittelu- ja toteutusprosessi muistuttaa ohjelmiston tuotantoprosessia [YOR02]. **Vaatimusmäärittelyvaiheessa** (Requirements specification phase) tunnistetaan vaatimukset jotka ontologian on täytettävä. Tässä vaiheessa työskentelee tyypillisesti alan ekspertit ontologia-eksperttien kanssa läheisessä yhteistyössä. Mahdollisisa käyttötapauksia, käyttäjiä ja valmiita ontologiaresursseja tunnistetaan. Vaatimukset kirjataan yläs vaatimusmäärittelydokumentiksi. Ontologiasta tehdään epämuodollinen luonnos tunnistettujen vaatimusten perusteella.

Jalostusvaiheessa (Refinement phase) ontologiasta tehdään valmis ja muodollisesti oikeellinen. Taksonomiat ja säännöt viimeistellään ja varmistetaan että ontologiasta tulee koherentti kokonaisuus.

Evaluointivaiheessa valmista ontologiaa tarkastellaan ensimmäisessä vaiheessa tehtyjä vaatimuksia vastaan ja varmistetaan että se täyttää ne.

Ontologioiden työstämiseen eri vaiheissa on olemassa kehitystyäkaluja, esimerkiksi

OntoEdit [YOR02]. Erilaisia visuaalisia malleja ontologioiden esittämiseen on järkevää käyttää eteenkin suunnitteluvaiheessa. UML ja Entity Relationship Model sopivat tähän tarkoitukseen hyvin, vaikkakin rajoittavat sellaisenaan jonkin verran esitettävää ontologiaa [BAC01].

3.2 Ontologioiden suunnitteluperiaatteita

Ontologioita suunnitellessa ja valmisellessa on hyvä noudattaa tiettyjä suunnitteluperiaatteita [GOM03]. Ontologiasta tulisi selkeästi käydä ilmi, mitä tarkoitusta varten se on ja mikä on kunkin komponentin merkitys. Toisin sanoen ontologian tulee olla itsensä selittävä, kuten hyvän koodinkin. Ontologian määrittelyssä ei saisi olla ympäristökohtaisia riippuvuuksia, esimerkiksi Price-luokan arvoalue (range) tulisi olla mieluummin määritelty "CurrencyQuantity"kuin "float". Ontologian tulisi olla laajennettavissa ilman että olemassa olevaa rakennetta tarvitsee muotoilla uudelleen. Sanomattakin on selvää, että ontologian tulee olla sisäisesti koherentti. Ontologiasta ei saa pystyä johtamaan päätelmiä jotka ovat ristiriidassa muiden määriteltyjen totuuksien kanssa. Käytetyissä nimissä ja sanastossa tulisi olla yhtenäinen siten että semanttisesti toisiinsa liittyvät asiat ovat nimetty yhdenmukaisesti.

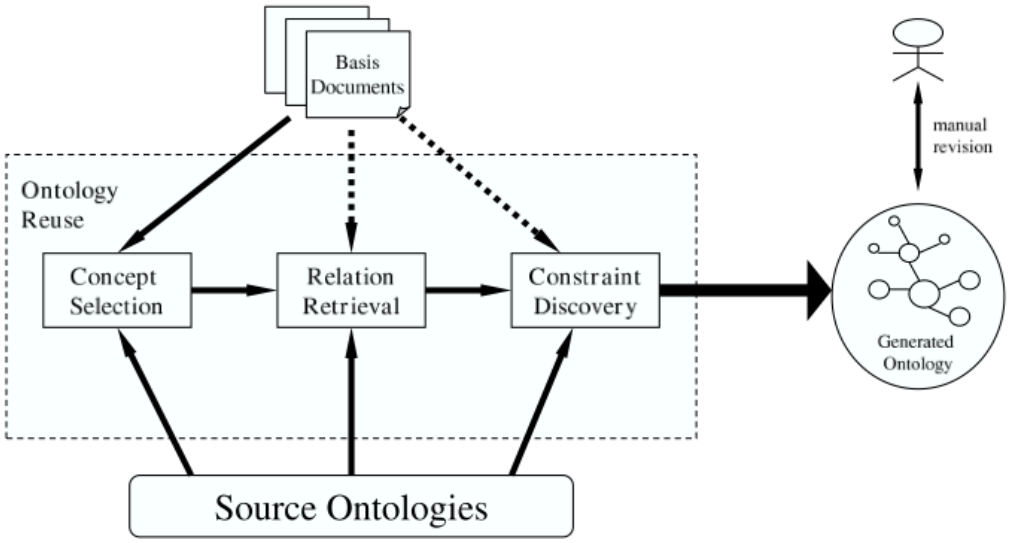
3.3 Ontologian oppiminen

Ontologioiden käyttäminen on vielä lapsen kengissään. Vielä jokin aika sitten valmiiksi tehdyt ontologiat olivat hyvin harvinaisia. Nykyäänkin ontologiat ovat hajallaan, ja niiden tunnistaminen sopiviksi omaa käyttötarkoitusta varten voi olla haasteellista, jopa niin haasteellista, että on kyseenalaista, onko sittenkään helpomaa koittaa automatisoida ontologioiden oppimista kuin rakentaa oma ontologia alusta asti [DING07].

Valmiit ontologiat tyypillisesti kattavat jonkin tietyn aihealueen. Esimerkiksi lääketieteestä, matematiikasta ja organsiaatio- ja yritysrakenteista on jo kehitetty ontologioita [GOM03, kalvot 64-70]. Tässä muodossa, tiettyyn aihealueeseen rajattuna modulina, ontologiat ovatkin parhaiten käytettävissä uudelleen [DING07]. Ongelmana ontologioiden uudelleen käytettävyydessä on se, että ne sijaitsevat hajanaisina eri paikoissa, ja jotkut voivat kattaa saman aihealueen hieman eri terminologialla, ja jotkut voivat olla osittain päällekkäisiä. Tällaisista ontologioista pitäisi pystyä tunnistamaan toisiaan vastaavat komponentit, ja integroimaan erilliset ontologiat päällekkäisyyksien välttämiseksi. Tämä, sekä potentiaalisten lähdeontologioi-

den tunnistaminen vaativat paljon ihmistyötä; siis ontologioiden generoiminen ei ole vielä lähelläkään täysin automaattista.

Useissa lähestymistavoissa ([DING07], [MAE01]) ontologioiden semiautomaattiseen generoimiseen kuuluu valmiiden ontologioiden integroimisen ja käyttämisen lisäksi luonnollisella kielellä kirjoitettujen lähdedokumenttien, kuten HTML-dokumenttien käyttäminen. Ding et al. [DING07] kuvaavat artikkelissaan kehittämänsä prosessin ontologian automaattiseksi generoimiseksi. Se koostuu kolmesta peräkkäisestä vaiheesta: konseptien valinta (concept selection), yhteyksien haku (relation retrieval) ja rajoitusten löytäminen (constraint discovery). Kuva 3 havainnollistaa prosessia.



Kuva 3: Ontologian generoiminen käyttäen uudelleenkäytettäviä ontologioita [DING07]

Merkittävä vaihe ennen varsinaista ontologian generointia on kuitenkin lähdeontologioiden hiominen ja integroiminen. Eri ontologioissa olevat semanttisesti samansisältöiset konseptit tulisi yhdistää yhdeksi konseptiksi, jotta ei konseptin valintavaiheessa valittaisi kahta samaa tarkoittavaa konseptia. Konseptien mappaaminen eri ontologioiden välillä ei ole triviaali tehtävä, ja siinä voi hyödyntää koneoppimimenetelmiä [DOA02]. Tämä on kuitenkin kokonaan oma ongelmansa, johon emme paneudu enempää tässä yhteydessä.

Ensimmäisessä vaiheessa, nimittäin konseptien valinnassa, tunnistetaan luonnollisella kielellä olevasta lähdemateriaalista avainsanoja ja niitä vastaavia konsepteja lähdeontologioista. Esimerkiksi jos lähdedokumentissa on lause "Afganistanin pää-

kaupunki on Kabul.", valitaan lähdeontologiasta konsepti Pääkaupunki. Tämän voi saada aikaiseksi myös sitä kautta, että lähdemateriaalista tunnistetaan sana Kabul, ja jos se esiintyy lähdeontologiassa konseptin Pääkaupunki instanssina, valitaan konsepti Pääkaupunki.

Yhteyksien hakuvaiheessa muodostetaan sopivat yhteydet tunnistettujen konseptien välille. Tämä on luontevaa tehdä tutkimalla yhteyksiä näiden konseptien välillä lädeontologiassa. Yksi mahdollisuus olisi tutkia kaikki mahdolliset yhteydet joidenkin kahden konseptin välillä, mutta tämä olisi aikaa vievää. Sen sijaan voidaan säätää kynnysarvo, jota lähempänä olevat yhteydet valitaan otettavaksi mukaan uuteen ontologiaan. Tämä on toteutettu mukautetulla Dijkstran algoritmilla, jota käytetään toistuvasti etsimään lyhin reitti, toiseksi lyhin reitti, jne. Kynnysarvo kannattaa säätää tarpeeksi korkealle, sillä on helpompaa karsia turhia yhteyksia ja konsepteja pois manuaalisessa tarkistusvaiheessa, kuin kehittää kokonaan puuttuvia yhteyksiä tyhjästä.

Rajoitteiden löytäminen on monimutkainen ongelma johtuen erilaisten rajoitteiden suuresta määrästä. Tästä syystä Ding et al. ovat keskittyneet vain yhden tyyppisten rajoitteiden automaattiseen generointiin, nimittäin lukumäärä rajoitteiden (cardianlity). Valitaan kaksi konseptia joiden välillä on yhteys ja tutkitaan niiden esiintymistä eri dokumenteissa. Jos dokumentissa D_1 esiintyy konsepti A, jolla on instanssi a_1, ja konsepti B, jolla ei esiinny instansseja siinä dokumentissa, voidaan päätellä, että konseptin A minimimäärä yhteydellä AB on 0.

Näiden vaiheiden jälkeen ontologian muodostaminen on helppoa, sillä ontologian perus rakennuspalikat ovat juuri konseptit (luokat), yhteydet, ja rajoitteet. Syntynyttä ontologiaa täytyy kuitenkin vielä hioa ja korjata ihmisvoimin. Liiat yhteydet pitää poistaa, nimetä sopimattomasti ja epäyhtenäisesti nimetyt asiat paremmin, korjata virheitä olemassa olevissa komponenteissa, ja lisätä puuttuvia komponentteja.

4 Yhteenveto ja johtopäätökset

Semanttisen webin idea on saada webissä oleva data sellaiseen muotoon, että koneet voivat käsitellä dataa sen semanttisen merkityksen mukaisesti ja muodostaa yhteyksiä konseptien välillä samoin kuin ihminen tekee. Tämä helpottaisi ja tehostaisi toteutuessaan webin käyttää ja tiedon hakua. Lukuisia teknologioita on kehitetty toteuttamaan tätä visiota. Kekseisessä osassa ovat ontologiat, jotka ovat formaali kuvaus tiedosta, sen hierarkiasta, semanttisista yhteyksistä ja rajoitteista.

Ontologioiden muodostaminen on haasteellinen ja paljon suunnittelua ja työtä vaativa tehtävä. Sitä tehostamaan on kehitetty muutamia semiautomaattisia menelemiä, joissa opitaan ontologioita käyttäen hyväksi olemassa olevia ontologioita. Tässäkin on haasteena olemassa olevien ontologioiden hajanaisuus ja vielä toistaiseksi suhteellisen pieni määrä.

Olemassa olevat ontologiat kuvaavat tyypillisesti jonkin tietyn rajatun aihealueen. Tällaisena ne sopivat parhaiten esimerkiksi tietyn organisaation kuvaamiseen sen omilla sivuilla ja omissa sovelluksissa. Jotta ontologioiden voima ulottuisi sivustorajojen, eli aihealueiden ulkopuolelle, täytyisi olla parempia tekniikoita ontologioiden integroimiseen. Toisin sanoen eri ontologioista pitäisi pystyä tunnistamaan semanttisesti samat komponentit. Tähän on kehitetty joitakin koneoppimista käyttäviä tekniikoita [DOA02], mutta kuten kuka tahansa webiä selaava voi huomata, semanttisen webin laajamittaiseen käyttöön on vielä matkaa.

5 Lähteet

[BAC01] Kenneth Baclawski, et al.: Extending UML to Support Ontology Engineering for the Semantic Web. Springer-Verlag Berlin Heidelberg 2001

[BRA06] Steve Bratt 2006 Emerging Web Technologies to Watch s. 19 (http://www.w3.org/2006/Talks/1023-sb-W3CTechSemWeb/Overview.html) (29.10.2010)

[DING07] Yihong Ding, Deryle Lonsdale, David W. Embley, Martin Hepp, and Li Xu: Generating Ontologies via Language Components and Ontology Reuse 2007

[DOA02] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy 2002: Learning to Map between Ontologies on the Semantic Web

[GOM03] Asuncion Gomez-Perez Mariano Fernandez-Lopez Oscar Corcho: Ontological Engineering and the Semantic Web tutorial joka perustuu samojen tekijäiden kirjaan Ontological Engineering. Springer Verlag 2003

[MAE01] Alexander Maedche and Steffen Staab: Ontology Learning for the Semantic Web. IEEE Intelligent systems 1094-7167/01

[MAN03] 2003 http://www.cs.man.ac.uk/ horrocks/ISWC2003/Tutorial/examples.pdf (1.11.2010)

[W3C04] W3C Recommendation 2004 OWL Web Ontology Language Overview (1.11.2010)

[YOR02] York Sure, Michael Erdmann, Juergen Angele, Steffen Staab, Rudi Studer, and Dirk Wenke 2002: OntoEdit: Collaborative Ontology Development for the Semantic Web. Springer-Verlag Berlin Heidelberg 2002.