hyväksymispäivä arvosana

arvostelija

Ontologiaperustainen RDF-annotaatio

Mikko-Ville Apiola

Helsinki 30.12.2004

Pro gradu-tutkielma

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Mikko-Ville Apiola

Työn nimi — Arbetets titel — Title

Ontologiaperustainen RDF-annotaatio

Oppiaine — Läroämne — Subject

Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu-tutkielma	30.12.2004	60 sivua + 10 liitesivua

Tiivistelmä — Referat — Abstract

Internetissä ja erilaisissa tietojärjestelmissä on valtava määrä tietoa, jonka löytäminen on kuitenkin useissa tapauksissa vaikeaa tai mahdotonta.

Semanttisessa webissä ideana on "rikastaa" koneen käsittelemä tieto metatiedolla, jolloin sekä ihmisen ja koneen että koneiden keskinäinen tiedon haku ja kommunikointi voi tehostua. Metatieto luokittelee käsitellyn tiedon tarkoituksenmukaisista näkökulmista sekä määrittelee kyseisen tiedon semanttiset suhteet muuhun maailmaan nähden.

Semanttisen metatiedon tuottamista eli annotaatiota varten on olemassa useita eri tarkoituksiin soveltuvia työkaluja. Automaattiset työkalut perustuvat metatiedon päättelemiseen tunnistamalla luonnollista kieltä sekä louhimalla tietoa. Manuaaliset työkalut tuottavat annotaatioita käyttäjän avustuksella.

Useat nykyiset annotaatioeditorit vaativat käyttäjiltään sekä teknistä että sisällöllistä asiantuntemusta, jotta niillä luodut annotaatiot olisivat järkeviä. Tutkimuksen osana on laadittu editori, jonka kohderyhmäksi on asetettu "tavalliset käyttäjät". Tavoitteena oli, että editori opastaa käyttäjää järkevien ja riittävän laajojen annotaatioiden tekemiseen. Editoria testattiin keltaiset sivut -tyyppisten palveluilmoitusten lisäämistä ja annotointia varten. Jotta editori pystyisi opastamaan käyttäjää, on sillä oltava käytössään eri ontologioiden suhteita kuvaava rajoiteontologia sekä tapauksiin liittyviä muiden käyttäjien tekemiä annotaatioita. Editorin käytettävyys voi muodostua ongelmaksi esim. käytettäessä oleellisesti testiympäristössä käytettyjä suurempia ontologioita.

Tässä tutkielmassa kartoitetaan ja vertaillaan olemassaolevia editoreita ja ratkaisuja, kuvataan tutkimusta varten laadittu editori, pohditaan näiden välisiä suhteita sekä esitetään arvioita suunnasta, johon editorien tulisi kehittyä.

ACM Computing Classification System (CCS): D.2 [Software Engineering],
H.3.2 [Information Storage], H.3.3 [Information Search and Retrieval],
I.2 [Artificial Intelligence]

Avainsanat — Nyckelord — Keywords

semantic web, annotaatio

Säilytyspaikka — Förvaringsställe — Where deposited

Kumpulan tiedekirjasto, sarjanumero C-2005-

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Internetissä ja erilaisissa tietojärjestelmissä on valtava määrä tietoa, jonka löytäminen on kuitenkin useissa tapauksissa vaikeaa tai mahdotonta.

Semanttisessa webissä [BLHL01] [1] ideana on "rikastaa" koneen käsittelemä tieto metatiedolla, jolloin koneet kykenevät paremmin ymmärtämään käsittelemäänsä tietoa. Näin sekä ihmisen ja koneen että koneiden keskinäinen tiedon haku ja kommunikointi voi tehostua. Metatieto luokittelee käsitellyn tiedon tarkoituksenmukaisista näkökulmista sekä määrittelee kyseisen tiedon semanttiset suhteet muuhun maailmaan nähden.

Semanttisen metatiedon tuottamista eli annotaatiota varten on olemassa useita erilaisiin tarkoituksiin soveltuvia työkaluja. Automaattiset työkalut perustuvat metatiedon päättelemiseen tunnistamalla luonnollista kieltä sekä louhimalla tietoa. Manuaaliset työkalut tuottavat annotaatioita käyttäjän avustuksella.

Annotaatioiden tuottaminen vaatii usein käyttäjiltä sekä teknistä että sisällöllistä asiantuntemusta. Yksinkertaisimmillaan annotaatioita voidaan tehdä lisäämällä annotaatioita suoraan annotoitaviin dokumentteihin tekstieditorilla, jolloin syntaksivirheiden riski kasvaa suureksi. Annotaatioeditorit huolehtivat siitä, ettei syntaktisesti virheellisiä annotaatioita ole mahdollista tehdä. Kuitenkin editoreilla on mahdollista tehdä semantiikan kannalta mielivaltaisia annotaatioita. Tämän takia ei ole mitään takeita luotujen annotaatioiden oikeellisuudesta tai riittävästä laajuudesta. Käyttäjätutkimuksissa [EMSS00] on käyttäjillä ilmennyt annotaatioissa käytettyyn terminologiaan liittyviä ymmärrysvaikeuksia ja ongelmia semanttisten suhteiden ymmärtämisessä.

Annotaatioiden käyttötarkoitukset ja annotaation suorittavien henkilöiden osaaminen vaihtelee paljon. Tehtäessä annotaatioita vaikkapa sairaalan potilaskertomuksia sisältäviin tiedostoihin, on järkevää ja usein välttämätöntä että annotaatioiden laatija on sekä erikoisalansa asiantuntija että tuntee tarkkaan annotaatioiden luomiseen liittyvän tekniikan. On kuitenkin monia tilanteita, joissa annotaatioita olisi mahdollista luoda tekniikkaa ja semantiikkaa tuntemattomien henkilöiden avustuksella. Esimerkiksi erilaisia palveluita tarjoavat yritykset voisivat annotoida palvelunsa sopivien luokitusten mukaisesti, mutta kyseisen kaltaisten luokitusten tekeminen on tekniikkaa tuntemattomille käyttäjille vaikeaa, joka johtaa helposti virheellisiin annotaatioihin.

[1] http://www.w3.org/2001/sw/

Tämän tutkielman tarkoituksena on ollut selvittää, olisiko mahdollista laatia työkalu jonka kohderyhmänä olisivat tavalliset käyttäjät siten, että editori opastaisi käyttäjää järkevien annotointien tekemiseen sekä pakottaisi käyttäjät tekemään riittävän laajoja annotaatioita.

Laadittavalle editorille on asetettu seuraavat vaatimukset:

- Täydentäminen ja rajoittaminen. Editori kykenee täydentämään käyttäjän tekemää annotaatiota sekä rajoittamaan puutteellisten virheellisten tekemistä. Virheellisellä annotaatiolla tarkoitetaan esimerkiksi palveluilmoitukseen liitettyä annotaatiota, jossa palvelun sijainniksi luokitellaan sisämaassa sijaitseva kohde, mutta kuitenkin luokitellaan tarjottu palvelu vesikuljetuspalveluksi.
- Suosittelu. Editori kykenee suosittelemaan muita mahdollisia annotaatioita perustuen muiden käyttäjien laatimiin annotaatioihin.
- Pakottaminen. Editori osaa pakottaa käyttäjää antamaan kaikki tarpeelliset tiedot.
- Geneerisyys. Editori on mahdollista sovittaa erilaisiin käyttöympäristöihin.

Laadittu editori täyttää asetetut vaatimukset tietyin rajoituksin. Editoria on testattu keltaiset sivut -tyyppisten palveluilmoitusten lisäämistä ja annotointia varten. Vaatimus annotaatioiden täydentämisestä toteutuu, jos editorilla on käytössään eri ontologioiden suhteita kuvaava rajoiteontologia, ja vaatimus suosittelusta täyttyy, jos annotaatiotietokannassa on tapaukseen liittyviä muiden käyttäjien tekemiä annotaatioita. Editorin käytettävyys voi muodostua ongelmaksi jos käytetään oleellisesti suurempia ontologioita, kuin testatussa ympäristössä käytettiin.

Tässä tutkielmassa kartoitetaan ja vertaillaan olemassaolevia editoreita ja ratkaisuja, kuvataan tutkimusta varten laadittu editori, pohditaan näiden välisiä suhteita sekä esitetään arvioita suunnasta, johon editorien tulisi kehittyä.

Luvussa 2 käydään lyhyesti läpi semanttisen webin perusidea, ontologioiden merkitys semanttisessa webissä sekä esitellään lyhyesti yleisimmät semanttisen webin standardit. Lukija, jolle nämä asiat ovat entuudestaan tuttuja, voi siirtyä suoraan lukuun 3, jossa tehdään tutustumiskierros annotaatio-ongelmaan sekä esitellään muita maailmalla tehtyjä annotaatiotyökaluja.

Luvussa 4 esitellään semanttisen webin sovellusten laadintaan soveltuvia tekniikoita, perustellaan editorin laatimista varten valitut tekniikat ja kuvataan käytetty testiaineisto.

Luvussa 5 esitellään laaditun annotaatioeditorin toiminta ja tekninen rakenne. Tulosten tarkastelu on luvussa 6. Yhteenveto tutkielmasta on luvussa 7.

2 Semanttinen web

Tällä hetkellä WWW:n tarjoamat sisällöt ovat vain ihmisen luettavissa. Tietokoneet eivät ymmärrä ihmisille välittämiensä sivujen sisältöjä, mikä asettaa huomattavia rajoituksia erilaisten Internetissä toimivien sovellusohjelmien toiminnalle. Semanttinen web on visio, jossa tietokoneet oppivat ymmärtämään enemmän käsittelemistään sisällöistä, jolloin sekä tiedon haku että erilaisten sovellusohjelmien yhteistoiminta paranee [BLHL01].

Toteutuakseen semanttinen web tarvitsee standardeja rakenteellisen tiedon suhteiden esittämiseen ja metatiedon kuvaamiseen. Suuria haasteita ovat yhteisymmärryksen saavuttaminen sovellusaluekohtaisten ontologioiden suhteen sekä semanttisen metatiedon tuottaminen.

Yleisin WWW:ssä tällä hetkellä välitettävä dokumentin tyyppi on rakenteiseen SGML-formaattiin [2] perustuva HTML-kieli [3], jota tukee laaja joukko Internet-selainohjelmia ja erilaisia editoreita. HTML-kielen avulla voidaan vaikuttaa dokumentin ulkoasuun jakamalla sisältö otsikoihin, kappaleisiin, listoihin, taulukoihin ja linkkeihin. Muita WWW:ssä liikkuvia dokumentteja ovat esimerkiksi erilaiset kuvaa, ääntä ja videota sisältävät tiedostot. Nykyisessä WWW:ssä tietokone kykenee kyllä jäsentämään käsittelemänsä dokumentit halutun ulkoasun mukaisesti, muttei ymmärrä käsittelemistään sisällöistä paljoakaan [BLHL01].

2.1 Tiedon haku

Jos ajatellaan nykyisiä WWW:ssä toimivia hakukoneita, voi halutun tiedon löytäminen niiden avulla olla vaikeaa. Annettaessa hakukoneille tiettyjä hakusanoja on hakutulosjoukko usein varsin suuri, mutta on mahdollista, että vain harva, jos yksikään, löydetyistä dokumenteista antaa haluttuun kysymykseen vastauksen.

Tiedon haku nykyisessä WWW:ssä perustuu usein käyttäjän antamien avainsanojen vertailuun WWW-dokumenteissa ilmenevien merkkijonojen kanssa. Tiedon louhinnassa (data mining) yritetään sisältöjä tulkita tunnistamalla dokumenteista eri-

[2] http://www.w3.org/MarkUp/SGML
[3] http://www.w3.org/MarkUp

laisia rakenteita tai tarkkailemalla verkossa toimijoiden käyttäytymistä [BLHL01]. Tiedon uuttamisella (information extraction) tarkoitetaan dokumenttien koneellista käsittelyä siten, että niiden merkitysrakenne saadaan selville [Hyv01]. Tulkitseminen kyseisillä menetelmillä on vaikeaa, koska jo tekstien rakenteiden tunnistaminen on vaikeaa luonnollisen kielen monimutkaisuuden vuoksi, merkityksistä puhumattakaan [SH01]. Lisäksi suuri osa sisällöistä ei ole tekstimuotoista, vaan esimerkiksi ääntä tai kuvaa. WWW- sivujen sisältöjä ei voida tulkita ainoastaan sivuilla olevan tiedon perusteella, vaan tulkintaan kuuluu oleellisena osana ulkopuolinen taustatieto ja "elämänkokemus". Tällaisen tiedon opettaminen koneelle on osoittautunut tekoälytutkimuksessa erittäin vaikeaksi [Hyv01].

On todennäköistä, että kaikkien WWW:n dokumenttien joukossa on tietoa, joka vastaa käyttäjän kysymykseen. Kuitenkaan nykyiset tiedon louhintaan, luonnollisen kielen tulkitsemiseen tai tiedon uuttamiseen perustuvat menetelmät eivät useissa tapauksissa pääse lähellekkään löytääkseen sitä. Lisäksi, on olemassa WWW:n kautta toimivia tietokantoja ja ohjelmia, jotka osaisivat vastata käyttäjän kyselyihin, mutta ne jäävät usein nykyisillä hakumenetelmillä saavuttamattomiin [SH01].

Nykyisessä Internetissä toimii jo nyt lukuisa joukko erilaisia palveluita, jotka on tarkoitettu tietokoneiden käytettäviksi. Esimerkkejä tällaisista ovat erilaiset yritysten B2B-sovellukset, hakukoneet, agenttisovellukset ja sähköisen kaupan sovellukset. Näiden palvelujen yhteistoiminta on kuitenkin nykyisellään hankalaa koska palvelut eivät löydä toisiaan eivätkä osaa kommunikoida keskenään.

2.2 Merkitysten Internet

Koska nykyisen WWW:n koneellinen tulkinta luonnollisen kielen monimutkaisuuden sekä erilaisten tiedon tyyppien vuoksi on vaikeaa, pyrkii semanttinen web ratkaisemaan ongelman siten, että "ihminen tulee konetta vastaan" tallettamalla tiedot alunperinkin muotoon, joka on koneelle helpompi tulkita. Tiedon tallettamista uuteen muotoon sekä vanhan tiedon "semanttista rikastamista", eli annotointia voidaan tehdä sekä manuaalisesti, että jossain määrin myös automaattisesti.

Jotta semanttinen webi toimisi, täytyy tietokoneiden saavutettavissa olla hyvin järjestettyä tietoa ja päättelysääntöjä, joiden perusteella koneet voivat suorittaa automaattista päättelyä. Semanttisen webin haaste on tarjota yhteinen kieli tarjolla olevan tiedon ja sääntöjen kuvaamiseen, sekä välineet joiden avulla minkä tahansa tietämysjärjestelmän säännöt voidaan tuoda semanttiseen webiin [BLHL01]. Toteu-

tuakseen semanttinen webi tarvitsee joukon ilmaisukeinoja sekä teknisiä ratkaisuja. Semanttinen webi voidaan jakaa seuraaviin tasoihin:

- Luottamus (trust)
- Päättely
- Ontologiat
- Metakuvaukset
- Rakenteen kuvaus
- Internet

Teknisenä perustana toimivat Internet, WWW ja rakenteiseen XML-kieleen (eXtensible Markup Language) [4] perustuvat teknologiat [Hyv01, BLHL01]. Nämä teknologiat ovat jo olemassa. XML on metakieli, jonka avulla voidaan määritellä sovelluskohtaisia merkkauskieliä (markup language) tietojen esittämiseksi. XML-dokumentista voidaan XSL-transformaatioiden (XML Stylesheet language) [XSL99] [5] avulla tuottaa erilaisia ulkoasultaan ja formaatiltaan toisistaan poikkeavia ilmentymiä kyseisestä dokumentista, esimerkiksi HTML-sivuja tai PDF-dokumentteja (Portable Document Format)[6]. XML antaa käyttäjille mahdollisuuden vapaasti määritellä dokumenttiensa rakenteen, muttei kerro mitään siitä, mitä rakenteella tarkoitetaan [BLHL01].

Metakuvausten tasolla alla olevan tason rakenteisiin dokumentteihin lisätään semanttinen merkitys. RDF-kieli (Resource Description Framework)[7] on jo vakiintunut standardi semantiikan kuvaamiseen. RDF:ssä voidaan esimerkiksi ilmaista, että tietyllä asialla (esim. ihminen) on ominaisuuksia, (esim. omistaa, on tekijänä) tietyn toisen asian (esim. toinen ihminen, web-sivu) suhteen [BLHL01]. Oleellista RDF-kielessä XML-kieleen verrattuna on huomata, että RDF:n tietomallina ei ole XML-kielen tavoin sarjallistettu (serialized) syntaktinen puu, vaan tietomallin voi ajatella graafisesti muodostavan suunnatun verkon [LS].

Mahdollisuus kuvata vapaasti semanttisia suhteita ei vielä riitä. Kahdessa eri järjestelmässä voidaan suhteita kuvata kussakin omilla tavoillaan, jolloin kuvaukset

[4] http://www.w3.org/XML/
[5] http://www.w3.org/Style/XSL/
[6] http://www.adobe.com/products/acrobat/adobepdf.html
[7] http://www.w3.org/RDF/

eivät ole yhteensopivia keskenään. Jotta saavutettaisiin yhteisymmärrys erilaisten semanttisten kuvausten kesken, tarvitaan yhteisiä asioiden välisiä suhteita kuvaavia sopimuksia, eli ontologioita. Ontologia voidaan määritellä formaaleiksi kuvauksiksi asioista ja niiden välisistä suhteista [BLHL01]. Käytännössä ontologiat ovat eri sovellusalojen terminologisia käsitehierarkioita, joissa määritellään alalla käytetyt termit ja käsitteet sekä näiden välisiä suhteita [Hyv01]. Ontologioiden kehittämiseen ja standardointiin liittyy monia ongelmia. Eri järjestelmien yhteistyö vaatii standardoituja ontologioita, joiden kehittäminen on hyvin vaikeaa.

Älykkäiden Internetissä toimivien ohjelmien pitäisi lisäksi vielä osata tehdä ontologioiden avulla semanttisesti rikkaan tiedon perusteella järkeviä päätelmiä. Luottamus Semanttisessa Webissä nousee oleelliseksi kysymykseksi. Nykyisessä WWW:ssä on valtavasti tietoa, jonka luotettavuus vaihtelee laidasta laitaan. Kuten nykyisessäkin WWW:ssä, jonkun tunnetun tahon tekemät semanttiset luokitukset ovat tuntematonta tahoa luotettavampia.

2.3 RDF-Tietomalli

RDF (Resource Description Framework) on standardiksi muodostumassa oleva metatietokuvauskieli semanttisessa webissä. RDF-kieltä voidaan käyttää resurssien kuvaamiseen sekä esimerkiksi uusien kuvauskielien, kuten sanaston kuvauskielen RDF(S) tai tätä edistyneempien ontologiakielien, kuten OWL, kuvaamiseen.

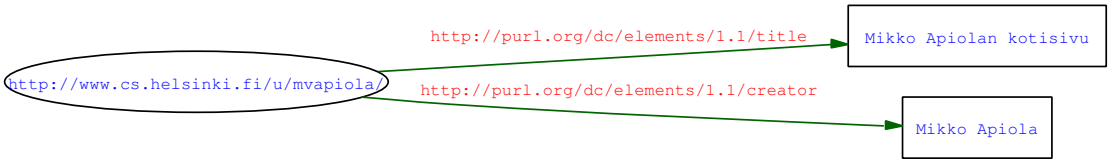
RDF-tietomalli koostuu RDF-lauseiksi (statement) kutsutuista subjekti-predikaatti-objekti- kolmikoista (triplet), joiden joukon voidaan ajatella muodostavan nimetyistä kaarista koostuvan suunnatun verkon. Verkossa resurssit (eli rdf:Resource-tyyppiset solmut) identifioidaan URI (Uniform Resource Indicator)-tunnisteiden avulla. URI:na voidaan käyttää esimerkiksi URL osoitetta, jos halutaan kuvata WWW-sivua, mutta mikä tahansa asia, jolle voidaan osoittaa yksilöllinen URI, voi olla resurssi.

Kuvassa 1 on esitetty esimerkkitapaus RDF-kuvauksesta. Kyseisessä resurssin kuvauksessa (eli annotaatiossa) määritellään Internet-sivun http://www.cs.helsinki.fi/u/mvapiola/ otsikko ja tekijä Dublin Core [WKLW98] metatietopredikaattien avulla. Dublin Core sisältää 15 kategoriaa, joiden avulla on tarkoitus voida määritellä yleisiä ominaisuuksia resursseille. Dublin Core- määrityksiä voidaan liittää HTML-, XML-, ja RDF- dokumentteihin. Esimerkin rivillä 5 määritellään kuvattavan resurssin otsikko ja rivillä 6 määritellään resurssin tekijä.

```
1: <?xml version="1.0"?>
2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:   xmlns:dc="http://purl.org/dc/elements/1.1/">
4:   <rdf:Description rdf:about="http://www.cs.helsinki.fi/u/mvapiola/">
5:     <dc:title>Mikko Apiolan kotisivu</dc:title>
6:     <dc:creator>Mikko Apiola</dc:creator>
7:   </rdf:Description>
8: </rdf:RDF>
```

Kuva 1: RDF kuvaus

Kuvassa 2 on visualisoitu sama annotaatio verkkona. Kuvasta voidaan selvästi havaita ne kaksi subjekti-predikaatti-objekti -kolmikkoa, joista verkko muodostuu. Kuva on tuotettu W3C:n RDF Validation Service-ohjelmalla [8]. Esimerkistä kannattaa huomioida, että tehtäessä järkeviä annotaatioita olisi syytä viitata tekijän kohdalla henkilöstä luotuun RDF-resurssiin eikä ainoastaan merkkijonoon “Mikko Apiola”.



Kuva 2: RDF verkko

Jotta RDF-muotoiset annotaatiot voisivat olla yhdenmukaisia keskenään, tarvitaan jonkinlaisia yhteisiä käsitteistöjä tiedon kuvaamiseen. RDF Vocabulary Description Language eli RDF Schema (RDFS) on tarkoitettu alimman tason ontologiakieleksi, joka käsittää vain välttämättömät primitiivit luokkien ja niiden ominaisuuksien kuvaamista varten. RDFS:llä voidaan mm. kuvata luokkien välisiä suhteita (rdfs:subClassOf), kuvata luokkaan kuuluvat ominaisuudet (rdfs:domain) sekä määritellä ominaisuuksien sallitut arvoalueet (rdfs:range). Vaikka RDFS on varsinaisesti tarkoitettu sanaston määrityskieleksi, sitä voidaan käyttää ja käytetään ontologioiden kuvaamiseen. Esimerkissä 3 on kuvattu yksinkertainen RDFS-ontologia.

Esimerkissä on kuvattu ontologia, joka sisältää Person-luokan, ja sen aliluokat Man

[8] http://www.w3.org/RDF/Validator

ja Woman. Person-luokalla on ominaisuudet “name”, joka saa arvokseen merkkijonon, sekä eri Person-instanssien välisiä suhteita kuvaava “marriedWith”-ominaisuus. Riveillä 33-41 on kyseisen ontologian mukainen annotaatio, jossa määritellään instanssit henkilöille “Susan Smith” ja “John Smith”, sekä että kyseisiin instansseihin liittyy keskinäinen suhde “marriedWith”.

```
1: <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3: <rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22/rdf-syntax-ns"
4:  xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303">
5:
6: <rdf:Description ID="Person">
7: <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Class"/>
8: <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource"/>
9: </rdf:Description>
10:
11: <rdf:Description ID="Man">
12: <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Class"/>
13: <rdfs:subClassOf rdf:resource="#Person"/>
14: </rdf:Description>
15:
16: <rdf:Description ID="Woman">
17: <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Class"/>
18: <rdfs:subClassOf rdf:resource="#Person"/>
19: </rdf:Description>
20:
21: <rdf:Description ID="name">
22: <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Property"/>
23: <rdfs:domain rdf:resource="#Person"/>
24: <rdfs:range rdf:resource="http://www.w3.org/TR/xmlschema-2/#string"/>
25: </rdf:Description>
26:
27: <rdf:Description rdf:ID="marriedWith">
28: <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Property"/>
29: <rdfs:domain rdf:resource="#Person"/>
30: <rdfs:range rdf:resource="#Person"/>
31: </rdf:Description>
32:
33: <Person rdf:ID="s_smith_x">
34:   <name>Susan Smith</name>
35:   <marriedWith rdf:resource="j_smith_x"/>
36: </Person>
37:
38: <Person rdf:ID="j_smith_x">
39:   <name>John Smith</name>
40:   <marriedWith rdf:resource="s_smith_x"/>
41: </Person>
42:
43: </rdf:RDF>
```

Kuva 3: RDFS esimerkki

OWL Web Ontology Language [BvHH+04] on RFDS-kieleen verrattuna huomattavasti ilmaisuvoimaisempi kieli. Kielen kehittäjien pyrkimyksenä on ollut mm. parannettu geneerisyys, laajennettavuus ja käytettävyys [BvHH+04]. OWL-kieli laa-

jentaa RDFS-kielen luokkamäärittelyä tarjoamalla mahdollisuuden määrittää luokkakohtaisia ominaisuusrajoitteita, useista muista luokista koostuvia yhdisteluokkia, yhden tai useamman luokan leikkauksesta koostuvia luokkia tai jonkin toisen luokan komplementtiluokkia.

Ominaisuuksien kuvausta on myös laajennettu siten, että se pohjautuu olio- ja tietotyyppiarvoisten ominaisuuksien erotteluun (owl:ObjectProperty) ja (owl:DatatypeProperty)-määrittelyjen avulla. OWL-kielen kehityksessä on myös otettu huomioon ontologioiden evoluutioon ja yhdistämiseen liittyviä tekijöitä [BvHH+04].

3 Katsaus annotaatioeditoreihin

Tässä luvussa käsitellään muutamien yleisimpien olemassaolevien annotaatioeditorien toimintaperiaatteet, sekä arvioidaan niiden käyttökelpoisuutta. Semanttisen metatiedon olemassaolo on perusedellytys semanttisen webin sovellusten toiminnalle. Metatietoa on eri järjestelmissä pyritty luomaan sekä automaattisesti että manuaalisesti.

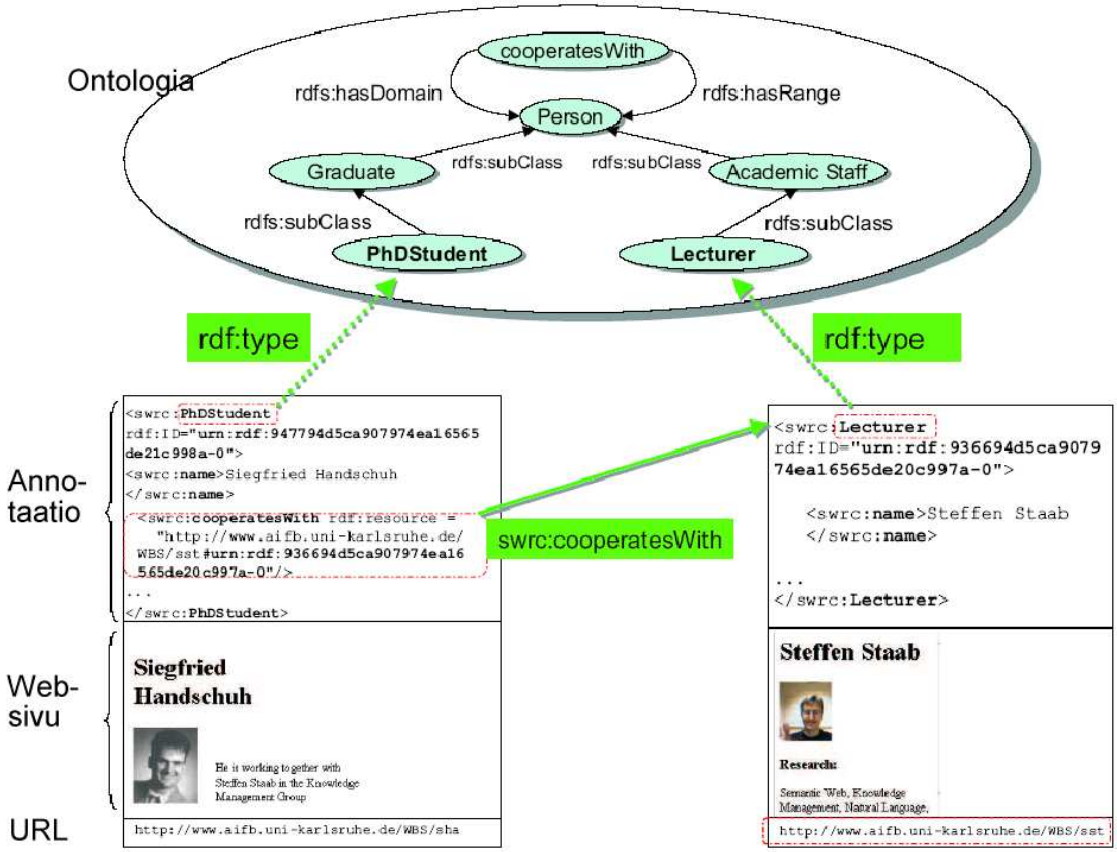
Automaattisella annotaatiolla tarkoitetaan prosessia, jossa jostakin olemassaolevasta tietomassasta yritetään mekaanisesti tietokoneen avulla tunnistaa dokumenttien rakenteita ja sen perusteella luoda annotaatioita.

Manuaalisen annotaation perustapauksessa ihminen tekee annotaatioita “käsin”, eli luokittelee vaikkapa verkkosivuja käyttäen apuna tarkoitukseen laadittua työkalua. Manuaalisen annotaation ongelmia automaattiseen nähden ovat työkalujen vaikeakäyttöisyys sekä prosessin hitaus, virhealttius ja monimutkaisuus. Automaattinen annotaatio voi toisaalta olla hyvinkin virhealtista, ja laadukkaaseen, täysin automaattiseen annotaatioon päästään usein ainoastaan hyvin rajatuissa ympäristöissä. Automaattiset työkalut ovat harvoin täysin automaattisia ja toisaalta manuaalisten työkalujen toiminta pyritään yleensä automatisoimaan mahdollisimman pitkälle.

Annotaation onnistuminen ja parhaan annotointimenetelmän valinta on aina kontekstiriippuvaista. Esimerkiksi koko WWW:n laajuinen annotaatio vaatisi ehdottomasti automatisointia, kun taas pienemmässä, suljetussa järjestelmässä ,voi olla järkevää pyrkiä käsin tehtyihin tarkkoihin annotaatioihin ajan ja vaivan kustannuksella. Työkalujen skaala vaihtelee siis käyttötarkoituksen mukaan manuaalisesta automaattiseen, suurimman osan työkaluista sijoittuen johonkin näiden välimaastoon.

3.1 Perusvaatimukset annotaatiolle

Annotaatiota voidaan myös tehdä semantiikan kannalta eri tavoilla. Yksinkertaisimmillaan annotaatiolla voidaan tarkoittaa metatiedon tuottamista vaikkapa "kommentoimaan" asiaa, esim. web-sivua [KKPR01]. Semanttisessa webissä annotaatioiden ehdottomana vaatimuksena pitäisi kuitenkin olla kytkeytyminen ontologioihin, jotta annotaatioiden avulla luokiteltuihin asioihin muodostuisi järkevä semanttisten suhteiden verkko. Lisäksi useat annotointityökalut tuottavat dokumenteille toisistaan erillistä metatietoa, joka voi kyllä olla kytketty ontologiaan, muttei välttämättä kytkeydy toisiin annotoituihin dokumentteihin. Jos esimerkiksi ollaan luomassa henkilöstä instanssia, jonka yhtenä ominaisuutena on henkilön yhteistyökumppani, pitäisi kyseisen ominaisuuden arvoksi voida asettaa toinen henkilöinstanssi, eikä ainoastaan esimerkiksi toisen henkilön nimen sisältävä merkkijono.



Kuva 4: Järkevä annotaatio [HS02]

Esimerkki järkevästä annotaatiosta on kuvassa 4, jossa kahteen verkkosivuun on liitetty annotaatiot, jotka ilmaisevat ensinnäkin kummankin verkkosivun yksilöivät tunnukset (urn:rdf:947794d5c...) ja (urn:rdf:936694d...). Lisäksi ilmaistaan, mihin kohtaan ontologiassa verkkosivu liittyy (PhD-student ja Lecturer), sekä instanssien keskinäinen suhde (cooperatesWith). Annotaatio on järkevä, koska kumpikin instanssi kytkeytyy ontologiaan ja instanssit ovat semanttisesti yhteydessä toisiinsa.

Perusvaatimukset järkeville annotaatioille on määritelty hyvin CREAM-projektissa [HS02]:

- Yhdenmukaisuus: annotaatioiden on kytkeydyttävä ontologioihin.
- Yksiselitteisyys: tietämyskannassa ei saa olla montaa samaa asiaa tarkoittavaa annotaatioinstanssia.
- Relatiivinen metadata: annotaatioiden on oltava semanttisesti yhteydessä toisiinsa. Monissa esim. Dublin Core:a [WKLW98] käyttävissä annotaatioympäristöissä luokitellaan esim. joku osa dokumentista tarkoittamaan henkilöä, muttei kyetä annotoimaan viittausta kyseistä henkilöä kuvaavaan instanssiin, kuten esimerkiksi kuvan 4 tapauksessa on tehty.
- Ylläpidettävyys: annotaatiot oltava ylläpidettävissä.
- Helppokäyttöisyys: tämä voi olla ongelmallista, koska metadatan luominen vaatii usein sekä teknistä että semantiikkaan liittyvää ymmärrystä.
- Tehokkuus: liittyy helppokäyttöisyyteen ja annotointiprosessin automatisointiin.

3.2 Automaattisia työkaluja

Laajojen tietomassojen annotointi vaatii automaattisuuden maksimoimista. Automaattisessa annotaatiossa käytettyjä menetelmiä ovat luonnollisen kielen tunnistus (recognition), tekstin rakenteen tunnistamiseen uuttamalla (information extraction), ryhmitteleminen (clustering) sekä louhiminen (mining). Koska jo luonnollisen kielen tunnistus on erittäin vaikeaa, on prosessi usein kovin virhealtis [BD03].

Toisaalta tiedon uuttamiseen perustuvien järjestelmien pitäisi pystyä mukautumaan alakohtaisiin käsitteisiin, kielen rakenteeseen, erilaisiin tekstilajeihin sekä dokumenttiformaatteihin [CW03]. Monessa tapauksessa nämä voivat olla ylivoimaisia vaatimuksia.

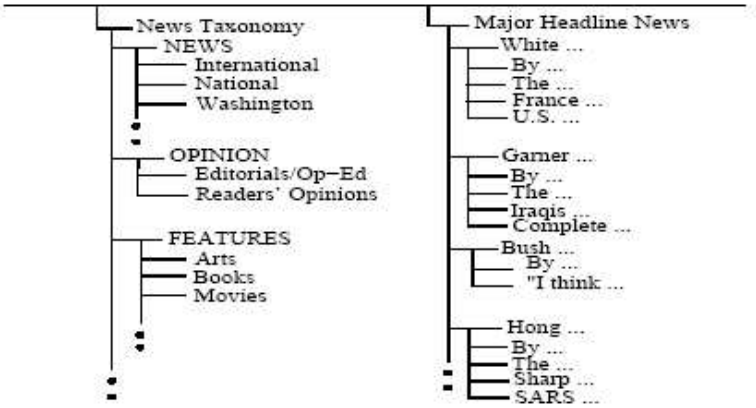
Yksi esimerkki automaattisesta annotaatiosta on ratkaisu, jossa pyritään automaattisesti annotoimaan joukko HTML-sivuja, joissa tieto on jaettu ihmislukijalle tietyn rakenteen mukaisesti, kuten esimerkiksi uutisportaalit [MYR03].

Kuva 5: The New York Times-portaalin etusivu 22.04.2003 [MYR03]

Tarkasteltaessa esimerkiksi The New York Times [9] (kts. kuva 5)- uutisportaalia, on havaittavissa että sen rakenne sisältää mm. vasemmassa laidassa olevan kiinteän taksonomian sekä sarakkeen pääuutisille sivun keskimmäisessä sarakkeessa. Yhden pääuutisen rakenne koostuu uutisen nimen sisältävästä hyperlinkistä, jota seuraa uutisen lähde. Lähteen jälkeen seuraa julkaisuaika ja tekstimuotoinen yhteenveto artikkelista, sekä mahdollisesti muutama linkki muihin asiaan liittyviin uutisiin.

Kyseisessä ratkaisussa pyrkimyksenä on ollut automaattisesti tunnistaa sivujen se-

[9] http://www.nytimes.com

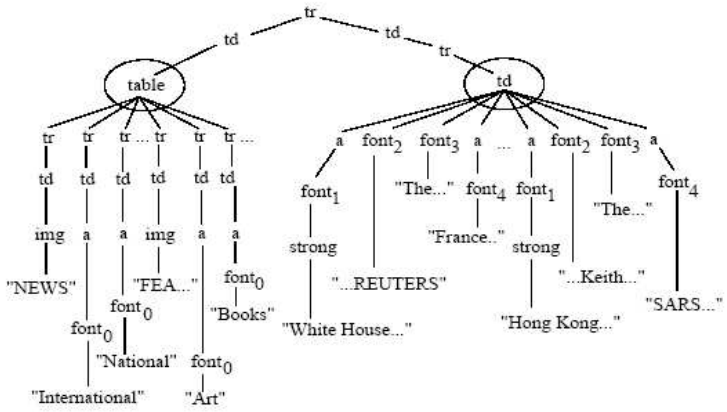


Kuva 6: Osa The New York Times sivuston semanttisesta rakenteesta [MYR03]

manttinen rakenne (kts.kuva 6) HTML-dokumentista. Rakenteen tunnistus on jaettu rakenneanalyysiin (structural analysis) sekä semanttiseen analyysiin (semantic analysis). Rakenneanalyysissä tunnistetaan dokumentin rakenne analysoimalla HTML-dokumenttia ja havaitsemalla esimerkiksi, että kaikilla uutistaksonomian juurilla, "NEWS", "OPINIONS", "FEATURES", ..., on kaikilla samanlainen rakenne HTML-dokumentin muodostamassa puussa (tr,td,table,tr,td,img), kts. kuva 7. Semanttisessa analyysissä pyritään liittämään sisältö tiettyyn ontologiaan löytämällä samoja sanoja rakenteesta sekä ontologiasta käyttäen apuna myös sanojen synonyymejä WordNet:n [Fel98] avulla. WordNet on tietokanta, jossa pyritään esittämään englannin kielisten sanojen välisiä merkityssuhteita. Yleensä sanakirjat ja sanastot on suunniteltu ainoastaan ihmisten luettaviksi. WordNetin kehittämisen päämääränä oli luoda sanojen semanttisia suhteita kuvaava tietokanta, joka olisi käytettävä myös tietokoneohjelmistoille. WordNet:ssä sanat kuuluvat synonyymijoukkoihin (synset), ja sen avulla voidaan hakea sanalle eri konteksteissa kuuluvat synonyymit. WordNet on kehitetty Princetonin yliopistossa, ja sen on käytettävissa Internetin välityksellä [10].

Ratkaisua testattiin 50:llä kahdeksasta eri uutisportaalista otetulla HTML-sivulla käyttäen kuvan 8 kaltaista ontologiaa [MYR03]. Tulokset olivat hyviä, koska lähes kaikki sivut annotoituivat suhteellisen järkevästi. Arvioitaessa ratkaisun hyvyyttä on syytä huomioida, että käytetty ontologia on varsin yksinkertainen, sekä että sivujen rakenteet ja sisällöt olivat hyvin toistensa kaltaisia. Huomion arvoinen asia

[10] http://www.cogsci.princeton.edu/~wn/



Kuva 7: Osa The New York Times sivuston DOM-puusta [MYR03]

on myös se, että dokumenttien rakenteet muuttuvat. Esimerkiksi verrattaessa The New York Times -portaalin [11] etusivua 22.3.2003 siihen, mitä se on tällä hetkellä, on rakenteessa selviä eroja.

Kuva 8: Uutisontologia [MYR03]

Esimerkki kehittyneestä automaattisesta annotaatiotyökalusta on Amilcare-järjestelmä [CW03], [12]. Se on tiedon uuttamiseen perustuva oppiva järjestelmä, joka mukautuu erilaisiin käyttöympäristöihin. Tavoitteena on ollut laatia järjestelmä, joka

[11] http://www.nytimes.com
[12] http://nlp.shef.ac.uk/amilcare/

muodostaa kullekin käyttöalueelle sopivat omat sääntönsä. Käyttöalueeseen kuuluvat alakohtaiset käsitteet, sanasto, tekstilajit (esim. lääketieteellinen teksti, poliisiraportit yms.), omanlainen tekstirakenne ja tekstiformaatti (esim. XML-dokumentit tai vapaata tekstiä sisältävät dokumentit).

Amilcare perustuu LP2-oppimisalgoritmiin [Cir01], joka pyrkii analysoimaan tekstiä eri tavoin hyödyntäen luonnollisen kielen tunnistusta. Järjestelmä saa syötteenä joukon annotoituja dokumentteja, joiden perusteella se muodostaa kyseisen tyyppisille dokumenteille soveltuvat annotaatiosäännöt. Seuraavassa vaiheessa järjestelmä yrittää luomiensa sääntöjen avulla uuttaa (extract) annotaatioita käyttäjän valitsemista dokumenteista.

Amilcare voidaan integroida manuaalisiin työkaluihin, jolloin järjestelmälle syötettävät annotoidut dokumentit saadaan toiselta järjestelmältä. Esimerkkejä työkaluista, joihin Amilcare on integroitu ovat MnM [VVMD+02] ja OntoMat [HSC02]-järjestelmät. Amilcaren toimintaa voi säädellä erilaisia käyttötilanteita varten, esimerkiksi pyrittäessä joko maksimoimaan automaattisuus tai minimoimaan annotaatiovirheet, antaen käyttäjän korjata ja muokata luotuja annotaatiosääntöjä [CW03].

Vaikka Amilcarella on päästy tietyissä tapauksissa hyviin tuloksiin, tiedon uuttamiseen perustuvat järjestelmät saavuttavat sataprosenttisen tarkkuuden vain erittäin yksinkertaisissa tapauksissa ja ympäristöissä [CW03]. Tarkkoihin annotaatioihin päästään ainoastaan puoliautomaattisen annotaation avulla, jossa käyttäjä tarkastaa ja korjaa luodut annotaatiot. Vaihtoehtona on manuaalinen annotaatio, mutta annotoidessa laajaa dokumenttien joukkoa, tiedon uuttamiseen perustuvat järjestelmät tarjoavat kuitenkin suurta apua [CW03].

Armadillo [CCDW04] on järjestelmä, joka pyrkii täysin automaattiseen annotaatioon käyttäen hyväkseen tiedon uuttamista tietolähteistä, kuten digitaalisista kirjastoista tai tietokannoista. Järjestelmää testattiin siten, että tarkoitus oli hakea tietyn yliopiston Internet-sivuilta kaikkien siellä työskentelevien tutkijoiden nimet. Järjestelmä päätteli erilaisten hakukoneiden, kuten CiteSeer [13] ja Unitrier [14] avulla, ketkä yliopiston sivuilla esiintyvistä henkilöiden nimistä olivat tutkijoita. Jos esim. henkilön nimellä ei löytynyt CiteSeeristä julkaisuja, henkilöä ei tulkittu tutkijaksi. Vaikka Armadillon tekijät haluavatkin nähdä järjestelmän "täysin automaattisena", vaatii esimerkinkin kaltaisen haun järjestäminen paljon käsityötä. Toiseksi on kyseenalaista kuinka moneen eri käyttötarkoitukseen on tarjolla sopivia tietokantoja.

[13] http://www.citeseer.org

[14] http://www.informatik.unitrier.de/ ley/db/

Yleisesti automaattinen annotaatio soveltuu parhaiten tilanteisiin, joissa annotoitavana on suuri joukko kielellisesti ja rakenteeltaan mahdollisimman samankaltaisia dokumentteja. Tällöinkin ihmisen apu annotaatiossa on usein välttämätöntä. Automaattiset annotaatioeditorit soveltuvat ainoastaan asiantuntijakäyttäjien käytettäviksi, eikä niiden kehittäminen tavallisten käyttäjien käytettäväksi ole mielekästä, koska niiden suoritus on kertaluontoista. Automaattisissa annotaatioeditoreissa on kuitenkin ominaisuuksia, joita voidaan soveltaa myös manuaalisissa työkaluissa. Esimerkiksi tässä tutkimuksessa laaditussa editorissa pyritään automaattisesti tunnistamaan muiden käyttäjien vastaavissa tilanteissa tekemiä annotaatioita, ja ehdottamaan niitä käyttäjälle.

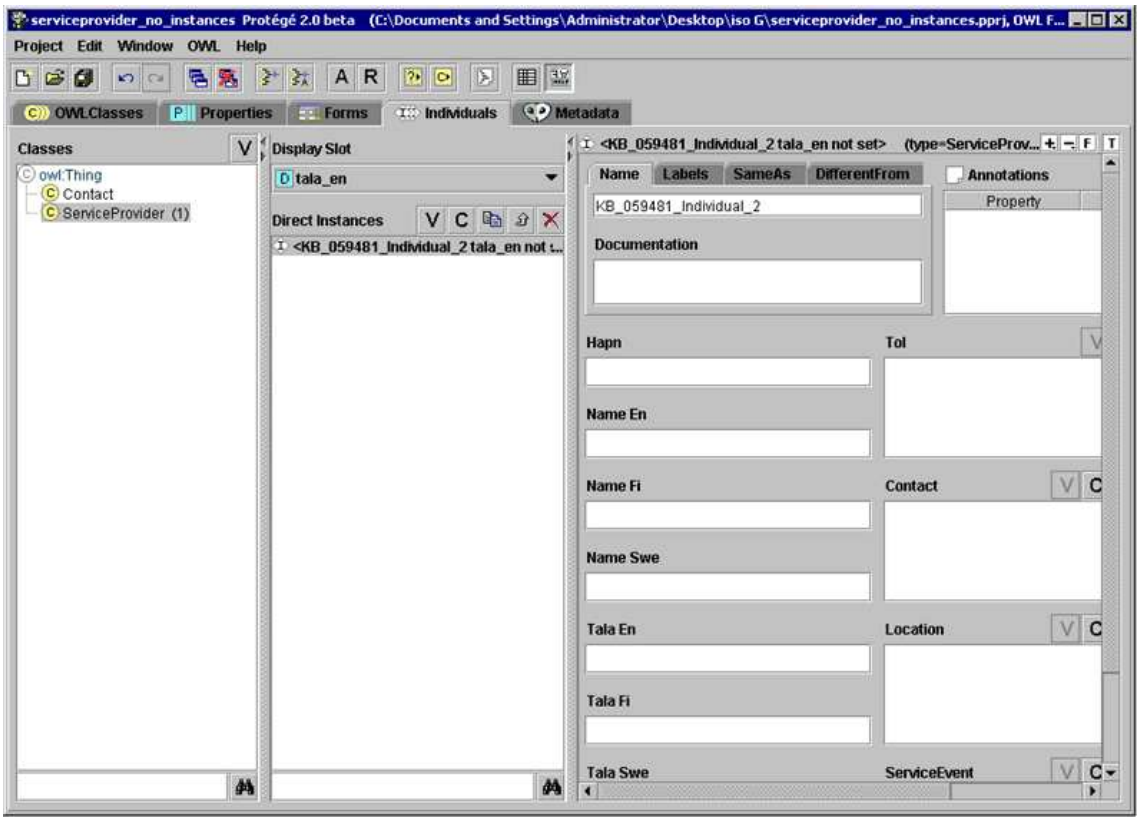
3.3 Manuaalisia työkaluja

Manuaalisessa annotaatiossa lähtökohtaisesti ihminen käsittelee annotoitavia dokumentteja yksi kerrallaan ja liittää niihin semanttisia luokitteluja. Annotaatioita voidaan yksinkertaisimmillaan tehdä liittämällä tekstieditorilla annotoitaviin dokumentteihin metatietoa esim. HTML-dokumenteissa <meta>-kenttien sisään. Tällä tavoin laaditut annotaatiot ovat varsin virhealttiita sekä semantiikan että myös syntaksin kannalta.

Käytettäessä annotaatioeditoria, saavutetaan huomattavasti parempi käytettävyys ja säästytään syntaksivirheiltä. Editorin käyttäjä esimerkiksi valitsee dokumentista tietyn tekstin ja liittää sen työkalun avulla sopivaan luokkaan ontologiassa. Useat manuaaliset annotaatioeditorit vaativat käyttäjältään joko tietoteknistä tai ontologiseen mallinnukseen liittyvää asiantuntemusta, jotta ne kykenisivät tuottamaan järkeviä ja riivvätän laajoja annotaatioita dokumenteille. Lisäksi manuaalinen annotaatio on usein työlästä sekä hidasta.

Protege [PROT] on tietämyseditori, jolla voidaan luoda ontologioita sekä niihin liittyviä annotaatioinstansseja. Protege on ollut jo pitkään suosittu mallinnustyökalu esim. lääketieteen alan asiantuntijoiden käytössä [NSD+01]. Protege on monipuolinen työkalu, jolla voidaan mallintaa tietoa ja muokata ontologioita. Kuvassa 9 on näkymä Protege-editorista, jolla on avattu luokat “Contact” ja “ServiceProvider” sisältävä “ServiceProvider”-ontologia (kts. vasen osio kuvasta). Kuvassa on avattu “Individuals”-välilehti, jossa ollaan luomassa “ServiceProvider”-tyyppistä annotaatioinstanssia. Kuvan keskimmäisessä osassa näkyy luotava instanssi, ja siitä voisi valita tarkasteltavaksi myös muita kyseisestä luokasta luotuja instansseja. Ruudun oikeassa laidassa näkyvät luokan “ServiceProvider” ominaisuudet, kuten “Hapn”,

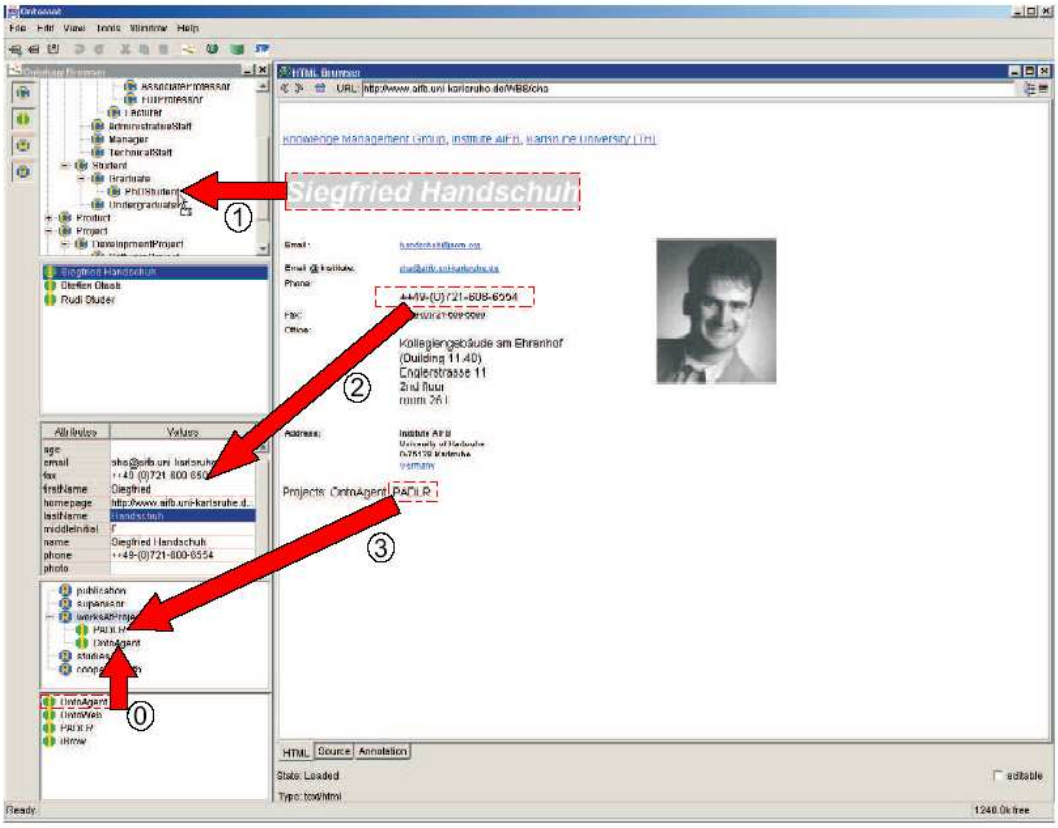
"Name En", "Name Fi", "Name Swe", "Tala En" ja "Tala fi" joihin käyttäjä voi tässä syöttää arvot. Edellä luetellut ominaisuudet ovat tekstiarvoisia ominaisuuksia. Ominaisuudet "Tol", "Contact" ja "Location" ja "ServiceEvent" ovat objektityyppisiä ominaisuuksia, joihin käyttäjä voi "C"-painiketta painamalla hakea objektityyppisen ominaisuuden. Protege on tarkoitettu ammattikäyttäjien käytettäväksi, ja vaikka asiansa osaava käyttäjä pystyy sillä luomaan haluamansa kaltaisia annotaatioinstansseja, ei sitä sen monimutkaisuuden vuoksi voi ajatella tekniikkaa tai alakohtaista semantiikkaa ymmärtämättömien annotoijien käyttöön.



Kuva 9: Protege tietämyseditori

Useat varsinaiset semanttista webiä varten laaditut annotaatioeditorit on tarkoitettu HTML-sivujen annotaatiota varten [HS02, KH01, KKPR01, DV00, Yee98, Hef01]. Yleinen toimintaperiaate on, että ontologioita sekä WWW-sivuista tehtäviä annotaatioita säilytetään ontologia- ja tietämyspalvelimilla, joiden kautta dokumentteja voidaan hakea annotaatioita hyödyntäen. Esimerkki kuvatun kaltaisesti toimivasta järjestelmästä on CREAM [HS02], joka suunniteltiin tuottamaan metatietoa tietyn

portaalin HTML-sivuille.



Kuva 10: CREAM Käyttöliittymäesimerkki [HS02]

CREAM sisältää graafisella käyttöliittymällä varustetun editorin, jolla voi maalata dokumenttien tekstejä ja linkittää niitä ontologioihin(kts. kuva 10). Kuvassa käyttäjä maalaa Internet-sivusta tekstin “Siegfried Handschuh” ja raahaa sen ontologian kohtaan “PhD Student”, jolloin kyseisestä ontologian luokasta luodaan uusi instanssi (kohta 1). Luodun instanssin ominaisuuksiksi voidaan asettaa esim. puhelinnumero raahaamalla se luokan ominaisuuslistaan (kts. kohta 2). Kohdassa 3 raahataan projektin nimi “PADLR” relaatioon “worksInProject” liittyvän instanssin (kohta 0) kohdalle, jolloin siis ilmaistaan että kyseinen henkilö työskentelee projektissa “PADLR”.

Järjestelmä sisältää myös mönkijän (crawler), jonka tarkoitus on hakea HTML-dokumenteissa olevaa annotaatiodataa muodostaen tästä tietämyskantaa. Lisäksi järjestelmässä on palvelin, johon annotaatiodata sekä ontologiat ovat tallennettu. Järjestelmän toimintaa on myös pyritty automatisoimaan integroimalla siihen aiem-

min esitelty Amilcare järjestelmä. Eräs varteenotettava työkalu CREAM:n lisäksi on MnM [VVMD+02], jonka toimintaperiaate on varsin samanlainen kuin CREAM:ssa.

Muita tunnettuja semanttista webiä varten kehitettyjä annotaatioeditoreita ovat esim. Annotea [KKPR01], ThirdVoice [15], Yawas [DV00], CritLink [Yee98], SHOE Knowledge Annotator [Hef01], AeroDAML [KH01] ja OntoMat [HSC02].

ThirdVoicessa, Yawas:ssa, CritLinkissä, sekä Annoteassa annotaatiolla tarkoitetaan web-sivuhin liitettäviä käyttäjien kommentteja, kuten neuvoja, muutosehdotuksia tai mielipiteitä. Ne perustuvat dokumenttilähtöiseen (document-centric) lähtökohtaan, jossa käyttäjät selailevat satunnaisesti dokumentteja ja tutkivat niihin liitettyjä annotaatioita (kommentteja). Annotaatiot ei kyseisissä järjestelmissä ole tarkoitettu helpottamaan tiedon hakua, eivätkä muutenkaan täytä mitään tämän luvun alussa määriteltyjä perusvaatimuksia semanttisen webin annotaatioeditoreille.

SHOE Annotator -työkalu [Hef01] pyrkii annotoimaan web-sivuja linkittämällä ne ontologioihin käyttäen omaa SHOE-kieltään. Annotaatiot talletetaan palvelimelle, ja niitä käytetään helpottamaan sivujen löytämistä. SHOE ei tue RDF-kieltä. Se on lähestymistavaltaan hyvin samankaltainen CREAM:n kanssa, mutta verrattuna CREAM:n sitä voidaan kuvailla enemmänkin "pieneksi apuohjelmaksi" kuin varteenotettavaksi annotaatioympäristöksi [HS02].

AeroDAML [KH01] on verkkopalvelu (Web Service [16]), joka annotoi automaattisesti annetun Web-sivun annetun ontologian mukaisesti käyttäen apunaan WordNet:ä. AeroDAML tunnistaa dokumentista yksittäisiä sanoja, ja liittää niitä ontologian luokkiin. Esimerkiksi se tunnistaa, että sana "Japan" on "nation"-luokan instanssi, tai että "gun" on "weapon"-luokan instanssi. AeroDAML:n lähestymistapa on varsin yksinkertainen, se kykenee ainoastaan tunnistamaan yksittäisiä sanoja, eikä osaa tulkita dokumenttien rakenteita sen tarkemmin.

3.4 Käytettävyysongelmat

Obtobroker-järjestelmää [EMSS00] varten pyydettiin käyttäjiä annotoimaan web-sivujaan lisäämällä tekstieditorilla HTML-sivuihinsa annotaatiotietoja <meta>-kentti-sisään. 30 testatun käyttäjän perusteella tehdyt tulokset annotaation suhteen olivat huonot: ainoastaan 15 käyttäjää kykenivät jollakin lailla annotoimaan sivunsa. Suuri osa annotaatioista oli syntaktisesti vääränmuotoisia. Lisäksi ilmeni käytettyyn

[15] http://www.thirdvoice.com
[16] http://www.w3.org/2002/ws/

terminologiaan liittyviä ymmärrysvaikeuksia sekä ongelmia semanttisten suhteiden ymmärtämisessä. Vaikka kyseessä oli ainoastaan web-sivuihin perustuva hyvin yksinkertainen annotaatio, on selvää että kyseiseen tehtävään tarvitaan jonkinlainen käyttäjää ohjaava työkalu. Syntaktiset ongelmat ratkeavat helposti, isompaa vaivaa koituu terminologisista ongelmista.

Jotkut kyseiseen annotaatioon osallistuneista henkilöistä eivät olleet ymmärtäneet ontologisia termejä kunnolla [EMSS00]. Tästä johtuen esimerkiksi eräs käyttäjä oli annotoinut julkaisuja (publications) sisältävän web-sivun henkilöön liittyväksi julkaisuksi (publication). Tämä oli järjestelmän kannalta oikeellinen annotaatio, sikäli että se tulkitsi julkaisulistan sisältävän sivun julkaisuksi. Tämä oli ongelma, koska kyseisessä kontekstissa julkaisulla tarkoitettiin nimenomaan tieteellistä artikkelia, eikä HTML-sivu jolla listataan tietyn henkilön tekemät julkaisut ole tieteellinen artikkeli.

Lisäksi käyttäjät eivät ymmärtäneet aina viitata asioihin URI-viittauksilla, vaan antoivat esimerkiksi henkilö-kenttään URI-viittauksen sijasta merkkijonomuotoisen nimen. Lisäksi URI-viittauksiin liittyen havaittiin ongelmia siinä, ettei osattu viitata aiemmin esimerkiksi tietystä henkilöstä laadittuun instanssiin, vaan luotiin uusi instanssi samasta asiasta. Lisäksi havaittiin haluttomuutta laatia kovinkaan laajoja annotaatioita, vaan tyydyttiin ainoastaan minimaaliseen luokitteluun. Vastaavankaltaisiin ongelmiin törmättiin myös Meedio-metadataeditorin [17] kehityksessä, ja samat ongelmat ovat kiteytyneet ongelmiksi monissa editoreissa, kuten Amilcaressa.

Annotaatioeditoreita käytetään monenlaisiin tarkoituksiin. Vaikka ne poikkeuksetta vaativat käyttäjiltään tietämystä ontologisista käsitteistä ja tekniikoista, ei se useissa tapauksissa ole ongelma. On kuitenkin tilanteita, joissa annotaation tekemistä voitaisiin helpottaa niin, että se olisi mahdollista myös “tavallisille käyttäjille”. Lisäksi annotaatioita olisi järkevää luoda jo siinä vaiheessa, kun järjestelmiin syötetään tietoa eikä vasta jälkikäteen.

3.5 Tarjottu ratkaisumalli

Tavoitteina laaditulle editorille asetettiin, että sen tulisi ensinnäkin täyttää editorien perusvaatimukset siten, että sillä on mahdollista tehdä syntaktisesti ja semanttisesti oikeellisia annotaatioita. Tutustuttaessa muihin laadittuihin annotaatioeditoreihin,

[17] http://www.cs.helsinki.fi/group/meedio

havaittiin, että vaikka asiantunteva käyttäjä voi niiden avulla helposti luoda annotaatioita, on niillä mahdollista luoda täysin mielivaltaisia annotaatioita. Tämän takia editorit eivät sovellu tavallisten käyttäjien käytettäväksi.

Laadittavan editorin kohderyhmäksi asetettiin tavalliset käyttäjät niin, että editorin olisi mahdollisimman pitkälle opastettava käyttäjää järkevien annotaatioiden tekemisessä ja pakotettava käyttäjä tekemään riittävän laaja ja järkevä annotaatio. Käyttäjän opastaminen ajateltiin toteuttaa siten, että editorin asetustiedostossa olisi määriteltävissä valintoja rajoittavia käyttöliittymäkomponentteja erilaisten instanssin ominaisuuksien kohdalle. Lisäksi editorilla olisi käytössään erillinen relaatioontologia, jonka perusteella käyttäjän tehdessä tietynlaisia valintoja, voitaisiin muita kyseisen instanssin ominaisuuksien mahdollisia valintoja rajoittaa. Tällöin käyttäjä ei pystyisi tekemään tiettyyn valintaan sopimatonta toista valintaa. Lisäksi editorin asetuksissa voitaisiin määritellä annotaation vaatimat pakolliset tiedot, jolloin editori pakottaisi käyttäjät antamaan riittävän laajoja annotaatioita. Lisäksi editoriin suunniteltiin automaattinen suositteluominaisuus, jolloin editori osaisi suositella annotaatioita perustuen muiden käyttäjien vastaavissa tilanteissa tekemiin annotaatioihin.

Editori on lähtökohtaisesti laadittu keltaiset sivut- tyyppisten palveluilmoitusten lisäämistä ja annotaatiota varten, mutta tärkeänä tavoitteena editorin laatimisessa on ollut geneerisyys siten, että editori olisi helposti sovitettavissa myös muihin ympäristöihin.

Laadittu editori rajautuu pääosin ratkaisemaan edellä mainittuja manuaaliseen annotaatioon liittyviä ongelmia, joskin muiden käyttäjien tekemiin annotointeihin perustuva suosittelu voidaan nähdä automaattiseen annotaatioon liittyvänä ominaisuutena. Laadittu editori toimii editoinnin ohella ilmoitusten syöttöjärjestelmänä siten, että annotaatiot luodaan samalla kun tietoja syötetään järjestelmään.

Luvussa 4 käydään lävitse yleisimmät semanttisessa webissä käytössä olevat tekniikat, perustellaan editorin laadintaan valitut tekniikat sekä käytetty testiaineisto. Lukija jolle nämä tekniikat ovat tuttuja, voi siirtyä lukuun 5, joka käsittelee tätä tutkimusta varten laadittua annotaatioeditoria.

4 Tekniikat ja toteutusympäristö

Tässä luvussa kuvataan ja perustellaan editorin laatimiseen valitut ympäristöt sekä käytetty testiaineisto.

Suunnittelussa lähdettiin siitä, että editori toimisi selainohjelmassa, jotta se olisi mahdollisimman käytettävä erilaisissa käyttöympäristöissä sekä että sen käyttämät ontologiat ja sen tallentamat annotaatioinstanssit sijaitsisivat tietyllä palvelimella jolloin ne olisivat yhdenmukaisia kaikkien käynnissä olevien editoriohjelmien välillä.

Editorin laatimista varten tarvittaisiin siten välineet RDF-tiedon käsittelyyn, Internetissä toimivan sovelluksen laadintaan sopiva ympäristö sekä tekniikka, joka mahdollistaisi palvelimen ja editorisovelluksen välisen kommunikaation Internetin välityksellä. Web-palvelut (Web Services) [BHM+03] ovat yleistymässä oleva tapa, jolla Internetissä toimivat ohjelmat voivat tarjota palveluitaan ja kommunikoida keskenään. Palvelimen toteuttaminen Web-palvelu -tekniikoiden avulla koettiin järkeväksi koska tällöin palvelimen tarjoamat palvelut voitaisiin helposti julkistaa myös muuta mahdollista käyttöä varten.

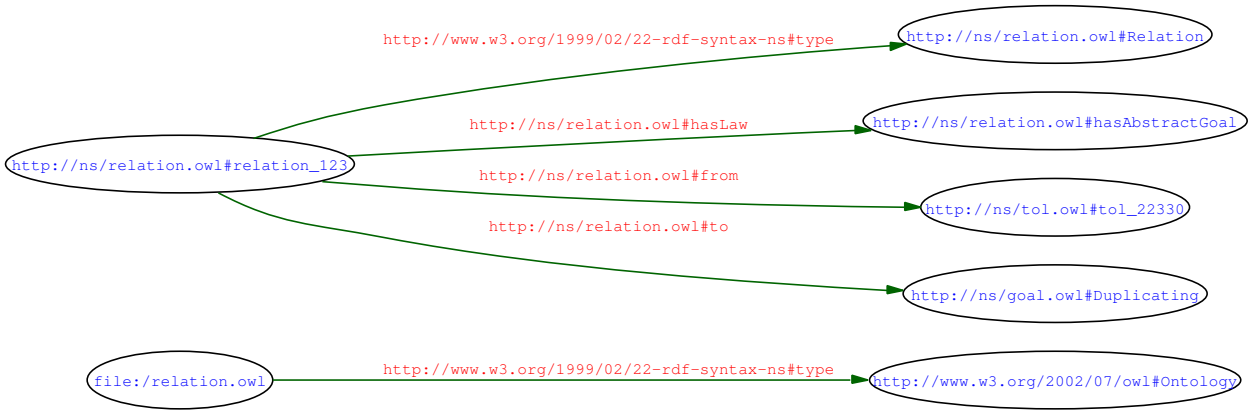
4.1 Tekniikat RDF-käsittelyyn

RDF-tiedon käsittelyyn on olemassa jo lukuisia työkaluja ja ohjelmointirajapintoja, kuten RDF API for PHP (RAP) [RRAP], Raptor RDF Parser Toolkit [RAPT] C-kielelle, Pyrple [PYRP] Python[18] -kielelle, SWI-Prolog:n RDF-parsintapakkaus [SWIP], sekä HP-Labs:n Java:lle kehittämä Jena-paketti [JEN03].

Ensisijaisesti lähdettiin hakemaan Java-kielistä parseria helppokäyttöisyyden ja monipuolisuuden ollessa tärkeimmät valintakriteerit esim. tehokkuuden jäädessä toisarvoiseksi. Jos olisi valittu muu kuin Java-kielinen parseri, olisi todennäköisesti jouduttu ratkaisemaan paljon varsinaisen tutkimusongelman kannalta epäolennaisia ongelmia.

Jena on suosittu Open source-lisenssin [19] alainen ohjelmointirajapinta, jota on kehitetty RDF:n alkuajoista lähtien läheisessä yhteistyössä W3C:n RDF-työryhmän (RDF Working Group) kanssa. Jenan ohjelmointirajapinnan arkkitehtuuri keskittyy RDF-malliin (model), eli joukkoon RDF-lauseita, joista koostuu yksi RDF-dokumentti, graafi tai sanasto. Yksinkertaisen RDF-dokumentin luominen onnistuu

[18] http://www.python.org/
[19] http://www.opensource.org/



Kuva 11: Esimerkki RDF-verkko

luomalla Model-luokan ilmentymä ja lisäämällä siihen vähintään yksi Statement-luokan ilmentymä. Jenassa RDF-mallia voidaan säilyttää tiedostossa, muistissa tai tietokannassa.

Mallissa olevaa RDF-tietoa voi käsitellä suoraan API-kutsuilla tai syntaksiltaan SQL-kieltä muistuttavan RDQL-kysekielen [RDQL] kautta. Jenassa on myös luokat DAML+OIL ja OWL-muotoisten ontologioiden helppoa käsittelyä varten sekä päättelykone. Edellä lueteltujen ominaisuuksien perusteella Jena on tarjolla olevista RDF-ohjelmointirajapinnoista monipuolisin ja helppokäyttöisin.

RDQL (A Query Language for RDF) on W3C:n suositus RDF-kyselykieleksi. Ensimmäinen RDQL-toteutus oli Jenan versiossa 1.2.0. Nykyään RDQL- toteutuksia on myös järjestelmissä, kuten RDFStore [20], Sesame [21], PHP XML Classes [22], 3Store [23] sekä RAP [RRAP]. Myös Joseki RDF-palvelin [JOSE] ottaa vastaan RDQL-muotoisia kyselyitä.

Kuvan 11 RDF-verkossa on määritelty että instanssilla “relation-123” on ominaisuuden “from” arvona “tol.owl#tol_22330” ja ominaisuuden “to” arvona “goal.owl#Duplicating”. Seuraavassa on esimerkki RDQL-kyselystä, jolla haetaan kyseisen kaltaisesta RDF-verkosta kaikki “to”-arvot sellaisille instansseille, joiden “from”-kentässä arvona on “tol_22330”.

```
SELECT ?x WHERE
 (?y, rdf:type, <http://ns/relation.owl#Relation>),
```

[20] http://rdfstore.sourceforge.net/
[21] http://www.openrdf.org/
[22] http://phpxmlclasses.sourceforge.net/
[23] http://sourceforge.net/projects/threestore/

```
(?y, <http://ns/relation#from>, "http://ns/tol.owl#tol_22330"),
(?y, <http://ns/relation#to>, ?x)
```

Jos edellistä kyselyä sovellettaisiin kuvan 11 RDF-verkkoon, saataisiin vastaukseksi arvo “http://ns/goal.owl#Duplicating”.

4.2 Web-tekniikat

Cocoon [COCO] on Apache Cocoon -projektin toteuttama avoimen lähdekoodin julkaisujärjestelmän kehys, jonka toiminta perustuu XML-kieleen, XSL-muunnoksiin ja joukkoon yleiskäyttöisiä komponentteja, joista kootaan putkilinjoja (pipeline). Putkilinjojen idea on, että kukin niiden sisältämä komponentti huolehtii tietystä operaatiosta, jolloin sovelluksia voidaan toteuttaa yhdistelemällä erilaisia tarkoitukseen soveltuvia komponentteja. Putkilinjoihin kuuluvat seuraavat komponentit: generaattorit (generator), muuntajat (transformer) ja serialisoijat (serializer).

Cocoon sisältää lukuisia mahdollisia tapoja Web-sovellusten laatimiseen. Seuraavassa käsitellään Cocoon:n Advanced Control Flow-ominaisuus, joka valittiin annotaatioeditorin kehittämisen perustaksi. Seuraavassa perustellaan tehty valinta.

Useimmat Web-sovellusten laatimiseen perustuvat ympäristöt mallintavat sovelluksen kontrollivuon (control flow) tila-automaattina (finite-state-machine) siten, että sovellus koostuu useista tiloista ja sovellus voi olla yhdessä tilassa kerrallaan. Sovellukset toimivat tapahtumaohjatusti (event-driven) siten, että selaimelta tuleva pyyntö aiheuttaa tietyn tapahtuman, joka siirtää sovelluksen toiseen tilaan. Siirtymän yhteydessä sovellus voi suorittaa toimenpiteitä, kuten päivittää muistissa tai tietokannassa olevaa tietoa.

Useissa ympäristöissä [STRU, TURB] toiminta mallinnetaan tila-automaattiin perustuvan Model-View-Controller (MVC)-suunnittelumallin mukaan, jossa käyttöliittymä ja itse sovelluksen toiminta pyritään jakamaan eri osiin. Tila-automaattina mallintaminen toimii mainiosti yksinkertaisissa web-sovelluksissa, mutta sovelluksen laajetessa eri tilojen ja siirtymien määrä kasvaa helposti hallitsemattomaksi.

Cocoon sisältää ainoana Web-sovellusten kehitysympäristönä jatkeiksi (continuations) kutsutun ominaisuuden, jonka avulla Web-sovellus voidaan mallintaa perinteisen ohjelman tavoin. Seuraavassa esimerkissä on havainnollistettu jatkeiden toiminta. Esimerkissä sovelluksen toiminta alkaa calculator()-funktiosta, jossa alustetaan muuttujat a, b sekä operator. Seuraavaksi lähetetään käyttäjälle lomake, joka

sisältää kentän numeron syöttämistä varten. Ohjelman kontrolli siirtyy nyt lomakkeeseen ja ohjelman suoritus jää odottamaan lomakkeelta palaamista. Kun käyttäjä on syöttänyt lomakkeeseen arvon, kontrolli palaa ohjelmalle, jonka suoritus jatkuu seuraavasta käskystä. Seuraavaksi ohjelma hakee vastaavalla tavalla arvot kentälle b, sekä operator. Lopuksi ohjelma joko laskee yhteen tai vähentää toisistaan luvut a ja b riippuen muuttujan operator arvosta sekä lähettää käyttäjälle vastauslomakkeen.

```
function calculator()
{
  var a, b, operator;

  cocoon.sendPageAndWait(''getA.html'');
  a = cocoon.request.get(''a'');

  cocoon.sendPageAndWait(''getB.html'');
  b = cocoon.request.get(''b'');

  cocoon.sendPageAndWait(''getOperator.html'');
  operator = cocoon.request.get(''op'');

  if (operator == ''plus'')
     cocoon.sendPage(''result.html'', {result: a+b});
  else if (operator == ''minus'')
     cocoon.sendPage(''result.html'', {result: a-b});
  else
     cocoon.sendPage(''result.html'', {result: 0});
}
```

Kullekin käyttäjälle käynnistetään oma kopio ohjelmasta ja sen paikallisista muuttujista, jolloin kaikki muuttujat ovat käyttäjäkohtaisia. Ohjelman tila voidaan tila-automaattina sovelluksen mallintaviin ohjelmiin verrattuna säilyttää huomattavasti yksinkertaisemmin normaaleissa muuttujissa. Toimintalogiikan sisältävä ohjelma voi olla toteutettu joko JavaScript-, tai Java-kielellä.

Cocoonissa käyttäjälle näytettävän lomakkeen muodostavan skriptin valintaan on myös lukuisia vaihtoehtoja, kuten Cocoon Forms, JXForms ja JXTemplate. Näistä JXTemplaten on kaikkein kevyin vaihtoehto sisältäen kuitenkin hyödyllisiä ominai-

suuksia. JXTemplaten käyttö mahdollistaa Java-, tai JavaScript-olioiden käytön lomakkeessa, joka on välitetty Cocoonin Advanced Control Flow:n kautta. JXTemplate sisältää joukon rakenteita, jotka mahdollistavat esimerkiksi Java-kokoelmien (collections) iteroinnin sivun sisällä.

Flow:ssa voidaan JXTemplate-skriptille välittää esimerkiksi Javan listatyyppinen tietorakenne, jota voidaan iteroida seuraavan esimerkin tapaan. Seuraava esimerkki muodostaa HTML-lomakkeen, joka sisältää HTML-taulukossa skriptille parametrina välitetyn Java:n Array-tietotyyppisen "java-taulukko-objekti":n kaikki alkiot.

```
<html><head></head><body>
<table>

<jx:forEach var=''item'' items=''${java-taulukko-objekti}''>
   <td> ${item} </td>
</jx:forEach>

</table>
</body></html>
```

Seuraavassa on monipuolisempi Java-kielinen esimerkki showForm-nimisestä funktiosta, ja siihen liittyvästä JXTemplate-skriptistä. Esimerkissä suoritetaan myös RDF-tiedon käsittelyä Jena-pakkauksen avulla.

Funktiolle showForm annetaan parametriksi OWL-malli String-muotoisena. Riveillä 3-4 luodaan Jenan OntModel-tietomalli, johon riveillä 5-6 luetaan parametrina annettu OWL-malli. Riveillä 9-10 lähetetään kontrolli lomakkeelle. Lomakkeelle välitetään parametrina OntModel-luokan ilmentymä model. Riveillä 12-20 kerätään lomakkeelta saadut paluuarvot hajautustauluun (HashMap), jossa avaimena toimii kentän nimi HTML-lomakkeessa, ja arvona käyttäjän lomakkeen kenttään syöttämä arvo. Hajautustaulusta voidaan hakea tietoa, kuten rivillä 22, jossa taulusta haetaan HTML-lomakkeen "location"-nimisen kentän arvo String-tyyppiseen muuttujaan s.

```
1: public void showForm ( String owl )
2: {
3:    OntModel model = ModelFactory.
4:                     createOntologyModel(OntModelSpec.OWL_MEM, null);
```

```
5:    ByteArrayInputStream is=
6:                        new ByteArrayInputStream(owl.getBytes());
7:    model.read(is,"");
8:
9:    cocoon.sendPageAndWait("editorForm'', new VarMap()
10:           .add("model",model));
11:
12:   Enumeration e = cocoon.getRequest().getParameterNames();
13:   HashMap parameters = new HashMap();
14:   while (e.hasMoreElements())
15:   {
16:       String p = (String)e.nextElement();
17:   parameters.put(p,cocoon.getRequest().getParameter(p));
18:   System.out.println("KLUDGE:"+p+":"+cocoon.getRequest().
19:                                              getParameter(p));
20:   }
21:
22:   String s = parameters.get(``location'');
23:
24:   return;
25: }
```

Seuraavassa on edelliseen esimerkkiin liittyvä JXTemplate-lomake, jolle kontrolli siirtyy edellisen esimerkin riveiltä 9-10. Skripti hakee parametrina välitetystä Jena-mallista OWL-luokan "ServiceProvider", ja muodostaa sen perusteella HTML-lomakkeen, jossa on kunkin OWL-luokan ominaisuuden nimi sekä kenttä, johon käyttäjä voi syöttää ominaisuudelle arvon.

Rivillä 5-7 oleva forEach-rakenne iteroi Jenan OntModel-luokan listDeclaredProperties-metodin palauttaman Javan Iterator luokkaa laajentavan ExtendedIterator-olion kaikki arvot läpi, suorittaen kullekin skriptin rivit 8-21. Metodin listDeclaredProperties palauttama iteraattori iteroi Jenan OntProperty-tyyppisiä olioita. Rivillä 9 on if-rakenne, jossa kutsutaan kulloinkin iteroitavan OntProperty-olion totuusarvotyyppistä isDataTypeProperty-metodia, joka kertoo onko käsiteltävä ominaisuus tyypiltään tekstikenttä. Jos kenttä on tekstikenttä, suoritetaan rivit 10-12, joissa lomakkeeseen tulostuu kentän nimi getLabel-metodikutsun kautta, sekä tyhjä kenttä, jonka nimeksi annetaan kyseisen olion public-tyyppisen localName-muuttujan arvo.

Jos ominaisuus on objektityyppinen (rivi 15), tulostetaan ominaisuuden nimi, sekä “Muokkaa”-nappi. Käyttäjän kuitattua lomakkeen joko “Muokkaa” napilla tai perinteisellä HTML-lomakkeen “Submit”-napilla kontrolli palaa showForm-metodille. Esimerkissä kannattaa huomioida, että “Muokkaa”-nappi ei aiheita esimerkissä minkäänlaista toimintaa, koska sitä ei oteta kiinni showForm-metodissa.

```
1:  <form xmlns:jx="&jx;">
2:   <form method="post" name="myForm"
3:                     action="${continuation.id}.cont">
4:    <table>
5:     <jx:forEach var="current_item"
6:             items="${model.getOntClass(‘‘ServiceProvider’’).
7:                      listDeclaredProperties()}">
8:      <tr/>
9:   <jx:if test="${current_item.isDatatypeProperty()}">
10:           <td/>${current_item.getLabel(‘‘fi’’)}
11:           <td/><input type="text"
12:                         name="${current_item.localName}"
13: </jx:if>
14:
15:     <jx:if test="${current_item.isObjectProperty()}">
16:        <td/>${current_item.getLabel(‘‘’fi’’)}
17:        <td/><button id="editobject"
18:                  value="${current_item}"
19:                   type="submit"
20:                   name="editobject">Muokkaa
21:             </button>
22:     </jx:if>
23:    </jx:forEach>
24:   </table>
25:  </form>
26: </form>
```

4.3 Web-palvelut

Web-palveluilla (Web Services) tarkoitetaan ohjelmakomponentteja, joita muut ohjelmat voivat kutsua Internetin välityksellä. Web-palvelut voivat vaihdella yksinkertaisista kyselyistä, kuten vaikkapa pörssikurssin haku, laajoihin järjestelmiin, kuten erilaisiin lipunvarausjärjestelmiin, jotka hyödyntävät eri tietojärjestelmistä saamaansa tietoa.

Palvelujen väliseen kommunikointiin tarvitaan eri standardeja koskevaa yhteisymmärrystä. UDDI (Universal Description, Discovery, and Integration) on standardi, jonka avulla palvelut voivat tuoda itsensä saataville. WSDL (Web Service Description Language) on toinen standardi, jolla palvelut kuvaavat transaktioitaan. Web-palvelujen viestit kulkevat useimmiten HTTP-protokollan välityksellä, ja ne ovat muodoltaan SOAP (Simple Object Access Protocol) [24] muotoisia. Yleisimmät Web-palveluiden toteutusympäristöt ovat kaupalliset Microsoft SOAP Client Toolkit, Microsoft .NET-Server ja Open Source -lisenssin alainen Apache Axis.

Apache Axis [AXIS] on ympäristö, jolla voidaan helposti toteuttaa SOAP-kielellä kommunikoivia ohjelmia, kuten asiakasohjelmia ja palvelimia. Palvelimia voidaan Axis:lla toteuttaa monella eri tavalla. Axis sisältää mm. apuohjelmat Java2WSDL, joka muodostaa Java-ohjelmista automaattisesti WSDL-kuvaukset. Yksinkertaisin tapa toteuttaa palvelin on nimetä itse tehty Java-tiedosto .jws-päätteeiseksi ja kopioida se tiettyyn hakemistoon, jolloin siitä muodostuu automaattisesti Web-palvelu. Näin luodun Web-palvelun kutsuttaviksi palveluiksi muodostuu automaattisesti kaikki tiedostossa määritellyt public-tyyppiset metodit. Tällöin rajoitteena on se, että sallittuina tietotyyppeinä, joita palvelin palauttaa ovat ainoastaan perustietotyypit, kuten merkkijonot, taulukot ja kokonaisluvut.

Seuraavassa on esimerkki palvelinohjelman queryRDF-nimisestä metodista, joka toimii Axis:ssa suoraan Web-palveluna, kunhan se sijaitsee virheettömässä Java-syntaksin mukaisessa .jws-päätteisessä tiedostossa. Metodi suorittaa RDQL-kyselyn Jena-malliin ja palauttaa tuloksen kutsujalle. Riveillä 3-7 muodostetaan RDQL-lause, johon sijoitetaan muuttujan sFrom arvo. Riveillä 9-24 suoritetaan RDQL-kysely Jena-malliin. Riveillä 26-28 muunnetaan Javan Vector-tyyppinen tietorakenne String-taulukoksi ja rivillä 30 palautetaan tulokset sisältävä taulukko kutsujalle.

```
1: public String[] queryRDF(String sFrom)
2: {
```

[24] http://ws.apache.org/soap/

```
3:      String queryString =
4:     ``SELECT ?x WHERE
5:      (?y, rdf:type, <http://ns/relation.owl'#Relation>),
6:      (?y, <http://ns/relation#from>, ''' + sFrom + ''`),
7:      (?y, <http://ns/relation#to>, ?x)'';
8:
9:      Query query        = new Query(queryString);
10:     query.setSource(relation_model);
11:     QueryExecution qe  = new QueryEngine(query);
12:     QueryResults results = qe.exec() ;
13:     QueryResultsFormatter fmt  = new QueryResultsFormatter(results);
14:
15:     Vector resultVector = new Vector();
16:
17:     for ( Iterator iter = results ; iter.hasNext() ; )
18:     {
19:     ResultBinding res = (ResultBinding)iter.next() ;
20:     Object obj = res.get("x") ;
21:     String value = (String)obj.toString();
22:     resultVector.add(value);
23:     }
24:     results.close();
25:
26:     String[] ret_array = new String[resultVector.size()];
27:     for (int i = 0; i < resultVector.size(); i++)
28:     ret_array[i] = (String)resultVector.elementAt(i);
29:
30:     return ret_array;
31: }
```

Seuraavassa on edellistä palvelua kutsuva asiakasohjelma. Metodi callQueryRDF kutsuu määrätyssä osoitteessa (rivi 3) sijaitsevan Web-palvelun queryRDF-metodia (rivi 8). Palvelun parametrin tyypiksi asetetaan merkkijono (rivi 9), ja paluuarvon tyypiksi merkkijonotaulukko (rivi 10). Riveillä 12-13 suoritetaan itse kutsu verkkopalvelulle ja palautunut merkkijonotaulukko palautetaan callQueryRDF-metodin kutsujalle (rivi 15).

```
1: public String[] callQueryRDF()
2: {
3:     String endpoint = "http://wrl-82.cs.helsinki.fi:8081/test/";
4:
5:     Service  service = new Service();
6:     Call call = (Call) service.createCall();
7:     call.setTargetEndpointAddress( new java.net.URL(endpoint) );
8:     call.setOperationName(``queryRDF''));
9:     call.addParameter ("op1", XMLType.XSD_STRING, ParameterMode.IN );
10:    call.setReturnType( XMLType.SOAP_ARRAY );
11:
12:    String[] ret = (String)call.invoke
13:        ( new Object[] { "http://ns/relation.owl#tol_22330" } );
14:    return ret;
15: }
```

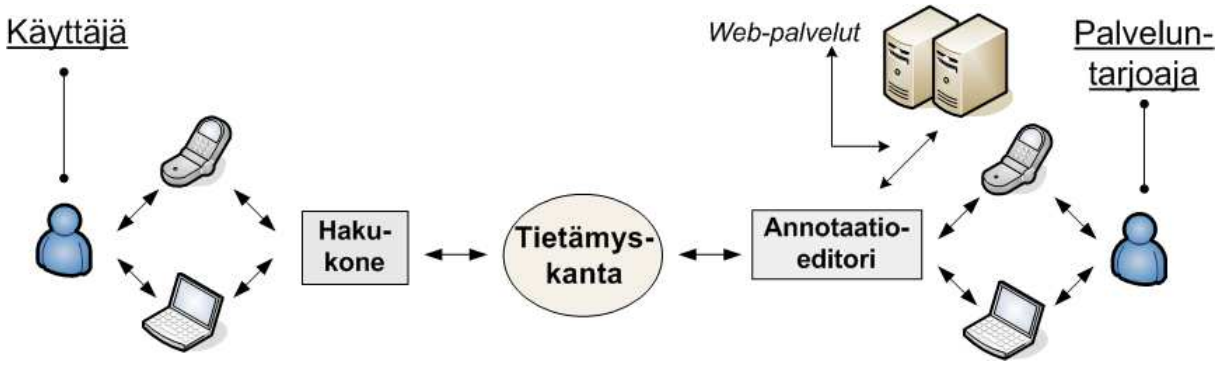
4.4 Testiaineisto

Editori toteutettiin osana IWebS (Intelligent Web Services)-projektia [25], joka on Helsingin yliopiston ja HIIT (Helsinki Institute for Information Technology):n yhteisen Semantic Computing -tutkimusryhmän [26] alainen kaksivuotinen projekti. Projekti käynnistyi syyskuussa 2002, ja sen rahoittajina toimivat Tekes, TeliaSonera Oy, Fonecta Oy, TietoEnator Oy, STAKES sekä Valtiovarainministeriö. Seuraavassa kuvataan perustiedot IWebS-järjestelmästä ja sen ontologioista.

Keltaiset sivut -tyyppiset hakemistopalvelut Internetissä ovat laajalti käytetty liiketoimintakonsepti, jonka tarkoitus on helpottaa yksityishenkilöitä löytämään palveluita ja tuotteita tarjoavia yrityksiä. Huolimatta lukuisista toteutusvaihtoehdoista voi yritysten löytäminen osoittautua vaikeaksi [GMV99, DKN+98, HVH02]. Toisaalta, palveluntarjoajan voi olla vaikea kuvata palvelu tavalla, jolla se olisi helposti löydettävissä. Ongelmat konkretisoituvat erityisesti tilanteissa, joissa loppukäyttäjä ei pysty tarkasti kuvaamaan tarvitsemaansa palvelua.

IWebS-projektin tavoitteena on tutkia mahdollisuutta hyödyntää semantic web ja web services [BHM+03] -tekniikoita sekä palveluilmoitusten annotaatiossa että käyttäjän tarvitsemien palveluiden haun helpottamisessa. Ideana on antaa järjestelmän

[25] http://www.cs.helsinki.fi/group/iwebs/
[26] http://www.cs.helsinki.fi/group/seco/



Kuva 12: IWebS-järjestelmän yleisarkkitehtuuri

käyttäjien kuvata palvelu termeillä ja käsitteillä, jotka ovat heille tuttuja. Järjestelmän on tarkoitus yhdistää nämä käsitteet käyttämiinsä ontologioihin. Järjestelmän yleisarkkitehtuuri esitetään kuvassa 12. Järjestelmän oleelliset osat ovat ontologioista sekä annotaatioista koostuva tietämyskanta, semanttinen hakukone palveluita hakevien loppukäyttäjien käyttöön sekä palveluntarjoajien käyttöön tarkoitettu semanttinen annotaatioeditori tarjottujen palveluiden lisäämistä varten.

Perinteisessä hakujärjestelmässä käyttäjän syöte kerätään implisiittisesti (käyttäjän konteksti ja profiili), eksplisiittisesti käyttäjän syöttämistä hakusanoista tai eksplisiittisesti navigointiin perustuvasta syötteestä. IWebS:n tapauksessa käyttäjän syötteeseen perustuen järjestelmän on kyettävä esittämään käyttäjän ongelma sellaisessa muodossa, että se voidaan ratkaista tarjottujen palveluiden avulla.

Ontoseek [GMV99] on järjestelmä, johon käyttäjä voi kuvata ongelmansa syöttäen erilaisia luonnollisen kielen termejä ja kuvaten termien välisiä suhteita leksikaalisina käsitekarttoina. Ontoseek hyödyntää haussa ontologioita, kuten WordNet:ä [Fel98] jolloin se kykenee löytämään myös annettujen termien synonyymeilla kuvattuja palveluilmoituksia. YPA-järjestelmä käyttää luonnollisen kielen prosessointia ja tiedon haun (information retrieval) tekniikoita etsiessään osittain rakenteisia palveluilmoituksia [DKN+98]. YPA:ssa käyttäjä syöttää järjestelmään kyselyitä, kuten “I need to get my camera repaired!”, joihin etsitään vastaus palveluilmoitusten ja WordNet-mallin [Fel98] avulla.

Sekä OntoSeek:n, että YPA:n avulla voidaan löytää mikä tahansa luonnollisella kielellä kuvattu ilmoitus. Mutta koska luonnollisen kielen tunnistus on erittäin vaikeaa ja virhealtista, voivat ilmoitukset jäädä löytymättä. OntoSeek käyttää leksikaalisia käsitekarttoja ongelmien esittämiseksi, mutta koska käsitteistö ja relaatiot voivat olla mitä tahansa, kyselygraafeja ei voida validoida mitenkään, minkä vuoksi kyselyt

voivat olla järjettömiä [GMV99].

IWebS-projektissa tavoitteena on käyttää rajoitettua joukkoa termejä ja relaatioita, jotka kuvataan ontologioissa, kyselyiden tekemistä ja palveluiden kuvaamista varten. Jos oikeat termit ja niiden väliset relaatiot löydetään, tämä auttaa sekä loppukäyttäjää että palveluntarjoajia tarjoamansa kuvaamisessa. Lisäksi, käytettäessä ontologioita käyttöliittymä voidaan rakentaa siten että se ohjaa käyttäjää semanttisesti oikeita valintoja kohden, kuten MuseoSuomi-järjestelmässä [HJK+04] on tehty.

IWebS-järjestelmän hakumoottori on rakennettu MuseoSuomi-kehyksen pohjalle. MuseoSuomen hakukäyttöliittymä perustuu moninäkymähakuun (multi view based search), jossa käyttäjä voi tehdä valintoja useista eri näkökulmista rajoittaen tulosjoukkoa eri näkökulmista. MuseoSuomi-järjestelmässä moninäkymähakua on laajennettu myös merkkijonoon perustuvaan hakuun, joka tarjoaa lisätavan määritellä rajoituksia kyselyyn.

IWebS:ssä palvelut pyritään kuvaamaan prosesseina, joilla on kohde (goal), tavoite (target), ja jotka toteutuvat tietyn ajan ja paikan puitteissa. Palvelut on IWebS-järjestelmässä kuvattu joukolla ontologioita. Ontologioita ovat kohde (goal), tavoite (target), palveluntarjoajan kuvaus, palvelukuvaus ja toimialaluokitus TOL [Sta02], sekä aika ja paikka. Kohde- ja tavoiteontologiat on tarkoitettu kuvaamaan loppukäyttäjän tarpeet. Tavoiteontologia koostuu kuluttajapalveluluokitus COICOP [Uni99], sekä projektin itse laatimasta elämänalueontologiasta. Palveluntarjoaja-, palvelukuvaus-, sekä luokitteluontologiat on yhdessä tarkoitettu kuvaamaan tarjottua palvelua.

Kohdeontologia koostuu abstrakteista käsitteistä, jotka ilmaisevat toimintoja kuten Kopioida, Tuottaa, Poistaa ja Siirtää. Kohde-ontologian termit määrittävät toimialueen tietyille alariippuvaisille termeille ja ovat tarkoitettu antamaan käyttäjälle keino kuvata palvelu “arkitietämystä” käyttäen. Käyttäjän ei tarvitse tietää mitään alariippuvaisia käsitteitä hakiessaan palveluja kohde-näkökulmasta. Hyvin samankaltaisia käsitteitä kuuluu SUMO:n (Standard Upper Merged Ontology) [NP01] prosessiontologiaan (process ontology). Nämä käsitteet on projektissa käännetty suomen kielelle järjestelmän kohde-ontologiaksi.

Esimerkikiksi käyttäjällä voisi olla ongelma, jossa hän haluaisi saada digitaalisessa muodossa olevan valokuvan tulostettua t-paitaan. Käyttäjä ei kuitenkaan tiedä mikä palveluntarjoaja (esim. kauppa tai tulostusliike) voisi tarjota hänelle halutun palvelun. Tavoitteena on, että käyttäjän ei tarvitse tietää, millä nimellä kuvan painamista t-paitaan pitäisi kutsua, vaan hän voisi kuvata palvelun määrittelemällä

haun kannalta oleelliset objektit, jotka tässä tapauksessa olisivat “kuva”, “painaa” ja “t-paita”. Lisäksi käyttäjä voisi määritellä, että palvelun on oltava saatavilla Helsingin ydinkeskustan alueella, ja että sen on oltava avoinna iltaisin.

Tuote- ja elämänalue- ontologiat on tarkoitettu kuvaamaan tarjottujen palvelujen kohteet. Tuotenäkökulman tarjoaa COICOP-ontologia. Korkein taso elämänalueontologiassa on Koti, Työ, Terveys, Koulutus, Viihde/Virkistys, Ulkonäkö, Omaisuus, Ruoka ja tarvikkeet sekä Sosiaaliset toiminnot.

Palveluntarjoaja-ontologian luokan instanssi voi olla mitä tahansa, joka voi tarjota yhden tai useamman palvelun. Palveluilmoitukset on mallinnettu tarjottujen palvelujen mukaan. Esimerkiksi parturiliike voi olla palveluntarjoaja, ja sillä on kaksi tarjottua palvelua: hiusten leikkaaminen sekä hiustenhoitotuotteiden myynti.

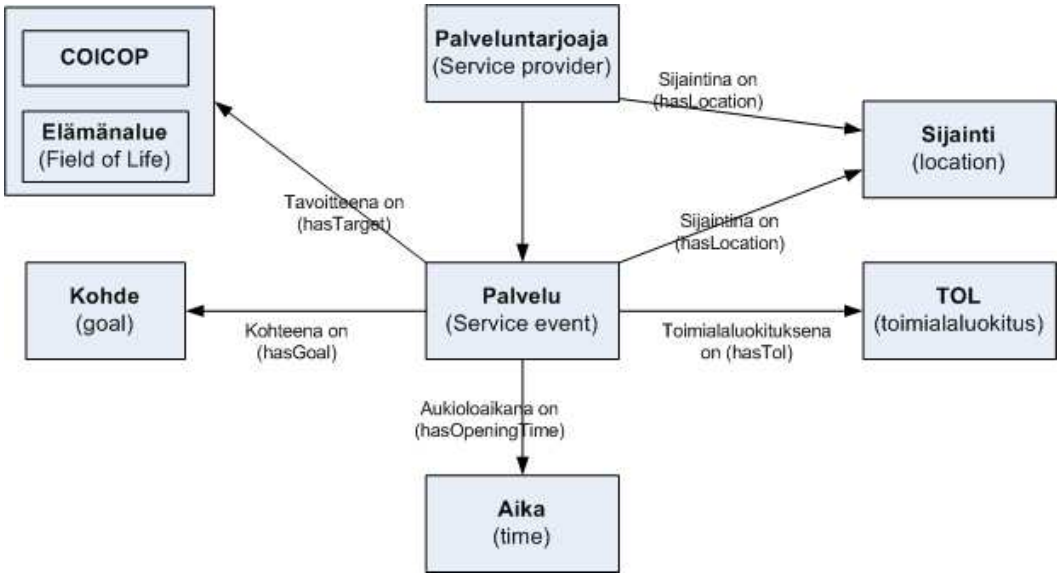
Kaikki ontologiat on kuvattu OWL-muodossa. Aikaontologia on IWebS-projektin itse laatima, ja sitä käytetään kuvaamaan esimerkiksi aukioloaikoja. Paikkaontologia on saatu MuseoSuomi-järjestelmästä [HJK+04].

Järjestelmän ontologioiden kytkeytyminen toisiinsa on nähtävissa kuvassa 13. Yhteen palveluntarjoajaan (service provider) liittyy yksi tai useampi tarjottu palvelu (service event), ja se sijaitsee tietyssä paikassa. Tarjottu palvelu sijoittuu myös tiettyyn paikkaan, joka voi olla eri kuin palveluntarjoajan sijainti. Lisäksi tarjottu palvelu on luokiteltu TOL-luokituksen mukaisesti, ja sillä voi olla kohde (goal) ja sen tavoitteena (target) voi olla joko COICOP- tai elämänalue-ontologian (field of life)-instanssi.

Annotaatioeditorin testausvaiheessa ontologioita on karsittu siten, että käytössä ovat ainoastaan palveluntarjoaja (service provider), palvelutapahtuma (service event), TOL-, kohde- (goal) ja tavoite- (target) ontologiat.

4.5 Yhteenveto

Järjestelmä päädyttiin toteuttamaan Apache Cocoon -kehyksessä käyttäen Cocoonin Control Flow -ominaisuutta, koska ohjelman mallintaminen tavallisena ohjelmana verrattuna muiden kehitysympäristöjen tapaan mallintaa Web-sovelluksen toiminta tila-automaattina katsottiin olevan huomattava etu ohjelman suunnittelun ja ylläpidettävyyden kannalta. Sivun näyttävän skriptin valinnassa päädyttiin Cocoonin sisältämiä eri vaihtoehtoja verrattaessa JXTemplate-skriptin käyttöön, koska sen käyttö todettiin helpoimmaksi sen sisältäessä kuitenkin kaikki toteutukseen vaadittavat ominaisuudet. Control Flow päädyttiin toteuttamaan Java-kielisenä ensin-



Kuva 13: IWebS-ontologiat ja niiden väliset relaatiot

näkin koska toisena vaihtoehtona olleeseen JavaScript-kieleen verrattuna Java on huomattavasti monipuolisempi, ja toiseksi muut toteutukseen valitut tekniikat, kuten RDF-käsittelyyn tarkoitettu Jena-paketti ja Axis:n Web-palvelukutsut toimivat parhaiten Java:lla.

Muihin RDF-käsittelyyn tarkoitettuihin tekniikoihin verratessa Jena osoittautui ylivoimaiseksi lähes kaikkien ominaisuuksiensa suhteen. Jenassa tietomalleja (model) voidaan säilyttää tiedostoissa, tietokannassa tai muistissa ja se tukee OWL-tietomallia, jolla testaukseen käytetyt ontologiat on kuvattu. Lisäksi editorin rajoitusominaisuuden ja muiden käyttäjien tekemiin annotaatioihin perustuvien suosittelujen hakuun RDQL-kyselykieli vaikutti sopivalta.

Järjestelmän annotaatiotietojen ja ontologioiden säilyttämiseen tarkoitettu palvelin katsottiin järkeväksi toteuttaa Web palvelut -tekniikoiden avulla. Ainoastaan Microsoft-yhtiön kaupalliset tuotteet olivat varteenotettavia vaihtoehtoja avoimen lisenssin Apache Axis-ympäristön rinnalla. Apache Axis on lisäksi monipuolinen ympäristö, jolla Web-palvelun pystyttäminen onnistuu pienellä vaivalla.

Edellämainittujen syiden perusteella annotaatioeditorin toteutusympäristöksi valittiin Java, Apache Cocoon, HP Labs:n Jena sekä Apache Axis.

Testiaineistoksi järjestelmälle saatiin IWebS-projektin käytössä oleva keltaiset sivut -tyyppisten palvelujen mallintamiseen tarkoitettu OWL-tietomalli. Editorin testaamista varten malliin kuuluvista ontologioista valittiin ainoastaan ne osat, joilla edi-

torin oleellisia vaatimuksia eli annotaatioiden täydentämistä, rajoittamista ja suosittelua katsottiin voitavat testata.

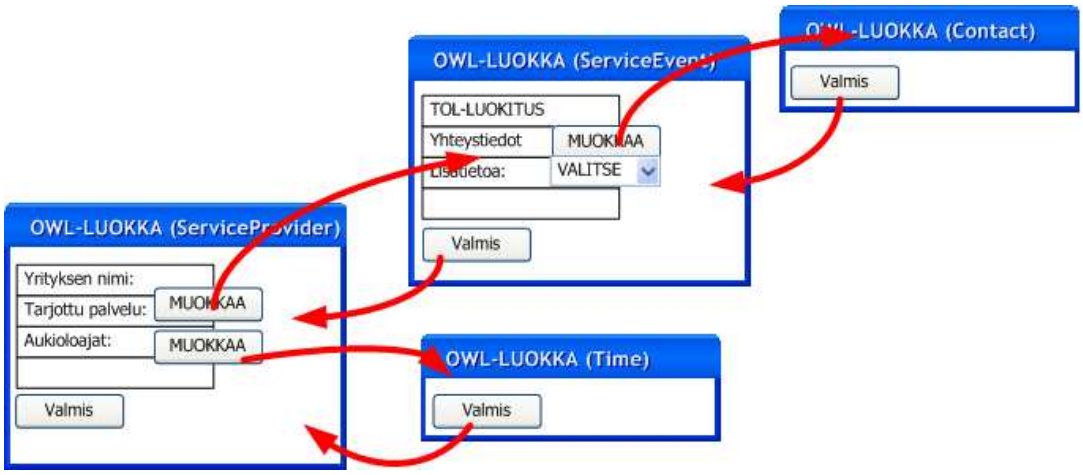
5 Laaditun annotaatioeditorin kuvaus

Tässä luvussa kuvataan suunnitelmat editorin laatimiseksi, editorin tekninen toteutus ja toimintaperiaate sekä testaus.

5.1 Lähtökohdat

Laadittavalle editorille asetetut vaatimukset olivat seuraavat: syntaktisesti ja semanttisesti oikeellisen RDF-metatiedon tuottaminen, annotaatioiden täydennys ja rajoitus-ominaisuus, muiden käyttäjien annotaatioihin perustuva suositteluominaisuus ja sopeutuminen erilaisiin aineistoihin eli geneerisyys. Lisäksi editorin vaatimuksiin kuului toiminta Internetissä ja että sen käsittelemät tiedot säilytetään keskitetysti palvelimella. Tällöin tiedot pysyvät yhdenmukaisina kaikkien ajossa olevien editorien kesken.

Suunnitelmissa päädyttiin toteuttamaan sellainen editori, jolla käyttäjä voi muokata yhden OWL-ontologian luokan tietoja kerrallaan ja siirtyä objektiarvoisten ominaisuuksien kohdalla muokkaamaan kyseisen OWL-luokan tietoja. Käyttäjä voisi siis vapaasti liikkua ontologioiden muodostamassa puuhierarkiassa täyttäen samalla annotaation vaatimia tietoja (katso kuva 14).

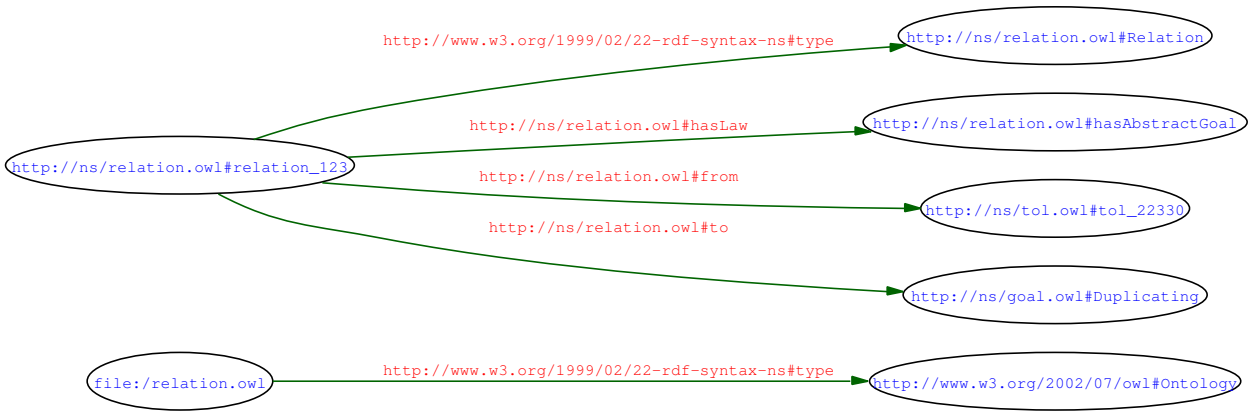


Kuva 14: Suunnitelma editorin toiminnasta, käyttäjä liikkuu OWL-luokkien välillä.

Koska ontologioiden objektiarvoisten ominaisuuksien kohdalla tulee tilanteita, joissa on mielekästä että ominaisuus valitaan joko olemassaolevien instanssien tai ontologian luokkien välillä, suunniteltiin editoriin mahdollisuus määrittää asetustiedostossa, minkälainen komponentti kunkin objektityyppisen ominaisuuden kohdalla käyttöliittymsäsä näytetään. Esimerkiksi käytetyn testiaineiston ServiceEvent-luokan yksi ominaisuus on toimialaluokitus. Tällöin ei ole mielekästä antaa käyttäjän luoda uutta toimialaluokitus-instanssia, vaan tarjota hierarkkinen lista, josta käyttäjä voi valita luokituksen olemassaolevien toimialaluokitus-instanssien joukosta.

5.1.1 Täydennys ja rajoitus

Täydennys ja rajoitusvaatimuksen täyttämiseksi editoriin suunniteltiin ominaisuus, joka toimii seuraavasti. Käyttäjän tehdessä valintoja yhden OWL-luokan ominaisuuksien sisällä, muita kyseisen luokan ominaisuuksien mahdollisia valintoja rajoitetaan erillisessä relaatio-ontologiassa määriteltyjen semanttisten suhteiden avulla. Samalla ominaisuuksiin, joihin käyttäjä ei ole vielä täyttänyt arvoa suositellaan relaatio-ontologian mukaisia arvoja. Jos käyttäjä esimerkiksi on luokittelemassa palvelunsa tarjoamaan vesikuljetuspalvelua, rajoittaa järjestelmä mahdollisen palvelun sijaintiluokituksen näyttämään ainoastaan veden äärellä sijaitsevat kohteet. Jotta järjestelmä osaisi tehdä kyseisen rajoituksen, täytyy relaatio-ontologiassa olla määritelty kaikki vesikuljetuspalveluun sopivat sijainnit.



Kuva 15: Esimerkki relaatio-ontologian instanssista.

Esimerkissä 15 kuvataan yksi relaatio-ontologian instanssi, jossa määritellään että ominaisuus “http://ns/tol.owl#tol_22330” liittyy ominaisuuteen “http://ns/goal.owl #Duplication”. “Tol_22330” tarkoittaa toimialaluokituksen luokkaa “Atk-tallenteiden

Kuva 16: Täydennys ja rajoitus

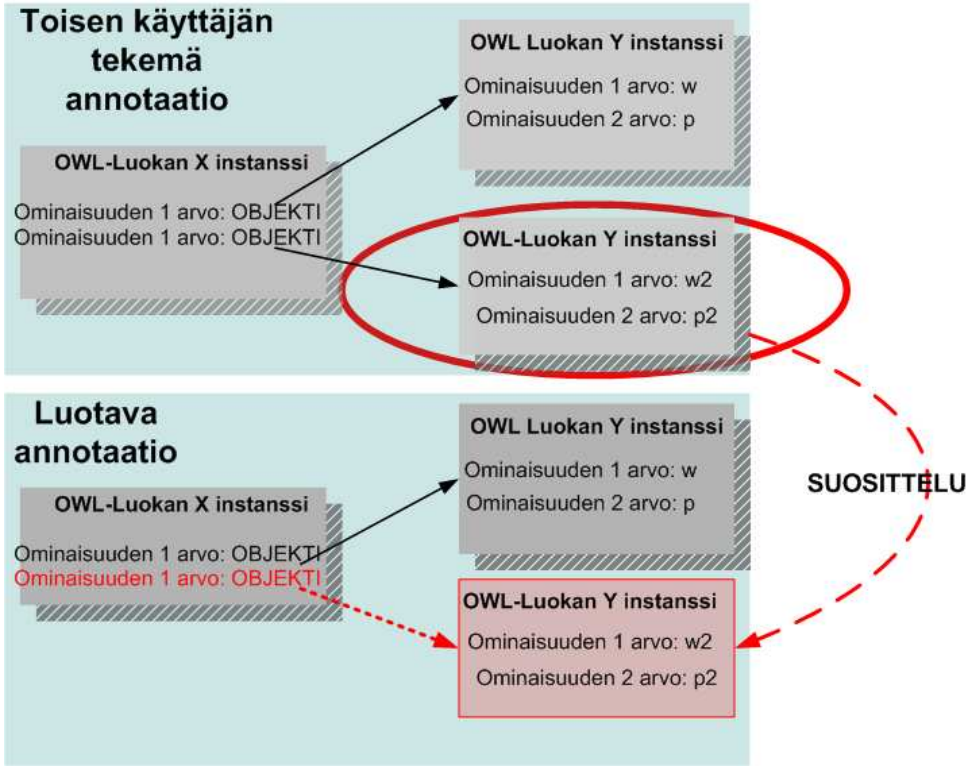
jäljentäminen", ja instanssi määrittelee siis, että se liittyy Goal-ontologiassa luokkaan "Duplicating" (kopioiminen). Jos editori on tilassa, jossa muokataan sellaista OWL-luokan instanssia, joka sisältää sellaiset objektityyppiset ominaisuudet, joiden sallittuina arvoalueina ovat TOL-ontologia sekä Goal-ontologia, niin käyttäjän valitessa TOL-ominaisuudeksi "tol_22330" editori suosittelisi Goal-ominaisuuden arvoksi "Duplicating".

Esimerkki edellä kuvatusta tilanteesta on kuvassa 16. Kuvan ensimmäisessä osassa (vasen ruutu) on alkutilanne, jossa käyttäjä on annotoimassa OWL-luokkaa, johon kuuluu objektityyppiset ominaisuudet TOL, GOAL ja TARGET. Toisessa osassa (keskimmäinen ruutu) käyttäjä valitsee TOL-hierarkiasta "Atk-tallenteiden jäljentäminen" (eli "tol_22330"). Kolmannessa osassa (oikeanpuoleinen ruutu) järjestelmä tämän seurauksena rajoittaa GOAL-hierarkiaa niin, että se sisältää ainoastaan relaatio-ontologiassa määritellyt sopivat instanssit, ja ehdottaa näistä ensimmäistä (ja tässä tapauksessa ainoaa) valintaa eli "Duplicating". Kannattaa huomioida, ettei kyseinen relaatio tässä esimerkissä koske ominaisuutta TARGET lainkaan. Esimerkissä oletetaan, että asetuksissa on määritelty ettei ominaisuus TARGET kuulu rajoitettaviin ominaisuuksiin (eikä siihen kyseisen tapauksen relaatio-ontologiassa liity yhtään relaatiota).

5.1.2 Suosittelu

Suositteluominaisuudella tarkoitetaan muiden käyttäjien tekemiin annotaatioihin perustuvaa suosittelua. Editorin suositteluominaisuus suunniteltiin siten että käyttäjän lisätessä OWL-luokan X objektityyppisen ominaisuuden arvoksi luokan Y instanssin, järjestelmä suosittelisi myös muita muiden käyttäjien järjestelmään tallentamia luokan Y instansseja, jotka kuuluvat sellaiselle luokan X instanssille, johon on liitetty samanlainen luokan Y instanssi kuin jonka käyttäjä syötti.

Esimerkiksi käyttäjän luokitellessa tarjotuksi palveluksi hiustenleikkuun, voisi järjestelmä suositella myös hiustenhoitotuotteiden myyntipalvelua, jos joku muu käyttäjä olisi luokitellut palvelun, joka tarjoaa sekä hiustenleikkuuta että hiustenhoitotuotteiden myyntiä. Kuva 17 havainnollistaa suositteluominaisuuden toimintaa.



Kuva 17: Suositteluominaisuus

5.1.3 Geneerisyys

Jotta editori olisi geneerinen, täytyy se voida asettaa toimimaan millä tahansa OWL-muotoisella aineistolla. Lisäksi käyttöliittymässä näytettäviä komponentteja, kuten hierarkisia listoja täytyy voida mukauttaa erilaisiin aineistoihin. Myös suositus-, ja täydennysominaisuuksien osalta täytyy voida asettaa minkä täydennettävän tai suositeltavan luokan ominaisuuksia käytetään suosittelussa ja rajoituksessa.

Editoriin suunniteltiin asetustiedosto, jossa voidaan asettaa editorin lukemien OWL-ontologioiden nimet sekä OWL-luokka josta annotaatio aloitetaan. Lisäksi asetustiedostossa voidaan määritellä miten luokan objektityyppisten ominaisuuksien kohdalla toimitaan. Alussa vaihtoehdoiksi asetettiin uuden instanssin luominen, jolloin käyttäjä voisi vapaasti luoda uuden kyseisen tyyppisen instanssin tai hierarkkinen pu-

dotuslista jossa näytetään ontologinen hierarkia. Ontologiset hierarkiat voivat myös muodostua eri tavoin, esim. OWL:n subClassOf-ominaisuuteen perustuen, tai RDF-instansseissa itse määritellyn ominaisuuden, kuten "hasParent"-ominaisuuden mukaisesti. Myös tämä on oltava määriteltävissä asetuksissa.

Jotta suositus- ja rajoitusominaisuudet toimisivat järkevästi, on asetuksissa voitava määritellä minkä luokan ominaisuuksien perusteella rajoituksia ja suosituksia tehdään. Esimerkiksi asetuksissa täytyy voida määritellä, että OWL-luokan "ServiceEvent" ominaisuuksia "tol", "goal" ja "target" rajoitetaan keskenään, ja että myös suosittelu tapahtuu näiden ominaisuuksien perusteella.

Asetuksissa on myös voitava määritellä kenttien järjestys käyttöliittymässä, sekä pakolliset kentät, joita käyttäjä ei saa jättää tyhjiksi. Esimerkki IWebS-testiaineistolle tehdystä asetustiedostosta on tutkielman liitteenä (liite 1). Asetustiedostossa määriteltävät oleelliset kohdat ovat:

ANNE_DATA Määrittelee käytettävät ontologiat.

ANNE_ROOT Määrittelee OWL-luokan, josta annotaatio aloitetaan.

CLASS_HIERARCHY, INSTANCE_HIERARCHY Määrittelevät, että halutun ominaisuuden kohdalla näytetään ontologinen hierarkia.

TITLE Määrittelee ominaisuuden otsikon.

SHOW Määrittelee, mitkä kentät OWL-luokasta näytetään.

ORDER Määrittelee, missä järjestyksessä ominaisuudet näytetään.

MANDATORY Määrittelee pakolliset kentät, joita käyttäjä ei saa jättää tyhjiksi.

ROOTURI, FOLLOWPROP, DISPLAYPROP Määrittelevät, millä perusteella ontologinen hierarkia muodostetaan.

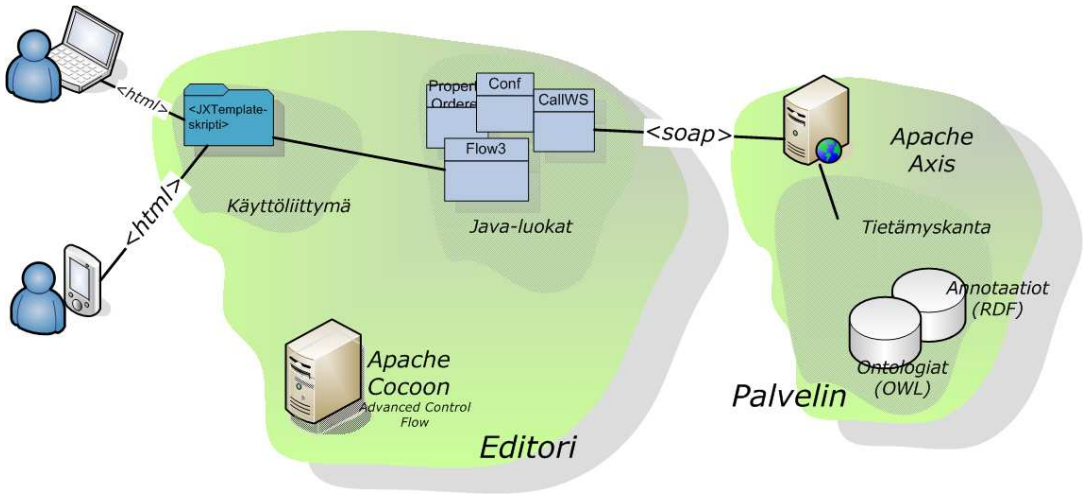
RESTRICT, PROPOSE Määrittelevät, miten suosittelu ja täydennysominaisuudet toimivat tietyssä OWL-luokassa.

5.1.4 Editorin ja palvelimen yhteistoiminta

Editorin ja palvelimen yhteistoiminta suunniteltiin siten, että kaikki tieto tallennetaan palvelimelle, jolle editori tekee kyselyjä. Myös relaatio-ontologiaan ja muiden käyttäjien annotaatioihin liittyvät kyselyt tapahtuvat palvelinpäässä.

5.2 Toteutus

Tässä aliluvussa kuvataan editorin tekninen toteutus. Editorin arkkitehtuuri (kuva 18) jakautuu varsinaiseen editoriosioon ja palvelinosioon. Editoriosio on toteutettu Apache Cocoon -kehyksellä käyttäen Javaflow-tekniikkaa, ja se jakautuu editorin toiminnallisuuden sisältäviin Java-luokkiin sekä käyttöliittymään, joka muodostetaan JXTemplate-skriptien avulla.



Kuva 18: Editorin arkkitehtuuri

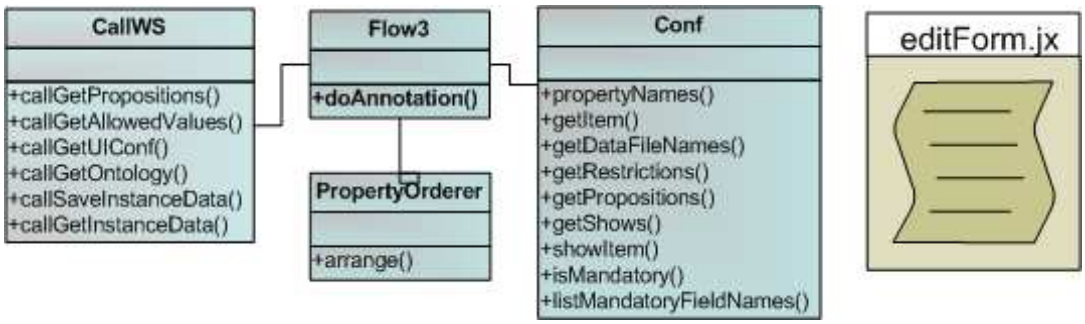
Palvelinosio on toteutettu web-palveluna Apache Axis:ta käyttäen. Palvelimen tehtävä on säilyttää ontologioita, relaatio-ontologiaa, luotuja annotaatioita ja editorin asetustiedostoa sekä tarjota kutsurajapinta, jolla editori pääsee näihin tietoihin käsiksi. Sekä editori- että palvelinosio käyttävät RDF-tiedon käsittelyyn HP Labs:n Jena-pakkausta.

5.2.1 Editori

Editoriosio koostuu toiminnallisesta osiosta, joka on toteutettu Java-kielellä sekä käyttöliittymän näyttävästä JXTemplate-skripteillä toteutetusta osiosta. Editoriosion luokkakaavio esitetään kuvassa 19. Toiminnalliseen osioon kuuluu Java-luokat Flow3, Conf, CallWS sekä PropertyOrderer. Flow3 on editorin pääluokka, jolle kontrolli siirtyy otettaessa selaimella yhteys Cocoon-palvelimeen. Flow3 pyytää CallWS ja Conf-luokilta ohjelman vaatimat ontologiat ja asetustiedot, muodostaa niiden perusteella käyttöliittymän tarvitsemat tiedot ja siirtää kontrollin käyttöliittymälle. Palatessa käyttöliittymältä Flow3 käsittelee käyttäjän käyttöliittymälle syöttämät

tiedot, tekee mahdollisesti ontologioihin liittyviä lisäkyselyitä ja muodostaa saatujen tietojen perusteella RDF-metatiedon.

CallWS-luokka vastaa kutsuista editorin palvelinosiolle. Luokka sisältää rajapinnan, jota kutsumalla voidaan pyytää palvelimelta editorin asetustiedosto, halutut ontologiat ja tehdä sekä relaatio-ontologiaan että muiden käyttäjien tekemiin annotaatioihin liittyviä kyselyitä.



Kuva 19: Editorin luokkakaavio

Conf on asetustiedoston käsittelyä varten laadittu luokka ja PropertyOrderer on pieni apuluokka jolla voidaan järjestää OWL-luokan ominaisuudet haluttuun järjestykseen. Luokkaa käytetään haluttaessa näyttää käyttöliittymässä OWL-luokan ominaisuudet tietyssä järjestyksessä.

Käyttöliittymä muodostetaan kahden JXTemplate-skriptin avulla: editForm sekä result. Editform-skripti on editorin käyttöliittymän pääskripti.

Seuraavassa kuvataan editoriosion toiminta korkealla tasolla.

1. Editori pyytää palvelimelta asetustiedoston ja tarvitsemansa ontologiat.

2. Editori katsoo asetustiedostosta, mistä ontologian luokasta annotaatio aloitetaan, hakee ontologiasta kyseisen luokan ja lähettää kontrollin käyttöliittymälle.

3. Käyttäjälle näytetään HTML-lomake, joka sisältää kyseisen ontologian luokan mukaiset kentät. Objekti-tyyppisten ominaisuuksien kohdalla näytetään asetustiedoston mukaisesti joko nappi, jolla käyttäjä voi luoda uuden kyseisen tyyppisen RDF-instanssin, tai pudotuslista, jossa näytetään asetusten mukaisesti tietty ontologinen puuhierarkia.

4. Jos käyttäjä luo luokan ominaisuudeksi uuden instanssin, kutsutaan rekursiivisesti kohtaa 2, jolloin käyttäjälle näytetään painamansa ominaisuuden luokan ominaisuuksia vastaava lomake.

5. Jos ollaan palaamassa rekursiivisesta kutsusta, tehdään muiden käyttäjien tekemiin annotaatioihin perustuva suosituskysely. Jos kysely tuottaa tuloksia, niin ehdotetaan käyttäjälle suositeltuja instansseja.

6. Jos käyttäjä on tehnyt valintoja hierarkkisista listoista, tarkistetaan rajoitteet, rajoitetaan muita samassa luokassa olevia listoja sen mukaisesti ja palataan kohtaan 3.

7. Kun käyttäjä on valmis, luotu RDF-metatieto lähetetään palvelimelle tallennettavaksi.

Käyttäjälle näytetään siis aina kerrallaan yhden ontologian luokan tiedot. Käyttäjä etenee luokkien muodostamassa puurakenteessa ja täyttää vaaditut tiedot editorin muodostaessa samalla semanttista RDF-metatietoa, joka lopulta käyttäjän ollessa valmis lähetetään palvelimelle tallennettavaksi.

Seuraavassa havainnollistetaan edellä esitetty toiminta esimerkin avulla. Kuvassa 20 on OWL-muotoinen kuvaus ServiceProvider eli palveluntarjoaja-luokasta. Luokalla on ominaisuudet contact (yhteystiedot), name-fi(yrityksen nimi), ytunnus (yrityksen y-tunnus) ja serviceEvent (palvelun kuvaus). Ominaisuudet “name-fi” ja “y-tunnus” (rivit 29-39) saavat arvokseen merkkijonot. Ominaisuus “contact” saa arvokseen “Contact”-tyyppisen objektin, ja “serviceEvent” saa arvokseen “ServiceEvent”-tyyppisen objektin (kts. rivit 26 ja 20).

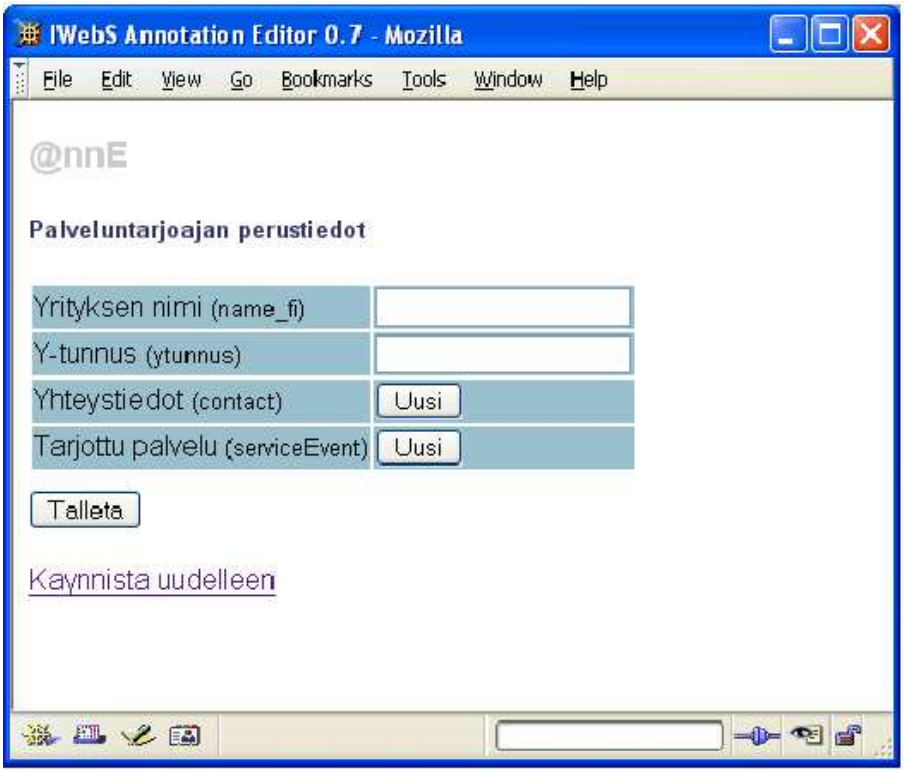
Kuvassa 21 on editorin kyseisen OWL-kuvauksen perusteella muodostama käyttöliittymä. Merkkijonotyyppisten ominaisuuksien kohdalla näytetään käyttäjälle tekstikenttä. Objektityyppisten ominaisuuksien kohdalla näytetään nappi, josta käyttäjä voi luoda kyseisen tyyppisen uuden ominaisuuden. Asetustiedostossa oltaisiin voitu myös määritellä, että käyttäjä voi valita objektityyppisen instanssin tietyin perustein muodostettavasta listasta.

Kuvassa 22 on oleellinen osa luokan ServiceEvent (palvelun kuvaus) OWL-kuvauksesta. Luokalla on tekstityyppinen ominaisuus “paymentMethod” (maksuväline) sekä objektityyppiset ominaisuudet “tol” (toimialaluokitus), “target” (palvelun tavoite) ja “goal” (palvelun kohde).

```
1: <?xml version="1.0"?>
2: <!DOCTYPE rdf:RDF [
3: <!ENTITY loc 'http://www.cs.helsinki.fi/group/iwebs/ns/location.owl#'>
4: <!ENTITY se 'http://www.cs.helsinki.fi/group/iwebs/ns/serviceevert.owl#'>
5: ]>
6: <rdf:RDF
7:     xmlns="http://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#"
8:     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
9:     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
10:     xmlns:owl="http://www.w3.org/2002/07/owl#"
11:     xmlns:loc="http://www.cs.helsinki.fi/group/iwebs/ns/locatior.owl#"
12:     xmlns:se="http://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#"
13:   xml:base="http://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl">
14:   <owl:Ontology rdf:about=""></owl:Ontology>
15:  <owl:Class rdf:ID="ServiceProvider">
16:     <rdfs:label xml:lang="en">Service provider</rdfs:label>
17:     <rdfs:label xml:lang="fi">Palveluntarjoaja</rdfs:label>
18:   </owl:Class>
19:   <owl:ObjectProperty rdf:ID="serviceEvent">
20:     <rdfs:range rdf:resource="&se;ServiceEvent"/>
21:     <rdfs:domain rdf:resource="#ServiceProvider"/>
22:     <rdfs:label xml:lang="fi">Tarjottava palvelu</rdfs:label>
23:   </owl:ObjectProperty>
24:   <owl:ObjectProperty rdf:ID="contact">
25:     <rdfs:domain rdf:resource="#ServiceProvider"/>
26:     <rdfs:range rdf:resource="#Contact"/>
27:     <rdfs:label xml:lang="fi">Yhteystiedot</rdfs:label>
28:   </owl:ObjectProperty>
29:   <owl:DatatypeProperty rdf:ID="name_fi">
30:     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
31:     <rdfs:domain rdf:resource="#ServiceProvider"/>
32:     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
33:     <rdfs:label xml:lang="fi">Yrityksen nimi</rdfs:label>
34:   </owl:DatatypeProperty>
35:   <owl:DatatypeProperty rdf:ID="ytunnus">
36:     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
37:     <rdfs:domain rdf:resource="#ServiceProvider"/>
38:     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
39:   </owl:DatatypeProperty>
40: </rdf:RDF>
```

Kuva 20: ServiceProvider-luokan RDF-kuvaus

Kuvassa 23 on editorin kyseisestä luokasta muodostama lomake. Huomioitavaa on, että toisin kuin kuvassa 21, ei objektityyppisten kenttien kohdalla näytetä “Uusi”-nappia, vaan asetustiedostossa määritellyn tyyppiset hierarkkiset pudotuslistat. Käyttäjän tehdessä valintoja listoista, esimerkiksi valitessa “Palvelun tavoite”-listasta tietyn luokituksen, tekee järjestelmä rajoitehaun ja rajoittaen muita kyseisessä lomakkeessa olevia listoja näyttää ainoastaan rajoiteontologiassa määritellyt sallitut arvot. Karsitut versiot ontologioista sekä kyseisillä ontologioilla toimimaan asetettu versio asetustiedostosta löytyvät tutkielman liitteistä (liitteet 1, 2, 3, 4).



Kuva 21: ServiceProvider-luokka käyttöliittymässä

5.2.2 Palvelin

Palvelinosio on tarkoitettu säilyttämään editorin tarvitsemia asetustietoja, ontologioita, annotaatioinstansseja ja relaatio-ontologiaa sekä tarjoamaan rajapinnan jonka kautta editoriosio voi pyytää siltä haluamiaan tietoja. Palvelinosio on toteutettu Apache Axis:lla.

Palvelimen rajapinta tarjoaa seuraavat palvelut:

Palvelu getUIConf Palauttaa kutsujalle editorin asetustiedoston.

Palvelu getOntology Palauttaa kutsujalle parametrina määritellyn nimisen ontologian.

Palvelu getInstanceData Palauttaa kutsujalle kaikki palvelimelle talletetut annotaatioinstanssit.

Palvelu getAllowedValues Palauttaa halutulle luokan ominaisuudelle kaikki relaatio-ontologiassa määritellyt sallitut arvot parametrina annettujen muiden saman luokan ominaisuuksien arvojen ollessa voimassa.

```
31:  <owl:Class rdf:ID="ServiceEvent">
32:     <rdfs:label xml:lang="fi">Palvelutapahtuma</rdfs:label>
33:     <rdfs:label xml:lang="en">Service event</rdfs:label>
34:   </owl:Class>
35:   <owl:ObjectProperty rdf:ID="tol">
36:     <rdfs:range rdf:resource="&tol;TolTaxonomyItem"/>
37:     <rdfs:domain rdf:resource="#ServiceEvent"/>
38:     <rdfs:label xml:lang="fi">Toimialaluokitus</rdfs:label>
39:   </owl:ObjectProperty>
40:  <owl:ObjectProperty rdf:ID="target">
41:     <rdfs:domain rdf:resource="#ServiceEvent"/>
42:    <rdfs:range rdf:resource="&coicop;CoicopTaxonomyItem"/>
43:     <rdfs:label xml:lang="fi">Palvelun tavoite</rdfs:label>
44:   </owl:ObjectProperty>
45:   <owl:ObjectProperty rdf:ID="goal">
46:     <rdfs:domain rdf:resource="#ServiceEvent"/>
47:     <rdfs:range rdf:resource="&goal;Tavoite_1"/>
48:     <rdfs:label xml:lang="fi">Palvelun kohde</rdfs:label>
49:   </owl:ObjectProperty>
50:   <owl:DatatypeProperty rdf:ID="paymentMethod">
51:     <rdfs:domain rdf:resource="#ServiceEvent"/>
52:     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
53:     <rdfs:label xml:lang="fi">Maksuvalineet</rdfs:label>
54:   </owl:DatatypeProperty>
```

Kuva 22: ServiceEvent-luokan RDF-kuvaus

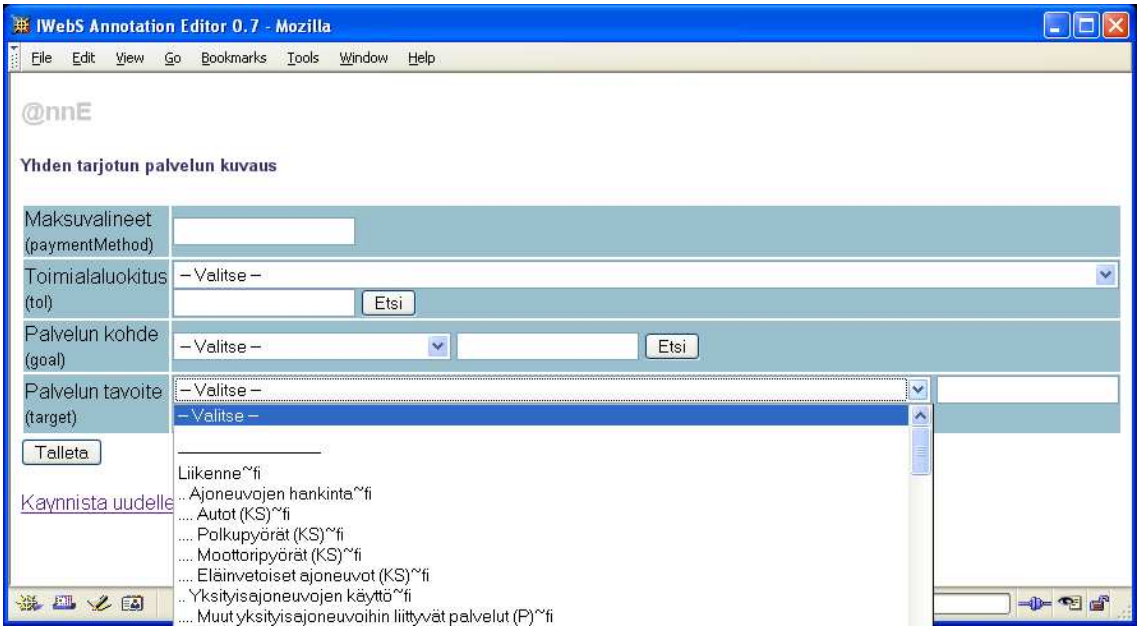
Esimerkkinä voidaan pyytää palvelimelta kaikki sallitut arvot ominaisuudelle "tol" (toimialaluokitus), kun ominaisuus "target" (palvelun kohde) =X sekä goal (palvelun tavoite) =Y.

Palvelu getPropositions Saa parametrinaan haettavan instanssin nimen, haettavien ominaisuuksien nimet ja haettavien ominaisuuksien arvot.

Tekee RDQL-kyselyn instanssitietokantaan hakien kaikki sellaiset muut instanssit, jotka liittyvät sellaiseen instanssiin, joka sisältää samat arvot kuin parametreina annetut. Palvelua käytetään muiden käyttäjien tekemien annotaatioiden suositteluun.

Palvelu saveInstanceData Tallettaa parametrina annetun instanssidatan annotaatiotietokantaan.

Palvelimen toiminnallisuus on toteutettu Java-kielellä, kaikki RDF-käsittely ja RDQL-kyselyt on toteutettu HP Labs:n Jena-pakkauksella. Jena:ssa RDF-mallit (model) voidaan säilyttää joko tiedostoissa, tietokannassa tai muistissa. Koska testiaineiston laajuus ei ole vaatinut tietokantapohjaista ratkaisua, on aineisto säilytetty tie-

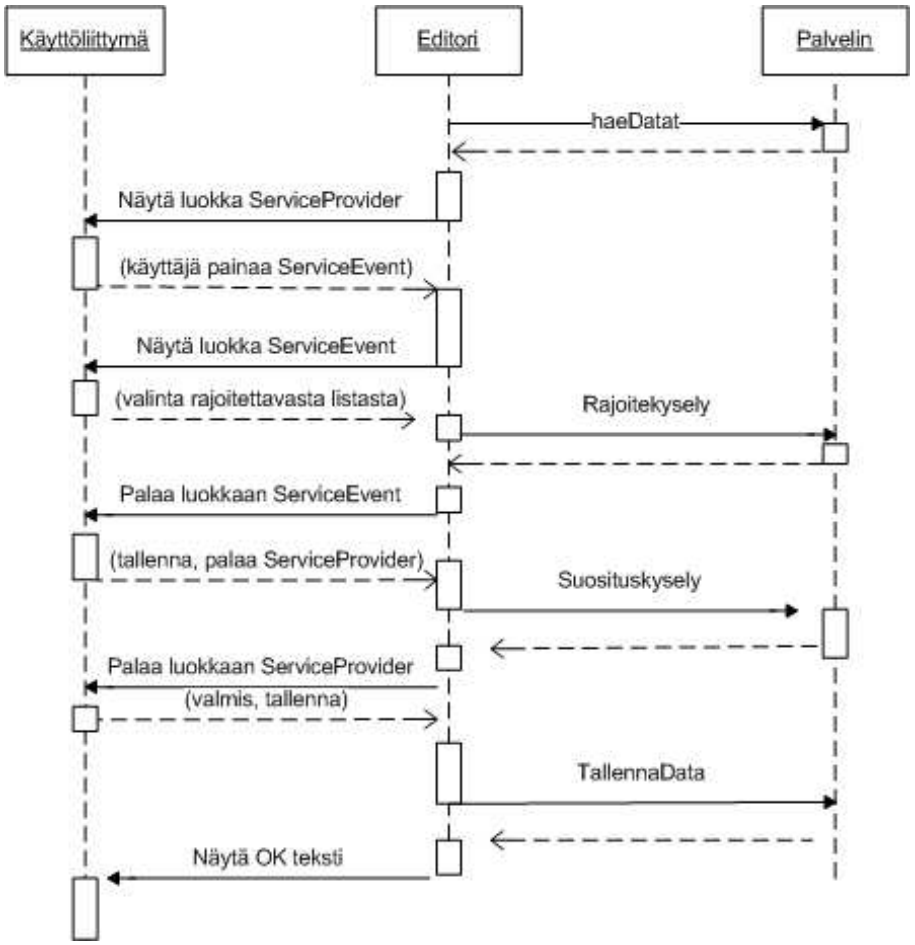


Kuva 23: ServiceEvent-luokka käyttöliittymässä

dostoissa. Aineisto on hyvin helppo siirtää tietokantaan, eikä palvelin vaadi kuin minimaaliset muutokset siirryttäessä käyttämään tietokantaa.

Seuraavassa näytetään kahden esimerkin avulla, miten rajoite- ja täydennyskyselyt toimivat. Oletetaan, että kutsuja haluaa kaikki sallitut arvot “ServiceEvent”-tyyppisen instanssin kentälle “tol”, kun “target”= “Varallisuus” ja “goal”= “Vaihtaa”. Parametrina getAllowedValues-palvelulle välitetään tällöin tieto siitä, mitä ominaisuutta haetaan, eli “tol”, joukko (ominaisuus,ominaisuuden arvo)-pareja sekä instanssin nimi, tässä tapauksessa “ServiceEvent”. Paluuarvona saadaan joukko “tol”:lle sallittuja arvoja.

Toisessa tapauksessa halutaan tietää kaikki muut sellaisille ServiceProvider-instanssille liittyvät ServiceEvent-instanssit, joihin liittyy ServiceEvent-instanssi jonka ominaisuuksien “tol”, “target” ja “goal”, arvoina ovat “Parturi-kampaamot”, “Välittää”, ja “Hiustenhoitoaineet”. Tällöin kutsutaan getPropositions-palvelua, jolle välitetään arvona luokka, johon liitettyjä instansseja haetaan eli “ServiceProvider”, sen instanssin nimi, jota haetaan eli “ServiceEvent”, ja mainittu joukko ominaisuuden (nimi+arvo)-pareja. Paluuarvoina saatuja instansseja suositellaan käyttäjälle.



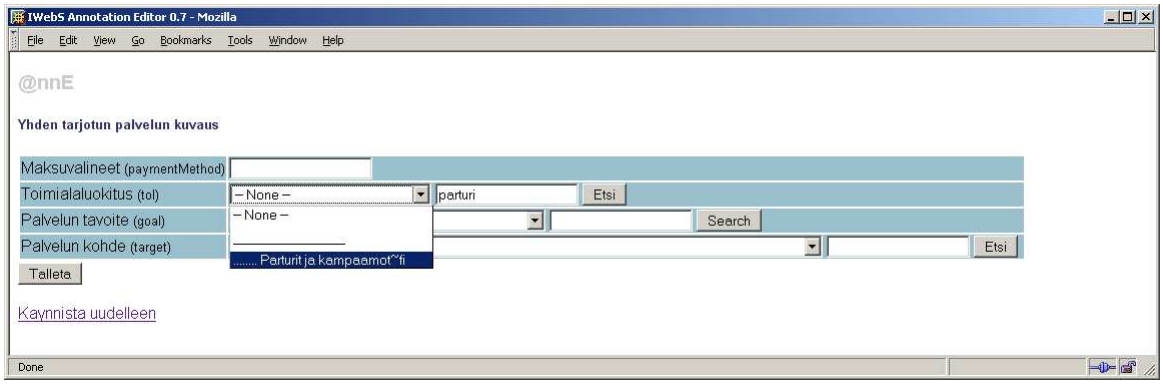
Kuva 24: Sekvenssikaavio

5.2.3 Koko järjestelmän toiminnan kuvaus

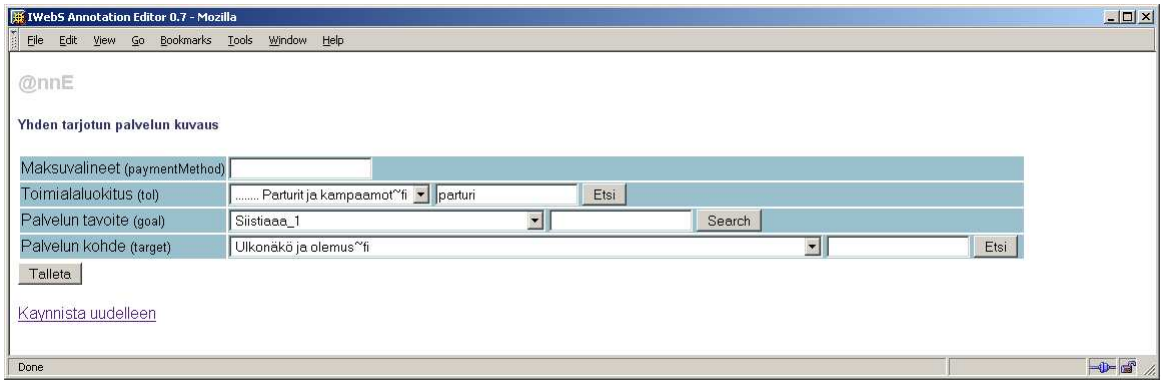
Kuvassa 24 on järjestelmän toimintaa kuvaava sekvenssikaavio, jossa kuvataan eräs mahdollinen käyttäjän tekemä annotaatio järjestelmällä. Sekvenssi alkaa sillä, että Editori pyytää Palvelimelta asetustiedoston sekä ontologiat. Tämän jälkeen kontrolli lähetetään käyttöliittymälle, jota pyydetään näyttämään käyttäjälle asetustiedostossa määritellyn annotaation aloitusluokka eli ServiceProvider. Kuva 21 vastaa käyttäjälle tässä vaiheessa näytettävää käyttöliittymää. Seuraavaksi käyttäjä painaa käyttöliittymästä serviceEvent-ominaisuuden kohdalla “Uusi”-nappia, jolloin kontrolli palaa Editorille. Editori lähettää kontrollin takaisin käyttöliittymälle, pyytäen tätä näyttämään ServiceEvent-luokan. Seuraaavaksi käyttäjä tekee valinnan listasta, joka on asetuksissa määritetty rajoitettavaksi listaksi. Tämän seurauksena kontrolli siirtyy Editorille, joka tekee rajoitekyselyn palvelimelle. Palvelimen vas-

tattua kontrolli lähetetään takaisin käyttöliittymälle pyytäen tätä palaamaan äskeiseen ServiceEvent-näkymään, mutta rajoittaen luokan ominaisuuksia halutulla tavalla. Seuraavaksi käyttäjä tallentaa ServiceEvent-instanssin, jolloin kontrolli palaa Editorille. Editori tekee tässä vaiheessa Palvelimelle suosituskyselyn koskien muiden käyttäjien vastaavalla tavalla tekemiä annotaatioita. Seuraavaksi kontrolli palautetaan Käyttöliittymälle, jota pyydetään palaamaan ServiceProvider-näkymään, ja näyttämään mahdollisesti löytyneet suositukset. Kuva 27 on esimerkki siitä, miltä käyttöliittymä tässä vaiheessa voi näyttää. Seuraavaksi käyttäjä tallentaa koko luomansa annotaation, jolloin kontrolli siirtyy Editorille, joka lähettää annotaatiotiedon Palvelimelle tallennettavaksi ja tämän jälkeen lähettää käyttöliittymälle tiedon talletuksen onnistumisesta.

5.3 Testaus



Kuva 25: Käyttäjä valitsee TOL-hierarkiasta “Parturit- ja kampaamot”

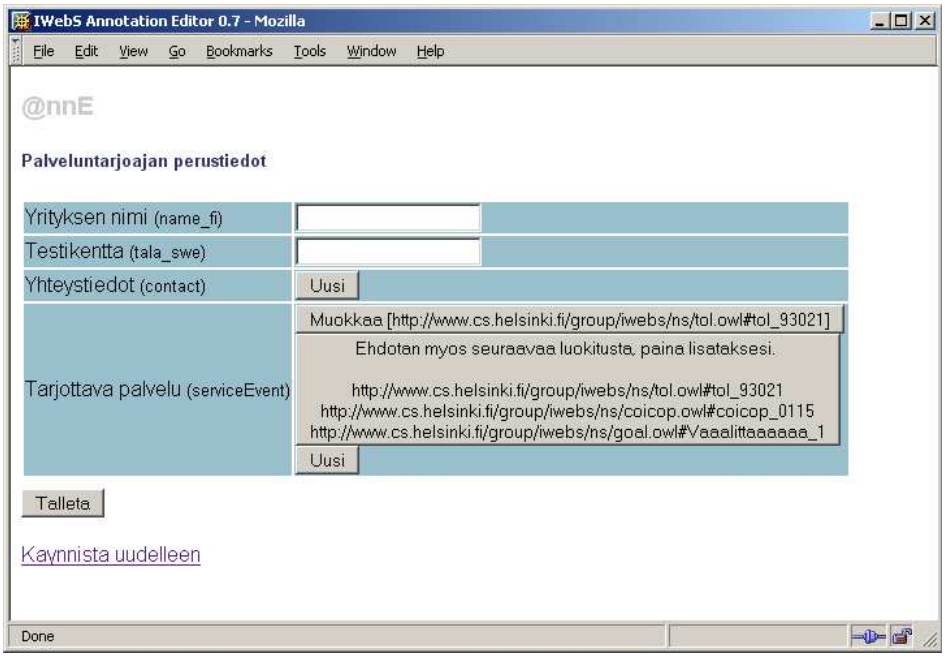


Kuva 26: Järjestelmä täydentää kohde- ja tavoite- hierarkiat.

Editoria käytettävyystestattiin muutamilla testikäyttäjillä käyttäen käyttötapausta,

jossa käyttäjiä pyydettiin lisäämään “parturi-kampaamo”-tyyppinen palveluilmoitus järjestelmään. Käyttäjille annettiin editorin käytöstä lyhyt käyttöohje.

Useimmat käyttäjät kykenivät annotoimaan palvelun järjestelmään toimintasekvenssin ollessa kullakin käyttäjällä jokseenkin seuraavanlainen: aluksi käyttäjä syötti yrityksen perustiedot editorin alkusivulle, jonka jälkeen käyttäjät siirtyivät lisäämään yrityksen tarjoamia palveluita valiten TOL-hierarkiasta “Parturit ja kampaamot” (kts. kuva 25). Tämän jälkeen järjestelmä rajoitti muita hierarkioita (goal, target) siten, että järjestelmä ehdotti tavoitehierarkiaan “Siistiä” ja kohde-hierarkiaan “Ulkonäkö ja olemus” (kts. kuva 26).



Kuva 27: Järjestelmä suosittelee uutta luokitusta.

Tämän jälkeen käyttäjät palasivat yrityksen perustiedot näyttävälle lomakkeelle, jossa järjestelmä ehdotti luokitukseksi myös hiustenhoitotuotteiden myyntipalvelua, perustuen järjestelmään valmiiksi annotoituun tapaukseen (kts. kuva 27).

Käytettävyystestin perusteella voidaan sanoa, että editorin ohjaus, suositus ja pakotusominaisuudet toimivat. Testauksessa käyttäjät pitivät hieman kummallisena monia siirtymiä sivuilta toisille. Lisäksi hierarkialistat koettiin hankaliksi käyttää, eikä yhdestä listasta tehdyn valinnan perusteella automaattisesti muihin listoihin tehtyä rajoitusta aina ymmärretty. Järjestelmän suositusominaisuuden yhteydessä käyttäjät kommentoivat sitä, että järjestelmä näyttää käyttäjälle turhaa teknistä tietoa, kuten instanssien URI-tietoja, jotka koettiin hämmentäviksi. Parannusehdo-

tuksina näihin käytettävyysongelmiin ehdotetaan käyttöliittymän uudelleensuunnittelua siten, että järjestelmä voitaisiin asettaa näyttämään yhdellä käyttöliittymän lomakkeella tietoja monista ontologian luokista. Hierarkkisten listojen näyttämiseen tarvitaan parempi käyttöliittymäkomponentti. Järjestelmään pitäisi myös lisätä enemmän käyttäjää opastavia tiedotuksia järjestelmän tehdessä rajoituksia, sekä poistaa käyttäjälle turhat ja tätä hämmentävät tekniset tiedot kuten URI-osoitteet käyttöliittymästä.

Järjestelmässä havaittiin myös muutamia pieniä teknisiä ongelmia mm. skandinaavisten merkkien näyttämisessä. Lisäksi havaittiin ongelma, jossa editorin alla toimiva Apache Cocoon-ympäristö kaatui jätettäessä editori pitkäksi aikaa käyttämättä. Ympäristön kaatumisongelmaan ei löytynyt ratkaisua.

Testiä arvioitaessa on otettava huomioon, että käyttäjät olivat poikkeuksetta tietotekniikkaa tuntevia henkilöitä. Lisäksi testaukseen käytetty aineisto on varsin pieni ja testitapausta varten järjestelmään oli lisätty tapaukseen sopivat relaatiot ja suositusinstanssit. Järjestelmän suositus- ja täydennysominaisuudet toimivat siis ainoastaan, jos järjestelmään on syötetty sopivia relaatioita ja suositusinstansseja. Jotta järjestelmän toimivuudesta voitaisiin vakuuttua, sitä täytyisi testata isommilla aineistoilla sekä erilaisilla käyttäjäryhmillä.

6 Tulosten tarkastelu

Tässä luvussa arvioidaan esitettyjä ratkaisuja. Ensin arvioidaan laadittua editoria (luku 6.1), sitten esitetään koko tutkimuksen tulokset (luku 6.2). Tämän jälkeen esitetään pohdintoja editorin jatkokehityksestä (luku 6.3).

6.1 Laaditun editorin arviointi

Laadittu editori soveltuu tällä hetkellä tarkoitukseen, jossa järjestelmään voidaan syöttää semanttisesti luokiteltua RDF-muotoista tietoa, kuten palveluilmoituksia. Seuraavaksi arvioidaan editorille asetettujen tavoitteiden täyttymistä, editorin käytettävyyttä sekä teknisiä ratkaisuja.

6.1.1 Tavoitteiden täyttyminen

Tavoitteena oli laatia annotaatioeditori, joka olisi käytettävä semantiikkaa ja tekniikkaa tuntemattomille henkilöille. Editorin oleellisimmat vaatimukset, eli syntaktisesti ja semanttisesti oikeellisen metatiedon tuottaminen, annotaation rajoitus- ja suositteluominaisuudet sekä geneerisyys saatiin täytettyä.

Editorin rajoitus- ja suositusominaisuudet suunniteltiin toimimaan ainoastaan yksittäisen OWL-luokan ominaisuuksien suhteen. Tämä havaittiin ongelmaksi siinä vaiheessa, kun IWebS-tietomallia suunniteltiin uudestaan siten, että palvelun kohdeja tavoitehierarkiat olisivat sijainneet eri OWL-luokissa. Testauksessa käytetyssä aineistossa molemmat sijaitsivat samassa tarjottua palvelua kuvaavassa OWL-luokassa ServiceEvent. Sama pätee myös suositushakulogiikkaan, jossa yhdessä haussa käytetään ainoastaan käyttäjän yhdessä OWL-luokassa tekemiä valintoja. Suositus- ja rajoitusominaisuudet osoittautuivat toimiviksi ratkaisuiksi, joskin niiden toimintaa on syytä laajentaa käytettäessä editoria isommilla aineistoilla.

Editorin asetuksissa on mahdollista vaikuttaa siihen, minkä tyyppinen kenttä käyttäjälle näytetään objektityyppisen OWL-ominaisuuden kohdalla. Vaihtoehtoina ovat uuden instanssin luominen tai olemassaolevan valinta pudotuslistasta. Testiaineistossa ontologiset hierarkiat määriteltiin joko OWL-luokkien välisenä subClassOf-hierarkiana, tai instanssihierarkiana, jossa luokkien keskinäinen suhde kuvattiin "hasParent"-ominaisuuden avulla. Pudotuslistassa voidaan näyttää tietty ontologinen hierarkia. Asetustiedostossa voidaan määritellä, millä perusteella pudotuslistassa näytettävä hierarkia muodostetaan.

Jos editori asetetaan luomaan uusia instansseja jonkin ominaisuuden kohdalla, luotavan instanssin tyyppi täytyy kiinnittää asetustiedostossa. Editoria olisi syytä laajentaa siten, että käyttäjä voisi valita luotavan instanssin tyypin. Esimerkiksi tilanteessa, jossa OWL-luokan ominaisuuden sallittu arvoalue on jonkin ontologisen hierarkian juuriluokka, olisi järkevää että käyttäjä voisi myös luoda instanssin mistä tahansa juuriluokan aliluokasta eikä ainoastaan juuriluokasta.

Asetustiedoston avulla voidaan periaatteessa asettaa editori toiminaan minkä tahansa OWL-muotoisen tiedon kanssa. Asetustiedostossa voidaan määrätä editorin käsittelemät OWL-tiedostot, luokka josta annotaatio aloitetaan, erilaisten OWL-luokan ominaisuuksien toiminta käyttöliittymässä, sekä suositus- ja rajoitushakujen logiikka. Asetustiedoston esitysmuoto olisi kenties viisasta muuttaa RDF-muotoon jopa siten, että ontologiat itsessään sisältäisivät editorin vaatimat tiedot. Tällöin pääs-

täisiin tilanteeseen, jossa lisäämällä mihin tahansa OWL-aineistoon sopivat asetustiedot olisi aineisto sellaisenaan kelvollinen annotaatioeditorille. Tällöin myös muut ontologioita prosessoivat RDF-käsittelijät voisivat lisätä annotaatioeditorin vaatimat tiedot asetuksiin.

6.1.2 Käytettävyys

Järjestelmää lähdettiin toteuttamaan ajatuksella, että käyttäjälle näytetään yhden OWL-luokan tiedot kerrallaan, ja annetaan käyttäjän vapaasti liikkua ontologian muodostamassa puurakenteessa. Kuitenkin tarkasteltaessa esimerkiksi IWebS-testiaineistoa käsittelemään asetetun editorin pääikkunaa, jossa näytetään palveluntarjoajan tiedot, on käytettävyyden kannalta kyseenalaista että yhteystietojen syöttämistä varten käyttäjä joutuu siirtymään uuteen ikkunaan. Monimutkaisemmilla aineistoilla tämä voi osoittautua ongelmaksi, ja editorin toimintalogiikkaa olisikin syytä laajentaa siten, ettei käyttöliittymälomake olisi riippuvainen käytetystä OWL-tietomallista. Tällöin yhdessä lomakkeessa voitaisiin näyttää kenttiä monista OWL-luokista.

Editori näyttää ontologiset hierarkiat pudotuslista- tyyppisen käyttöliittymäkomponentin avulla. Valintojen tekeminen listasta voi käydä hankalaksi jos hierarkiat kasvavat isoiksi. Editoriin laadittiin tätä varten listoihin liittyvä hakuominaisuus, mutta uusien käyttöliittymäratkaisujen miettiminen voi tulla oleelliseksi haluttaessa laajentaa editoria toimimaan isommilla aineistoilla.

6.1.3 Tekniset ratkaisut

Valitut tekniikat osoittautuivat erittäin käyttökelpoisiksi annotaatioeditorin laadintaan. HP Labs Jena -paketti osoittautui monipuoliseksi RDF-käsittelyyn soveltuvaksi työkaluksi ja myöskin Apache Cocoon:n Control Flow -ominaisuus todettiin erinomaiseksi. Moittimisen arvoista Cocoon:ssa oli kunnollisten dokumentaatioiden puuttuminen. Esimerkiksi toteutuksessa ilmenneeseen session vanhenemisongelmaan ei löytynyt dokumenteista ratkaisua. Session vanhenemisongelma ilmenee siten, että käynnistetty editori kaatuu, jos se jätetään käyttämättä pitemmäksi aikaa.

Editorin ja siihen kuuluvan palvelimen kommunikaatio toteutettiin web-palveluna käyttäen Apache Axis-ympäristöä. Editorin testauksessa käytettiin kooltaan varsin pienikokoista aineistoa; minkään palvelimen ja editorin välillä siirrettävän tiedoston koko ei ylittänyt kymmentä megatavua. Lisäksi testauksessa palvelin ja itse edi-

tori toimivat samalla tietokoneella. Koska editorin ja palvelimen välillä siirretään kerralla koko aineisto niin aineistojen koon kasvaessa tietoliikenne editorin ja palvelimen välillä muodostuu todennäköisesti ongelmaksi. Editorin ja palvelimen välinen liikenne on täten syytä suunnitella uudelleen siten, että tietoa siirretään ainoastaan minimaalinen tarvittava määrä. Joseki RDF-palvelin on todennäköisesti sopiva komponentti tähän tarkoitukseen, ja nyt käytössä oleva Axis-palvelin onkin syytä vaihtaa Josekiin siirryttäessä käyttämään laajempaa aineistoa. Käytettäessä laajempaa aineistoa, on aineisto myös syytä säilyttää tietokannassa eikä tiedostoissa.

Ratkaisun hyvyyttä arvioitaessa myös ohjelmointitekniset näkökulmat on syytä ottaa huomioon. Editorin toiminnallisuusosion, eli Java- luokkien sekä palvelinosion toteutukset pysyivät hyvin hallinnassa. Ongelmia ilmeni käyttöliittymän näyttävän skriptin koon alkaessa kasvaa hallitsemattoman suureksi. Ongelma johtuu siitä, että vaikka editorin toiminnallisuus sijoittuu pääosin Java-koodiin, tehdään myös skriptissä jonkin verran RDF-prosessointia. Editoria laajennettaessa käyttöliittymäskriptin toiminta on syytä suunnitella uudestaan siten, ettei skriptissä tehdä RDF-prosessointia lainkaan.

6.2 Tulokset

Tutkimuksessa laadittu annotaatioeditori toteuttaa sille asetetut vaatimukset joskin parannuskohteita on helppo osoittaa. Editorin täydennys-, ja suosittelu- ominaisuudet ovat riippuvaisia olemassa olevasta relaatio-ontologiasta ja käytössä olevien annotaatioiden määrästä. Kyseiset ominaisuudet ovat asetettavissa erilaisiin aineistoihin sopiviksi, mutta rajoituksia ja suosituksia voidaan tehdä ainoastaan yhden OWL-luokan ominaisuuksien suhteen kerrallaan. Tutkimuksessa huomattiin, että tämä voi tulla ongelmaksi käytettäessä editoria monimutkaisemmilla kuin tässä tutkimuksessa käytetyllä aineistolla.

Useimmat annotaatioeditorit on laadittu varsin erilaisiin käyttötarkoituksiin kuin tässä tutkimuksessa laadittu editori. Useimmat editorit on tarkoitettu jo olemassaolevien HTML-sivujen annotaatiota varten. Tästä syystä laadittua editoria on hyvyyden suhteen vaikea verrata muihin editoreihin. Lisäksi editoria on testattu ainoastaan yhdellä aineistolla, joten editorin geneerisyydestä muilla aineistoilla ei voida olla varmoja. Toisaalta editorissa laaditun kaltaisia suositus- ja rajoitusominaisuuksia ei ole missään muussa tarjolla olevassa annotaatioeditorissa. Laadittua editoria ei siten ole hyvyyden suhteen syytäkään verrata muihin editoreihin, vaan editorin laatimisesta saatu anti on esitellä uusia annotaatioon liittyviä ominaisuuksia, joi-

den avulla voidaan laatia käyttäjää ohjaavia annotaatiotyökaluja. Esitellyt ominaisuudet antavat huomionarvoisen lähtökohdan laadukkaampien annotaatioeditorien laatimista varten, ja näin osaltaan edistävät semanttisen webin kehittymistä. Lisäksi tutkimuksessa on esitelty ja arvioitu tekniikoita, joiden avulla voidaan laatia laadukkaita semanttisen webin sovelluksia.

6.3 Jatkokehitys

Laadittua editoria tullaan mahdollisesti jatkokehittämään IWebS-tutkimusryhmässä. Oleellisimmat jatkokehityskohteet ovat editorin käytettävyyden parantaminen uusilla käyttöliittymäelementeillä sekä käyttäjälle näytettävän lomakkeen uudelleensuunnittelulla, palvelinosion muuntaminen Joseki-palvelimeksi, käyttöliittymäskriptin uudelleensuunnittelu, suositus- ja täydennys-kyselyjen laajentaminen, asetustiedoston muokkaaminen RDF-muotoiseksi sekä editorin testaaminen useilla eri aineistoilla.

7 Yhteenveto

Tämän tutkimuksen tavoitteena oli selvittää, olisiko mahdollista laatia sellainen käyttäjää ohjaava manuaalinen annotaatiotyökalu, jonka kohderyhmänä olisivat käyttäjät, jotka eivät tunne annotaatio-ongelmaa eivätkä ontologisia käsitteitä.

Tutkimuksessa on kartoitettu muut olemassa olevat annotaatioeditorit, joiden joukossa on erilaisiin tarkoituksiin varsin käyttökelpoisia työkaluja. Kuitenkin mikään kyseisistä editoreista ei sovellu tekniikkaa tai semantiikkaa tuntemattomien käytettäväksi.

Tutkimuksen osana laadittiin annotaatioeditori, jonka kohderyhmäksi asetettiin annotaatiota tuntemattomat henkilöt. Editorille asetettiin seuraavat tavoitteet:

- Täydentäminen ja rajoittaminen. Editori kykenee täydentämään käyttäjän tekemää annotaatiota sekä rajoittamaan virheellisten annotaatioiden tekemistä.
- Suosittelu. Editori kykenee suosittelemaan muita mahdollisia annotaatioita perustuen muiden käyttäjien laatimiin annotaatioihin.
- Pakottaminen. Editori osaa pakottaa käyttäjää antamaan kaikki tarpeelliset tiedot.
- Geneerisyys. Editori on mahdollista sovittaa erilaisiin käyttöympäristöihin.

Pohjatyönä editorin laatimiselle tehtiin tekniikoihin liittyvä kartoitus, jonka perusteella valittiin editorin toteutukseen käytettävät työkalut ja ympäristöt. Editori toteutettiin osana IWebS-järjestelmää, jonka aineistoja käytettiin editorin testaamiseen. Tuloksena tekniikoihin liittyvästä kartoituksesta löydettiin joukko laadukkaita välineitä, joiden avulla semanttisen webin sovelluksia voidaan laatia.

Tutkielmassa on kuvattu suunnitelma editorin laatimiseksi, ja laaditun editorin tekninen toteutus ja testaus. Tuloksia arvioitaessa on havaittu editorin täyttävän sille asetetut vaatimukset rajatussa ympäristössä. Editorilla on oltava käytössään tietynlainen relaatio-ontologia, jotta täydennys- ja rajoitusominaisuus toimisi. Jotta suositusominaisuus toimisi, on editorin tietokannassa oltava tapaukseen liittyviä muiden käyttäjien tekemiä annotaatioita. Editorin arvioinnissa on myös havaittu puutteita käytettävyydessä, teknisessä toteutuksessa, laajennettavuudessa ja geneerisyydessä. Kaikkiin puutteisiin on esitetty parannusehdotuksia jatkokehitystä varten.

Tämän tutkimuksen pääasiallinen anti oli esitellä uusia ominaisuuksia ja ajatuksia laadukkaampien, käyttäjää ohjaavien semanttisen webin annotaatioeditorien laatimista varten.

Lähteet

AXIS Apache Web Services Project - Axis. URL: http://ws.apache.org/axis/.

BD03 Buitelaar, P. ja Declerck, T., Linguistic annotation for the semantic web. *Annotation for The Semantic Web*, IOS Press. ISBN: 1 58603 345 X, 2003, sivut 99-111.

BHM+03 Booth, D., Haas, H., McCabe, F., Champion, M., Ferris, C., Newcomer, E. ja Orchard, D., Web Services Architecture. W3C Working Draft 8, 2003. URL: http://www.w3.org/TR/2003/WD-ws-arch-20030808/.

BLHL01 Berners-Lee, T., Hendler, J. ja Lassila, O., The Semantic Web. *Scientific American*, vol 284, no. 5, 2001, sivut 34-43.

BvHH+04 Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. ja Stein, L. A., OWL Web Ontology Language Reference, 2004. W3C Recommendation URL: http://www.w3.org/TR/owl-ref/.

CCDW04 Ciravegna, F., Chapman, S., Dingli, A. ja Wilks, Y., Learning to harvest information for the semantic web. *The Proceedings First European Semantic Web Symposium, ESWS 2004, Heralkion, Crete, Greece*, 2004, sivut 312-326.

Cir01 Ciravegna, F., Adaptive information extraction from text by rule induction and generalisation. *The Proceedings 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

COCO Apache Cocoon Project. URL: http://cocoon.apache.org/.

CW03 Ciravegna, F. ja Wilks, Y., Designing adaptive information extraction for the semantic web in amilcare. *Annotation for The Semantic Web*, IOS Press. ISBN 1 58603 345 X, 2003, sivut 112-127.

DKN+98 De Roeck, A., Kruschwitz, U., Neal, P., Scott, P., Steel, S., Turner, R. ja Webb, N., YPA - An Intelligent Directory Enquiry Assistant. *BT Technology Journal*, vol 16, no. 3, 1998, sivut 145-155. URL: citeseer.ist.psu.edu/roeck98ypa.html.

DV00 Denoue, L. ja Vignollet, L., An annotation tool for web browsers and its applications to information retrieval. *In Proceedings of RIAO2000, Paris*, 2000, URL: citeseer.ist.psu.edu/denoue00annotation.html.

EMSS00 Erdmann, M., Maedche, A., Schnurr, H. ja Staab, S., From manual to semi-automatic semantic annotation: About ontology-based text annotation tools *In Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content, Luxembourg* , 2000. URL: citeseer.ist.psu.edu/erdmann00from.html.

Fel98 Fellbaum, C., toimittaja, *WordNet: An Electronic Lexical Database*. The MIT Press, ISBN 0-262-06197-X, 1998.

GMV99 Guarino, N., Masolo, C. ja Vetere, G., OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems*, 1999, sivut 70-80.

Hef01 Heflin, J and Hendler, J, A Portrait of the Semantic Web in Action. *IEEE Intelligent Systems*, vol. 16, no. 2, 2001.

HJK+04 Hyvönen, E., Junnila, M., Kettula, S., Mäkelä, E., Saarela, S., Salminen, M., Syreeni, A., Valo, A. ja Viljanen, K., Finnish Museums on the Semantic Web. User's Perspective on MuseumFinland. *Museums and the Web 2004 (MW2004)*, Arlington, Virginia, USA, 2004.

HS02 Handschuh, S. ja Staab, S., Authoring and annotation of web pages in cream. *In Proceedings The Eleventh International World Wide Web Conference (www2002), Honolulu, Hawaii, USA* , 2002. URL: citeseer.ist.psu.edu/handschuh02authoring.html.

HSC02 Handschuh, S., Staab, S. ja Ciravegna, F., S-cream — Semi-automatic Creation of Metadata. *In Proceedings 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, October 2002. URL: citeseer.ist.psu.edu/handschuh02scream.html.

HVH02 Hyvönen, E., Viljanen, K. ja Hätinen, A., Yellow Pages on the Semantic Web. *Towards the Semantic Web and Web Services, In Proceedings of XML Finland 2002 Conference* , 2002, sivut 3-14.

Hyv01 Hyvönen, E., Semantic web – Kohti uutta merkitysten Internetiä. *Presentation at the semantic web kick-off seminar, University of Helsinki*

and Helsinki Institute for Information Technology, Helsinki, 2. Lokakuuta, 2001.

JEN03 Hp Labs Semantic Web Research, Jena, 2003. URL: http://jena.sourceforge.net/.

JOSE Joseki - The Jena RDF Server. URL: http//www.joseki.org/.

KH01 Kogut, P. ja Holmes, W., AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. *In Proceedings The First International Conference on Knowledge Capture (K-CAP 2001). Workshop on Knowledge Markup and Semantic Annotation*, Victoria, B.C., Canada, 2001.

KKPR01 Kahan, J., Koivunen, M., Prud'Hommeaux, E. ja R.Swick, Annotea: Open RDF Infrastructure for Shared Web Annotations. *In Proceedings of the WWW10 International Conference*, Hong Kong, 2001.

LS Lassila, O. ja Swick., R., Resource Description Framework (RDF) Model and syntax specification, 1999. URL: http://www.w3.org/tr/1999/rec-rdf-syntax-19990222.

MYR03 Mukiherjee, S., Yang, G. ja Ramakrishnan, I. V., Automatic annotation of content-rich html. em In Proceedings of the International semantic web conference (ISWC), Florida, USA, 2003. URL: citeseer.ist.psu.edu/668883.html.

NP01 Niles, I. ja Pease, A., Towards a Standard Upper Ontology. *The Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001.

NSD+01 Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W. ja Musen., M. A., Creating semantic web contents with protege-2000. *IEEE Intelligent Systems*, vol. 16, no. 2, 2001, sivut 60-71.

PROT Protege Knowledge Editor. URL: http://protege.stanford.edu/.

PYRP Pyrple, Python RDF API. URL http://infomesh.net/pyrple/.

RRAP RDF API for PHP (RAP). URL: http://sourceforge.net/projects/rdfapi-php/.

RAPT Raptor RDF Parser Toolkit. URL: http://librdf.org/raptor/.

RDQL A Query Language For RDF (RDQL). URL: http://www.w3.org/submission/2004/subm-rdql-20040109/.

SH01 Swartz, A. ja Hendler, J., The semantic web: A network of content for the digital city. *In Proceedings Second Annual Digital Cities Workshop, Kyoto, Japan, October*, 2001.

Sta02 Statistics Finland, *Standard Industrial Classification TOL 2002.* Valopaino, Helsinki, 2002. ISBN 952-467-097-6, Available at: http://www.stat.fi/tk/tt/luokitukset/index_talous_keh_ en.html.

STRU Apache Struts Project. URL: http://struts.apache.org/.

SWIP Swi prolog home. url http://www.swi-prolog.org/.

TURB Jakarta Turbine Web Application Framework URL: http://jakarta.apache.org/turbine/index.html.

Uni99 United Nations, Statistics Division, *Classification of Individual Consumption by Purpose (COICOP).* New York, USA, 1999. URL: http://unstats.un.org/unsd/cr/registry/regcst.asp?Cl=5 &Lg=1.

VVMD+02 Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A. ja Ciravegna, F., MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. *The Proceedings of the 13th International Conference on Knowledge Engineering and Management (EKAW 2002)* , A. Gomez-Perez, toimittaja. Springer Verlag, 2002.

WKLW98 Weibel, S., Kunze, J., Lagoze, C. ja Wolf, M., Dublin core metadata for resource discovery. Internet RFC 2413., 1998. URL: http://www.ietf.org/rfc/rfc2413.txt, 1998.

XSL99 Xsl99 XSL Transformations (XSLT) Version 1.0, 1999. URL: http://www.w3. org/tr/xslt.html., 1999.

Yee98 Yee, K.-P., Critlink: Better hyperlinks for the WWW, 1998. URL: http://crit.org/ ping/ht98.html.

LIITE 1: ASETUSTIEDOSTO + SELOSTUS

Annotaatioeditorin asetustiedosto, joka on asetettu toimimaan IWebS-aineistolla.
Määriteltävät asiat:

- **ANNE_DATA:** editorin käsittelemien ontologiatiedostojen nimet
- **ANNE_ROOT:** OWL-luokka, josta annotaatio aloitetaan
- **CLASS_HIERARCHY:**

  Määritys ”URI=CLASS_HIERARCHY” kertoo, että ominaisuuden kohdalla, jonka sallittuna arvoalueena (range) on annettu URI, näytetään käyttöliittymässä subClassOf-suhteeseen perustuva luokkahierarkia, jonka juuriluokkana annettu URI on. Seuraavassa esimerkissä määritellään, että URILLE ”…#Ihminen_1” näytetään subClassOf-luokkahierarkia, jonka juuriluokka on ”..#Ihminen_1”.

  ```
  http\://www.cs.helsinki.fi/group/iwebs/ns/human.owl#Ihminen_1=CLASS_HIERARCHY
  ```

- **TITLE:** Ominaisuuden käyttöliittymässä näytettävä otsikko.
- **SHOW:** Mitkä ominaisuudet luokasta näytetään.

  Seuraavassa esimerkissä määritellään, että luokasta “…#ServiceProvider” näytetään ominaisuudet “serviceEvent”,”contact”,”name_fi” ja “ytunnus”.

  ```
  SHOW_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider=
  serviceEvent,contact,name_fi,ytunnus
  ```

- **ORDER:** Näytettävien kenttien järjestys.
- **MANDATORY:** Pakolliset kentät. Näitä käyttäjä ei saa jättää tyhjäksi.
- **EDIT_BUTTON_TEXT:** Käyttöliittymässä tietystä luokasta luodun instanssin editointinappiin asetettava teksti.
- **INSTANCE_HIERARCHY:**

  Määritys ”URI=INSTANCE_HIERARCHY” kertoo, että ominaisuuden kohdalla, jonka sallittuna arvoalueena (range) on annettu URI, näytetään hierarkia, joka perustuu **ROOTURI**, **FOLLOWPROP** ja **DISPLAYPROP** määrittelyihin.

  - ROOTURI: Juurio minaisuuden URI:
  - FOLLOWPROP: Mitä ominaisuutta seurataan (esim. hasParent)
  - DISPLAYPROP: Mikä ominaisuus näytetään käyttöliittymässä.

  Seuraavassa esimerkissä määritellään, että URI:lle ”…#CoicopTaxonomyItem” näytetään hierarkia, jonka juurio minaisuutena on ”…#coicop_1-12”, seurattavana ominaisuutena on ”…#hasParent”, sekä käyttöliittymässä näytettävänä ominaisuutena on ”http://www.w3.org/2000/01/rdf-schema#label”.

  ```
  http\://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#CoicopTaxonomyItem=INSTANCE_HIERARCHY

  ROOTURI_http\://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#CoicopTaxonomyItem=
  http://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#coicop_1-12
  ```

```
FOLLOWPROP_http\://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#CoicopTaxonomyItem=
http://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#hasParent

DISPLAYPROP_http\://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#CoicopTaxonomyItem=
http://www.w3.org/2000/01/rdf-schema#label
```

- **RESTRICT:** Rajoitusominaisuus. Tämän ominaisuuden avulla määritellään, mitä tietyn luokan ominaisuuksia rajoitetaan keskenään. Käytännössä tämä määrää, mitkä ominaisuudet otetaan rajoitekyselyyn mukaan. Esimerkin tapauksessa määritellään luokalle ”…#ServiceEvent” rajoitettavat ominaisuudet ”tol, target ja goal”. Tämä määritys kertoo, että käyttäjän valitessa arvon johonkin kyseiseen ominaisuuteen, suoritetaan rajoitekysely, jossa lopuille arvoille. Esim. käyttäjän valitessa arvon tol:lle, tehdään rajoitekysely target:lle ja goal:lle. Edelleen käyttäjän valitessa target:lle arvon, tehdään rajoitekysely goal:lle.

```
RESTRICT_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent=tol,target,
goal
```

- **PROPOSE:** Kertoo, minkä luokan ominaisuutta ehdotetaan, ja millä perusteella. Tässä esimerkissä ”…#ServiceProvider” luokalle ehdotetaan muita ”serviceEvent”-instansseja, joilla on samat tol, target ja goal, kuin käyttäjän syöttämällä. Tässä siis ensimmäinen arvo (serviceEvent) kertoo ehdotettavan ominaisuuden nimen, ja loput (tol,target,goal) ne kyseisen luokan ominaisuudet, joiden perusteella ehdotus tehdään.

```
PROPOSE_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider=
serviceEvent,tol,target,goal
```

```
%
% Annotaatioeditori, asetustiedosto
%
%
% **** data (ontologiat), ja mistä instanssista aloitetaan

ANNE_DATA=serviceprovider_no_instances.owl, serviceevent_no_instances.owl, \
         location_no_instances.owl, goal.owl, coicop.owl, \
         human.owl, tol.owl

% time.owl

ANNE_ROOT=http://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider

%
% ****
% URI --> miten toimitaan kälissä
%
% CLASS_HIERARCHY - näytetään subClass-hierarkia pudotuslistana
% NEW_INSTANCE - luokasta luodaan uusi instanssi
% INSTANCE_HIERARCHY - näytetään instanssihierarkia pudotuslistana
%   ROOTURI_%%URI%% --> instanssihierarkian juuri
%   FOLLOWPROP_%%URI%% --> mitä ominaisuutta seurataan
%   DISPLAYPROP_%%URI%% --> mikä ominaisuus näytetään listassa
%
%

http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider=NEW_INSTANCE
http\://www.cs.helsinki.fi/group/iwebs/ns/human.owl#Ihminen_1=CLASS_HIERARCHY
http\://www.cs.helsinki.fi/group/iwebs/ns/goal.owl#Process=CLASS_HIERARCHY

http\://reliant.teknowledge.com/DAML/SUMO.owl#Entity=CLASS_HIERARCHY
%
% Otsikot
```

```
%

TITLE_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider=
Palveluntarjoajan perustiedot

TITLE_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#Contact=
Palveluntarjoajan yhteystiedot

TITLE_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent=
Yhden tarjotun palvelun kuvaus

%
% Mitkä kyseisen luokan propertyt näytetään, aj missä järjestyksesä
%

SHOW_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider=
serviceEvent,contact,name_fi,ytunnus

ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider_serviceEvent=4
ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider_contact=3
ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider_name_fi=1
ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider_ytunnus=2

MANDATORY_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider=
serviceEvent,contact,name_fi,ytunnus

%

SHOW_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#Contact=info,type
MANDATORY_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#Contact=info
ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#Contact_type=1
ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#Contact_info=2

EDIT_BUTTON_TEXT_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#Contact=info

%

SHOW_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent=
tol,goal,target,paymentMethod

ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent_tol=2
ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent_goal=3
ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent_target=4
ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent_paymentMethod=1
%ORDER_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent_sumo=5

MANDATORY_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent=tol,goal,target
EDIT_BUTTON_TEXT_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent=tol
%
%

http\://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#TolTaxonomyItem=INSTANCE_HIERARCHY
ROOTURI_http\://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#TolTaxonomyItem=
http://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#tol_TOL

FOLLOWPROP_http\://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#TolTaxonomyItem=
http://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#hasParent

DISPLAYPROP_http\://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#TolTaxonomyItem=
http://www.w3.org/2000/01/rdf-schema#label

%

http\://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#CoicopTaxonomyItem=INSTANCE_HIERARCHY
ROOTURI_http\://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#CoicopTaxonomyItem=
http://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#coicop_1-12

FOLLOWPROP_http\://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#CoicopTaxonomyItem=
http://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#hasParent

DISPLAYPROP_http\://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#CoicopTaxonomyItem=
http://www.w3.org/2000/01/rdf-schema#label

%
% Rajoitukset, suositukset
%
% RESTRICT_%%URI%% --> mitä ominaisuuksia verrataan instanssidataan
```

```
% PROPOSE_%%URI%% --> ensimmäinen kertoo, mitä ominaisuutta ehdotetaan, loput minkä perusteella
haetaan datasta
%

RESTRICT_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#ServiceEvent=tol,target,goal

%
% PROPOSE kertoo, minkä luokan ominaisuutta ehdotetaan, ja minkä perusteella.
%   Esim. tässä serviceprovider luokalle ehdotetaan muita serviceevent-instansseja, joilla samat
tol,target,goal kuin
%   käyttäjän syöttämällä
%

PROPOSE_http\://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#ServiceProvider=
serviceEvent,tol,target,goal

%
%
%
```

LIITE 2: RELAATIO-ONTOLOGIA (relation.owl)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
  This file is automatically generated by ...
  This file is subject to changes or overwriting without a notice!
-->
<!DOCTYPE rdf:RDF [
<!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
<!ENTITY sumo 'http://reliant.teknowledge.com/DAML/SUMO.owl#'>
<!ENTITY field_of_life 'http://www.cs.helsinki.fi/group/iwebs/ns/human.owl#'>
<!ENTITY time 'http://www.cs.helsinki.fi/group/iwebs/ns/time.owl#'>
<!ENTITY tol 'http://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#'>
<!ENTITY coicop 'http://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#'>
<!ENTITY goal 'http://www.cs.helsinki.fi/group/iwebs/ns/goal.owl#'>

]>
<rdf:RDF
    xmlns="http://www.cs.helsinki.fi/group/iwebs/ns/relation.owl#"
xmlns:goal="http://www.cs.helsinki.fi/group/iwebs/ns/goal.owl#"
    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:sumo="http://reliant.teknowledge.com/DAML/SUMO.owl#"
    xmlns:field_of_life="http://www.cs.helsinki.fi/group/iwebs/ns/human.owl#"
    xmlns:tol="http://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#"
    xmlns:coicop="http://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#"
  xml:base="http://www.cs.helsinki.fi/group/iwebs/ns/relation.owl">
  <owl:Ontology rdf:about="file:/home/pmlindgr/work/iwebs/ontologies/relation.owl" />
<!-- CLASSES -->
  <owl:Class rdf:ID="Relation">
    <rdfs:label xml:lang="fi">Relaatio</rdfs:label>
    <rdfs:label xml:lang="en">Relation</rdfs:label>
    <owl:Restriction>
      <owl:onProperty rdf:about="#from"/>
      <!-- <owl:cardinality>1</owl:cardinality> -->
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:about="#to"/>
      <!-- <owl:cardinality>1</owl:cardinality> -->
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:about="#hasLaw"/>
      <!-- <owl:cardinality>1</owl:cardinality> -->
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:about="#hasWeight"/>
      <!-- <owl:cardinality>1</owl:cardinality> -->
    </owl:Restriction>
    <rdfs:comment>"an aspect or quality (as resemblance)
    that connects two or more things or parts as being
    or belonging or working together or as being of the
    same kind" Merriam-Webster</rdfs:comment>
  </owl:Class>

  <owl:Class rdf:ID="RelationLaw">
    <rdfs:label xml:lang="fi">Relaatiolaki</rdfs:label>
    <rdfs:label xml:lang="en">Relation law</rdfs:label>
  </owl:Class>

<!-- PROPERTIES -->
  <owl:ObjectProperty rdf:ID="hasLaw">
    <rdfs:range rdf:resource="#RelationLaw"/>
    <rdfs:domain rdf:resource="#Relation"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="from">
    <rdfs:label xml:lang="fi">lähtö</rdfs:label>
    <rdfs:label xml:lang="en">from</rdfs:label>
    <rdfs:domain rdf:resource="#Relation"/>
    <rdfs:range  rdf:resource="&owl;Thing"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="to">
```

```
    <rdfs:label xml:lang="fi">maali</rdfs:label>
    <rdfs:label xml:lang="en">to</rdfs:label>
    <rdfs:domain rdf:resource="#Relation"/>
    <rdfs:range  rdf:resource="&owl;Thing"/>
  </owl:ObjectProperty>

  <owl:DatatypeProperty rdf:ID="hasWeight">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#Relation"/>
  </owl:DatatypeProperty>

<!-- INSTANCES -->
  <!-- RELATION LAWS -->
  <RelationLaw rdf:ID="hasAbstractGoal">
    <rdfs:label xml:lang="fi">on abstraktilta tavoitteeltaan</rdfs:label>
    <rdfs:label xml:lang="en">has an abstract goal</rdfs:label>
  </RelationLaw>

  <RelationLaw rdf:ID="hasTarget">
    <rdfs:label xml:lang="fi">kohdistuu tuotteeseen</rdfs:label>
    <rdfs:label xml:lang="en">has a target product</rdfs:label>
  </RelationLaw>

  <RelationLaw rdf:ID="hasDomain">
    <rdfs:label xml:lang="fi">on alueeltaan</rdfs:label>
    <rdfs:label xml:lang="en">has domain</rdfs:label>
  </RelationLaw>

  <!-- RELATIONS -->
  <!-- Muuttopalvelu -->
  <Relation rdf:ID="relation_1">
    <hasLaw rdf:resource="#hasAbstractGoal"/>
    <from rdf:resource="&tol;tol_60242" />
    <to rdf:resource="&sumo;Transportation" />
  </Relation>

  <Relation rdf:ID="relation_2">
    <hasLaw rdf:resource="#hasAbstractGoal"/>
    <from rdf:resource="&tol;tol_60242" />
    <to rdf:resource="&sumo;Carrying" />
  </Relation>

  <Relation rdf:ID="relation_3">
    <hasLaw rdf:resource="#hasTarget"/>
    <from rdf:resource="&tol;tol_60242" />
    <to rdf:resource="&coicop;coicop_05" />
  </Relation>

  <Relation rdf:ID="relation_4">
    <hasLaw rdf:resource="#hasDomain"/>
    <from rdf:resource="&tol;tol_60242" />
    <to rdf:resource="&field_of_life;Home" />
  </Relation>

<!-- Parturi ja kampaamo -->

<Relation rdf:ID="relation_25">
    <to rdf:resource="&tol;tol_93021" />
    <from rdf:resource="&goal;Siistiaaa_1" />
  </Relation>

<Relation rdf:ID="relation_26">
    <from rdf:resource="&tol;tol_93021" />
    <to rdf:resource="&target;UlkonaaakoooJaOlemus_1" />
  </Relation>

</rdf:RDF>
```

LIITE 3: PALVELUNTARJOAJA-ONTOLOGIA (serviceprovider.owl)

Tässä on määritelty palveluntarjoaja-ontologia. Ontologia sisältää luokat Contact sekä ServiceProvider. Luokassa ServiceProvider on ominaisuudet: Location, ServiceEvent, Contact, name_fi, tala_swe, tol, ytunnus, type, name_swe, tala_en, …

Editoria testattaessa asetuksilla on määrätty, että ontologiasta käytetään kentia Contact (yhteystiedot), ServiceEvent (yhden tarjotun palvelun kuvaus, kts. liite 4), name_fi (yrityksen nimi), tala_swe (testausta varten käytetty kenttä).

```
<?xml version="1.0"?>
<!--
  This file is automatically generated by ClassInstanceCombiner.java
  on Tue Jun 01 21:40:18 EEST 2004 by pmlindgr.
  This file is subject to changes or overwriting without a notice!

  DO NOT MAKE ANY CHANGES TO THIS FILE!

-->
<!DOCTYPE rdf:RDF [
<!ENTITY loc 'http://www.cs.helsinki.fi/group/iwebs/ns/location.owl#'>
<!ENTITY se 'http://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#'>
]>
<rdf:RDF
    xmlns="http://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#"
    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:loc="http://www.cs.helsinki.fi/group/iwebs/ns/location.owl#"
    xmlns:se="http://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#"
  xml:base="http://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl">
  <owl:Ontology rdf:about="">
    <cvstag>$Name:  $</cvstag>
    <cvsversion>$Revision: 1.3 $</cvsversion>
    <lasteditedby>$Author: mtlaukka $</lasteditedby>
  </owl:Ontology>
<!--
  <owl:Class rdf:ID="Location"/>
  <owl:Class rdf:ID="ServiceEvent"/>
-->
  <owl:Class rdf:ID="Contact"/>
  <owl:Class rdf:ID="ServiceProvider">
    <rdfs:label xml:lang="en">Service provider</rdfs:label>
    <rdfs:label xml:lang="fi">Palveluntarjoaja</rdfs:label>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="location">
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdfs:range rdf:resource="&loc;Location"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="serviceEvent">
    <rdfs:range rdf:resource="&se;ServiceEvent"/>
    <rdfs:domain rdf:resource="#ServiceProvider"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="contact">
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdfs:range rdf:resource="#Contact"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="name_fi">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="tala_swe">
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
```

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

<!--  <owl:DatatypeProperty rdf:ID="tol">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#ServiceProvider"/>
  </owl:DatatypeProperty>
-->

  <owl:DatatypeProperty rdf:ID="ytunnus">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:ID="type">
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:first>PUHELINNUMERO</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:first>FAKSINUMERO</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:rest rdf:parseType="Resource">
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                <rdf:first>WWW</rdf:first>
              </rdf:rest>
              <rdf:first>EMAIL</rdf:first>
            </rdf:rest>
          </rdf:rest>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
    <rdfs:domain rdf:resource="#Contact"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="name_swe">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="tala_en">
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="name_en">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="tala_fi">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#ServiceProvider"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="hapn">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#ServiceProvider"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="info">
    <rdfs:domain rdf:resource="#Contact"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```

LIITE 4: PALVELUN KUVAUS-ONTOLOGIA (serviceevent.owl)

Tässä on määritelty yhden tarjotun palvelun kuvausta varten tarkoitettu ontologia. Editoria testattaessa luokasta "..#ServiceEvent" on käytetty ominaisuuksia tol (toimialaluokitus), target (tavoite, koostuu coicop:sta sekä ihminen-ontologiasta), goal (kohde) sekä paymentMethod (maksuväline).

```
<?xml version="1.0"?>
<!--
  This file is automatically generated by ClassInstanceCombiner.java
  on Tue Jun 01 21:40:20 EEST 2004 by pmlindgr.
  This file is subject to changes or overwriting without a notice!

  DO NOT MAKE ANY CHANGES TO THIS FILE!

-->
<!DOCTYPE rdf:RDF [
<!ENTITY loc 'http://www.cs.helsinki.fi/group/iwebs/ns/location.owl#'>
<!ENTITY goal 'http://www.cs.helsinki.fi/group/iwebs/ns/goal.owl#'>
<!ENTITY human 'http://www.cs.helsinki.fi/group/iwebs/ns/human.owl#'>
<!ENTITY sp 'http://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#'>
<!ENTITY time 'http://www.cs.helsinki.fi/group/iwebs/ns/time.owl#'>
<!ENTITY coicop 'http://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#'>
<!ENTITY tol 'http://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#'>
]>
<rdf:RDF
    xmlns="http://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl#"
    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:loc="http://www.cs.helsinki.fi/group/iwebs/ns/location.owl#"
    xmlns:goal="http://www.cs.helsinki.fi/group/iwebs/ns/goal.owl#"
    xmlns:human="http://www.cs.helsinki.fi/group/iwebs/ns/human.owl#"
    xmlns:sp="http://www.cs.helsinki.fi/group/iwebs/ns/serviceprovider.owl#"
    xmlns:time="http://www.cs.helsinki.fi/group/iwebs/ns/time.owl#"
    xmlns:coicop="http://www.cs.helsinki.fi/group/iwebs/ns/coicop.owl#"
    xmlns:tol="http://www.cs.helsinki.fi/group/iwebs/ns/tol.owl#"
  xml:base="http://www.cs.helsinki.fi/group/iwebs/ns/serviceevent.owl">
  <owl:Ontology rdf:about="">
    <cvstag>$Name:  $</cvstag>
    <cvsversion>$Revision: 1.3 $</cvsversion>
    <lasteditedby>$Author: mtlaukka $</lasteditedby>
  </owl:Ontology>
<!--
  <owl:Class rdf:ID="Location"/>
  <owl:Class rdf:ID="Goal"/>
-->
  <owl:Class rdf:ID="ServiceEvent">
    <rdfs:label xml:lang="fi">Palvelutapahtuma</rdfs:label>
    <rdfs:label xml:lang="en">Service event</rdfs:label>
  </owl:Class>
<!--
  <owl:Class rdf:ID="Human"/>
  <owl:Class rdf:ID="Product"/>
-->
  <owl:Class rdf:ID="Contact"/>
<!--
  <owl:Class rdf:ID="Time"/>
-->
  <owl:ObjectProperty rdf:ID="openingTimes">
    <rdfs:range rdf:resource="&time;Time"/>
    <rdfs:domain rdf:resource="#ServiceEvent"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="location">
    <rdfs:range rdf:resource="&loc;Location"/>
    <rdfs:domain rdf:resource="#ServiceEvent"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="tol">
```

```
    <rdfs:range rdf:resource="&tol;TolTaxonomyItem"/>
    <rdfs:domain rdf:resource="#ServiceEvent"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="target">
    <rdfs:domain rdf:resource="#ServiceEvent"/>

<!--    <rdfs:range rdf:resource="&human;Ihminen_1"/> -->
<!--    <rdfs:range rdf:resource="&coicop;CoiCopTaxonomyItem"/> -->
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="&human;Ihminen_1"/>
          <owl:Class rdf:about="&coicop;CoicopTaxonomyItem"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="contact">
    <rdfs:range rdf:resource="&sp;Contact"/>
    <rdfs:domain rdf:resource="#ServiceEvent"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="goal">
    <rdfs:domain rdf:resource="#ServiceEvent"/>
    <rdfs:range rdf:resource="&goal;Tavoite_1"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="brand">
    <rdfs:domain rdf:resource="#ServiceEvent"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="paymentMethod">
    <rdfs:domain rdf:resource="#ServiceEvent"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="trafficInstruction">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#ServiceEvent"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:ID="hapn">
    <rdfs:domain rdf:resource="#ServiceEvent"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="fact">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#ServiceEvent"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```