Maiju Virtanen

RDF-TIETOMALLI TOIMINTAPROSESSIN TIEDONHALLINNAN TUKENA

Esimerkkinä suomalainen lainsäädäntöprosessi

Tietojärjestelmätieteen
pro gradu -tutkielma
08.10.2004

Jyväskylän yliopisto
Tietojenkäsittelytieteen laitos
Jyväskylä

TIIVISTELMÄ

Virtanen, Maiju Anniina
RDF-tietomalli toimintaprosessin tiedonhallinnan tukena. Esimerkkinä suomalainen lainsäädäntöprosessi / Maiju Virtanen
Jyväskylä: Jyväskylän yliopisto, 2004.
118 s.
Pro gradu -tutkielma

Tutkielma käsittelee RDF-tietomallia (Resource Description Framework), jonka avulla voidaan esittää tietoa webin resursseista. Tutkielman tavoitteena on analysoida, miten RDF.ää voidaan hyödyntää toimintaprosessien tiedonhallinan tukemisessa. Toimintaprosesseilla tarkoitetaan tässä tutkielmassa työprosesseja, joihin osallistuu monia eri henkilöitä ja yhteisöjä ja jotka koostuvat monesta erilaisesta toiminnosta. Esimerkkinä toimintaprosessista käytetään suomalaista lainsäädäntöprosessia.

Tutkielma on käsitteellis-teoreettinen, johon sisältyy konstruktiivinen osuus. Tutkielmassa esitellään esimerkein RDF-malli ja siihen liittyvä RDF Schema –kieli. Arviointiosuudessa verrataan keskenään RDF:n ja XML:n erilaisia käyttötarkoituksia. RDF:n ja RDF Scheman mahdollisuuksia arvioidaan muun muassa metatietoskeemojen esittämisessä ja älykkäiden semanttisen webin palvelujen toteuttamisessa. RDF-skeemojen suunnittelu esitetään seitsemän vaiheen mukaan. Skeemojen suunnittelun tueksi annetaan yleisiä ohjeita sekä esimerkkejä työkaluista.

Tutkielman tuloksena saatiin RDF-mallin arviointi, suomalaista lainsäädäntöprosessia tukeva RDF-skeemaehdotus sekä RDF-mallin hyödyntämisideoita lainsäädäntöprosessiin liittyen. Tutkielmassa saatiin selville, että RDF:n avulla pystytään esittämään käyttäjälle erilaisia näkymiä tiedosta ja toteuttamaan profilointia sekä älykkäitä palveluja, kuten semanttista samoilua ja semanttista suosittelua.

AVAINSANAT: RDF, RDF Schema, toimintaprosessit, tiedonhallinta, semanttinen web

ABSTRACT

Virtanen, Maiju Anniina
Supporting information management in work processes by RDF. The case of the Finnish legislative process/Maiju Virtanen
Jyväskylä: University of Jyväskylä, 2004.
118 p.
Master's Thesis

The thesis examines the RDF data model (Resource Description Framework) which is a language for represeting information about resources in the World Wide Web. The aim of the thesis is to analyze the possibilities of RDF for the information management in work processes. A work process in this study is understood as a process consisting of set of activities and participated by a number of persons and organisations as actors. The Finnish legislative process is used as an example case.

In the thesis the RDF model and the RDF Schema language are described with several examples. In the evaluation part, the RDF is compared with XML. Futhermore, the possibilities of the RDF and RDF Schema for representing metadata schemas and for implementing intelligent Semantic Web services are analyzed. The thesis also describes the design process of RDF schemas.

The main outcome of the study is the evaluation of the RDF model, a suggestion for an RDF schema supporting the Finnish legislative process, and ideas how to exploit the RDF model in the legislative process. The study indicates that with the help of the RDF model it is possible to extract different views from the data and to implement intelligent services like semantic browsing and semantic recommendation.

KEYWORDS: RDF, RDF Schema, working processes, data management, Semantic Web

SISÄLLYS

KUVIOT

TAULUT

1 JOHDANTO

Webin monet käyttömahdollisuudet ja sen käytön helppous ovat johtaneet sen suureen suosioon. Web-dokumentin löytää helposti URL-osoitteen avulla ja hyperlinkkejä seuraamalla pääsee käsiksi muuhun aiheeseen liittyvään tietoon. (Koivunen & Miller 2001.) Nykyisen webin yksinkertaisuus on kuitenkin johtanut siihen, että webissä on helppo eksyä ja oleellisen tiedon löytäminen helposti ja nopeasti on usein hankalaa. Nykyisestä webistä puuttuu *metatietoa*: nimiöintiin, luokitteluun ja tiedon kuvailuun liittyvää tietoa, joka mahdollistaisi web-resurssien täsmällisen hakemisen ja prosessoinnin (Swick 2000). Jos halutaan esimerkiksi hakea dokumentit, joíden kirjoittajana on Maiju Virtanen, on niiden hakeminen nykyisillä hakukoneilla hankalaa. Hakusanalla ”Maiju Virtanen” hakukone antaa tulokseksi kaikki mahdolliset dokumentit, jossa kyseinen sana esiintyy. Jos metatieto ”dokumentin kirjoittaja” olisi saatavilla tietokoneen ymmärrettävässä muodossa, pystyttäisiin saamaan täsmällinen hakutulos siihen mitä haettiin.

HTML-kieli kuvailee webissä tiedon ulkoasun, mutta ei kerro mitään tiedon sisällöstä tai merkityksestä. Vaikka ihmiset pystyvät tällaisen tiedon hyvin ymmärtämään, on sitä koneiden vaikea prosessoida. *Semanttinen web* on nykyisen webin laajennus ja ajankohtainen tutkimuksen kohde, jossa tieto on niin formaalissa muodossa, että sovelluksetkin pystyvät tulkitsemaan sen merkityksen ja ”ymmärtämään” tietoa (Berners-Lee, Hendler & Lassila 2001). Semanttisen webin teknologioilla ja kielillä pystytään luomaan tietoon rikasta semantiikkaa ja metatietoa. *Ontologiat* ovat eräs semanttisen webin teknologia, joilla voidaan määritellä formaalisti sovellusalakohtainen käsitteistö (Gruber 1993). Ontologioiden avulla sovellukset pystyvät tulkitsemaan tietoresurssien välisiä suhteita ja tekemään ontologisia päätelmiä (Berners-Lee, Hendler & Lassila 2001).

RDF (Resource Description Framework) on tietomalli ja metatiedon esitystapa, jonka avulla pystytään esittämään resurssien välisiä suhteita ja resurssien ominaisuuksia. RDF:llä on myös XML-syntaksi. *RDF Schema* –kieltä käytetään määrittelemään RDF-kuvailuissa tarvittava sanasto. (Manola & Miller 2004.) RDF Schema -määrittelyjä kutsutaan usein ”kevyiksi ontologioiksi” (Volz, Oberle & Studer 2003; Boulos, Abdul & Carson 2002, 134).

RDF:ää voidaan pitää semanttisen webin de facto –standardina, koska monet semanttisen webin kielet perustuvat RDF-malliin (Connolly, Harmelen, Horrocks ym. 2001; McGuinness & Harmelen 2004) ja sitä luonnehditaan semanttisen webin laajennettavaksi *ydinkieleksi* (Staab, Erdmann, Maedche, & Decker 2002). RDF:n on kehittänyt W3C (World Wide Web Consortium), joka koostuu monista jäsenorganisaatioista ympäri maailmaa ja joka kehittää tulevaisuuden webin visioita ja teknologioita (W3C 2004).

Tutkielman ensisijaisena tutkimusongelmana on: Miten RDF-mallia voidaan hyödyntää toimintaprosesessien tiedonhallinnassa? Osaongelmat tähän liittyen ovat: Minkälaisia mahdollisuuksia ja rajoitteita RDF-malli pitää sisällään ja kuinka RDF-skeemojen suunnittelu ja toteutus tehdään käytännössä?

Esimerkkinä toimintaprosesseista käytetään suomalaista lainsäädäntöprosessia. Toimintaprosesseilla tarkoitetaan tässä tutkielmassa työprosesseja, joihin osallistuu monia eri henkilöitä ja yhteisöjä ja jotka koostuvat monesta erilaisesta toiminnosta. Lainsäädäntöprosessin tiedonhallinta on monimutkaista, koska prosessiin osallistuu asiantuntijoita monesta eri organisaatiosta, siinä tarvitaan paljon aikaisemmissa lainsäädäntöprosesseissa tuotettua materiaalia ja työskentelyssä käytetään kymmeniä tietojärjestelmiä (Lehtinen, Salminen & Huhtanen 2004).

RDF-mallin hyötyjen analysoimisessa koskien suomalaista lainsäädäntöprosessia tutkielmassa rakennetaan RDF-skeemaehdotus ja keskitytään RDF:n käyttötapojen ideoimiseen skeemaehdotuksen pohjalta. RDF-kuvauksien käytännön toteutukseen liittyen kerrotaan erityisesti RDF-skeemojen suunnittelusta, sillä skeemoja tarvitaan yhtenäisten RDF-kuvailujen luontiin. Tavat ja tekniikat varsinaisten RDF-kuvailujen toteuttamiseen ja mahdolliseen automaattiseen generoimiseen on jätetty tämän tutkielman ulkopuolelle.

Tutkimuksen lähestymistapa on käytännönläheinen. Esimerkkinä käytetään todellista sovellusaluetta toimijoineen ja järjestelmineen. Tutkielma on käsitteellis-teoreettinen, johon sisältyy konstruktiivinen osuus (RDF-skeemaehdotus). Tutkimusmenetelmänä on käytetty tieteellisen ja muun kirjallisuuden analysoimista tutkimusaiheeseen liittyen.

Tutkielman tuloksena on RDF:n mahdollisuuksien ja rajoitteiden arviointi sekä yleisellä tasolla, että suomalaiseen lainsäädäntöprosessiin liittyen. Tutkielman konkreettisin tulos on RDF-prosessiskeema, johon on mallinnettu lainsäädäntöprosessin asiakirjoja, toimijoita ja vaiheita. Prosessiskeeman pohjalta tehtyjen kuvauksien avulla on mahdollista jäljittää esimerkiksi missä vaiheessa tietty lainsäädäntöhanke on menossa ja mikä on prosessissa syntyneiden asiakirjojen järjestys. Saman skeemasanaston pohjalta tehdyt kuvaukset ovat keskenään yhteensopivia. Niitä voivat siis käyttää erilaiset järjestelmät edellyttäen, että niihin on rakennettu ohjelmallinen tuki RDF-tiedon prosessointiin. Yhtenäinen ja yhteensopiva tiedon muoto tukee järjestelmien integrointia (Shen & Yang 2004; Candan, Liu & Suvarna 2001; Hjelm 2001, 4), koska sen välittäminen järjestelmien välillä on helpompaa verrattuna siihen, että jokaisen järjestelmän välillä tiedon siirtomuoto ja käsittelytapa suunniteltaisiin erikseen. RDF-skeeman avulla tehtyjen kuvailujen pohjalta sovellukset pystyvät tekemään automaattisia päätelmiä, jotka mahdollistavat asiaan liittyvien muiden tietoresurssien linkkien näyttämisen käyttöliittymässä ja sellaisen tiedon esittämisen, jota käyttäjä ei alun perin kuvitellut hakevansa (Hyvönen, Saarela, Viljanen ym. 2004).

Tutkielman toisessa luvussa esitetään keskeisimmät käsitteet. Kolmannessa ja neljännessä luvussa esitellään esimerkein RDF-tietomallin ja RDF Scheman ominaisuudet. Viidennessä luvussa verrataan RDF-mallia XML:ään ja arvioidaan RDF:n etuja ja mahdollisuuksia sekä rajoitteita ja haasteita. Kuudennessa luvussa esitetään RDF-skeemasuunnittelun vaiheet sekä ohjeita ja esimerkkejä työkaluista RDF-skeemojen suunnittelussa. Seitsemännessä luvussa analysoidaan RDF-mallin mahdollisuuksia ja hyötyjä suomalaisen lainsäädäntöprosessin tiedonhallinnan tukemisessa. Lopuksi yhteenvedossa kootaan tutkimuksen tulokset ja keskeinen sisältö.

2 KESKEISET KÄSITTEET

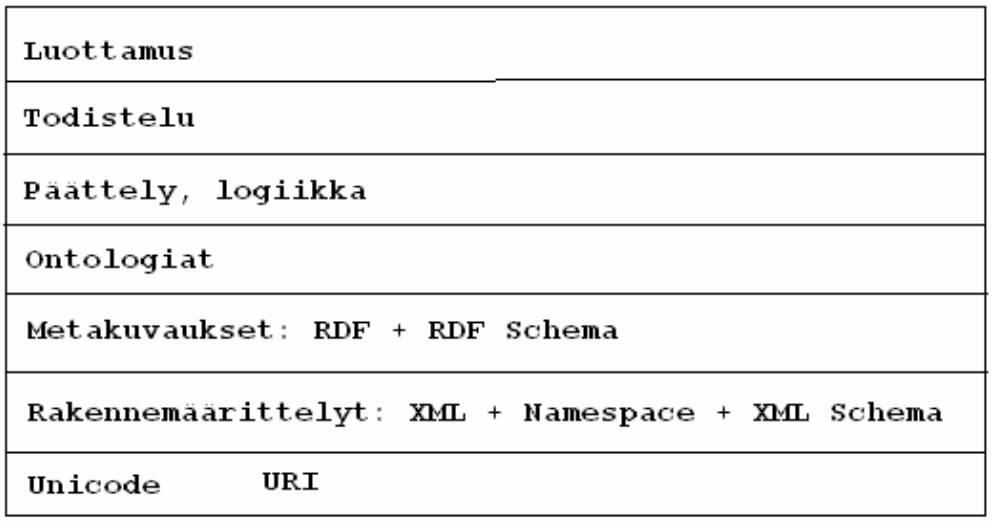
Tässä luvussa esitellään tutkielman kannalta keskeiset käsitteet semanttinen web, metatieto ja ontologiat. Käsitteiden ymmärtäminen auttaa seuraavien lukujen ymmärtämisessä.

2.1 Semanttinen web

Suurin osa webissä olevasta tiedosta on tänä päivänä suunniteltu ihmisten luettavaksi. Tietokoneetkin voivat joiltain osin prosessoida webin tietoa, mutta se rajoittuu dokumentin rakenteen (otsikko, linkki) ja ulkoasun prosessoimiseen. Tietokoneet eivät pysty ymmärtämään tiedon todellista *semantiikkaa*. Semanttinen web antaa välineet semanttisen sisällön esittämiseen webissä. Tällaista sisältöä tietokoneet voivat ja sovellukset käyttää automaattisesti ja itsenäisesti. (Berners-Lee, Hendler & Lassila 2001.)

Semanttisella webillä ei tarkoiteta mitään erillistä webiä, vaan nykyisen webin kehittämistä ja laajentamista. Tieto määritellään semanttisessa webissä formaalisti koneiden ymmärrettävässä muodossa. (Berners-Lee, Hendler & Lassila 2001.) Asia, jota erityisesti semanttisessa webissä tarvitaan, on metatieto. Metatiedolla voidaan kuvailla tietoa ja esittää tiedon semantiikkaa formaalissa muodossa. (Swick 2000.)

Semanttisen webin teknologiat voidaan esittää kerrosmallin avulla. Kuviossa 1 nähdään, että alimpana ovat Unicode ja URI –kerrokset, jotka takaavat, että Internetissä käytetään kansainvälisiä merkkejä ja että resurssit pystytään identifioimaan webissä. XML yhdessä nimiavaruus- ja XML Schema -määrittelyiden kanssa varmistavat rakenteellisen tai syntaktisen yhteentoimivuuden, jotta semanttisen webin määrityksiä voidaan integroida muiden XML-pohjaisten standardien kanssa. Seuraavalla tasolla ovat metakuvaukset RDF ja RDF Schema –muodossa, jotka mahdollistavat väittämien tekemisen URI-resursseista sekä sanastojen luomisen. Sanastoilla voidaan antaa resursseille tyyppejä. Ontologiakerros tukee sanastojen kehittämistä. Täällä voidaan määritellä käsitteitä ja niiden välisiä suhteita. Logiikkatasolla voidaan tehdä päättelysääntöjä alla olevan ontologian pohjalta. Todistelukerros (proof) suorittaa säännöt ja arvioi luottamuskerroksen kanssa sovellusten luotettavuutta. (Koivunen & Miller 2001.)



KUVIO 1. Semanttisen webin kerrokset Berners-Lee:tä (2000) mukaillen.

2.2 Metatieto

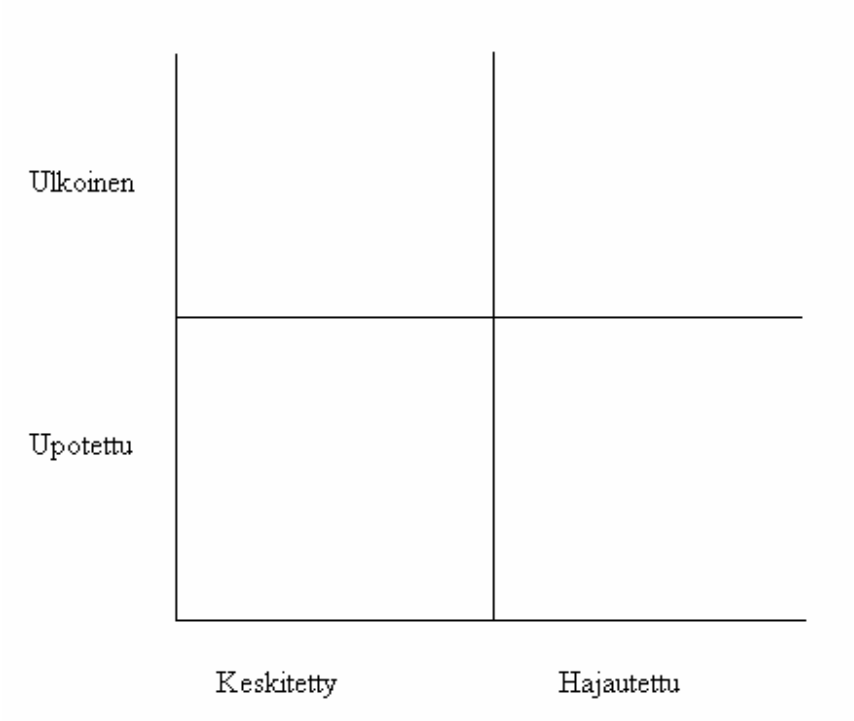
Metatieto on yksinkertaisesti määriteltynä ”tietoa tiedosta”, mutta se voidaan käyttäjäryhmästä riippuen ymmärtää hyvin monella eri tavalla. Gilliland-Swetland (2000) määrittää metatiedon seuraavasti: Metatieto on sen kaiken summa, joka voidaan sanoa tiedosta millä tahansa tavalla koostettuna (”The sum of total of what one can say about any information object at any level of aggregation”). Metatieto on tärkeää tiedon hallinnassa, tiedon saatavuuden varmistamisessa, tietokokoelmien käytössä ja tiedon hauissa, kirjastoissa sekä tiedon arkistoinnissa. (Gilliland-Swetland 2000.)

Metatietorakenteet voivat olla monimutkaisia ja kalliita ylläpitää, mutta niistä saatava hyöty on usein vaivan arvoinen. Metatieto parantaa tiedon saatavuutta ja tietoon tehtävien hakujen tehokkuutta. Metatieto mahdollistaa haut useisiin eri tietovarastoihin ja sen avulla voidaan tehdä ”virtuaalisia tietovarastoja” tietoresursseista, jotka ovat tallennettuna hajallaan toisistaan. Metatiedoissa pystytään säilyttämään tiedon *konteksti*, joka tarkoittaa tietoresurssin suhteita muihin resursseihin, ihmisiin (esimerkiksi tekijätieto), tapahtumiin (esimerkiksi versiot ja muutoshistoria) ja aikaan (koska resurssi on luotu). Metatiedot helpottavat tiedon profilointia eli tiedon esittämistä eri tavalla eri käyttäjille. Metatietojen avulla voidaan luokitella resursseja, osoittaa resurssien käyttöoikeudet ja niiden näyttämiseen tarvittava lait-

teisto ja ohjelmisto, kerätä resurssin käyttötietoa ja –historiaa sekä tallentaa tietoa siitä, kuinka kauan ja missä muodossa resurssia pitäisi säilyttää. (Gilliland-Swetland 2000.) Gillian-Swetland luokittelee metatiedot hallinnollisiin (administrative), kuvaileviin (desciptive), säilytykseen liittyviin (preservation), teknisiin (technical) ja käyttöön (use) liittyviin metatietoihin.

Ahmedin, Ayersin, Birbeckin ym. (2001, 327) mukaan metatietoa voidaan luokitella kuvion 2 osoittamalla tavalla. Kuvion pystyakseli kuvaa metatietoa resurssin itsensä kannalta: metatieto voi olla *upotettuna* (embedded) resurssiin itseensä tai tallennettuna resurssista erilliseen paikkaan (*ulkoinen* metatieto). Kuvion vaaka-akseli kuvaa metatietoa sovelluksen kannalta: metatieto voi olla tallennettuna *keskitetysti* yhteen paikkaan, johon kaikki kyselyt voidaan kohdistaa, tai *hajautetusti*, jolloin metatieto haetaan useasta eri paikasta.

Upotetusta metatiedosta esimerkkinä on <META>-elementti HTML-dokumentin sisällä. Upotetun metatiedon etu on sen luomisen helppous, koska yleensä sen luominen tapahtuu samalla välineellä kuin itse resurssin luominen. Ulkoinen metatieto on tallennettuna esimerkiksi tietokantaan, tavalliseen tiedostoon tai rakenteiseen tiedostoon. Ulkoista metatietoa on yleensä helpompi ylläpitää ja sen avulla voidaan tietoon luoda useita eri näkymiä (views) ilman, että alkuperäiseen resurssiin tehdään muutoksia. Esimerkkinä keskitetystä metatiedosta ovat webin hakukoneiden metatietovarastot. Hakukoineiden tarvitsee kohdistaa kyselyt vain yhteen metatietovarastoon sen sijaan, että niiden pitäisi yrittää tehdä kyselyjä kaikkiin maailman web-sivuihin. Hajautettua metatietoa varten sovelluksen täytyy osata muodostaa hakuja moneen paikkaan ja osata yhdistää hakujen tulokset. (Ahmed, Ayers, Birbeck ym. 2001.)



KUVIO 2. Metatiedon luokittelu Ahmedin, Ayersin, Birbeckin ym. (2001) mukaan.

Salminen (2003a) luokittelee dokumenttien metatietoja *bibliografisiin* metatietoihin (esimerkiksi Dublin Core –metatietoelementeillä ilmaistuna), ontologioiden avulla ilmaistuihin *semanttisiin* metatietoihin, dokumenttien luomis- ja käyttöympäristöjä kuvaaviin *kontekstuaalisiin* metatietoihin sekä *rakennemetatietoihin*, joita esitetään esimerkiksi XML Scheman avulla.

Metatietojen standardointi on tärkeää, jotta erilaiset ja eri lähteistä tulevat metatiedot olisivat yhteentoimivia keskenään (Gilliland-Swetland 2000). Metatiedon standardointi ja kontrolloitu semantiikka ja syntaksi ovat myös edellytys automaattiseen metatiedon tuottamiseen. (JUHTA 2004) Koko semanttisen webin onnistumisessa on oleellista metatiedon luomisen helppous. Metatiedon luomista varten pitäisi löytyä kunnollisia työvälineitä ja sen luomisprosessista tulisi tehdä mahdollisimman automaattinen. (Hyvönen, Salminen, Junnila & Kettula 2004.)

Tärkeimpiä olemassa olevia metatietostandardeja tämän tutkielman kannalta ovat Dublin Core ja JHS143. *Dublin Core* on kansainvälinen metatietoformaatti, jonka yksinkertaisin muoto (Simple Dublin Core) sisältää yhteensä 15 metatietokenttää (title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, rights). Laajempi Dublin Coren muoto (Qualified Dublin Core) sisältää yhden lisäelementin (audience) ja joukon elementtien tarkenteita (esimerkiksi available, created). Dublin Coren tavoitteena on olla mahdollisimman yleinen ja yksinkertainen, jotta sitä voitaisiin käyttää mahdollisimman laajasti. Sitä voidaan laajentaa vastaamaan sovellusalan tarpeita ja toisaalta kaikkia sen metatietoelementtejä ei ole pakko käyttää, vaan osa voidaan jättää tarpeen mukaan pois. Dublin Corelle on myös olemassa RDF-skeema. (DCMI 2004.) *JHS 143 -suositus* on julkisen hallinnon neuvottelukunnan (JUHTA) määrittelemä suomalainen julkisen hallinnon asiakirjojen metatietoformaatti, joka perustuu Dublin Core – standardiin. Suositus on tarkoitettu asiakirjahallinnon tarpeisiin sekä julkisen hallinnon asiakirjojen julkaisemiseen erityisesti verkkojulkaisuna. Se sisältää yhteensä 24 metatietoelementtiä ja lisäksi useita tarkenteita. (JUHTA 2004.)

2.3 Ontologiat

Semanttisen webin kerroksissa (kuvio 1) ontologiat ovat rakennemäärittelyjen ja metakuvauksien yläpuolella. Ontologioille on kirjallisuudessa esitetty monenlaisia määritelmiä. Luultavasti perinteisin ja tieteellisessä kirjallisuudessa käytetyin on Gruberin (1993) määritelmä: Ontologia on eksplisiittinen käsitteiden määritelmä (”An ontology is an explicit specification of a conceptualization”). Useissa ontologian määritelmissä viitataan formaaliin määrittelyyn käsitteistä ja niiden suhteista (Gruninger & Lee 2002; Klein 2002; Boulos, Roudsari & Carson 2002). Yleensä ontologia liittyy tiettyyn sovellusalaan (Gruninger & Lee 2002).

Stoffel, Taylor ja Hendler (1997) jakavat ontologiat *perinteisiin* (traditional) ja *sekamuotoisiin* (hybrid) ontologioihin. Perinteisillä ontologioilla he tarkoittavat ontologioita, jotka sisältävät ainoastaan käsitteet ja niiden määritykset. Sekamuotoiset ontologiat taas koostuvat sekä ontologisista käsitteistä ja suhteista että niiden pohjalta tehdyistä ilmentymistä. Sekamuotoiset ontologiat saattavat Stoffelin ym. (1997) muiden mukaan koostua suhteellisen

pienestä ontologiaosuudesta ja suuremmasta ilmentymien esimerkkikannasta.

Perinteisen *tesauruksen* ero ontologiaan on se, että tesaurus määrittelee sanoja ja ontologia käsitteitä. Esimerkiksi YSA – yleinen suomalainen asiasanasto (Helsingin yliopiston kirjasto 1999) on tällainen tesaurus. Tesauruksessa on määriteltynä sanojen synonyymit, homonyymit, rinnakkaiset käsitteet helpottamaan tiedonhakua ja ohjaamaan tiedonhakijoita käyttämään sopivia sanoja tiedonhaussa. Ontologia määrittelee abstraktimmalla tasolla käsitteitä, joten esimerkiksi yksi käsitteellinen ontologia saattaa sisältää joukon tesauruksia eri kielillä. (Hyvönen, Salminen, Junnila & Kettula 2004.) Wielingan, Schreiberin, Wielemakerin ja Sandbergin (2001) mukaan perinteiset sanastot ovat vain osa tiedosta, jota tarvitaan semanttisen webin rikkaiden kuvauksien tekemiseen. Rikkaita metakuvauksia saadaan heidän mukaansa aikaan ontologioiden avulla.

Kiinnostus ontologioihin on kasvamassa koko ajan ja Klein (2002) uskaltaa jopa väittää, että ontologioiden kehitystyö on siirtymässä asiantuntijatehtävistä kenen tahansa web-julkaisijan tehtäväksi. Vaikka tällä hetkellä ontologioita on olemassa jo paljon, on vain vähän sellaista tietoa, joka todella on yhteydessä johonkin ontologiaan. (Klein 2002.)

Käsitteet ontologia, sanasto ja metatietoskeema ovat hyvin lähellä toisiaan tämän tutkielman aihepiirissä. RDF Schemaa sanotaan usein ontologiaksi (Klein 2002) tai kevyeksi ontologiaksi (light weight ontology language) (Volz, Oberle & Studer 2003; Boulos, Roudsari & Carson 2002, 134). RDF–spesifikaatioissa RDF Schema –määrittelyjä kutsutaan sanastoiksi (vocabularly). Skeema-nimitys on yleinen ja sitä käytettäessä voidaan puhua RDF-skeemoista, XML-skeemoista tai esimerkiksi tietokannan rakenteen skeemasta. Metatietoskeemalla tarkoitetaan sanastoa metatiedon esittämiseen. RDF Schema –kielellä tehdyt skeemat ovat yleensä metatietoskeemoja. Ontologiakielissä, kuten DAML, ei käytetä käsitettä skeema ja siksi tässä tutkielmassa käytetään RDF-skeemoista joskus myös käsitettä ontologia osoittamaan yhteys ontologiakieliin.

3 RDF-TIETOMALLI

Tässä luvussa esitellään RDF-malli. Johdanto kertoo RDF-mallin perusidean ja sen jälkeen RDF-malli ja RDF-graafiesitystapa esitellään tarkemmin. Seuraavaksi kuvataan RDF/XML-syntaksin tärkeimmät piirteet, sitten tyhjät solmut sekä tietotyyppien käyttö RDF:ssä ja lopuksi kaksi yleistä URI-viitteiden lyhennystapaa RDF/XML-syntaksissa.

Luku sisältää monia esimerkkejä, joiden tarkoitus on näyttää konkreettisesti, miten tietoa esitetään RDF-muodossa. Luvun ei ole tarkoitus olla täydellinen kooste kaikista mahdollisista RDF- ja RDF/XML-muodoista, vaan luvussa esitetään tämän tutkielman kannalta oleelliset asiat. Tarkat kuvaukset RDF:stä löytyvät RDF-spesifikaatioista.

W3C:n RDF-spesifikaatio koostuu kuudesta eri dokumentista:
1) RDF/XML Syntax Specification (Beckett 2004),
2) RDF Vocabulary Description Language 1.0: RDF Schema (Brickley & Guha 2004),
3) RDF Primer (Manola & Miller 2004),
4) Resource Description Framework (RDF): Concepts and Abstract Syntax (Klyne & Carroll 2004)
5) RDF Semantics (Hayes 2004) ja
6) RDF Test Cases (Grant & Beckett 2004).

Tässä tutkielmassa lähteenä on käytetty kolmea ensimmäistä dokumenttia. Pääasiallisena lähteenä tässä luvussa on käytetty RDF Primer –dokumenttia (Manola & Miller 2004), joten lähdeviitteet on merkitty vain, jos on käytetty tästä poikkeavaa lähdettä.

3.1 Johdanto

RDF (Resource Description Framework) on W3C:n kehittämä kieli webin resurssien, kuten www-sivujen kuvaamiseen ja niiden metatietojen esittämiseen. RDF:n avulla voidaan ilmaista metatietoa myös sellaisista resursseista, jotka eivät ole haettavissa suoraan webistä, esimerkiksi henkilöt ja elektronisen kauppapaikan tuotteet. RDF-spesifikaatiot ovat W3C:n

suosituksia (recommendation). (Becket 2004.)

RDF perustuu ideaan, että kuvatuilla resursseilla on ominaisuuksia. Esimerkiksi resurssilla ”kirja” on ominaisuus ”kirjoittaja”. Ominaisuuksilla on edelleen arvoja. Esimerkiksi ominaisuudella ”kirjoittaja” on arvo ”Väinö Linna”.

Resurssit voidaan kuvata tietokoneiden ymmärtämillä *väittämillä* (statement). Väittämät ovat aina yksinkertaisia (resurssi, ominaisuus, arvo)-*kolmikoita*. Toiselta nimeltään RDF-mallin kolmikon osat ovat (subjekti, predikaatti, objekti). Esimerkki väittämästä voisi olla: ”Kirjan [resurssi/subjekti] on kirjoittanut [ominaisuus/predikaatti] Väinö Linna [arvo/objekti]”.

3.2 Resurssien identifioiminen URI-tunnisteiden avulla

RDF:n perusmääritelmä resurssille on: Resurssi on mikä tahansa asia, joka voidaan identifioida URI-tunnisteen avulla. URI-tunnisteilla (Uniform Resource Identifier) voidaan identifioida internet-osoitteellisia eli URL:n (Uniform Resource Locator) omistavia asioita tai sellaisia asioita, jotka eivät sijaitse internetissä. (Manola & Miller 2004.) URI-tunnistetta kutsutaan lyhemmin URI:ksi.

URI-spesifikaation mukaan (Berners-Lee, Fielding & Masinter 1998, 2) URL on URI:n alalaji, jolla identifioidaan resursseja ensisijaisesti niiden sijainnin kautta (”location” = sijainti), eikä esimerkiksi resurssin nimen kautta. Toinen URI:n alalaji on URN (Uniform Resource Name), jolla resurssit identifioidaan niiden nimien avulla ja jonka on tarkoitus säilyä globaalisti ainutlaatuisena ja pysyvänä, vaikka resurssi ei enää ole olemassa tai saatavilla. RDF:n käyttämä identifiointiväline URI voidaan katsoa olevan joko sijainti, nimi tai molemmat. (Berners-Lee ym. 1998, 2) RDF ja internet-selaimet siis tulkitsevat URI-tunnisteet hieman eri tavalla. RDF:n URI-tunnisteita ei välttämättä aina pysty paikantamaan selaimen avulla, koska RDF käyttää URI:a ainoastaan *identifioimaan* resursseja. (Manola & Miller 2004.)

Usein URI-tunnisteeseen liitetään *osatunniste* (fragment identifier) ja tällöin tunnistetta kutsutaan yleisemmin *URI-viitteeksi* (URIref, URI reference). URI-viitteen muoto on esitetty URI-spesifikaation mukaan (Berners-Lee ym. 1998, 14) taulussa 1. Taulussa oleva *absoluuttinen URI* on täydellinen osoitin resurssiin (esimerkiksi http://www.it.jyu.fi/raske/projekti.html) ja *suhteellinen URI* tulkitaan kontekstin avulla (tässä esimerkiksi projekti.html). Osatunniste erotetaan URI:sta merkillä ”#”.

TAULU 1. URI-viitteen muoto.

URI-viite = [absoluuttinen URI	suhteellinen URI] ["#" osatunniste]

Käsitettä *URI-viite* käytetään siis yleisenä nimenä resurssin tunnisteelle. URI-viite voi olla absoluuttinen tai suhteellinen ja siihen voi liittää vapaaehtoisen osatunnisteen. Varsinainen *URI-tunniste* (URI) taas syntyy URI-viitteen pohjalta ja sillä tarkoitetaan absoluuttista URI:a, josta osatunniste on otettu pois.

Kuten edellä mainittiin, RDF ja selain eivät tulkitse URI-tunnisteita samalla tavalla. Samoin on asia myös osatunnisteiden kanssa. HTML-dokumentissa osatunnisteet usein viittaavat tiettyyn kohtaan URL:n paikantamassa dokumentissa. Selain tulkitsee esimerkiksi kahden eri URI-viitteen http://www.oma.fi/index.html ja http://www.oma.fi/index.html#osa2 olevan toisiinsa liittyviä. Koska RDF käyttää URI-tunnisteita ainoastaan identifioimaan resursseja, ei hakemaan niitä, RDF ei näe mitään yhteyttä näiden kahden URI-viitteen välille. RDF:lle ne ovat vain kaksi eri resurssia. On tietenkin mahdollista, että URI-tunniste osoittaa dokumentin sijainnin, mutta joka tapauksessa RDF ei oleta, että näin on. (Manola & Miller 2004.)

3.3 RDF-malli

Graafisesti RDF-kuvauksia voidaan esittää RDF-graafina solmuina ja nuolina. Graafissa nuoli esittää RDF-väittämän predikaattia ja soikio subjektia tai objektia. Predikaattinuoli

suunnataan subjektisolmusta objektisolmuun. Suorakulmiolla esitetään objekti, joka on puhtaassa tekstimuodossa (literal) URI-muodon sijaan. Tekstimuodossa voidaankin väittämän osista kuvata ainoastaan objekti – subjekti ja predikaatti pitää esittää aina URI-tunnisteena.

Kuviossa 3 on esimerkki koneelle ymmärrettävästä RDF-graafista, joka kuvaa resurssia http://www.jyu.fi. Resurssin ominaisuutta ”otsikko” voidaan kuvata Dublin Core -termillä http://purl.org/dc/elements/1.1/title. Tämän ominaisuuden arvo on teksti ”Jyväskylän yliopiston kotisivu”. Näistä osasista koostuva ihmisen ymmärtämä väittämä kokonaisuudessaan on: ”Web-sivun http://www.jyu.fi otsikko on Jyväskylän yliopiston kotisivu.”

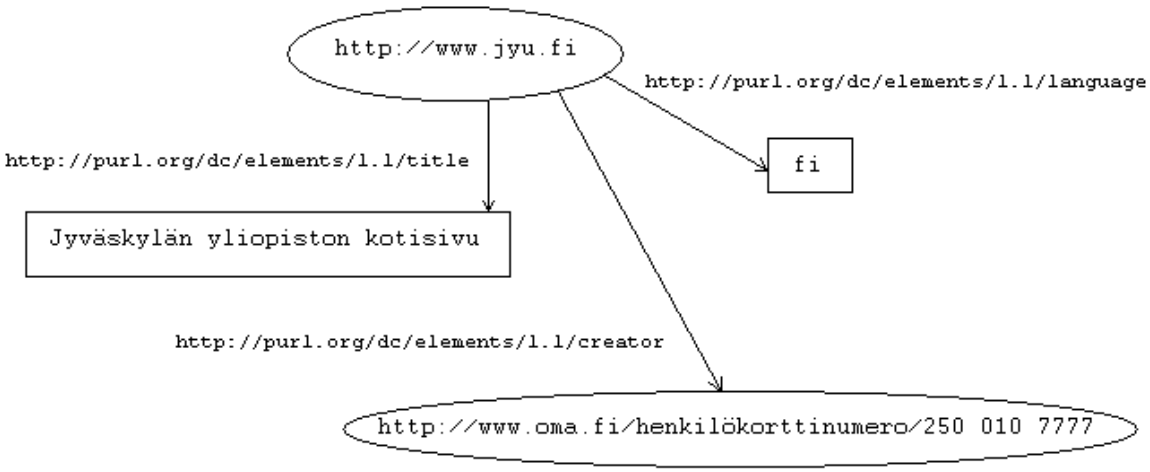
KUVIO 3. RDF-graafi.

Joskus graafien piirtäminen monimutkaisista väittämistä on hankalaa. Vaihtoehtoinen tapa väittämien esittämiseen ovat *kolmikot* (triples). Kolmikko sisältää subjektin, predikaatin ja objektin tässä järjestyksessä. Kolmikko-notaatiossa URI-viitteet pitää kirjoittaa kulmasulkeissa (<- ja >-merkit). Tekstimuotoiset objektisolmut kirjoitetaan lainausmerkkeihin. Kuvio 3 voidaan esittää kolmikko-notaationa esimerkin 1 mukaisesti.

ESIMERKKI 1. Kolmikko-esitystapa.

```
<http://www.jyu.fi> <http://purl.org/dc/elements/1.1/title> ”Jyväskylän yliopiston kotisivu”.
```

Samasta resurssista voidaan tehdä niin monta väittämää kuin halutaan. Jos esimerkiksi resurssista http://www.jyu.fi halutaan vielä kertoa sivuston tekijä ja se, että se on suomenkielinen, voidaan kaikki resurssia koskevat väittämät esittää kuvion 4 tapaan. Sivuston tekijä on tässä identifioitu URI:lla, joka sisältää henkilökortin numero.



KUVIO 4. Useiden väittämien esittäminen samasta resurssista.

Eri ominaisuussanastojen termit erotetaan toisistaan *nimiavaruuksien* avulla. Esimerkiksi kaikki elementit ja attribuutit, jotka alkavat nimiavaruuden `dc:`-etuliitteellä, kuuluvat Dublin Coren nimiavaruuteen eli Dublin Coren määrittämään sanastoon (URI: http://purl.org/dc/elements/1.1/). Nimiavaruuksien avulla käytetylle sanastolle voidaan antaa tietty merkitys ja erottaa kahdessa eri sovelluksessa käytetyt termit. Esimerkiksi sanaa ”osoite” voitaisiin käyttää merkityksessä ”jonkun ihmisen lähiosoite ja kadun numero” ja toisessa sovelluksessa taas merkityksessä ”www-sivun osoite”.

Täydet URI-viitteet voidaan korvata lyhennetyllä muodolla, niin sanotulla tarkennetulla nimellä (XML qualified name, QName). Tarkennettu nimi sisältää *etuliitteen* (prefix) ja *paikallisen nimen* (local name), esimerkiksi `esimerkki:täällä`. Etuliite nimeää URI-tunnisteen. Esimerkiksi jos etuliite `esimerkki` nimeää URI:n

http://www.esimerkki.fi/jotain/, tarkennettu nimi `esimerkki:täällä` voisi olla lyhenne URI-tunnisteelle http://www.esimerkki.fi/jotain/täällä.

Tämän tutkielman esimerkeissä käytetään muun muassa seuraavassa listassa lueteltuja etuliitteitä. Listassa olevat etuliitteet `rdf:` ja `rdfs:` viittaavat RDF:n omaan sanastoon.

- etuliite `rdf`, URI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- etuliite `rdfs`, URI: `http://www.w3.org/2000/01/rdf-schema#`
- etuliite `dc`, URI: `http://purl.org/dc/elements/1.1/`
- etuliite `oma`, URI: `http://www.oma.fi/termit`
- etuliite `xsd`, URI: `http://www.w3.org/2001/XMLSchema#`

Käyttämällä dc-etuliitettä ja kolmikkoesitystapaa, kuvio 3 voidaan esittää esimerkin 2 mukaisesti.

ESIMERKKI 2. URI:n lyhentäminen tarkennetun nimen avulla.

```
<http://www.jyu.fi> dc:title ”Jyväskylän yliopiston kotisivu”.
```

3.4 RDF/XML-syntaksi

RDF:lle on olemassa XML-syntaksi, jota kutsutaan nimellä RDF/XML. Esimerkissä 3 on annettu kuvion 3 mukaista graafia vastaava RDF/XML-kuvaus, joka selitetään jäljempänä tulevassa kappaleessa. Rivinumerot eivät kuulu syntaksiin, vaan ne ovat selityksen apuna.

ESIMERKKI 3. RDF/XML-syntaksi kuvion 3 graafille.

```
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.jyu.fi”>
5.       <dc:title>Jyväskylän yliopiston kotisivu</dc:title>
6. </rdf:Description>
7. </rdf:RDF>
```

Rivillä 1 on XML-määritys (XML declaration), joka kertoo, että sisältö on XML-muotoista. Rivi 2 alkaa `rdf:RDF`-elementillä, joka kertoo, että esimerkissä oleva sisältö on RDF:ää. Sen jälkeen tulevat nimiavaruuksien määrittelyt. Tässä on määritelty RDF:n ja Dublin Coren nimiavaruudet etuliitteillä `rdf:` ja `dc:`. Rivillä 4 elementti `rdf:Description` osoittaa resurssin kuvauksen alun. Elementin sisällä oleva `rdf:about`-attribuutti määrittää RDF-väittämän subjektina olevan resurssin URI-tunnisteen. Rivillä 5 on ominaisuuselementti `dc:title`. Tämän ominaisuus-elementin sisältönä on RDF-väittämän tekstimuotoinen objekti ”Jyväskylän yliopiston kotisivu”. Ominaisuuselementin määrittäminen `rdf:Description`-elementin sisälle kertoo, että ominaisuuselementti viittaa `rdf:about`-attribuutin ilmaisemaan resurssiin. Lopuksi kuvaus suljetaan `rdf:RDF`-loppuelementillä.

Kuvion 4 ilmaiseva RDF/XML-kuvaus useille väittämille on vastaavasti esimerkin 4 mukainen. Esimerkissä `rdf:Description`-elementit voitaisiin toistaa kaikille kolmelle väittämälle, mutta tässä on käytetty lyhennettyä muotoa (abbreviation), jossa kaikki ominaisuuselementit on merkitty saman `rdf:Description`-elementin sisään. Huomaa, että kun väittämän objektilla on oma URI, kuten tekijä-ominaisuuden arvolla tässä esimerkissä, URI merkitään `rdf:resource`-attribuutin arvoksi. Attribuutti `rdf:resource` osoittaa, että objekti on toinen resurssi.

ESIMERKKI 4. Useiden väittämien esittäminen samasta resurssista.

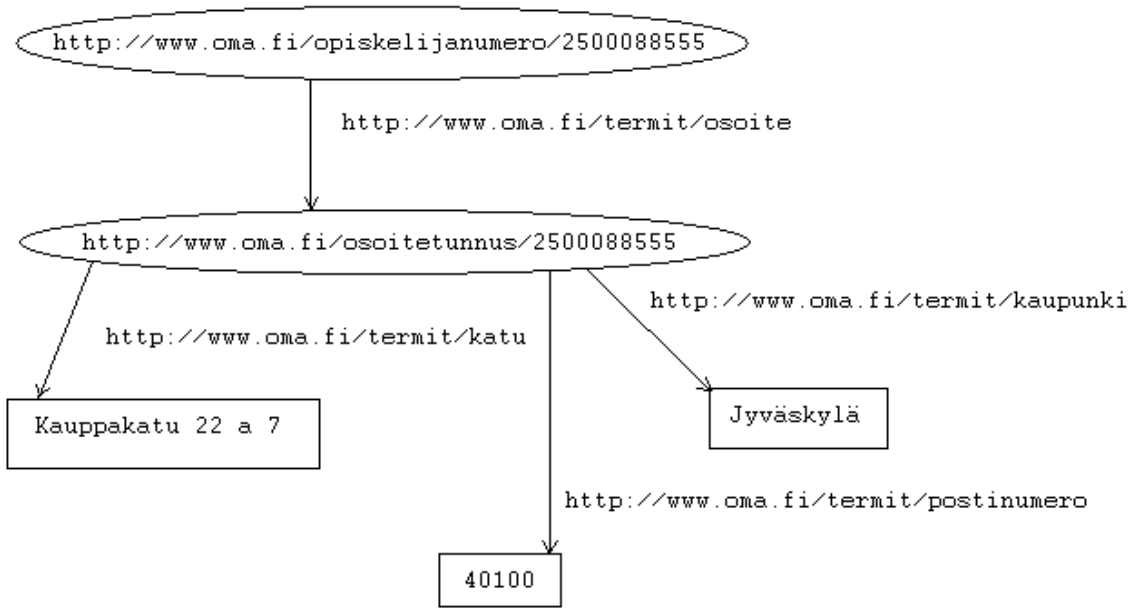
```
<?xml version=”1.0”?>
<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:dc=”http://purl.org/dc/elements/1.1/”>

<rdf:Description rdf:about=”http://www.jyu.fi”>
   <dc:title>Jyväskylän yliopiston kotisivu</dc:title>
   <dc:creator   rdf:resource=”http://www.oma.fi/henkilökorttinumero/250
   010 7777”/>
   <dc:language>fi</dc:language>
</rdf:Description>
</rdf:RDF>
```

3.5 Moniosaiset ominaisuusarvot ja tyhjät solmut (blank nodes)

Yksinkertaisimmat RDF-kuvaukset eivät riitä, kun halutaan esittää monimutkaisempia arvoja, joissa yhteen ominaisuuteen voi kuulua monta osaa. Esimerkiksi osoitteeseen kuuluu lähiosoite, kadun numero, postinumero ja postitoimipaikka tai päivämäärään päivä, kuukausi ja vuosi. Tällaiset rakenteiset ominaisuuden arvot voidaan RDF:ssä esittää erillisen *”väliresurssi”* (intermediate URIref) avulla (tässä esimerkissä osoite). Näin kaikkiin osaarvoihin pystytään myöhemmin viittaamaan erikseen.

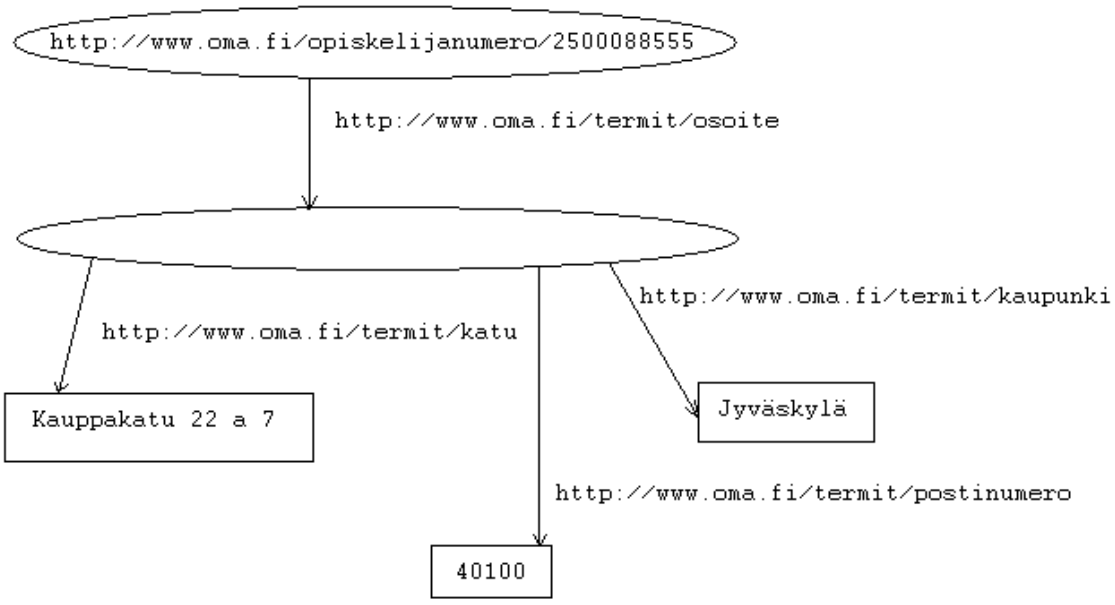
Kuviossa 5 on esitetty erään opiskelijan osoite muodostamalla sille erikseen väliresurssi http://www.oma.fi/osoitetunnus/2500088555 ja liittämällä siihen kolme osoitteen osien arvoa tekstimuodossa. Opiskelijaan viitataan URI-tunnisteella, joka sisältää opiskelijan opiskelijanumeron.



KUVIO 5. Moniarvoisen ominaisuuden esittäminen RDF-graafissa.

Tällainen moniarvoisten ominaisuuksien esittäminen voi synnyttää valtavia määriä URI-viitteellisiä väliresursseja, kuten tässä osoitesolmussa oleva URI http://www.oma.fi/osoitetunnus/2500088555. Jos tällaiseen välilliseen URI-viitteeseen ei

ole erikseen tarvetta viitata myöhemmin, RDF-kuvaus voidaan esittää myös ilman sitä. Kuviossa 6 kuvion 5 osoite esitetään tyhjänä solmuna (blank node).



KUVIO 6. Moniarvoisen ominaisuuden esittäminen tyhjänä solmuna.

Kolmikkoesitystavassa tyhjät solmut ilmaistaan esimerkin 5 mukaisesti. Tyhjä solmu esitetään *tyhjän solmun tunnisteen* (blank node identifier) avulla. Tälle tunnisteelle keksitään sopiva nimi ja se erotetaan muista _:-merkinnällä.

ESIMERKKI 5. Tyhjän solmun ilmaiseminen kolmikko-esitystapana (kuvion 6 RDF-graafin mukaisesti).

```
omaopiskelija:2500088555   oma:osoite          _:matinosoite
_:matinosoite              oma:katu            ”Kauppakatu 22 a 7”
_:matinosoite              oma:postinumero     ”40100”
_:matinosoite              oma:kaupunki        ”Jyväskylä”
```

RDF/XML tarjoaa muutamia erilaisia esitystapoja tyhjille solmuille. Suorin tapa on esittää tyhjät solmut antamalla niille tyhjän solmun tunniste. Tunniste esitetään RDF/XML-

dokumentissa `rdf:nodeID`-attribuutin avulla.

Esimerkissä 6 on kuvattu kuvion 6 tiedot RDF/XML-syntaksin muodossa. Tyhjän solmun tunnisteena käytetään merkkijonoa ”matinosoite”. Esimerkissä tyhjä solmu osoitetaan olevan ensimmäisen väittämän objektina rivillä 6 ja toisen väittämän subjektina rivillä 8 (kuvion 6 mukaisesti).

ESIMERKKI 6. Tyhjän solmun ilmaiseminen RDF/XML-syntaksilla (kuvion 6 RDF-graafin mukaisesti).

```
1. <?xml version=”1.0”?>
2. <rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
3. xmlns:oma=”http://www.oma.fi/termit/”>

4. <rdf:Description
5. rdf:about=”http://www.oma.fi/opiskelijanumero/2500088555”>
6.       <oma:osoite rdf:nodeID=”matinosoite”/>
7. </rdf:Description>

8. <rdf:Description rdf:nodeID=”matinosoite”>
9.   <oma:katu>Kauppakatu 22 a 7 </oma:katu>
10.   <oma:postinumero>40100</oma:postinumero>
11.   <oma:kaupunki>Jyväskylä</oma:kaupunki>
12. </rdf:Description>

13. </rdf:RDF>
```

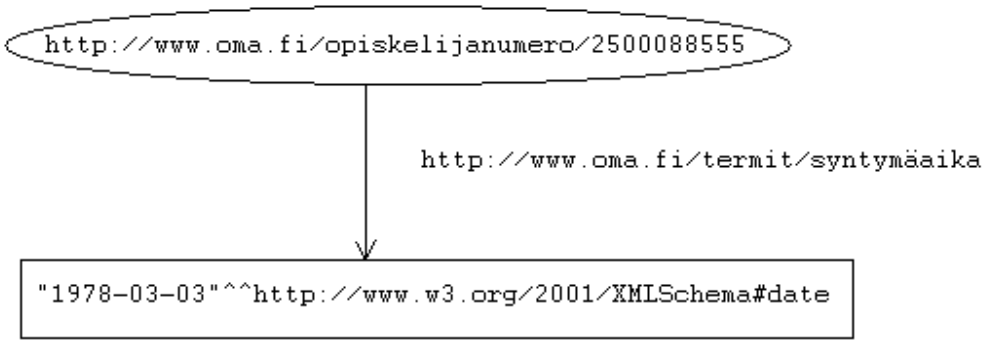
Tyhjiä solmuja ei pidetä RDF-kuvauksen todellisina osina URI-viitteiden ja merkkijonojen tapaan. Tyhjien solmujen tunnisteita kolmikko- tai RDF/XML-esitystavoissa voidaan käyttää vain paikallisesti ja siksi niihin ei voi viitata dokumentin ulkopuolella.

Tyhjät solmut tarjoavat joissain tilanteissa tarkemman tavan ilmaista reaalimaailman asioita kuin URI-viitteiset solmut. Esimerkiksi henkilön kuvaaminen käy luonnollisemmin tyhjän solmun avulla. Henkilö itsessään ei ole URI-viite, kuten hänen kotisivunsa on. Hän ei ole myöskään pelkkä merkkijono, kuten hänen nimensä on, joten tyhjä solmu kuvaa henkilöä resurssina paremmin kuin URI-viite tai merkkijono.

3.6 Tietotyypit RDF:ssä

RDF tarjoaa vain yhden sisään rakennetun tietotyypin `rdf:XMLLiteral`:n, joka esittää tekstimuotoista XML-tietoa. Muuten RDF:ssä joudutaan viittaamaan muualla määriteltyihin tietotyyppeihin, esimerkiksi XML Scheman määrityksiin, joita suositellaan käytettävän. Tekstimuotoista ominaisuusarvoa, jonka tietotyyppi on määritelty, sanotaan RDF:ssä *tyypitetyksi merkkijonoksi* (typed literal).

Kuviossa 7 on esitetty opiskelijan syntymäaika käyttämällä XML Scheman date-tietotyyppiä. RDF-väittämän objekti eli syntymäaika on esitetty tekstimuotoisena solmuna, jossa ovat päivämäärä ja tietotyypin URI-viite merkeillä ^^ erotettuna.



KUVIO 7. XML Scheman tietotyypin käyttäminen RDF-väittämässä.

RDF/XML:ssä tietotyypit esitetään `rdf:datatype`-attribuutin avulla. Kuvio 7 voidaan esittää RDF/XML-muodossa esimerkin 7 mukaan. Esimerkissä itse päivämäärä on `oma:syntymäaika`-ominaisuuselementin arvona. Ominaisuuselementillä on `rdf:datatype`-attribuutti, joka määrittelee tietotyypin. Attribuuttien arvoja ei voi lyhentää nimiavaruuden etuliitteen (`xsd:`) avulla, joten siksi esimerkissä on XML Schemaan viittaava URI täytynyt kirjoittaa kokonaisena.

ESIMERKKI 7. Tietotyyppien esittäminen.

```
<?xml version=”1.0”?>
<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:oma=”http://www.oma.fi/termit/”>

<rdf:Description
rdf:about=”http://www.oma.fi/opiskelijanumero/2500088555”>
   <oma:syntymäaika
   rdf:datatype=”http://www.w3.org/2001/XMLSchema#date”>1978-03-
   03</oma:syntymäaika>
</rdf:Description>

</rdf:RDF>
```

Vaikka URI-muotoisten attribuutin arvoja ei voikaan lyhentää suoraan nimiavaruuden etuliitteen avulla, RDF/XML:ssä URI-tunnisteita voidaan lyhentää *XML-entiteettien* avulla. Tällöin kokonaiseen XML Schemaan viittaavan URI-tunnisteen voi kirjoittaa lyhyemmässä muodossa `&xsd;`. Entiteetti voidaan määritellä esimerkkiin 7 esimerkin 8 mukaisesti. Tätä lyhennystapaa käytetään myöhemmissäkin esimerkeissä.

ESIMERKKI 8. URI:n lyhentäminen XML-entiteetin avulla.

```
<?xml version=”1.0”?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd ”http://www.w3.org/2001/XMLSchema#”>]>

<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:oma=”http://www.oma.fi/termit/”>

<rdf:Description
rdf:about=”http://www.oma.fi/opiskelijanumero/2500088555”>
   <oma:syntymäaika rdf:datatype=”&xsd;date”>1978-03-03
   </oma:syntymäaika>
</rdf:Description>

</rdf:RDF>
```

3.7 URI:n lyhentäminen (rdf:ID-määrite) ja resurssin tyypin ilmaiseminen (rdf:type-määrite)

Tähän mennessä URI-tunnisteiden lyhentämistapoina on esitelty kappaleessa 2.3. tarkennetut nimet (nimiavaruuden etuliitteet) ja kappaleessa 2.6. XML-entiteetit. Myös tässä kappaleessa esiteltäviä määritteitä `rdf:ID` ja `rdf:type` voidaan käyttää tähän tarkoitukseen. Määritettä `rdf:type` käytetään lisäksi resurssin tyypin ilmaisemiseen.

Määrite `rdf:ID` sijoittuu kuvauksissa `rdf:Description`-elementin attribuutiksi esimerkin 9 mukaisesti. Esimerkissä 9 on esimerkin 8 opiskelijaresurssin URI ilmaistu `rdf:ID`-määritteellä. Määritteen arvona on osatunniste (tässä ”2500088555”), joka viittaa yleensä kyseisen dokumentin URI-viitteeseen. Jos halutaan viitata dokumentin ulkopuoliseen URI-viitteeseen, kuten esimerkissä 9, se ilmaistaan `xml:base`-määritteellä muiden nimiavaruuksien määrittelyn yhteydessä. Tällöin kaikki kuvauksen suhteelliset URI:t viittaavat `xml:base`-määritteen osoittamaan ”perus-URI:iin”.

ESIMERKKI 9. URI:n lyhentäminen määritteillä rdf:ID ja xml:base.

```
<?xml version=”1.0”?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd ”http://www.w3.org/2001/XMLSchema#”>]

<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:oma=”http://www.oma.fi/termit/”
xml:base=”http://www.oma.fi/opiskelijanumero#”>

<rdf:Description rdf:ID=”2500088555”>
   <oma:syntymäaika rdf:datatype=”&xsd;date”>1978-03-03
   </oma:syntymäaika>
</rdf:Description>

</rdf:RDF>
```

Joskus halutaan kertoa resurssin olevan tiettyä tyyppiä tai kuuluvan tiettyyn kategoriaan. Tyypit, kategoriat ja luokat voidaan esittää **`rdf:type`**-määritteen avulla.

Esimerkissä 10 on kuvattu opiskelijan olevan perustutkinto-opiskelija URI-tunnisteen http://www.oma.fi/termit/Perustutkinto-opiskelija avulla. Kyseinen URI-tunniste saattaa osoittaa RDF Schema –kielellä tehtyyn sanastoon, jossa on määritelty erilaiset luokat opiskelijoille (esimerkiksi perustutkinto-opiskelija, jatko-opiskelija jne.). RDF Schema -kieli esitellään tarkemmin seuraavassa kappaleessa.

ESIMERKKI 10. Opiskelijan tyypin kuvaaminen.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:oma="http://www.oma.fi/termit/"
xml:base="http://www.oma.fi/opiskelijanumero/">

<rdf:Description rdf:ID="2500088555">
   <rdf:type   rdf:resource="http://www.oma.fi/termit/Perustutkinto-
   opiskelija"/>
   <oma:syntymäaika rdf:datatype="&xsd;date">1978-03-03
   </oma:syntymäaika>
</rdf:Description>

</rdf:RDF>
```

On hyvin yleistä, että resursseilla on `rdf:type`-ominaisuus. Tällaisia resursseja kutsutaan *tyypitetyksi solmuiksi* (typed nodes). RDF/XML tarjoaa lyhennystavan tyypitettyjen solmujen kuvaamiseen. Tässä lyhennystavassa `rdf:type`-ominaisuus poistetaan ja `rdf:Description`-elementti korvataan `rdf:type`-määritteen arvon tarkennetulla nimellä (tässä tapauksessa `oma:Perustutkinto-opiskelija`). Esimerkki 10 voidaan merkitä tätä lyhennystä käyttäen esimerkin 11 mukaisesti. Lyhennystapaa käytetään vastedes muissakin esimerkeissä.

ESIMERKKI 11. Tyypitettyjen solmujen lyhennetty esitystapa.

```
<?xml version=”1.0”?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd ”http://www.w3.org/2001/XMLSchema#”>]

<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:oma=”http://www.oma.fi/termit/”
xml:base=”http://www.oma.fi/opiskelijanumero/”>

<oma:Perustutkinto-opiskelija rdf:ID=”2500088555”>
   <oma:syntymäaika rdf:datatype=”&xsd;date”>1978-03-03
   </oma:syntymäaika>
</oma:Perustutkinto-opiskelija>

</rdf:RDF>
```

4 RDF SCHEMA

Tässä luvussa esitellään RDF Schema siltä osin, kuin sen käsittely tutkielman kannalta on tarpeellista. Johdannon jälkeen kerrotaan kuinka RDF Scheman avulla voidaan luoda sanastoja määrittelemällä luokkia ja ominaisuuksia. Kahdessa viimeisessä alaluvussa esitellään, kuinka resurssiryhmiä voidaan ilmaista säiliö- ja kokoelmaluokkien avulla ja kuinka olemassa olevista RDF-väittämistä voidaan tallentaa lisätietoa tekemällä niistä uusia väittämiä.

Käsitettä RDF Schema käytetään tässä viittaamaan RDF Schema –spesifikaatioon ja käsitettä RDF-skeema viittaamaan johonkin tiettyyn RDF Schema –kielellä tehtyyn skeemaan, RDF Scheman ilmentymään. RDF-skeemaa voidaan joissain yhteyksissä kutsua myös RDF-sanastoksi. Tässä luvussa on käytetty päälähteenä RDF Primer –spesifikaatiota (Manola & Miller 2004), joten lähdeviitteet on merkitty vain tästä poikkeavista lähteistä.

4.1 Johdanto

RDF Schema -spesifikaation mukaan (Brickley & Guha 2004) RDF Scheman avulla voidaan määritellä RDF-kuvailuissa käytetty sanasto sekä varmistaa sen yhtenäisyys. RDF Schema (RDFS) on RDF-muotoinen syntaksiltaan ja se käyttää omaa spesifikaatiossa määriteltyä käsitteistöään.

RDF Schema määrittelee luokkia (classes) ja ominaisuuksia (properties). Luokka vastaa yleismaailmallista käsitettä kategoria tai tyyppi (Manola & Miller 2004). RDF Schema ei tarjoa sovellusriippuvaisia luokkia, kuten `oma:Henkilö` tai `oma:Opiskelija` tai sovellusriippuvaisia ominaisuuksia, kuten `oma:tekijä` tai `oma:osoite`. RDF Scheman avulla voidaan sen sijaan määritellä tällaiset luokat ja ominaisuudet, antaa niille nimet sekä osoittaa, mitä luokkia ja ominaisuuksia oletetaan käytettävän yhdessä. (Brickley & Guha 2004.)

RDF Scheman määritelmät luokista ja ominaisuuksista ovat lähellä olio-ohjelmoinnin käsitteistöä. RDF:ssä esimerkiksi resurssi voidaan määritellä jonkin luokan *ilmentymäksi.* RDF Scheman avulla luokille voidaan määritellä *hierarkia*: Esimerkiksi luokka Opiskelija voi-

daan määritellä luokan Henkilö *aliluokaksi*. Aliluokka *perii* yliluokkansa ominaisuudet samaan tapaan kuin olio-ohjelmoinnissa.

RDF Schemassa ajattelutapa eroaa kuitenkin hieman olio-ohjelmoinnin ajattelutavasta. Olio-ohjelmoinnissa määritellään luokka, johon kuuluu tietyt metodit ja ominaisuudet. (Määritellään esimerkiksi luokka Henkilö, jolle kuuluu ominaisuudet nimi ja ikä). RDF Schemassa ominaisuudet ovat itsenäisempiä ja määrittely lähteekin ominaisuudesta eikä luokasta: RDF määrittelee ominaisuuden, joka on liitettävissä joihinkin luokkiin. (Määritellään esimerkiksi ominaisuus nimi, joka voi kuulua luokille Henkilö ja Asiakirja). RDF:n ominaisuudet ovat siis globaaleja ja voivat kuulua useampiin luokkiin, kun taas olio-ohjelmoinnissa ne määritellään paikallisesti jonkin luokan sisällä. Tämän ominaisuuskeskeisen lähestymistavan etu on, että kuka tahansa voi määritellä luokille lisää ominaisuuksia ja näin laajentaa resurssien kuvauksia.

Esimerkki 12 esittää, miksi RDF Schemaa tarvitaan. Esimerkissä on RDF-kuvaus Orkesteri Soittelijat –nimisestä orkesterista. Se sisältää ominaisuudet nimi, paikkakunta, kapellimestari ja lämpötila.

ESIMERKKI 12. RDF-kuvaus eräästä orkesterista.

```
<rdf:Description rdf:about=”http://www.oma.fi/Soittelijat”>
   <oma:nimi>Orkesteri Soittelijat</oma:nimi>
   <oma:kapellimestari>68</oma:kapellimestari>
   <oma:paikkakunta>Jokilaakso</oma:paikkakunta>
   <oma:lämpötila>30 astetta</oma:lämpötila>
</rdf:Description>
```

Kapellimestari-ominaisuuden arvona on luku 68, mikä ei tietenkään ole mahdollista. Orkesterin ominaisuutena lämpötila on myös mahdoton reaalimaailmassa. RDF/XML-jäsentimen mukaan kuvaus on kuitenkin täysin hyväksyttävä. RDF Scheman avulla voidaan tällaisissa tapauksissa tarkasti määrittää, mitkä ominaisuudet ovat sopivia tietylle resurssille (tässä esimerkiksi orkesterille sopii ominaisuus kapellimestari, mutta ei ominaisuus lämpötila) ja

mitkä arvot ovat sopivia tietylle ominaisuudelle (tässä esimerkiksi ominaisuudelle kapellimestari ei sovi arvo 68).

4.2 Luokat (Classes)

RDF Scheman käsite luokka vastaa yleiskäsitteitä kategoria tai tyyppi. Luokat määritellään määritteiden `rdfs:Class`, `rdf:type` ja `rdfs:subClassOf` avulla. RDF Schemassa luokka on mikä tahansa resurssi, jonka ominaisuuden `rdf:type` arvo on `rdfs:Class`. Resurssit, jotka kuuluvat tiettyyn luokkaan, ovat luokan ilmentymiä. Resurssi voi olla yhden tai useamman luokan ilmentymä. Täten RDF:ssä *moniperintä* on mahdollista.

Otetaan esimerkiksi soitinliike, joka myy monia eri soittimia. Kuvatakseen tuotteitaan, soitinliike tarvitsee ensin luokan edustamaan koko tuoteryhmää. Viitataan tähän luokkaan tarkennetulla nimellä `omasoitin:Soitin` ja kuvataan tätä resurssia `rdf:type`-ominaisuudella, jonka arvo on `rdfs:Class`. Tämä RDF-väittämä on kuvattuna esimerkissä 13.

ESIMERKKI 13. Luokan määritteleminen RDF Schemassa.

```
omasoitin:Soitin    rdf:type     rdfs:Class
```

Ominaisuutta `rdf:type` käytetään yllä olevan luokkamäärityksen lisäksi ilmaisemaan, että resurssi on jonkin luokan ilmentymä. Resurssi `omat:pasuuna123` voidaan kuvata luokan `omasoitin:Soitin` ilmentymäksi esimerkin 14 väittämällä.

ESIMERKKI 14. Luokan ilmentymän määrittäminen.

```
omat:pasuuna123     rdf:type     omasoitin:Soitin
```

Soitinliikkeessä myydään monia erilaisia soittimia, joita voidaan kuvata luokilla: `omasoitin:Pasuuna`, `omasoitin:Rummut`, `omasoitin:Viulu` jne. Nämä ovat eräänlaisia soittimia, joten ne voidaan kuvata luokan `omasoitin:Soitin` *aliluokkina*. Luokka voi olla yhden tai useamman luokan aliluokka (esimerkiksi Luokka Rumpukapula voi kuulua luokkiin Rummut ja MuutTarvikkeet). Aliluokat kuvataan RDF:ssä määritteen `rdfs:subClassOf` avulla.

Esimerkissä 15 on RDF-skeema, joka määrittelee soitinliikkeen tuotteiden pasuuna, viulu ja rummut luokkahierarkian. Aluksi määritellään Soitin-luokka ja sitten määritellään soittimet tämän luokan aliluokiksi. Esimerkissä on käytetty `rdf:type`-määritteen lyhennystapaa, joka esitettiin kappaleessa 2.6.

ESIMERKKI 15. RDF-skeema soittimista.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.oma.fi/skeemat/soittimet">

<rdfs:Class rdf:ID="Soitin"/>

<rdfs:Class rdf:ID="Pasuuna"/>
<rdfs:subClassOf rdf:resource="#Soitin"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Viulu"/>
<rdfs:subClassOf rdf:resource="#Soitin"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Rummut"/>
<rdfs:subClassOf rdf:resource="#Soitin"/>
</rdfs:Class>

</rdf:RDF>
```

4.3 Ominaisuudet (Properties)

Ominaisuudet kuvaavat luokkia. Esimerkiksi luokkaa `omasoitin:Pasuuna` voisi kuvata ominaisuus `oma:omistaja`. RDF Schemassa ominaisuudet kuvataan määritteillä `rdf:Property`, `rdfs:domain`, `rdfs:range` ja `rdfs:subPropertyOf`.

Kaikki ominaisuudet ovat luokan `rdf:Property` ilmentymiä. Esimerkki 16 kuvaa `omasoitin:omistaja`-ominaisuuden `rdf:Property`-luokan ilmentymäksi.

ESIMERKKI 16. Ominaisuuden määritteleminen RDF Schemassa.

```
omasoitin:omistaja  rdf:type     rdf:Property
```

RDF Schemassa voidaan määritellä, miten luokat ja ominaisuudet liittyvät toisiinsa. Määritettä **`rdfs:range`** käytetään osoittamaan, minkälaisia arvoja ominaisuus voi saada. Arvoväli määritellään jonkin luokan avulla. Määrite `rdfs:range` siis antaa arvoalueen RDF-väittämän objektille. Esimerkiksi soitinliike voisi haluta määritellä, että ominaisuuden `omasoitin:omistaja` arvot ovat luokan `omasoitin:Henkilö` ilmentymiä. Tämä voidaan kirjoittaa RDF-väittämien muotoon esimerkin 17 mukaisesti.

ESIMERKKI 17. Ominaisuuden arvovälin rajaaminen.

```
omasoitin:Henkilö   rdf:type     rdfs:Class
omasoitin:omistaja  rdf:type     rdf:Property
omasoitin:omistaja  rdfs:range   omasoitin:Henkilö
```

Esimerkin 17 väittämät ilmaisevat, että resurssi `omasoitin:Henkilö` on luokka, `omasoitin:omistaja` on ominaisuus ja että `omasoitin:omistaja`-ominaisuudella kuvattavat resurssit ovat Henkilö-luokan ilmentymiä.

Ominaisuus **`rdfs:domain`** ilmaisee minkä tyyppistä resurssia ominaisuus voi kuvata tai toisin sanoen minkä luokan kanssa ominaisuutta voidaan käyttää. Resurssin tyyppi ilmaistaan siis luokan avulla. Voidaan esimerkiksi määrittää esimerkin 18 mukaisesti, että ominaisuus `omasoitin:omistaja` voi kuvata luokan `omasoitin:Soitin`-tyyppisiä resursseja.

ESIMERKKI 18. Ominaisuuteen liittyvän resurssin tyypin määritteleminen.

```
omasoitin:Soitin    rdf:type     rdfs:Class
omasoitin:omistaja  rdf:type     rdf:Property
omasoitin:omistaja  rdfs:domain  omasoitin:Soitin
```

Esimerkin 18 väittämät määrittelevät, että resurssi `omasoitin:Soitin` on luokka, `omasoitin:omistaja` on ominaisuus ja että `omasoitin:omistaja`-ominaisuudella kuvattavat resurssit ovat soittimia eli `omasoitin:Soitin`-luokan ilmentymiä.

Samoin kuin luokille, myös ominaisuuksille voidaan määritellä *aliominaisuuksia* **subPropertyOf**-määritteen avulla. RDF Schemassa aliominaisuus perii yliominaisuutensa ominaisuudet (esimerkiksi range- ja domain -määritykset). Ominaisuus voi olla yhden tai useamman ominaisuuden aliominaisuus.

Esimerkiksi soittimelle voidaan määritellä ominaisuus `omasoitin:käyttäjä`, jonka aliominaisuus on `omasoitin:pääkäyttäjä`. Nämä ominaisuudet voisivat kuvata esimerkiksi tilanteen, jossa jonkun orkesterin omistuksessa oleva soitin on jollain jäsenellä lainassa. Tämä jäsen on soittimen pääkäyttäjä, mutta soittimella voi olla myös useita muita käyttäjiä, jotka lainaavat soitinta satunnaisesti. Ominaisuudet `omasoitin:käyttäjä` ja `omasoitin:pääkäyttäjä` sekä niiden aliominaisuudet määritellään esimerkissä 19.

ESIMERKKI 19. Aliominaisuuden määritteleminen.

```
omasoitin:käyttäjä        rdf:type     rdf:Property
omasoitin:pääkäyttäjä     rdf:type     rdf:Property
omasoitin:pääkäyttäjä     rdfs:subPropertyOf  omasoitin:käyttäjä
```

Esimerkissä 20 on kokonainen RDF-skeema, jossa on kaikki tähän asti esillä olleet määritelmät soitinten hierarkiasta ja niiden ominaisuuksista. Lisäksi esimerkissä määritellään tietotyyppi `xsd:date` ja ominaisuus `omasoitin:huollettu`. RDF Schema luokalla **`rdfs:Datatype`** määritellään, että `xsd:date` on tietotyyppi. Tätä tietotyyppiä käytetään `omasoitin:huollettu`-ominaisuudessa. Ominaisuuden `omasoitin:huollettu` range-määrite kuvaa, että kaikki ominaisuuteen liittyvät arvot ovat tietytyypiltään xsd:date-tyyppisiä ja domain-ominaisuus kuvaa, että `omasoitin:huollettu`-ominaisuutta käytetään viittaamaan Soitin-tyyppisiin resursseihin. Kyseinen RDF-skeema luo nimiavaruuden http://www.oma.fi/skeemat/soittimet, joka on määritelty skeemassa `xml:base`-määritteellä.

ESIMERKKI 20. RDF-skeema soittimista ja niiden ominaisuuksista.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.oma.fi/skeemat/soittimet#">

<rdfs:Class rdf:ID="Soitin"/>

<rdfs:Class rdf:ID="Pasuuna">
<rdfs:subClassOf rdf:resource="#Soitin"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Viulu">
<rdfs:subClassOf rdf:resource="#Soitin"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Rummut">
<rdfs:subClassOf rdf:resource="#Soitin"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Henkilö"/>

<rdfs:Datatype rdf:about="&xsd;date"/>

<rdf:Property rdf:ID=”nimi”>
  <rdfs:domain rdf:resource="#Henkilö"/>
</rdf:Property>

<rdf:Property rdf:ID="omistaja">
  <rdfs:domain rdf:resource="#Soitin"/>
</rdf:Property>

<rdf:Property rdf:ID="käyttäjä">
  <rdfs:domain rdf:resource="#Soitin"/>
  <rdfs:range rdf:resource="#Henkilö"/>
</rdf:Property>

<rdf:Property rdf:ID="pääkäyttäjä">
  <rdfs:subPropertyOf rdf:resource="#käyttäjä"/>
</rdf:Property>

<rdf:Property rdf:ID="huollettu">
  <rdfs:domain rdf:resource="#Soitin"/>
  <rdfs:range rdf:resource="&xsd;date"/>
</rdf:Property>

</rdf:RDF>
```

Nyt kun luokat ja ominaisuudet on kuvattu esimerkin 20 soittimet-skeemassa, skeeman luokkien ja ominaisuuksien ilmentymiä voidaan esittää RDF-kuvauksissa. Esimerkissä 21 kuvataan luokan `omasoitin:Pasuuna` ilmentymä ja sen ominaisuuksia. Koska ominaisuuden `omasoitin:käyttäjä` arvo on skeemassa määritelty range-määritteen avulla Henkilö-tyyppiseksi, pitää Henkilö ”Kalle Kiiski” määritellä kuvauksessa erikseen Henkilö-luokkaan kuuluvaksi, jotta kuvaus on skeeman mukainen. Soittimet-skeema liitetään RDF-kuvaukseen nimiavaruusmäärittelyn `xmlns:omasoitin` kautta (esimerkissä mustattuna). Nimiavaruusmäärittelyllä osoitetaan skeeman URI.

ESIMERKKI 21. Soittimet-skeemaa vastaava RDF-kuvaus.

```
<?xml version=”1.0”?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd ”http://www.w3.org/2001/XMLSchema#”>]>
<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:omasoitin="http://www.oma.fi/skeemat/soittimet#"
xml:base="http://www.oma.fi/asiat">

<omasoitin:Henkilö rdf:about=”
http://www.oma.fi/opiskelijanumero/2500088555”>
   <omasoitin:nimi>Kalle Kiiski</omasoitin:nimi>
</omasoitin:Henkilö>

<omasoitin:Pasuuna rdf:ID=”pasuuna123”>
   <omasoitin:omistaja>Orkesteri       Soittelijat</omasoitin:omistaja>
   <omasoitin:käyttäjä rdf:resource=”
   http://www.oma.fi/opiskelijanumero/2500088555”>
   </omasoitin:pääkäyttäjä>
   <omasoitin:huollettu rdf:datatype="&xsd;date">1994-03-
   06</omasoitin:huollettu>
</omasoitin:Pasuuna>

</rdf:RDF>
```

Määrittely `rdf:ID=”pasuuna123”` viittaa URI:iin http://www.oma.fi/asiat/pasuuna123 `xml:base`-määrityksen mukaisesti. Esimerkin 20 soittimet-skeema kuvaa ominaisuuden `omasoitin:huollettu` arvon `xsd:date`-tyyppiseksi, joten ominaisuuden arvo RDF-kuvauksessa täytyy olla tämän tietotyypin mukainen vastatakseen skeemaa. (Tietotyypin

range-määrittely skeemassa ei siis automaattisesti nimeä tekstimuotoisen arvon tietotyyppiä, vaan tietotyyppi pitää merkitä eksplisiittisesti näkyviin RDF-kuvauksessa).

4.4 Resurssiryhmien esittäminen RDF:ssä

Monesti halutaan viitata useampaan resurssiin yhtä aikaa. Halutaan esimerkiksi ilmaista, että jollakin artikkelilla on useampia kirjoittajia. Tällaisia resurssiryhmiä voidaan RDF:ssä esittää tietorakenteiden kaltaisilla säiliö- ja kokoelmaluokilla. Nämä luokat esitellään seuraavassa kahdessa alaluvussa.

4.4.1 Säiliöluokat (Containers)

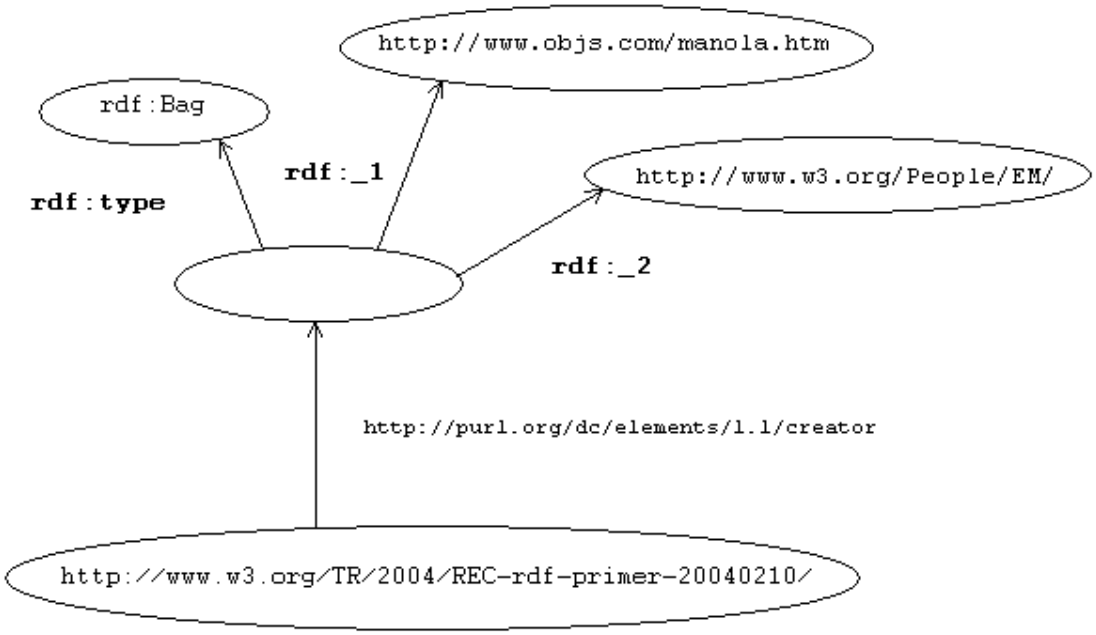
Säiliö (Container) on resurssi, joka sisältää ryhmän asioita. Säiliössä olevia asioita kutsutaan *jäseniksi* (member). Jäsenet voivat olla toisia resursseja tai tyhjiä solmuja.

RDF:ssä on kolme säiliöluokkaa: Bag (pussi), Sequence (sarja) ja Alternative (valintalista). *Pussi* on järjestämätön ryhmä asioita ja *sarja* järjestetty ryhmä. Sekä pussi että sarja voivat sisältää toistuvia arvoja. *Valintalista* on ryhmä, joka esittää vaihtoehtoja yksittäiselle arvolle tai ominaisuudelle. Valintalistaa käytetään usein esimerkiksi kielivarianttien esittämisessä.

Säiliöluokka merkitään RDF/XML:ssä rdf:type-ominaisuuden avulla. Säiliön jäsenet voidaan merkitään *jäsenominaisuudella* (container membership property), joka on muotoa `rdf:_n` (esimerkiksi `rdf:_1`, `rdf:_2` jne.). Toinen merkintätapa jäsenominaisuudelle on `rdf:li`.

Kuviossa 8 on RDF-graafi ja sen perässä esimerkissä 22 vastaava RDF/XML-kuvaus, jotka ilmaisevat väitelauseen: ”Osoitteessa http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ olevan spesifikaation ovat kirjoittaneet Frank Manola ja Eric Miller.” Säiliöresurssina tässä esimerkissä on tyhjä solmu. Sen ominaisuus `rdf:type` osoittaa resurssin ole-

van rdf:Bag-tyyppinen. Kuviossa 8 on käytetty jäsenominaisuuksille rdf:_n-merkinätapaa ja esimerkissä 22 ne on kirjoitettu muotoon rdf:li. Kirjoittajiin on viitattu URI-viitteillä.



KUVIO 8. Viittaaminen kahteen RDF-spesifikaation kirjoittajaan rdf:Bag-säiliöluokan avulla.

ESIMERKKI 22. Säiliöluokan rdf:Bag ilmaiseminen RDF/XML:ssä.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:dc=”http://purl.org/dc/elements/1.1/”>

   <rdf:Description    about=”http://www.w3.org/TR/2004/REC-rdf-primer-
   20040210/”
          <dc:creator>
           <rdf:Bag>
            <rdf:li resource=”http://www.objs.com/manola.htm”>
            <rdf:li resource=”http://www.w3.org/People/EM/”>
           </rdf:Bag>
          </dc:creator>
   </rdf: Description>
</rdf:RDF>
```

RDF:n säiliöluokat ovat yleisiä malleja resurssiryhmien esittämiselle. Käyttäjät ovat vapaita valitsemaan myös oman tapansa kuvata resurssiryhmiä. Säiliöluokkien avulla voidaan kuitenkin parantaa sovellusten yhteistoiminnallisuutta.

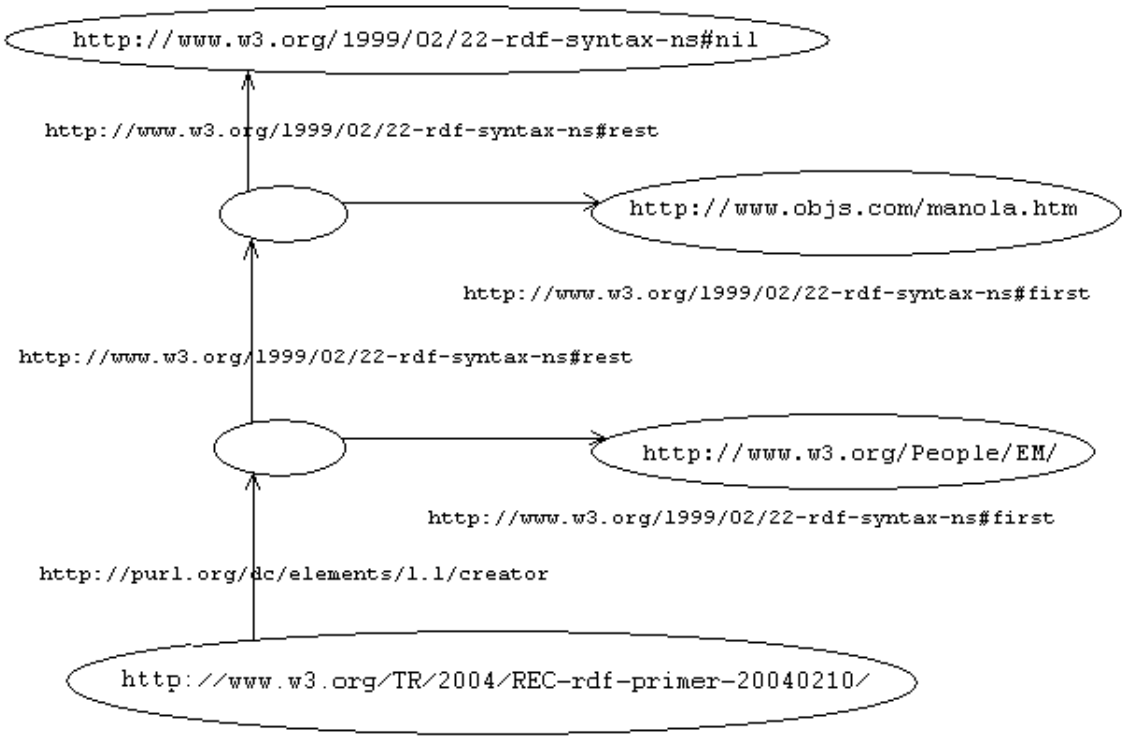
Säiliöt eivät ole varsinaisia tietorakenteita, niin kuin tietorakenteet ohjelmointikielissä. Ne vain *kuvailevat* ryhmän resursseja. Kun käytetään säiliöitä, ei ole myöskään varmaa, että jäsenet ovat ainoita jäseniä ryhmässä.

4.4.2 Kokoelma-luokat (Collection)

Kokoelma (collection) kuvaa ryhmän resursseja, niin kuin säiliökin. Kokoelmien ero säiliöihin on se, että kokoelma on suljettu: kokoelmassa olevat jäsenet ovat ainoita jäseniä ryhmässä. Sillä, onko kokoelman solmujen järjestyksellä väliä, ei määritetä RDF:ssä, vaan se on sovelluskohtaisesti määriteltävä asia. Kokoelman solmujen rakenne tulkitaan joka tapauksessa aina siinä järjestyksessä, missä solmut ovat RDF/XML-syntaksissa. RDF-kokoelma esitetään listarakenteena RDF-graafissa. Kokoelmaluokkia kuvataan ominaisuuksilla `rdf:first`, `rdf:rest` ja resurssilla `rdf:nil`.

Kuvio 9 kuvaa kokoelmasanaston käyttöä. Kuviossa on esitetty RDF Primer -spesifikaation kirjoittajat Frank Manola ja Eric Miller kokoelma-tyypin avulla. Kirjoittajiin on viitattu heidän kotisivujensa URL-osoitteiden avulla. Rakenne ilmaisee, että spesifikaation kirjoittamiseen ovat osallistuneet ainoastaan nämä kaksi henkilöä. Kokoelman jäsenet ovat `rdf:first`-ominaisuuden objekteja. Subjektina on resurssi (tässä tyhjä solmu), joka edustaa listarakennetta. Tämä listaresurssi linkitetään loppulistaan `rdf:rest`-ominaisuudella. Resurssi `rdf:nil` edustaa tyhjää listaa ja on aina viimeisenä listassa.

Tällainen listarakenne on käytössä myös Lisp-ohjelmointikielessä. Rakenteen osat helpottavat listan ohjelmallista käsittelyä. Rakenne osoittaa, että `rdf:first`-ominaisuuden arvo on listan jäsen, `rdf:rest` osoittaa seuraavan solmun paikan ja `rdf:nil`-ominaisuudella osoitetaan aina listan loppu.



KUVIO 9. RDF-spesifikaation kirjoittajat esitettynä kokoelma-tyypin avulla.

RDF/XML:ssä kokoelmat voidaan esittää attribuutilla **`rdf:parseType="Collection"`** tai pidemmän esitystavan mukaan. Tässä tutkielmassa esitetään ainoastaan lyhempi esitystapa. Attribuuttia `rdf:parseType` käytetään ilmaisemaan, että elementin sisältö tulkitaan erityisellä tavalla. Tässä tapauksessa sillä ilmaistaan, että attribuutin sisällä olevilla elementeillä on tarkoitus muodostaa kokoelman mukainen listarakenne. Esimerkissä 23 on kuvattu kuvion 9 kokoelma RDF/XML-syntaksin mukaisesti.

ESIMERKKI 23. Kokoelmaluokan ilmaiseminen RDF/XML:ssä.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:dc=”http://purl.org/dc/elements/1.1/”>

   <rdf:Description about=”http://www.w3.org/TR/2004/REC-rdf-primer-
   20040210/”
          <dc:creator rdf:parseType=”Collection”>
                 <rdf:Description
                 rdf:about=”http://www.objs.com/manola.htm”/>
                 <rdf:Description
                 rdf:about=”http://www.w3.org/People/EM/”/>
          </dc:creator>
   </rdf: Description>
</rdf:RDF>
```

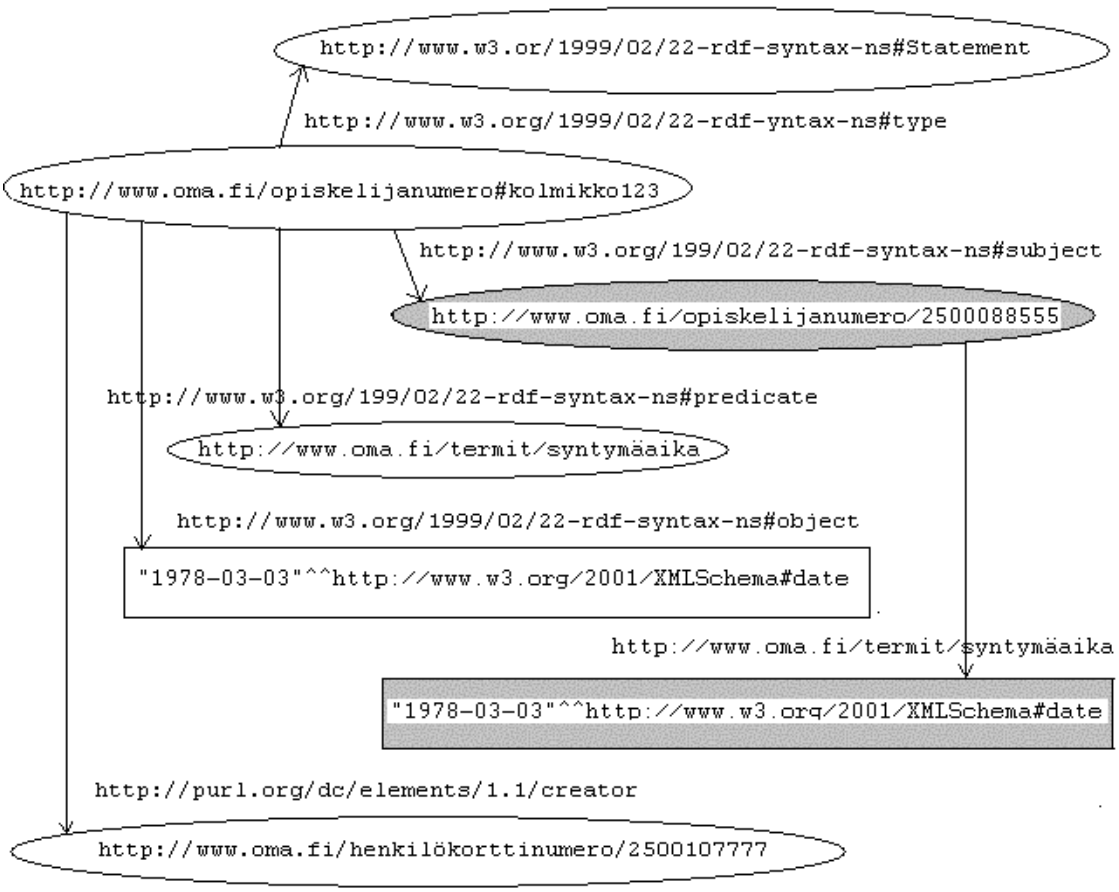
4.5 Väittämien kuvaaminen uusilla väittämillä rdf:Statement-luokan avulla

Joskus halutaan kuvailla kokonaisia RDF-väittämiä uusilla väittämillä. Tämä on mahdollista `rdf:Statement`-luokan avulla. Voidaan esimerkiksi haluta tallentaa lisätietoa siitä, milloin väittämä on tehty tai kuka sen teki. Tällaisesta väittämästä voi olla suurta hyötyä esimerkiksi alkuperäisen väittämän luotettavuuden arvioinnissa.

RDF:ssä on oma sanastonsa tällaisten väittämien kuvaamiseen. RDF-spesifikaatiossa väittämien kuvaamista tällä sanastolla kutsutaan englannin kielen termillä *reification.*

Kuviossa 10 palataan kuvion 7 (luku 3.6.) esimerkkiin opiskelijan syntymäajan esittämisestä. Kuviossa esitetään sekä alkuperäinen väittämä (opiskelija on syntynyt 3.3.1978), että uusi väittämä koskien alkuperäistä. Alkuperäisen väittämän subjekti ja objekti ovat kuvassa esitetty harmaalla pohjalla. Uusi väittämä kuvaa alkuperäistä esittämällä alkuperäisen väittämän tekijän URI-viitteellä http://www.oma.fi/henkilökorttinumero/2500107777.

Uudesta väittämästä on tehty resurssi nimeltä ”kolmikko 123”, joka ilmaistaan `rdf:type`-määritteellä olevan luokan `rdf:Statement` ilmentymä. Uudelle väittämälle määritellään subjekti, predikaatti ja objekti, jotka viittaavat alkuperäiseen väittämään. Nämä neljä väittämää ilmaistuna `rdf:type`, `rdf:subject`, `rdf:predicate` ja `rdf:object` –määritteillä ovat tyypillisesti aina tällaisessa väittämä väittämästä –rakenteessa.



KUVIO 10. Väittämän ilmaiseminen olemassa olevasta väittämästä. (Alkuperäinen väittämä harmaalla pohjalla.)

Esimerkissä 24 on kuvio 10 esitetty RDF/XML-muodossa. RDF/XML:ssä käytetään samaa sanastoa kuin RDF-graafissa.

ESIMERKKI 24. Väittämä väittämästä.

```
<?xml version=”1.0”?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd ”http://www.w3.org/2001/XMLSchema#”>]

<rdf:RDF
xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:oma=”http://www.oma.fi/termit/”
xmlsn:dc=”http://purl.org/dc/elements/1.1/”
xml:base=”http://www.oma.fi/opiskelijanumero”>

<rdf:Description rdf:ID=”2500088555”>
   <oma:syntymäaika rdf:datatype=”&xsd;date”>1978-03-03
   </oma:syntymäaika>
</rdf:Description>

<rdf:Statement rdf:about=”#kolmikko123”>
      <rdf:subject rdf:resource=”
      http://www.oma.fi/opiskelijanumero/2500088555”/>
      <rdf:predicate
      rdf:resource=http://www.oma.fi/termit/syntymäaika/>
      <rdf:object rdf:datatype=”&xsd;date”>1978-03-03</rdf:object>

      <dc:creator rdf:resource=”
      http://www.oma.fi/henkilökorttinumero/2500107777”>
</rdf:Statement>

</rdf:RDF>
```

5 RDF-MALLIN ARVIOINTI

Tutkielman kolmannessa ja neljännessä luvussa esiteltiin RDF-tietomalli, sen RDF/XML-syntaksi sekä RDF Schema RDF-spesifikaatioiden mukaisesti. Tässä luvussa arvioidaan RDF-mallia. Aluksi verrataan RDF:ää ja XML:ää keskenään ja selvennetään, miksi RDF:ää tarvitaan XML:n rinnalla. Seuraavaksi käsitellään RDF:n etuja ja mahdollisuuksia ja lopuksi sen rajoitteita ja haasteita metatietojen kuvaamisessa.

5.1 RDF-mallin vertailua XML:ään

RDF:n tarkoitus on eristää metatieto dokumentista ja esittää se sellaisessa yksinkertaisessa muodossa, johon on helppo tehdä kyselyjä. RDF ei pyri korvaamaan XML:ää, vaan rakentaa tason XML:n päälle. (Berners-Lee 1998.) Tässä kappaleessa verrataan RDF:n ja XML:n erilaisia ominaisuuksia ja tehtäviä keskenään.

RDF tarjoaa yksinkertaisen tietomallin, jota sovellusten on helppo prosessoida ja manipuloida (Klyne & Carroll 2004). XML tarjoaa tavan tiedon rakenteen esittämiseen, eikä sano mitään tiedon merkityksestä. RDF:n avulla pystytään ilmaisemaan tiedon merkitystä, tietopalasten välisiä suhteita sekä semantiikkaa periytymisestä (Hjelm 2001, 5; Volz, Oberle & Studer 2003). Asioiden välisten suhteiden tarkan kuvaamisen avulla saadaan metatietokuvaukset paremmin reaalimaailmaa vastaavaksi (Volz, Oberle & Studer 2003). RDF ja XML soveltuvatkin eri tarkoituksiin: XML tiedon rakenteen esittämiseen sekä tiedonsiirtoon sovellusten välillä ja RDF metatiedon ja semantiikan esittämiseen. XML:n avulla esitetystä rakenteisesta tiedosta puuttuu RDF:n semantiikka periytymisestä.

RDF-mallia voidaan edelleen käyttää, vaikka XML-syntaksi muuttuisi tai katoaisi. RDF on ensisijaisesti tietomalli ja sen XML-syntaksi vain yksi mahdollinen esitystapa. Tietomallina RDF on täysin riippumaton XML:stä. (Decker, Melnik, Harmelen ym. 2000.)

Jos dokumentit on esitetty XML-muodossa, metatieto pystyttäisiin yleensä poimimaan myös XML-dokumenttien elementeistä kyselyjä tekemällä. XML-dokumenttien hierarkiat

ovat joka tapauksessa monesti monimutkaisia ja syviä, joten kyselyistä tulee välttämättä monimutkaisempia verrattuna kyselyiden tekemiseen RDF-rakenteeseen. (Berners-Lee 1998.) RDF:n avulla metatieto pystytään esittämään erillään varsinaisista XML-dokumenteista ja saamaan näin kyselyt metatietoihin tehokkaammaksi. RDF-mallina esitettynä metatieto on yksinkertaisessa muodossa (Ensel & Keller 2002).

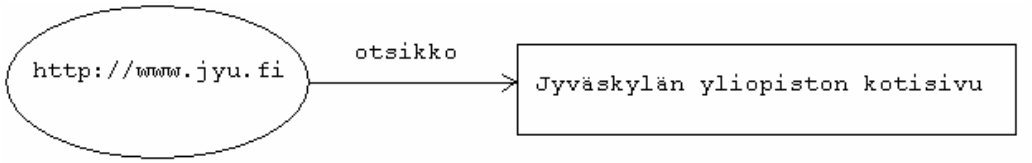
Voidaan pitää makuasiana, onko järkevämpää hakea XML-dokumenteissa oleva metatieto dokumenttien elementeistä kyselyjen avulla, vai poimia metatieto ja semantiikka dokumentista erilleen.

Seuraaviin esimerkkeihin on saatu idea Berners-Leen (1998) artikkelista. Esimerkit 26-28 kuvaavat, kuinka XML-muodossa voidaan kolmella eri tavalla esittää tutkielman alussa esitetty esimerkki: ”Sivun http://www.jyu.fi otsikko on Jyväskylän yliopiston kotisivu”. Kuvio 11 esittää saman asian RDF-mallina. Tieto voidaan XML-muodossa esittää monella eri tapaa, mikä monimutkaistaa kyselyjen tekemistä. Erilaiset XML-dokumentit voidaan esittää RDF-väittämien avulla yksikäsitteisesti, jolloin kyselyjä on helpompi tehdä. (Berners-Lee 1998.)

```
ESIMERKKI 26.
<otsikko>
     <uri>http://www.jyu.fi</uri>
     <nimi>Jyväskylän yliopiston kotisivu</nimi>
</otsikko>

ESIMERKKI 27.
<dokumentti href="http://www.jyu.fi">
   <otsikko>Jyväskylän yliopiston kotisivu</otsikko>
</dokumentti>

ESIMERKKI 28.
<dokumentti>
   <tiedot>
    <uri>href="http://www.jyu.fi"</uri>
    <otsikko>
        <nimi>Jyväskylän yliopiston kotisivu</nimi>
    </otsikko>
    </tiedot>
</dokumentti>
```



KUVIO 11. XML-esimerkkien 26-28 esittäminen RDF-mallina.

RDF Scheman etu metatietojen kuvaamisessa on sen luontainen graafirakenne ja sen suuntautuneisuus metatasoon, eikä tietotasoon XML:n tapaan. XML Schemaa voitaisiin periaatteessa käyttää myös RDF:n syntaksin kuvailemiseen, mutta koska XML Schema on tarkoitettu *puumaisen* XML-rakenteen syntaksin määrittelemiseen, *graafimallisen* RDF:n määritteleminen onnistuu helpommin RDF Schema -kielellä. (Ensel & Keller 2002.) Graafirakenteen seurauksena on, että dokumenttiosien järjestyksellä ei ole merkitystä niin kauan kuin solmut graafissa on identifioitu yksikäsitteisesti. XML kuvailee tiedon rakennetta, ja tällöin taas elementtien järjestyksestä tulee merkitsevä. (Ensel & Keller 2002; Berners-Lee 1998.) Vaikka ihmiselle puumainen järjestysrakenne helpottaa lukemista, pystyy kone hyvin selviämään graafimuodon lukemisesta niin kauan kuin solmut on identifioitu yksikäsitteisesti. (Berners-Lee 1998.)

XML-dokumenteissa kuvataan usein sama asia moneen kertaan, kun taas RDF-mallina asia pystytään esittämään ilman toistoa (Ensel & Keller 2002). Tämä perustuu siihen, että RDF-mallissa asia pystytään esittämään resurssina graafissa kerran ja samaan URI-tunnisteella identifioituun resurssiin voidaan viitata monessa eri väittämässä, kun taas XML-dokumentissa asiaa kuvaava XML-elementti pitää toistaa yleensä niin monessa paikassa kuin siihen tarvitsee viitata.

RDF:ssä ja XML:ssä eroavia asioita ovat siis tietomalli, näkökulma järjestykseen, kyselyjen muodostaminen sekä toisto elementtejä esitettäessä. Nämä asiat on koottu yhteenvedoksi tauluun 2.

TAULU 2. XML:n ja RDF:n vertailua.

	XML	***RDF***
Esitettävä tieto	Tietotaso	Metataso
Tietomalli	Puumalli	Graafi, kolmikkomalli
Järjestys	Järjestyksellä merkitystä	Järjestyksellä ei merkitystä
Kyselyjen monimutkaisuus	Kyselyt sitä monimutkaisempia, mitä syvemmälle puurakenteessa mennään.	Kyselyt yksinkertaisempia tietomallin takia.
Elementtien ja solmujen toisto	Saman elementin toisto usein monissa eri dokumenteissa	Yksi solmu voidaan kuvata yhdessä paikassa

5.2 RDF:n etuja ja mahdollisuuksia

Tässä kappaleessa arvioidaan RDF:n etuja ja mahdollisuuksia. Eduista kuvaillaan sen käyttökohteiden monipuolisuutta, XML-syntaksia, laajennettavuus- ja yhdistelymahdollisuutta sekä URI-viitteiden käyttöä. RDF:n mahdollisuuksia ovat ainakin mahdollisuus määritellä sanastoja ja metatietoskeemoja sekä toteuttaa semanttisen webin älykkäitä palveluja.

Käyttökohteiden monipuolisuus

Shen ja Yang (2004) sekä Horrocks ja Patel-Schneider (2003) esittävät RDFS:n olevan erityisen varteenotettava työväline ontologian ja tietämyksen kuvaamiseen. Shen ja Yang (2004) mainitsevat sen olevan myös lupaava tapa tiedon integrointiin tulevaisuudessa. RDF-skeemasanastojen avulla tieto voidaan kuvailla yhtenäisesti ja näin tietoa voidaan helpommin siirtää sovellusten välillä. Guhan, McCoolin ja Millerin (2003) mukaan RDF sopii tiedon semantiikan ilmaisemiseen.

Candan, Liu ja Suvarna (2001) toteavat RDF:n olevan täysin riippumaton sovellusalasta. RDF:ää voidaan käyttää esimerkiksi web-sivujen sisällön kuvaamiseen, dokumenttien versionhallintaan, sisällön luotettavuuden arviointiin, tekijänoikeuksien ja tietosuojamääritysten (privacy preferences) ilmaisemiseen ja digitaalisiin allekirjoituksiin. RDF-kuvauksia voivat käyttää hyödykseen hakukoneet ja älykkäät ohjelmistoagentit. (Candan ym. 2001.)

XML-syntaksi

XML-syntaksin ansiosta RDF-muotoista tietoa on mahdollista prosessoida ohjelmallisesti (Manola & Miller 2004). XML-syntaksista on myös monia muita etuja. XML käyttää koodistonaan Unicodea, joka on maailmanlaajuisin merkkijärjestelmä tällä hetkellä, koska se ei rajoitu vain latinalaisiin aakkosiin kuten ASCII. XML:ää on tutkittu paljon, joten ratkaisut sen tallentamiseen tietokantoihin, tavat sen siirtämiseen tietoverkoissa ja mallit sitä käyttävien sovelluksien rakentamiseen ovat jo olemassa. Sitä voidaan sanoa vakiintuneeksi tekniikaksi, koska sitä käyttäviä toimivia sovelluksia on jo niin paljon olemassa. (Hjelm 2001.) XML takaa tiedon yhtenäisen esitystavan ja täten mahdollisuuden sovellusten keskinäiseen kommunikointiin ja tiedon vaihtoon (Ebenhoch 2001; Klyne & Carroll 2004). XML-syntaksin etuna on alustariippumattomuus, jonka johdosta metatiedon jakaminen erilaisten ohjelmistojen ja laitealustojen kesken on mahdollista.

Mahdollisuus yhteisten sanastojen ja metatietoskeemojen määrittelemiseen

Deckerin, Mitran ja Melnikin (2000) mukaan tulevaisuuden internetissä sanastot ja niiden jakaminen tulevat yhä tärkeämmäksi. RDF Schema –kielellä voidaan uusia URI-pohjaisia sanastoja määritellä hyvin RDF Scheman joustavuuden ja rikkaan semantiikan takia (Decker, Mitra & Melnik 2000; Shen & Yang 2004).

Vanha tapa sanastojen määrittelemiseen ovat esimerkiksi kirjaston luokitteluskeemat. Tällaisten luokittelujen tavoite on olla mahdollisimman kattavia ja tarjota universaali keskitetty skeema, mutta ne ovat väkisinkin vain tiettyyn aihealueeseen tai kulttuuriin sopivia. RDF:ssä ratkaisuna tähän ongelmaan ovat hajautetut skeemat. RDF Schema -kielen avulla kaikki voivat luoda omia sanastojaan ja skeemojaan omiin tarkoituksiinsa. (Hjelm 2001, 4.) Hunter ja Lagoze (2001) ottavat kantaa tähän samaan asiaan: Yksinkertaiset sanastot ja standardit kuten Dublin Core eivät enää riitä, vaan tarvitaan monimutkaisempia sovelluskohtaisia metatietokuvauksia, joita RDF Schema -kielellä on mahdollista toteuttaa.

Tavalliseen luokitteluun verrattuna, RDF-kuvausten etu on se, että ilmentymä voi kuulua useampaan kuin yhteen luokkaan (Manola & Miller 2004). Tällaisen moniperinnän ansiosta

voidaan kuvata sellaisia rakenteita, jotka tavallisesti eivät olisi mahdollisia. Voidaan esimerkiksi kuvata tietyn henkilön olevan sekä luokan `Tutkija` että `Professori` ilmentymä.

Ontologioihin rinnastettavien RDF-skeemojen avulla käsitteet ja niiden suhteet voidaan esittää niin, että koneet osaavat prosessoida tietoa samaan tapaan kuin ihmiset, vaikka sovellukset tai tietokoneet eivät todellisuudessa koskaan pystykään ”ymmärtämään” tietoa. (Berners-Lee, Hendler & Lassila 2001.) Metatietoelementtien ja niiden suhteiden esittäminen formaalisti voidaan tehdä rakenteistamalla ne skeeman avulla (ISO/PDTS 23081 2003). RDF-skeemassa voidaan käsitteet määritellä luokiksi ja ominaisuuksiksi ja osoittaa näin käsitteiden merkitystä (Noy, Sintek, Decker ym. 2001).

Metatietoskeemojen hyötyjä ISO/PDTS 23081-luonnoksen (2003, 14) mukaan ovat:

- yhtenäinen metatiedon hallinta
- yhteentoimivuuden mahdollistaminen yhdistämällä erilaiset metatietojoukot
- metatietoelementtien suhteiden ja semantiikan ilmaiseminen
- yhtenäisyyden varmistaminen ja ylläpitäminen tietojärjestelmissä
- modulaarisen tietojärjestelmien kehityksen mahdollistaminen: järjestelmien hajottaminen osiin tai niiden yhdistäminen keskenään ovat mahdollisia.
- perustan tarjoaminen tietojärjestelmien tai tietokantojen kehittämiseen

Ontologioilla voidaan parantaa web-hakujen tarkkuutta, koska hakukone voi etsiä vain niitä sivuja, jotka viittaavat täsmälliseen käsitteeseen sen sijaan, että käytettäisiin monimerkityksisiä hakusanoja (Berners-Lee, Hendler & Lassila 2001). Perinteinen tiedon haku pohjautuu lähes kokonaan sanojen esiintymislukumäärään dokumenteissa. RDF-muodossa olevan eksplisiittisen semantiikan lisääminen ja rakenteisen, koneille ymmärrettävän tiedon saatavuus semanttisessa webissä tarjoaa mahdollisuuksia parantaa perinteistä tiedon hakua. (Guha, McCool & Miller 2003.) Kehittyneemmät sovellukset pystyvät käyttämään ontologioita ja päättelysääntöjä hyväkseen ja tehostamaan näin hakuja entisestään (Berners-Lee, Hendler & Lassila 2001).

Ontologioiden käyttäminen tarjoaa Fernandesin, Mouran ja Porton (2003) mukaan hyvän

mahdollisuuden parantaa merkittävästi tiedon jakamista yhteisöjen kesken. Ontologioilla tuettu tiedonvälitys voi tapahtua sekä ihmisten että koneiden välillä (Gruninger & Lee 2002). Jos ajatellaan eduskuntaa ja ministeriöitä, on jokaisessa näistä yhteisöistä oma käsitteistönsä omissa tietojärjestelmissään ja web-sivuissaan. Ihmisten omilla koneilla on tietty tiedostojen nimeämiskäytäntö ja hakemistorakenne, jotka jollain tavalla organisoivat tietoa. Kaikkea tällaista *piilotettua tietämystä* voitaisiin paremmin hyödyntää käyttämällä systemaattisesti hyödyksi metatietoa (Fernandes, Moura & Porto 2003). Vaikka tiedostonimet ja hakemistot parantavat henkilökohtaista tiedonhallintaa, erilaiset käsitteistöt eri yhteisöjen ja eri ihmisten välillä saattavat olla esteenä sisällön jakamiselle (Fernandes, Moura & Porto 2003).

Kun ontologiasanasto on kerran muodostettu, voi sitä tai sen osia Noyn ja McGuinnessin (2001) mukaan hyödyntää muuallakin kuin tiedon kuvailussa. Sanastosta voivat heidän mukaansa hyötyä esimerkiksi uudet työntekijät, joiden tavoitteena on opetella, mitä sovellusalan käsitteet tarkoittavat. Ontologiatietoa voidaan käyttää uudelleen myös tietovarastojen rakenteistamiseen ja järjestämiseen (Gruninger & Lee 2002). Sovellusalakohtaisia oletamuksia ja tietämystä on helpompi muuttaa ja ylläpitää ontologiassa verrattuna kiinteästi ohjelmointikielellä koodattuun tietoon (Noy & McGuinness 2001).

Hunter ja Lagoze (2001) toteavat, että sovellusten metatietojen yhteistoiminnallisuus helpottaa tiedon löytymistä internetistä. RDF parantaa metatietojen yhteistoiminnallisuutta sovellusten välillä, koska se tarjoaa yhteiset käytännöt metatiedon semantiikan, syntaksin ja rakenteen esittämiseen (Candan, Liu & Suvarna 2001; Hjelm 2001, 4). RDF-kuvauksissa voidaan esittää arvoja XML Schema –tietotyyppien mukaan, mikä helpottaa tiedon vaihtoa RDF- ja XML-sovellusten välillä (Klyne & Carroll 2004). Metatietojen yhteistoiminnallisuus kuitenkin edellyttää, että sovellukset hyväksyvät yhteisen tavan metatiedon esittämiseen sekä yhteisen metatietoskeeman – oli metatietoskeeman esittämistapa sitten RDF tai joku muu (Niemijärvi 2002).

Tiettyjen sovittujen käsitteiden käyttäminen tukee Noyn ja McGuinnessin (2001) mukaan järjestelmien yhteentoiminnallisuutta ja täten myös järjestelmien integrointia. Esimerkiksi

jos eduskunnan ja valtioneuvoston web-sivut jakaisivat ja käyttäisivät samaa sanastoa, voitaisiin sivuilta ohjelmallisesti poimia ja koostaa tietoa. Sovellukset voisivat käyttää tätä koostettua tietoa vastatessaan käyttäjän hakuihin tai jakaessaan tietoa muille sovelluksille (Noy & McGuinness 2001). Jos eri yhteisöt käyttävät eri ontologioita, erilaiset termit voidaan tehdä yhteensopivaksi linkittämällä lokaalit termit yhteiseen jaettuun ontologiaan. (Hyvönen, Salminen, Junnila & Kettula 2004). Viimeksi mainittu on tosin hyvin paljon työläämpää, kuin se, että sovellukset jo alusta alkaen käyttäisivät samaa ontologiaa. Jaetut ja yhteiset ontologiat vähentävät tarvetta paikallisille sanastoille (Hyvönen, Salminen, Junnila & Kettula 2004).

RDF-kuvailujen laajennettavuus ja yhdisteleminen

Klynen ja Carrollin (2004) mukaan RDF sopii Internet-maailmaan hyvin, koska se sallii jokaisen tehdä RDF-väittämiä ja –kuvailuja resursseista. Kuvailuja tietystä resurssista voidaan aina joustavasti laajentaa (Hunter & Lagoze 2001; Candan, Liu & Suvarna 2001; Klyne & Carroll 2004). RDF-malli ei oleteta, että resurssista olisi kaikki tieto saatavissa tai toisaalta että epäyhtenäisiä tai paikkansapitämättömiä tietoja ei esiintyisi kuvailuissa (Klyne & Carroll 2004). Tällainen rajaton laajennettavuus ei tosin sovi suljettuun järjestelmään, kuten lainsäädäntöprosessin järjestelmiin. Suljetuissa järjestelmissä on yleensä tarpeellista käyttää yhtenäistä sanastoa (RDF-skeemaa) kuvausten tekemiseen ja sopia käytännöt, joiden mukaan kuvauksia tehdään (esimerkiksi mistä dokumenteista ja milloin).

RDF-kuvauksia voidaan joustavasti yhdistellä. Kuvauksia pystytään muodostamaan monien eri lähteistä tulevien skeemojen avulla. (Hunter & Lagoze 2001; Candan, Liu & Suvarna 2001.) Täten RDF-kuvausten ja -skeemojen jaettavuus ja uudelleenkäyttö eri tahojen kesken onnistuvat joustavasti. RDF pohjautuu voimakkaasti XML-nimiavaruusmekanismiin. Nimiavaruuksien avulla eri käsitteet voidaan erottaa toisistaan. Nimiavaruudet mahdollistavat RDF-skeemojen uudelleenkäytön ja skeemojen yhdistämisen. (Staab, Erdmann, Maedche & Decker 2002.)

URI-viitteiden käyttö

URI-viitteiden käyttämisellä RDF:ssä on monia etuja. Subjektin, objektin ja predikaatin tallentaminen URI-viitteenä mahdollistaa sanastojen ja skeemojen jakamisen internetissä. Onkin kannattavaa käyttää olemassa olevia sanastoja, esimerkiksi Dublin Corea, kun sen käyttö on mahdollista. Saman käsitteistön käyttäminen parantaa sovellusten yhteentoimivuutta. (Manola & Miller 2004.)

URI-viitteisiin liittyen RDF:n tärkeänä etuna on se, että niin RDF:n väittämiä, ominaisuuksia kuin ominaisuuksien arvojakin voidaan käsitellä resursseina. Näin niihin voidaan tallentaa lisätietoa tarvittaessa. (Ebenhoch 2001; Manola & Miller 2004.) URI-viitteiden käyttö siis mahdollistaa RDF-kuvausten laajentamisen.

URI-viitteiden avulla asiat pystytään lisäksi identifioimaan tarkasti: esimerkiksi henkilöön liittyvällä URI:lla voidaan viitata juuri tiettyyn henkilöön, eikä mihin tahansa merkkijonoon tai kehen tahansa samannimiseen ihmiseen. (Manola & Miller 2004.) URI-viitteillä voidaan tehdä eksakteja linkkejä webin ulkoiseen maailmaan. Voidaan siis esimerkiksi viitata henkilöihin ja yhteisöihin, vaikka ne eivät olekaan web-resursseja. (Hyvönen, Salminen, Junnila & Kettula 2004.)

Mahdollisuus semanttisen webin älykkäiden palvelujen toteuttamiseen

RDF-muotoisen tiedon ja pohjalta on mahdollista toteuttaa ontologisia päättelysääntöjä sekä erilaisia semanttisen webin älykkäitä palveluja, kuten näkymien esittäminen, semanttinen samoilu ja semanttinen suosittelu. Tässä kappaleessa kerrotaan, minkälaisia tällaiset palvelut voisivat olla ja miten niitä voidaan toteuttaa RDF:n avulla.

Käyttöliittymätasolla RDF-muodossa kuvailtuun tietoon voidaan antaa eri näkymiä, joista käyttäjä itse saa valita mieleisensä. Voidaan esimerkiksi esittää kaikki mahdollinen RDF-tieto ja RDF-ominaisuudet, joka liittyvät resurssiin tai vain tietyn ominaisuuden ja ominaisuuden arvon mukaan suodatetut resurssit. (Quan & Karger 2004.)

Näkymällä tarkoitetaan yleensä sitä, että tietojoukosta on valittu tietty luokka ja luokan jollekin ominaisuudelle määrätty tietty arvo. Esimerkki näkymästä voisi olla ”Valtiokuntamietinnöt, joiden tekijänä on ympäristövaliokunta”. Näkymät voivat auttaa käyttäjää monin tavoin tiedon löytämisessä. Taustalla olevan RDF-skeeman luokkahierarkian ja ominaisuuksien esittäminen näkymähierarkiana antaa käyttäjälle yleiskuvan, mitä tietoa ylipäätään tietovarastossa on. Hierarkiat ohjaavat käyttäjää luomaan hakuja ja osoittavat, millä käsitteillä asioista puhutaan ja mitkä hakusanat täten ovat sopivia haun tekemiseen. (Esimerkiksi käsitteet `Vaihe2`, `Yleisistuntokäsittely`, `Presidentinesittely`, `vaiheAlkanut`, `vaihePäättynyt` jne.) Käyttäjä voi valita ensin tietyn tietokategorian ja mennä vähitellen syvemmälle hierarkiassa ja tarkentaa hakua. Näkymät auttavat käyttäjää navigoinnissa, kun hän selaa tietovaraston sisältöä. (Hyvönen, Saarela, Viljanen ym. 2004.) Perinteisen haun tuloksena syntyvää dokumenttilistaa voidaan RDF-muodossa kuvatulla tiedolla rikastaa esittämällä dokumenttilista valmiiksi kategorisoituna RDF-skeemassa olevien luokkien mukaan (Guha, McCool & Miller 2003). Lakiasiakirjoja etsittäessä haun tuloksena saadut lakiasiakirjat voitaisiin esimerkiksi listata lakihankkeiden mukaisesti kategorisoituna.

Tiedonhaun tuki, *semanttinen samoilu* (semantic browsing), jossa käyttäjä selailee tietovarastoa tarkentaen kyselyä kategorioiden avulla vähitellen, on erityisen hyvä silloin, kun käyttäjä ei tarkalleen tiedä mitä etsii. Tällainen tilanne lainsäädäntöprosessissa voi olla esimerkiksi silloin, kun ei olla varmoja, liittyykö aloitettuun lakihankkeeseen muita siihen liittyviä lakihankkeita vai ei. Järjestelmä voisi tällöin automaattisesti näyttää liittymät muihin hankkeisiin, joten käyttäjä näkisi tiedon, vaikkei ole sitä alun perin hakenutkaan. Muussa tapauksessa, kun etsitään esimerkiksi tiettyä asiakirjaa, suora Google-tyyppinen sanahaku on parempi. (Hyvönen, Saarela, Viljanen ym. 2004.)

Kun tiedon kuvailuun käytettävä ontologia on olemassa, tarvitaan tähän ontologiaan pohjautuva hakusovellus (search engine), joka pystyy hakemaan tietoresursseja niiden ominaisuuksien mukaan. Stuckenschmidtin ja Harmelen (2001) esittämän tiedonhakumallin mukaan hakukone kysyy käyttäjältä käsitettä, josta halutaan löytää tietoa. Tämän käsitteen ontologiassa olevaan määrittelyyn pohjautuen muodostetaan hakukäyttöliittymä, johon käyttäjä saa syöttää rajoitteita käsitteen ominaisuuksille. Käytännössä RDF Schemassa tämä tar-

koittaa tietyn luokan ja ominaisuuden valitsemista: halutaan esimerkiksi etsiä hankkeita luokasta `Lakihanke` ominaisuuden `vaihe` perusteella. Hakusovellus etsii tämän jälkeen metatietokuvailuista kaikki tietoresurssit, joilla ominaisuusrajoitteet pitävät paikkansa. Hakusovellus toimii tällaisessa mallissa yhtenä komponenttina järjestelmässä. (Stuckenschmidt & Harmelen 2001.)

Kun tieto on semanttisesti merkattu RDF-muodossa, loppukäyttäjien sovellukset voivat hyödyntää RDF-kuvailuja ja tehdä niiden avulla älykkäitä päätelmiä siitä, miten tieto halutaan esittää käyttötarkoituksesta ja itse käyttäjästä riippuen. RDF:n formaali semantiikka tarjoaa luotettavan pohjan päättelysääntöjen (inference rules) muodostamiseen (Klyne & Carroll 2004). Hovyn (2003) mukaan ontologian tukeman ohjelmallisen päättelyn avulla tietokoneohjelmat voivat auttaa käyttäjiä löytämään ja yhdistämään tietoa sekä löytämään vastaavuuksia saman aihealueen tiedoista, vaikka tiedot tulisivat eri lähteistä. Ontologiset päättelysäännöt rikastavat Hyvösen, Saarelan, Viljasen ym. (2004) mukaan tiedon semantiikkaa. Eri sovellukset ja käyttöliittymät voivat näyttää päättelysääntöjen avulla esimerkiksi eri määrän tietoa – toinen yksityiskohtaisemmin kuin toinen (Quan & Karger 2004).

RDF:n ainut päättelymekanismi perustuu hierarkiasuhteisiin luokkien välillä (Ribiére & Charlton 2000). Syvälliseen ontologiseen päättelyyn RDF Schema tarvitsee tuekseen lisäkerroksen (Broekstra, Klein, Decker ym. 2001). RDF:n pohjalta tehdyt päättelyt voivat perustua esimerkiksi RDF-kyselykieliin, tietokantaratkaisuun (Karvounarakis 1999), logiikkaohjelmointikieliin kuten Prolog (Hyvönen, Saarela, Viljanen ym. 2004) tai RDF:n laajentamiseen ontologiakielten avulla (Broekstra ym. 2001).

Ontologian päälle voidaan rakentaa älykkäitä palveluja, kuten MuseoSuomi-portaalin (portaali Suomen museoiden kokoelmien ja aineiston selaamiseen, MuseoSuomi 2004) käyttämä *semanttinen suosittelu* (semantic recommendation). Semanttinen suosittelu tarkoittaa sitä, että varsinaisen haun rinnalla esitetään muita asiaan liittyviä linkkejä RDF-skeeman kategorioiden (luokkien) perusteella (Hyvönen, Saarela, Viljanen ym. 2004).

Ihmiset kokevat usein vaikeaksi ilmaista, mitä he haluavat tai mitä he etsivät. Toisaalta hei-

dän on yleensä hyvin helppo tunnistaa hakemansa asiat, kun he näkevät ne (Middleton, Alani, Shadbolt & De Roure 2002). Tämän takia semanttinen suosittelu voi olla tehokas väline tiedon löytymiseen. MuseoSuomi-portaalissa semanttista suosittelua käytetään niin, että esimerkiksi kokoelmaan kuuluvan yksittäisen astian tietoja tarkasteltaessa sovellus näyttää rinnalla linkkejä muihin esineisiin, jotka joko liittyvät samaan aiheeseen, ovat tehty samasta materiaalista tai ovat valmistettu samassa paikassa. MuseoSuomi-portaalissa suosittelua käytetään myös tietokategorioiden esittämisessä: Käyttäjälle näytetään tietokategorioiden hakuosumat jo etukäteen. Tällä tavoin voidaan estää käyttäjää tekemästä hakuja, jotka antavat tyhjän tuloksen sekä ”suositella” käyttäjälle sellaisia kategorioita, joissa osumia on monta. (Hyvönen, Saarela, Viljanen ym. 2004.)

Semanttisessa suosittelussa resurssin rinnalla näytettävät resurssiin liittyvät asiat voidaan poimia käyttöliittymään suoraan resurssin RDF-ominaisuuden mukaan (esimerkiksi näytetään eduskuntakäsittelyvaiheen valiokunnan lausunnon rinnalla muut samaan vaiheeseen kuuluvat asiakirjat) tai päättelyä käyttäen ns. *virtuaalisten ominaisuuksien* mukaan. Virtuaaliset ominaisuudet tarkoittavat ominaisuuksia, jotka voidaan päätellä normaalien ominaisuuksien kautta. (Quan & Karger 2004.) Esimerkiksi jos Pekka työskentelee Jyväskylän yliopistolla tutkimusapulaisena, voidaan päätellä, että Pekka kuuluu yliopiston henkilökuntaan. Esimerkkejä virtuaalisista ominaisuuksista liittyen lainsäädäntöprosessiin esitellään luvussa 7. Hyvönen, Saarela, Viljanen ym. (2004) kutsuvat virtuaalisia ominaisuuksia *implisiittisiksi linkeiksi* ja RDF-väittämissä suoraan esiintyviä ”tavallisia” ominaisuuksia *eksplisiittisiksi linkeiksi*.

5.3 RDF:n rajoitteita ja haasteita

Tässä kappaleessa arvioidaan RDF:n rajoitteita ja haasteita. Aiheena ovat muun muassa RDF:n ilmaisuvoima, graafiesitystapa ja RDF/XML-syntaksi, URI-viitteiden käyttö ja RDF:n soveltuvuus koneille ymmärrettävän semantiikan ilmaisemiseen. Lisäksi arvioidaan kokemuksia RDF:n käytöstä sekä työkalujen ja valmiiden skeemojen saatavuutta.

Käyttökokemusten ja työkalujen vähäisyys

Niemijärven mukaan (2002) RDF:n heikkoutena on sen uutuus ja se, että sitä on testattu vain vähän käytännössä. XML:n ja RDF:n erilainen tietomalli voi aiheuttaa ongelmia tulevissa ratkaisuissa. Tietomallien erilaisuuden takia XML-muotoista tietoa voi olla vaikea kuvailla RDF-tietomallin avulla (Klein 2002).

Candan, Liun ja Suvarnan (2001) toteavat konkreettisten sovellusten olevan tarpeen RDF:n käytettävyyden osoittamiseksi. Muita RDF:n menestymisen edellytyksiä ovat heidän mukaansa työkalujen ja tietokantajärjestelmien kehittäminen sekä tietoturvaratkaisut, käytön helppous ja yhteensopivuus muiden sovellusympäristöjen kanssa. Tietokantajärjestelmiä tarvitaan RDF-tiedon hallitsemiseen ja paikantamiseen. (Candan ym. 2001.) Yleisesti hyväksyttyjä luonti- ja jäsennintyökaluja täytyy löytyä, että RDF-skeemojen ja –kuvausten suunnittelu ja virheetön toteutus on mahdollista (Candan ym. 2001; Maganaraki ym. 2002; Niemijärvi 2002).

Niemijärvi (2002) huomauttaa tutkimuksessaan, että loppujen lopuksi ei kaikkea metatietoa millään voi kuvata vain yhdellä ja samalla syntaksilla. Hän toteaa, että esimerkiksi CASE-työkalut tuottavat hyvin paljon metatietoa ja kaikenlaisten ohjelmien ja komponenttien tuottaman metatiedon muuntaminen RDF-muotoon RDF/XML-syntaksin mukaiseksi toisi varmasti lisäkustannuksia ja aiheuttaisi paljon vaivaa.

Rajoittunut ilmaisuvoima verrattuna rikkaampiin ontologiakieliin

RDF:llä on rikkaampiin ontologiakieliin nähden rajoittunut ilmaisuvoima. Resurssien hyvin yksityiskohtainen kuvaaminen ja täysien ontologioiden rakentaminen tarvitsee RDF(S):n laajentamista rikkaammilla ontologiakielillä, kuten OIL (Ontology Inference Layer), DAML (The DARPA Agent Markup Language), DAML+OIL ja OWL (Web Ontology Language). (Horrocks & Patel-Schneider 2003; Broekstra, Klein, Decker ym. 2001.) Nämä ontologiakielet pohjautuvat RDF-malliin ja RDF/XML-syntaksiin, mutta niillä voidaan kuvata monipuolisemmin esimerkiksi rajoitteita ominaisuuksille (Connolly, Harmelen, Horrocks ym. 2001; McGuinness & Harmelen 2004).

Manolan ja Millerin (2003) mukaan rikkaammilla ontologiakielillä on seuraavia lisäominaisuuksia RDF:ään verrattuna:

- kardinaalisuus (esimerkiksi Henkilöllä on *ainoastaan yksi* biologinen isä)
- tietyn ominaisuuden transitiivisuuden määrittely (esimerkiksi jos Aapolla `onEsi-isä` Pekka ja Pekalla `onEsi-isä` Kalle, siitä seurauksena Aapolla `onEsi-isä` Kalle.)
- jonkin ominaisuuden määrittäminen ainutlaatuiseksi tunnisteeksi (unique identifier, key) tietyn luokan ilmentymälle.
- Kahden eri luokan (joilla on eri URI-viitteet) määrittäminen saman luokan edustajiksi.
- Kahden eri ilmentymän (joilla on eri URI-viitteet) määrittäminen saman ilmentymän edustajiksi.
- Mahdollisuus määrittää ominaisuuden arvo-alueelle (range) sellaisten rajoitteita, jotka ovat riippuvaisia resurssin luokasta (esimerkiksi jalkapallojoukkueen ominaisuudella onPelaajia voi olla 11 arvoa, kun taas koripallojoukkueella samalla ominaisuudella voi olla vain 5 arvoa).
- Mahdollisuus kuvata uusia luokkia muiden luokkien yhdistelmien kautta (esim. unioni ja leikkaus) tai kuvata se tilanne, että kahden luokan leikkaus on tyhjä (yksikään resurssi ei ole molempien luokkien instanssi).

Staab, Erdmann, Maedche ja Decker (2002) myöntävät, että RDF tarjoaa vain perusmallinnusmenetelmät ontologioiden suunnitteluun. Heidän mukaansa RDF:n ilmaisuvoima on suunniteltu tarkoituksella suhteellisen alhaiseksi, koska sen tavoitteena on pystyä kuvailemaan mahdollisimman suurta osaa webin resursseista. RDF:n ei ole tarkoituskaan olla täydellinen vastaus kaikkiin tiedon esittämisen ongelmiin, vaan olla *laajennettava ydinkieli* (Staab, Erdmann, Maedche, & Decker 2002). Muutenkaan aina ei edes haluta rakentaa monimutkaista ja laajaa ontologiaa. RDF:n soveltuvuus riippuu käyttötarkoituksesta ja kuvattavan sovellusalan monimutkaisuudesta.

Skeemojen saatavuus

Yksi RDF:n käytännön haasteista Candanin, Liun ja Suvarnan (2001) mukaan on tehdä RDF-skeemat helposti saataviksi ja löydettäviksi skeemojen uudelleenkäyttöä varten. Myös Magkanaraki, Alexaki, Christophides ja Plexousakis (2002) toteavat, että skeemarekistereitä tarvittaisiin skeemojen jakamisen mahdollistamiseksi. Heidän mukaansa monet samankin toimialan yhteisöt ovat toteuttaneet itsenäisesti omat metatietosanastonsa. Tietämyksen jakaminen skeemarekistereiden avulla säästäisi paljon turhaa työtä ja parantaisi sovellusten yhteentoimivuutta.

Candanin, Liun ja Suvarnan (2001) mukaan valmiita RDF-skeemoja pitäisi toteuttaa ainakin yleisimmille resursseille. Tämä helpottaisi heidän mielestään RDF:n käytön yleistymistä. Esimerkiksi skeemojen toteuttaminen kuva-, ääni- ja videotiedostoille helpottaisi tiedon hakua myös tällaisista ei-tekstimuotoisista tiedostoista. (Candan, Liu & Suvarna 2001.) Jos skeemoja olisi valmiina, ei niiden käyttämiseen olisi niin suurta kynnystä kuin jos skeemat pitäisi alusta asti toteuttaa itse.

URI:n ongelmat

RDF:n käyttämä identifioimisväline URI:ssa on oman kokemukseni mukaan myös haittapuolia. Ensinnäkään pitkät URI-nimet eivät ole helposti luettavia ihmiselle. Lyhennettyjen nimien käyttäminen (esimerkiksi `dc:name`) toki helpottaa luettavuutta.

Lisäksi URI-tunnisteet näyttävät URL-osoitteilta, joita Internet-selain osaa tulkita. Vaikka RDF-syntaksissa käytettävät URI-tunnisteet saattavat kertoa myös resurssin sijainnin, kuten selvitetty kappaleessa 2.2., RDF käyttää ensisijaisesti URI-tunnisteita ainoastaan *identifioimaan* resursseja. Ensimmäinen assosiaatio ihmisellä usein URI-viitteen nähdessään on, että kyseisestä Internet-osoitteesta löytyisi jotain sisältöä. Kun näin ei kaikissa tapauksissa olekaan, saattaa asia aiheuttaa hämmennystä. Esimerkiksi URI http://www.paapaa.fi ei ole oikea Internet-osoite, mutta RDF-spesifikaation (Manola & Miller 2004) mukaan sitä voidaan silti käyttää RDF:ssä resurssien identifioimiseen. Jos identifiointitapa olisi joku muu kirjain- tai numerosarja, ei sekaannusta URL-osoitteisiin tulisi.

Asiaa voi ajatella edelleen eteenpäin. Kuinka paljon sekaannusta saattaakaan aiheuttaa seuraavanlainen tilanne. URI:a http://www.oma.fi on käytetty tämän tutkielman esimerkeissä. Myöhemmin huomataan, että kyseinen URI on jo käytössä Internet-osoitteena (kuten tässä tapauksessa kävi) tai vaihtoehtoisesti se joskus kymmenien vuosien kuluttua otetaan käyttöön Internet-osoitteena jossain aivan muussa yhteydessä. RDF-spesifikaatio ei ota kantaa tällaiseen tilanteeseen. RDF-kuvauksia tehdessä olisi kuitenkin varmasti parempi ottaa myös URI-tunnisteen mukainen URL-osoite ”haltuun” eli ottaa käyttöön sellaiset URI-tunnisteet, jotka toimivat URL-osoitteena omassa tai organisaation web-palvelimessa. Tällä vältettäisiin sekä sekaannuksia että se, että joku muu taho identifioisi omiin RDF-kuvauksiinsa omat resurssinsa samoilla URI-tunnisteilla, joita on esimerkiksi tässä tutkielmassa käytetty.

Ongelmallinen on myös sellainen tilanne, jossa resurssin URL-osoite muuttuu ja RDF-kuvaukseen jää vanha URL-osoite. Periaatteessa vanhallakin URL-osoitteella resurssi pystytään yhä identifioimaan, mutta jos kerran resurssilla on URL, jota voidaan käyttää URI:na resurssin identifioimisessa, on selkeämpää käyttää tätä samaa tunnusta sekä identifiointiin ja sijainnin ilmaisemiseen. On kuitenkin suorastaan todennäköistä, että resurssin URL-osoite jossain vaiheessa muuttuu. Muutoksen voi aiheuttaa esimerkiksi yritysostot, jolloin yrityksen nimen muutos aiheuttaa lähdes aina myös yrityksen URL-osoitteet muutoksen.

Se, että URI-viitteitä ja osatunnisteita käytetään RDF:ssä myös ilman, että URI kertoisi dokumentin sijainnin, on osittain ristiriidassa alkuperäisen URI-syntaksispesifikaation kanssa osatunnisteisiin liittyen. URI-spesifikaatiossa (Berners-Lee ym. 1998, 14) osatunnisteista sanotaan seuraavasti: “*A fragment identifier is only meaningful when a URI reference is intended for retrieval and the result of that retrieval is a document for which the identified fragment is consistently defined*.” Eli osatunniste on ainoastaan merkitsevä ja mielekäs silloin, kun URI-viitettä käytetään dokumentin sijainnin osoittamiseen ja tämän sijaintitiedon avulla haetussa dokumentissa on määritelty kyseinen osatunniste. (Berners-Lee ym. 1998, 14) RDF:ssä osatunnistetta kuitenkin käytetään myös ilman sijaintitiedon osoittamista, joten voitaisiin esittää kysymys, eikö RDF-kuvauksien sisältämien osatunnisteiden käyttö ole

aina mielekästä? Tähän ristiriitaan viitataan RDF-spesifikaatiossakin. Asia kuitataan spesifikaatiossa sillä, että koska RDF:n perusajatus mukaan resurssin ei tarvitse olla saatavissa internetissä, osatunnisteella viitataan RDF:ssä URI-spesifikaatiosta poikkeavasti sekä webin että muihin resursseihin (Klyne & Carroll 2004).

Loppujen lopuksi siinä mahdollisuudessa, että URI-tunniste *osaa* tarvittaessa kertoa dokumentin sijainnin, on omat etunsa. Se antaa paljon lisäarvoa RDF-kuvauksille ja -skeemoille, koska näin esimerkiksi niiden julkaiseminen on mahdollista Internetissä suoraan identifiointitunnuksen avulla. Esitetyt haittapuolet kompensoituvat mielestäni URI-tunnisteen käytön merkittävillä eduilla.

RDF-graafien monimutkaistuminen ja luettavuus

RDF:n etuna mainittiin edellisessä kappaleessa, että kaikki RDF-kolmikon jäsenet voivat olla resursseja ja niihin kaikkiin voidaan täten liittää lisää kuvailutietoa uusien väittämien ja ominaisuuksien muodossa. Se, että RDF-spesifikaatio tällä tavoin sallii RDF-kuvailujen rajattoman laajentamisen ilman rajoitteita graafin rakenteelle, saattaa koitua joissain tapauksissa myös ongelmaksi. Näin graafit voivat esimerkiksi muodostaa silmukoita ja muita rakenteita, joita saattaa olla ohjelmallisesti vaikea käsitellä. Tämä on hyvä pitää mielessä todellisia sovelluksia rakennettaessa, vaikka RDF-spesifikaatio ei graafien rakennetta rajoitakaan.

Esitystapana graafimalli on selkeä ja luettava yksinkertaisissa RDF-kuvauksissa, mutta monimutkaisemmissa kuvauksissa erityisesti graafissa olevat ominaisuusnuolet menevät niin sotkuisesti päällekkäin, että ihmislukijan on enää vaikea saada kuvasta selvää. Kolmikkomalli tai RDF/XML on tällöin ihmiselle luettavampi esitystapa. Koneille graafien käsitteleminen ei ole ongelma (Berners-Lee 1998).

Monet vaihtoehtoiset esitystavat RDF/XML-syntaksissa

RDF/XML-syntaksissa on monia vaihtoehtoja esittää samaa tarkoittavia asioita. Esimerkit 30 ja 31 esittävät kaksi vaihtoehtoista tapaa määrittää resurssi jonkin luokan ilmentymäksi. Esimerkissä 30 resurssi määritetään `Description`- ja `rdf:type`-määritteillä ja esimerkissä 31 yksinkertaisesti luokan nimen avulla kokonaan ilman Description-elementtiä.

ESIMERKKI 30. Resurssin luokan osoittaminen rdf:type-määritteellä.

```
<rdf:Description rdf:about="http://www.oma.fi/opiskelijat/2500088555">
<rdf:type rdf:resource="http://www.oma.fi/termit/Opiskelija"/>
</rdf:Description>
```

ESIMERKKI 31. Resurssin luokan osoittaminen suoraan luokan nimellä.

```
<oma:Opiskelija rdf:about="http://www.oma.fi/opiskelijat/2500088555">
</oma:Opiskelija>
```

Erilaiset merkintätavat saattavat aiheuttaa ongelmia tulkinnassa niin ihmiselle kuin tietokoneohjelmallekin. On aivan makuasia, onko pidempi vai lyhyempi esitystapa ihmiselle luettavampi. Omasta mielestäni lyhyempi tapa on selkeämpi. Joka tapauksessa RDF/XML-syntaksiin perehtyjälle voivat erilaiset merkintätavat aiheuttaa hämmennystä, sillä esimerkiksi artikkeleissa esiintyvissä RDF-esimerkeissä saatetaan käyttää jopa saman RDF-kuvauksen sisällä kahta erilaista merkintätapaa.

Kaikki sovellukset ja XML-pohjaiset kielet eivät pysty käsittelemään erilaisia merkintätapoja. Esimerkiksi XPath on kieli, jolla voidaan viitata XML-dokumentin tiettyihin rakenneosiin. Vaikka RDF voidaan esittää puhtaasti XML-muodossa, useat syntaksivaihtoehdot hankaloittavat XPathin käyttöä RDF-tietoon tehtäviin kyselyihin. XPathia voidaan käyttää, jos sovitaan käytettäväksi vain tiettyä esitystapaa – joko pitempää tai lyhyempää tapaa. (Ensel & Keller 2002.)

RDF ja merkitys – ymmärrettävää ihmiselle ja koneelle?

RDF:n avulla voidaan kuvata tiedon *merkitystä* sekä ihmisille että koneille (Manola & Miller 2004). Joissain tapauksissa on kyseenalaista, ovatko RDF-kuvaukset ymmärrettäviä kaikille ihmisille vai esimerkiksi vain jonkin tietyn kielen osaaville ihmisille. Jos esimerkiksi käytetään standardia Dublin Core ilmaisua päivämäärästä, `dc:date`, menetetään RDF-kuvauksessa alkuperäinen termi ”päivämäärä”, jolla on merkitystä suomenkieliselle lukijalle. Tiedon merkitys on siis monesti kieliriippuvaista. RDF-kuvauksiin on mahdollista haluttaessa lisätä merkitystä ihmislukijalle `rdf:comment`-elementin avulla.

RDF-spesifikaation esittämien RDF-rakenteiden yhteydessä korostetaan monessa kohtaa sitä, että RDF-sanaston merkitys määräytyy vasta varsinaisessa sovelluksessa. RDF:ssä ei ole sisäänrakennettua (build-in) ymmärrystä RDF-rakenteille sen kummemmin kuin muillekaan rakenteille (Manola & Miller 2004). Täytyy siis huomata, että RDF on XML-syntaksinsa ansiosta prosessoitavissa sovellusten avulla (machine-processable) (Manola & Miller 2004), mutta ei itsessään ole automaattisesti koneille ymmärrettävää (machine-understandable). Esimerkiksi rakennetta `rdf:Seq`, joka kuvaa järjestettyä ryhmää, käytetään spesifikaation (Brickley & Guha 2004) mukaan osoittamaan *ihmislukijalle*, että säiliön numeerinen järjestys on tarkoitettu olevan merkityksellinen.

Tällaiset kohdat spesifikaatiossa voivat aiheuttaa hämmennystä ja tässä on vaarana, että erilaiset sovellukset tulkitsevat RDF:ää eri tavalla. RDF-spesifikaatioiden tiedoista voi jäädä sellainen käsitys, että tieto voitaisiin yhtä hyvin esittää millä tahansa muullakin tavalla, kuin RDF:llä, jos kerran loppujen lopuksi riippuu vasta sovelluksesta, kuinka RDF-rakenteita tulkitaan. Tai voidaan ajatella, että RDF:n tarkoitus on määritellä joustavia rakenteita, joista sovelluksen kehittäjä voi itse päättää kuinka niitä tulkitaan. RDF:ää on kuitenkin tarkoitettu käytettävän juuri spesifikaatiossa tarkoitetulla tavalla. Spesifikaatiossa halutaan vain korostaa, että sovelluksen kehittäjällä on vastuu siitä, että RDF:ää tulkitaan oikein. Kun yleisesti käytetään tällaista samaa ”standardia” esitystapaa, tuetaan sovellusten yhteistoiminnallisuutta, koska on sovittu yhteiset periaatteet ja käytännöt, minkä mukaan sovellukset tulkitsevat tietoa.

6 RDF-SKEEMOJEN SUUNNITTELU

Tässä kappaleessa esitetään suunnitteluperiaatteita RDF-skeemoille ja joitain yleisiä ongelmia, joita suunnittelussa saattaa tulla vastaan. Aluksi kuvataan skeemasuunnittelun vaiheita. Toisessa kappaleessa esitetään yleisiä ohjeita skeemojen luomiseen ja kolmannessa kaksi hyödyllistä työkalua skeemojen suunnittelun tukemiseen.

6.1 Vaiheet RDF-skeemojen suunnittelussa

Ontologiatyyppisen skeeman suunnittelun vaiheet esitellään alla Noyn ja McGuinnessin (2001) mukaan. Kukin vaihe kuvataan tarkemmin vaihelistan jälkeen.

1) Määrittele skeemalle sovellusala ja tavoite.
2) Tarkista olemassa olevien skeemojen uudelleenkäyttömahdollisuus.
3) Luettele tärkeät termit skeemaa varten.
4) Määrittele luokat ja luokkahierarkia.
5) Määrittele ominaisuudet luokille.
6) Määrittele ominaisuuksien rajoitteet.
7) Luo ilmentymiä luokille.

Skeeman sovellusalueen ja tavoitteen määrittäminen

Ensimmäisessä (1) vaiheessa määritellään skeeman sovellusalue ja tavoite sekä halutaan löytää vastaus kysymyksiin: Mihin tarkoitukseen skeemaa halutaan käyttää? Kuka käyttää skeemaa ja mihin sovellusalaan se on tarkoitettu? (Noy & McGuinness 2001.) Yksi keino määrittää skeeman tarkoitus, on listata kysymyksiä, joihin skeeman ja sen pohjalta tehtyjen kuvausten pitäisi antaa vastaus. Näitä kysymyksiä voidaan kutsua pätevyyskysymyksiksi (competence questions) (Gruninger & Fox 1995). Lainsäädäntöprosessin skeemaa suunniteltaessa voisivat pätevyyskysymykset olla esimerkiksi:

- Missä vaiheessa jokin lainsäädäntöhanke on?
- Mihin lainsäädäntöhankkeisiin tietty ministeriö tai valiokunta on osallistunut?

- Mitä asiakirjoja on tietyn lain valmistelu- ja käsittelyvaiheissa syntynyt?

Näiden kysymysten perusteella voisi päätellä, että lainsäädäntöprosessin skeema voisi sisältää ainakin tietoa lainsäädäntöprosessin vaiheista, siihen osallistuvista toimijoista ja laki-asiakirjoista.

Olemassa olevien skeemojen uudelleenkäyttö

Toisessa (2) vaiheessa olemassa olevien skeemoja etsiminen on usein hyödyllistä ja se säästää aikaa ja työtä. Joskus olemassa olevien skeemojen uudelleenkäyttö saattaa olla jopa edellytyksenä skeeman luomiselle, jos halutaan järjestelmän toimivan yhteen muiden sovellusten kanssa, jotka jo käyttävät jotain skeemaa tai sanastoa. (Noy & McGuinness 2001.) Esimerkiksi Dublin Core –sanastoa (DCMI 2004) voidaan jo sanoa metatietosanastojen de facto –standardiksi, joten on suositeltavaa ottaa se skeeman pohjaksi. Seuraavassa on listattu muutamia RDF-skeemarekistereitä, joista valmiita RDF-skeemoja voi etsiä.

- SchemaWeb. Sisältää hyvin monipuolisesti RDF-skeemoja eri aiheista ja käyttää skeemojen esitystapana RDF/XML-syntaksin lisäksi muun muassa kolmikko-muotoa ja luokka/ominaisuuslistoja. <http://www.schemaweb.info/>
- CORES Registry. A Forum on Shared Metadata Vocabularies. Sisältää 39 RDF-skeemaa <http://www.cores-eu.net/registry/schema/>
- RDF Schema Registry. (FORTH Institute of Computer Science.) Sisältää useita RDF-skeemoja eri aiheista, muun muassa lääketieteestä, elektronisesta oppimisesta ja koulutuksesta. <http://139.91.183.30:9090/RDF/Examples.html>
- WebODE. Sisältää 17 sanastoa toteutettuna RDF(S)- ja OIL-kielillä. Käytettävissä rekisteröinnin jälkeen osoitteessa <http://delicias.dia.fi.upm.es/webODE/>.

Tässä kohtaa täytyy huomata, että skeemat ovat aina vain tekijänsä tai tekijöidensä kuvaus sovellusalasta. Jos valmis sovellusalan skeema ei tunnu sopivalta, on aina mahdollista luoda oma skeema, jotta se mahdollisimman hyvin palvelisi tavoitteena olevan sovelluksen tarpeita. (Noy & McGuinness 2001.)

Oleellisimpien termien luetteleminen

Kolmannessa (3) vaiheessa luetellaan vapaasti sovellusalan keskeisimmät termit ja käsitteet välittämättä niiden päällekkäisyydestä, suhteista tai ominaisuuksista (Noy & McGuinness 2001). Esimerkiksi tärkeitä asiakirjaan liittyviä käsitteitä voisivat olla otsikko, identifioiva tunniste ja tekijä. Jos sovellusalueella on joukko olemassa olevia XML-dokumentteja, voidaan XML-dokumenttien elementti- ja attribuuttinimiä käyttää tärkeimpien käsitteiden etsimiseen (Klein 2002).

Termien jakaminen luokkiin ja ominaisuuksiin

Neljäs (4) ja viides (5) vaihe eli luokkien ja ominaisuuksien määrittely ovat hyvin lähekkäisiä ja usein päällekkäisiä tehtäviä. Luokkien määrittämiseen voidaan käyttää eri menetelmiä. Ylhäältä alaspäin –menetelmässä (top-down) määritellään ensin kaikista yleisimmät käsitteet luokiksi ja siirrytään muodostamaan niistä aliluokkia. Lainsäädäntöprosessiin liittyen voitaisiin esimerkiksi määritellä ensin luokka `Vaihe` ja tarkentaa siitä aliluokiksi käsitteet `HE_Valmistelu`, `HE_Valtioneuvostokasittely`, `Eduskuntakasittely` ja `HyvaksytynLainValtioneuvostokasittely`. Alhaalta ylöspäin –menetelmässä (bottom-up) aloitetaan tarkimmista käsitteistä ja yleistetään siitä käsitteet laajemmiksi. Näiden yhdistelmässä määritellään keskeisimmät käsitteet ensin ja muodostetaan niistä sekä yleisempiä että tarkempia käsitteitä. (Noy & McGuinness 2001.) Huhtanen (2003, 95) sanoo ylhäältä alaspäin –metodin ongelmana olevan helposti epätarkat korkean tason luokat ja alhaalta ylöspäin –metodin ongelmana liiallinen yksityiskohtaisuus. Hänen mielestään keskeltä molempiin suuntiin lähtevä metodi onkin suositeltavin ja vaatii vähiten turhaa työtä ja tarpeettomien ylä- tai alakäsitteiden määrittelyä. Kehittäjä voi tietenkin vapaasti valita itselleen luonnollisimman metodin luokkien määrittelemiseen.

Kullekin ominaisuudelle pitää määritellä, mihin luokkaan se voidaan liittää. Ominaisuudet pitää liittää kaikista yleisimpään luokkaan, koska aliluokat perivät yliluokan ominaisuudet. (Noy & McGuinness 2001). Esimerkiksi luokan `Lakiasiakirja` ominaisuus `otsikko` periytyy aliluokalle `Lausunto`, joten kyseinen ominaisuus liitetään ainoastaan luokkaan `Lakiasiakirja`.

Jos keskeisiä käsitteitä on etsitty olemassa olevien XML-dokumenttien avulla, voidaan XML-dokumentin elementtinimet usein kääntää luokiksi ja attribuuttinimet ominaisuuksiksi (Klein 2002). Esimerkissä 25 on osa XML-muotoisesta perustuslakivaliokunnan lausunnosta numero 61 vuodelta 2001. Esimerkkiin on lihavoitu mahdolliset luokkien nimet (vkl - valiokunnan lausunto ja edustaja) ja kursivoitu sekä lihavoitu mahdolliset ominaisuusnimet (tila, versio, kieli, tunniste ja numero). Tosin esimerkiksi elementtiä etunimi ei välttämättä kannattaisi mallintaa luokaksi, vaan ominaisuudeksi, vaikka se ei XML-attribuutti olekaan. On siis syytä tarkasti miettiä, mitkä käsitteet sopivat luokiksi ja mitkä ominaisuuksiksi sekä mitkä käsitteet XML-dokumentissa todella ovat oleellisia ontologian kannalta.

ESIMERKKI 25. XML-dokumentista löytyviä luokkien (lihavoitu) ja ominaisuuksien (kursivoitu ja lihavoitu) nimiä.

```
<vkl tila = "ok" versio = "Versio 2.0" kieli = "suomi"
    tunniste = "PeVL 61&sol;2001 vp">
…
<edustaja>
       <henkilo numero="467">
       <asema>pj.</asema>
       <etunimi>Paula</etunimi>
       <sukunimi>Kokkonen</sukunimi></henkilo>
       <ekryhma>kok</ekryhma>
</edustaja>
…
</vkl>
```

Rajoitteiden määrittely ominaisuuksille

Vaiheessa kuusi (6) kuvataan ominaisuusrajoitteiden avulla, minkälaisia arvoja ominaisuus voi saada (Noy & McGuinness 2001). RDF:ssä voidaan käyttää XML Schema – tietotyyppejä rajoittamaan ominaisuusarvoja tai voidaan määritellä, että ominaisuusarvon täytyy olla aina jonkin luokan ilmentymä. Esimerkiksi `Lakiasiakirja`-luokan ominaisuus `tekijä` voitaisiin määritellä olevan aina luokan `Henkilö` ilmentymä.

Ilmentymien luominen

Tässä vaiheessa luodaan yleensä sekä skeemaan suoraan kuuluvia ilmentymiä tai varsinaisia ilmentymiä. Skeemaan suoraan kuuluvia ilmentymiä tarvitaan varsinaisten ilmentymien luomisessa. Suomalaisiin asiakirjoihin saatetaan esimerkiksi määritellä luokalle `Kieli` valmiiksi kielet suomi ja ruotsi. Skeemaan kuuluvat ilmentymät voidaan Hyvösen, Salmisen, Junnilan ja Kettulan (2004) mukaan luoda käsin, tai joissain tapauksissa kerätä suoraan tietokannasta automaattisesti tai puoliautomaattisesti. Esimerkiksi yksityiskohtaisen henkilöontologian henkilöt voitaisiin kerätä suoraan yhteisön henkilötietokannasta, jos sellainen on olemassa (Hyvönen, Salminen, Junnila & Kettula 2004).

Kun itse skeema on valmis, luodaan varsinaisia ilmentymiä luokille. RDF:n kohdalla tämä tarkoittaa RDF-kuvauksien luomista RDF-skeeman luokkien ja ominaisuuksien mukaisesti. Voidaan esimerkiksi luoda luokalle `Lakiasiakirja` ilmentymän, jolla on ominaisuudet `nimi: HE 273/2002` ja `päivämäärä: 24.1.2003`. Ilmentymien luomisen jälkeen luodaan tarvittaessa päättelysääntöjä, joiden avulla voidaan osoittaa lisää suhteita asioiden välille. (Hyvönen, Salminen, Junnila & Kettula 2004.)

6.2 Huomioitavaa skeemojen suunnittelussa

Tässä kappaleessa kerrotaan, mikälaisia asioita pitää ottaa huomioon skeemoja suunniteltaessa. Pohditaan skeeman kehittämisprosessia, XML-nimiavaruuksien huomioimista, skeemojen julkaisemiseen liittyviä asioita, luokkien ja ominaisuuksien nimeämistä sekä sitä, miten tarkkoja käsitteiden tulisi skeemassa olla ja mitä käsitteitä ylipäätään skeemaan kannattaa mallintaa. Lisäksi annetaan ohjeita luokkahierarkian tarkistamiseen ja siihen, pitäisikö käsite mallintaa skeemaan luokkana, ominaisuutena vai ilmentymänä. Lopuksi käsitellään moniarvoisia suhdekuvauksia, joissa ominaisuus mallinnetaan skeemaan omaksi luokakseen.

Yleistä

Skeeman tai sanaston suunnittelussa täytyy muistaa, että ei ole olemassakaan yhtä oikeaa

tapaa kuvata sovellusaluetta. Paras ratkaisu riippuu tarkoituksesta ja sovelluksesta, johon skeemaa halutaan käyttää. Skeeman kehittäjän täytyy miettiä kuinka yksityiskohtainen tai yleinen skeemasta halutaan, millainen ratkaisu tukisi parhaiten skeeman tavoitetta ja millainen olisi intuitiivisin, laajennettavin ja ylläpidettävin skeeman muoto. Skeeman pitäisi kuvata todellisuutta, joten sen käsitteet ja suhteet pitäisivät olla todella lähellä sovellusalaa. (Noy & McGuinness 2001.)

Yhden skeeman ei pidä mallintaa kaikkea mahdollista tietoa sovellusalasta, vaan vain oleellinen. Skeemaan ei siis tarvitse koota luokkien kaikkia mahdollisia ominaisuuksia, vaan vain ne, jotka tavoitteena olevassa sovelluksessa tarvitaan. (Noy & McGuinness 2001.) Esimerkiksi lainsäädäntöprosessin toimijoilla saattaa olla ominaisuuksia kuten ikä ja paino, mutta ne ovat kuitenkin täysin irrelevantteja lainsäädäntöprosessin kontekstissa.

Skeemamäärittelyssä pitäisi tulla mahdollisimman hyvin esille, kuinka luokkia ja ominaisuuksia kuuluu tulkita. Määrää ilmaiseville ominaisuusarvoille kannattaa esimerkiksi määritellä, missä yksikössä arvosta puhutaan. Esimerkiksi onko pituuden yksikkönä senttimetri, maili vai kilometri. RDF-kuvauksissa saman ominaisuuden yksikkö täytyy pitää aina samana. (Hjelm 2001, 115.)

Ensimmäisen skeeman version jälkeen skeemaa voidaan arvioida ja korjata ongelmanratkaisumenetelmillä ja sovellusalan asiantuntijoiden kanssa. Skeeman kehittäminen onkin lähes aina iteratiivinen prosessi. (Noy & McGuinness 2001.)

Metatietoskeemoja täytyy päivittää jatkuvasti organisaation ja liiketoiminnan muutoksien mukaan ja muutenkin metatietojen tallentaminen pitäisi tehdä koko asiakirjan elinkaaren ajan (ISO/PDTS 23081). Muutokset tietotekniikassa, organisaatiossa ja liiketoimintamahdollisuuksissa aiheuttavat tarpeen päivittää metatietomäärittelyjen aikaisemmat versiot (Salminen 2003b).

Erilaiset käsitteet asiantuntijoille ja maallikoille

Hovyn (2003) mukaan ontologian olisi hyvä sisältää sanasto sekä asiantuntijoille että maallikoille. Näin taataan, että asiantuntijat pääsevät käsiksi yksityiskohtaiseen asiantuntijatietoon ja maallikot pystyvät nopeasti saamaan tiedon käsiinsä, ilman erityiskäsitteiden tuntemista (Hovy 2003). Esimerkiksi lakiteksteihin ja lainsäädäntöprosessiin liittyy paljon juridisia käsitteitä, joita tavallisen kansalaisen on vaikea ymmärtää. Erilaisten ontologioiden tekeminen sekä lainsäädännön asiantuntijoille, että kansalaisille helpottaisi molempia ryhmiä löytämään haluamansa tiedon nopeasti tuntemillaan käsitteillä.

Jos kuitenkin monen eri ontologian luominen ei ole esimerkiksi sovelluksen monimutkaistumisen takia järkevää, toisenlainen ratkaisu on mallintaa samaan skeemaan sekä yleisiä että yksityiskohtaisempia käsitteitä. Esimerkiksi lainsäädäntöprosessin vaihetietoon liittyen tavallisille kansalaisille saattaa riittää tieto, missä päävaiheessa (esimerkiksi eduskuntakäsittely) lakihanke on, mutta kansanedustaja haluaa tietää tarkemman vaiheen (esimerkiksi lähetekeskustelu).

XML-nimiavaruuksien huomioon ottaminen ja skeemojen julkaisu

RDF-määrittelyt pohjautuvat vahvasti XML-nimiavaruusmäärittelyihin. Nimiavaruuksia pitää käyttää XML-nimiavaruuksien sääntöjen mukaan. Kun halutaan käyttää muuta kuin RDF:n omia nimiavaruuksia (`rdf:` ja `rdfs:`) ja näistä poikkeavaa sanastoa, täytyy uudet sanastot esitellä XML-nimiavaruuksien kautta. (Hjelm 2001, 115.) Nimiavaruuksia skeemassa on siis ainakin kolme: RDF Scheman ja RDF-syntaksin sekä skeeman oma nimiavaruus. Lisäksi voidaan esitellä muita nimiavaruuksia, joita skeema käyttää (esimerkiksi Dublin Core).

Jos halutaan, että skeemaa voidaan vapaasti käyttää uudelleen, se täytyy julkaista. Julkaisemisen edellytyksenä on, että skeemalla on tietty URL, josta skeeman voisi periaatteessa löytää pitkän aikaa jälkeenpäinkin. Monet RDF-skeemat sijaitsevat yksinkertaisesti HTTP/FTP-palvelimella. Jos skeemaa käyttää useampi taho, täytyy varmistaa, että palvelin kestää tarpeellisen määrän liikennettä. (Hjelm 2001, 119.) Skeemaa käytettäessä se kannat-

taa joissain tapauksissa kyselyiden tehostamiseksi ladata palvelimelta lokaaliin tietokantaan (Quan & Karger 2004, 258).

Nimeämiskäytäntö luokille ja ominaisuuksille

RDF-skeeman luokille ja ominaisuuksille kannattaa kehittää jonkinlainen nimeämiskäytäntö. Se parantaa skeeman luettavuutta ja ymmärrettävyyttä. (Noy & McGuinness 2001.)

Yleinen ja suositeltava nimeämiskäytäntö RDF-kuvauksissa RDF Primer –spesifikaation (Manola & Miller 2004) mukaan on, että luokkien nimet kirjoitetaan isolla kirjaimella ja ominaisuuksien nimet pienellä. Käsitteiden nimissä ei saa käyttää mitään XML-merkkaukselle varatuista merkeistä, kuten merkit ”&” ja ”<” (Hjelm 2001, 115). Kaksisanaisista käsitenimistä on hyvä määritellä, kirjoitetaanko ne esimerkiksi alaviivalla (`Prosessin_vaihe`), tyhjällä välillä (`Prosessin vaihe`) vai erottamalla sanat isoilla kirjaimilla (`ProsessinVaihe`). On syytä miettiä, käytetäänkö luokkanimistä yksikköä vai monikkoa, esimerkiksi `Lakiasiakirja` vai `Lakiasiakirjat`. Kun käytetään systemaattisesti jompaakumpaa, suunnittelija välttyy virheeltä mallintaa samasta käsitteestä kaksi eri luokkaa. Sanoja ”luokka” tai ”ominaisuus” ei kannata käyttää käsitenimissä. Kontekstista käy joka tapauksessa ilmi kumpi on kyseessä. Yleensä on hyvä välttää lyhennyksiä luettavuuden parantamiseksi (käytetään esimerkiksi mieluummin ominaisuusnimessä sanaa `vaihenimi` kuin `vnimi`). Lisäksi on syytä valita skeemassa käytetty kieli skeeman käyttäjäryhmä mukaisesti. Yhtenäisestä nimeämiskäytännöstä on syytä pitää kiinni koko skeeman laajuudella. (Noy & McGuinness 2001.)

Luokkahierarkian tarkistaminen

Synonyymit samalle käsitteelle eivät edusta eri luokkia, vaan skeemassa on valittava yksi käytettävä termi yhtä asiaa kohden. Luokkasilmukoita pitäisi myös välttää. Jos määritellään luokan A:n olevan luokan B aliluokka ja lisäksi luokan B olevan luokan A aliluokka, loogisesti päätellen luokat A ja B samoja. (Noy & McGuinness 2001.)

Aliluokkasuhde on *transitiivinen*: Jos B on A:n aliluokka ja C on B:n aliluokka (A→B→C), silloin C on A:n aliluokka. Täytyy siis muistaa, että A:n ominaisuudet periytyvät luokan B lisäksi myös luokalle C. (Noy & McGuinness 2001.)

Sisarukset (siblings) ovat luokkia, jotka ovat suoria aliluokkia samalle luokalle. Kaikkien sisaruksien pitää olla samalla yleisyystasolla (esimerkiksi luokka `Vaihe` ei voi olla sisar luokalle `LakiesityksenEduskuntakasittely`, koska luokka `Vaihe` on yleisempi käsite). Yleisenä nyrkkisääntönä aliluokkien lukumäärästä voi pitää sitä, että aliluokkia olisi 2-12. Jos jollain luokalla on vain yksi aliluokka, on kyseessä todennäköisesti suunnitteluvirhe ja jos aliluokkia on enemmän kuin 12, saatetaan tarvita muutama välikategoria lisää. (Noy & McGuinness 2001.)

Sääntöjä päätöksenteon tueksi: Käsitteen mallintaminen joko luokkana, aliluokkana, ominaisuutena tai ilmentymänä

Noy & McGuinness (2001) esittävät artikkelissaan muutaman päätöksentekoa helpottavan nyrkkisäännön, jotka liittyvät skeeman suunnitteluun. Tässä kappaleessa säännöt esitetään lihavoituna.

Skeemaa suunnitellessa saattaa olla vaikea päättää, muodostetaanko luokalle aliluokkia vai mallinnetaanko tietty käsite muulla tavalla. Noyn ja McGuinnessin **sääntö 1: Aliluokilla on yleensä lisäominaisuuksia tai eri tavalla rajoitettuja ominaisuuksia yliluokkiin verrattuna**. Toisaalta taas joskus luokkaan ei liitetä ominaisuuksia ollenkaan, kun esimerkiksi halutaan vain erottaa tietyt tärkeät käsitteet toisistaan.

Joskus ei ole ollenkaan yksiselitteistä muodostetaanko asiasta uusi luokka vai esitetäänkö asia ominaisuuden avulla. Esimerkiksi muodostetaanko omat luokat eri asiakirjatyypeille (`Valiokuntamietintö`, `Hallituksen esitys` jne.) vai esitetäänkö asia antamalla luokalle `Lakiasiakirja` ominaisuus `asiakirjatyyppi`. Noyn ja McGuinnessin **sääntö 2**: **Uusi luokka voidaan muodostaa, jos käsitteen erottaminen on sovellusalalla tärkeää ja sovellusalalla ajatellaan näiden eri käsitteiden olevan erilaisia olioita.** Lakiasiakirjan tapauk-

sessa ominaisuus asiakirjatyyppi saattaisi olla parempi ratkaisu, sillä eri asiakirjatyyppien voidaan ajatella olevan samanlaisia Lakiasiakirja-olioita.

Skeemaa suunniteltaessa voidaan joutua pohtimaan, pitäisikö joku käsite ilmaista luokkana vai luokan ilmentymänä. Esimerkiksi luokalle `Kieli` voidaan määritellä ominaisuudet `kielen nimi` ja `kielen tunnus` ja ilmaista kielet näin ilmentyminä. Toinen vaihtoehto on määritellä kielet suoraan luokan kieli aliluokaksi (aliluokiksi `Suomi, Ruotsi` jne.). Noyn ja McGuinnessin **sääntö 3**: **Ilmentymät ovat aina kaikista tarkimpia käsitteitä skeemassa**. Tällä perusteella kieli, esimerkiksi suomi, on kieli-luokan tarkin käsite, joten se voidaan kuvata skeeman ilmentymänä. Toinen Noyn ja McGuinnessin esittämä ohjeistus aiheeseen on **sääntö 4**: **Uusi luokka voidaan muodostaa, jos käsitteet muodostavat luonnollisen hierarkian** (esimerkiksi muodostetulla luokalla voidaan ajatella olevan vielä aliluokkia). Noy ja McGuinness käyttävät esimerkkiä eri viinialueista, jaottelemalla ne maiden mukaan (Yhdysvallat, Ranska, Saksa jne.). Näiden alueiden sisällä on monenlaisia viinialueita, kuten Bourgogne Ranskan alueella, mutta koska vielä Bourognessakin on omia alueitaan, voidaan siitäkin muodostaa luokka, jotta sille saadaan tehtyä aliluokkia.

Moniarvoiset suhdekuvaukset RDF:ssä

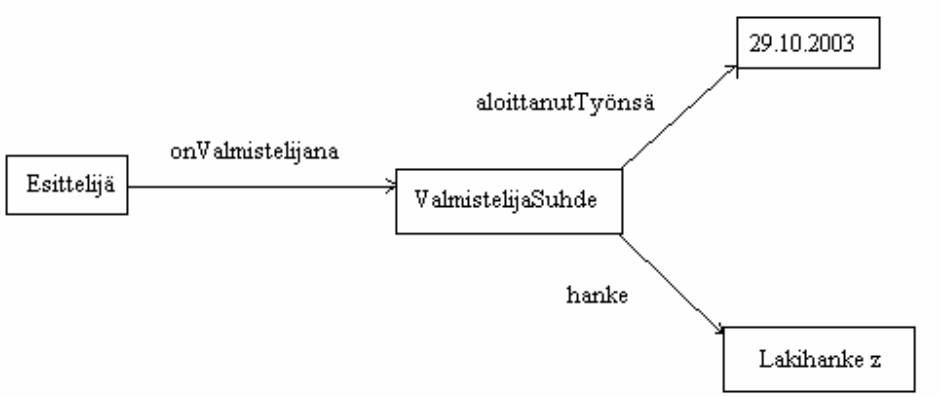
Noy ja Rector (2004) ovat pohtineet artikkelissaan (W3C Working Draft) moniarvoisten suhteiden määrittämistä semanttisessa webissä. Tämän luvun tiedot perustuvat suurimmaksi osaksi heidän havaintoihinsa.

RDF-suhdekuvauksissa suhde ei usein ole pelkkä ominaisuus, vaan suhteesta tehdään uusi luokka RDF-skeemaan ja uusi solmu RDF-kuvailuun (Ensel & Keller 2002). Yksi esimerkki aiheesta oli kappaleessa 2.4., jossa ominaisuus osoite esitettiin erillisen *tyhjän solmun* avulla rakenteisena arvona sisältäen osat katu, postinumero ja kaupunki. Suhdeluokkien nimet voivat olla oikeastaan mitä tahansa tai ne voidaan esittää tyhjinä solmuina, sillä ne ovat vain apuna kokoamassa yhteen liittyvät jäsenet kiinni toisiinsa.

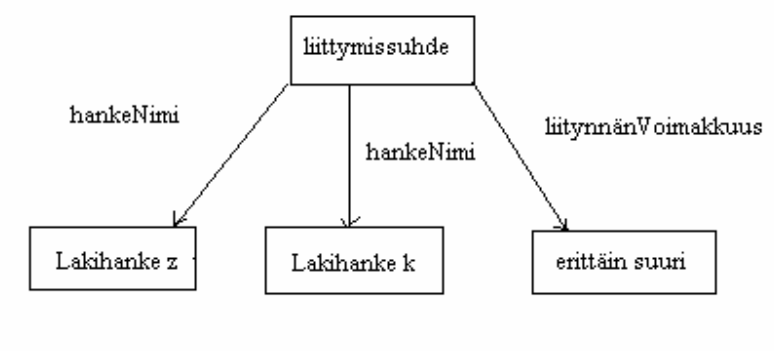
Suhdeluokan tavoitteena on yleensä se, että suhteesta itsestään voidaan tallentaa lisätietoa

tai että kuvauksen rakenteesta tulee selkeämpi ja siihen voidaan helpommin kohdistaa kyselyjä. (Ensel & Keller 2002) Suhteesta tallennettavia tietoja voivat olla esimerkiksi suhteen *voimakkuuden* tai *todennäköisyyden* kertova tieto (Valiokunnan mietintö saadaan pian valmiiksi erittäin suurella todennäköisyydellä) tai suhteen *tarkoituksen* kertova tieto (henkilö x on tehnyt ostoksen y häälahjaksi) tai suhdekuvauksen *luontitieto* (lakihanke z liittyy lakihankkeeseen k henkilön x mielestä).

Aina ei ole yksiselitteistä, minkälaisena rakenteena suhdekuvaus esitetään. Kuviossa 12 on esitetty tapa, jossa suhdesolmusta erotetaan yksilö (tässä: Esittelijä), joka on suhdekuvauksen ”alkuunpanija”. Kuviossa 13 on esitetty toinen tapa, jossa moniarvoinen suhde (tässä: liittymissuhde) edustaa suhteeseen kuuluvien jäsenten verkkoa, jossa jäsenillä on oma roolinsa, mutta samanveroinen merkitys.

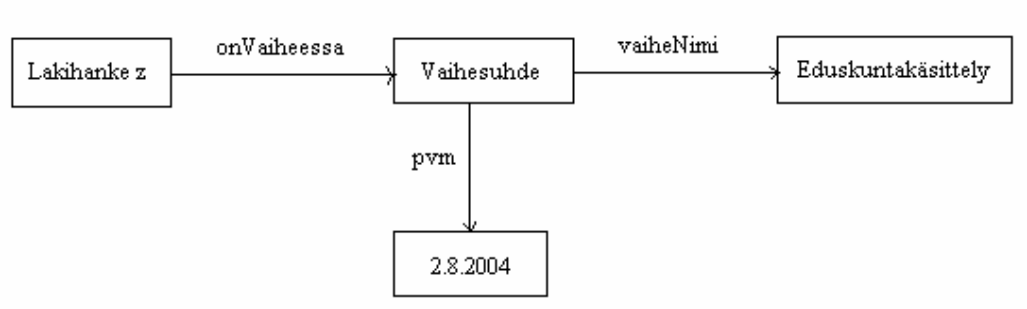


KUVIO 12. Moniarvoisen suhteen esittäminen, tapa 1: Esittelijä suhdekuvauksen ”alkuunpanijana”.

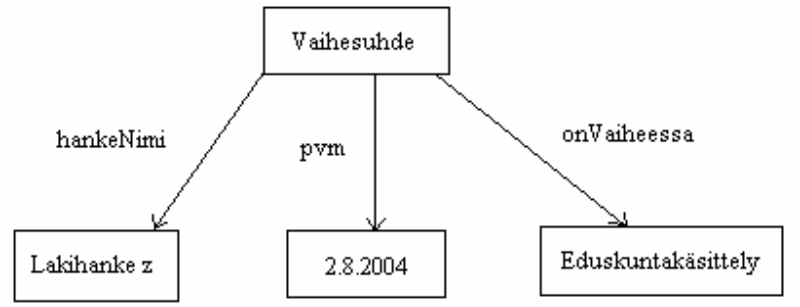


KUVIO 13. Moniarvoisen suhteen esittäminen, tapa 2: Kaikki suhdekuvauksen jäsenet samanarvoisia.

Joissain tapauksissa yllä esitetyistä tavoista molemmat tuntuvat sopivilta. Tällöin kuvauksien tekijä päättää itse, kumman tavan haluaa valita. Jos halutaan esimerkiksi kuvata, että tietty lakihanke on vaiheessa eduskuntakäsittely ja tämä väittämä on sanottu tiettynä päivämääränä, voidaan se esittää kahdella eri tavalla kuvioiden 14 ja 15 mukaisesti. Kuviossa 14 `Lakihanke` esitetään ”alkuunpanijana”, josta ominaisuusnuolet lähtevät kohti muita resursseja. Kuviossa 15 esitetään kaikki ryhmän jäsenet (`hankeNimi, onVaiheessa, pvm`) samanarvoisina `Vaihesuhde`-resurssissa kiinni.



KUVIO 14. Vaihesuhteen esittäminen tavalla 1, jossa Lakihanke z toimii ns. ”alkuunpanijana” kuvion vasemmalla puolella.



KUVIO 15. Vaihesuhteen esittäminen tavalla 2, jossa kaikki ryhmän jäsenet ovat samanarvoisia ja kiinni Vaihesuhde-resurssissa.

Vaihtoehtoiset esitystavat tämäntyyppisille suhderakenteille ovat RDF:n omat rakenteet `rdf:Statement`-elementti (esiteltiin luvussa 4.5.) ja `rdf:value`-elementti (tarkemmat tiedot RDF-spesifikaatioissa).

6.3 Työkalut RDF-skeemojen suunnittelussa

Tässä kappaleessa kerrotaan mitä työkaluja RDF-skeemojen suunnittelussa tarvitaan ja miksi työkalut ovat hyödyllisiä. Lopuksi esitellään kaksi esimerkkityökalua, joita tarvitaan skeemojen RDF/XML-syntaksin jäsentämiseen, RDF-kuvailujen validointiin vastaavan skeeman mukaiseksi sekä skeeman luokkien ja ominaisuuksien suunnittelun helpottamiseen.

Yleistä työkaluista

Työkalut ovat tärkeitä RDF-skeemojen suunnittelussa ja RDF-kuvauksien tekemisessä. Esimerkiksi HTML:ään verrattuna RDF:n luominen on paljon monimutkaisempaa (Quan, Huynh & Karger 2003). Tietokonemaailmassa RDF:n kaltaisesta tiedon esittämismuodosta ei ole hyötyä, jos kaikki pitää tehdä käsin (Hjelm 2001, 254).

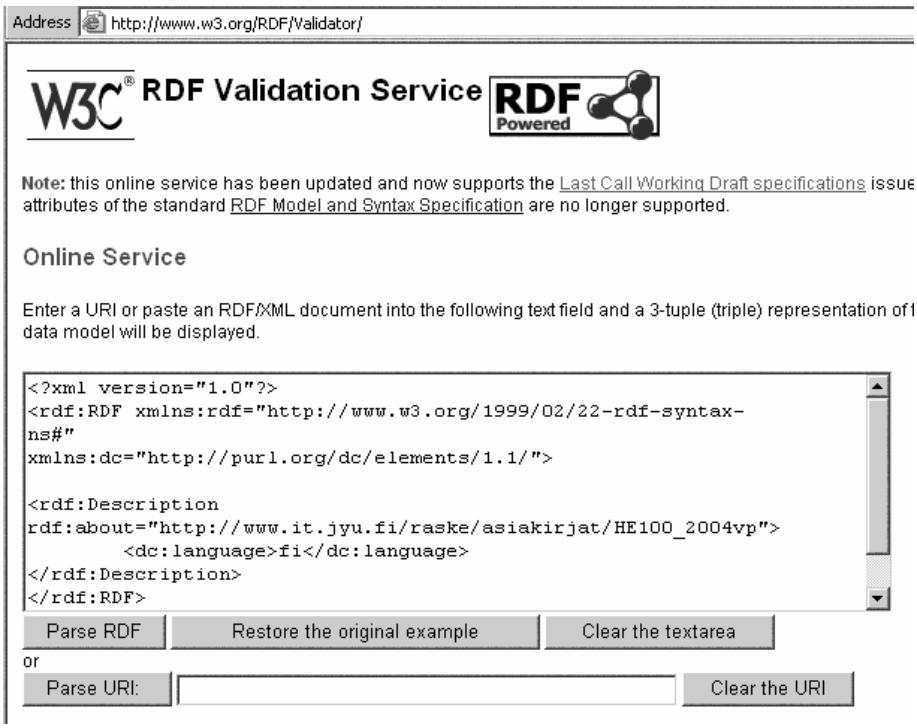
RDF-jäsennin (parser) on välttämätön lähes jokaisessa RDF-sovelluksessa. Jäsennin käy läpi koko RDF-dokumentin ja selvittää siinä mahdollisesti esiintyvät syntaktiset virheet. RDF-jäsennintä tarvitaan, että sovellus osaa tulkita RDF/XML-muodon abstraktiksi RDF-tietomalliksi (Ahmed, Ayers, Birbeck ym. 2001, 205; Hjelm 2001, 255). RDF/XML-muotohan on vain yksi esitystapa taustalla olevalle tietomallille.

Jos samassa dokumentissa on sekä XML että RDF/XML-muotoista merkkausta, RDF-jäsennin tutkii dokumentista vain `rdf-` ja `rdfs`-alkuisten elementtien sisältämät osat ja XML-jäsennin hoitaa loput (Ensel & Keller 2002). Pelkkä XML-jäsennin ei riitä tulkitsemaan RDF/XML:ää, koska se osaa tulkita XML-muodosta vain elementtien hierarkkisen rakenteen eikä resursseja ja niiden ominaisuuksia, niin kuin RDF-malli vaatisi. Tuloksena RDF-jäsennyksestä saadaan RDF/XML-muotoiset väittämät kolmikkomuotoisina. (Ahmed, Ayers, Birbeck ym. 2001, 204-206.) Käytännössä RDF-jäsentimet (esimerkiksi seuraavassa kappaleessa esitelty W3C:n RDF-jäsennin) hoitavat yleensä XML-jäsennyksen sekä validoinnin RDF/XML-syntaksin skeemaa (`rdf:`) ja RDF Schemaa (`rdfs:`) vasten.

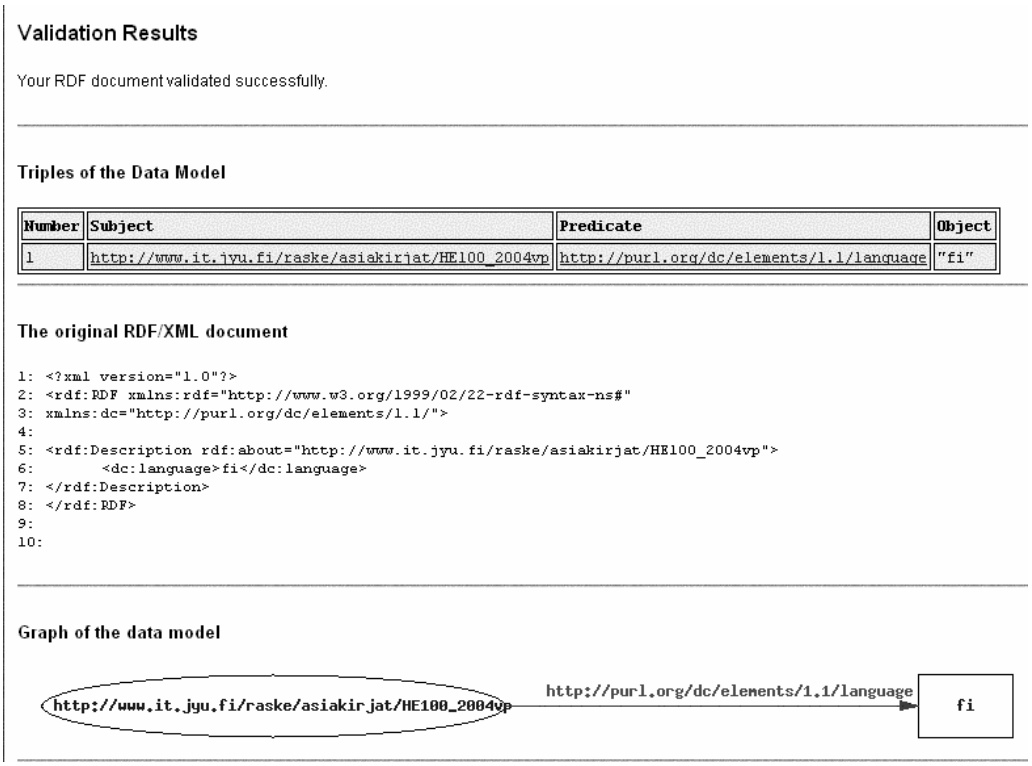
RDF/XML-syntaksin jäsentämisen jälkeen voidaan suorittaa vielä RDF-kuvauksen validointi vastaavan skeeman suhteen. Validoinnilla tarkistetaan, onko kuvauksessa käytetty skeeman määrittelemää sanastoa ja onko sanastoa käytetty oikein.

W3C:n RDF-jäsennin

W3C ylläpitää omaa RDF-jäsennintä URL-osoitteessa http://www.w3.org/RDF/Validator/. Siihen syötetään RDF/XML-muoto RDF-skeemasta tai –kuvauksesta. Jäsentäjä tarkistaa RDF/XML:n oikeellisuuden sekä muodostaa siitä kolmikkoesitystavan ja RDF-graafin RDF-tietomallin mukaisesti. Kuviossa 16 on RDF-jäsentimen käyttöliittymä, jonka tekstilaatikkoon voidaan syöttää haluttu RDF/XML-dokumentti tai antaa RDF/XML-dokumentin URL sille varattuun kenttään. Tässä tapauksessa tekstilaatikkoon on syötetty RDF/XML-muoto erään hallituksen esityksen kielen kuvaamisesta. Jäsennyksen tulos näkyy kuviossa 17. Tulos sisältää tiedon RDF/XML-dokumentin virheellisyydestä tai virheettömyydestä sekä dokumentin pohjalta generoidut kolmikko- ja RDF-graafimuodot.



KUVIO 16. W3C:n RDF-jäsentäjän käyttöliittymä.



KUVIO 17. W3C:n RDF-jäsentimen tulos: RDF/XML-muodon mahdolliset virheet sekä kolmikko- ja graafiesitystavat.

Protégé-2000 RDF-skeemasuunnittelun tukena

Protégé-2000-työkalu on yksi vaihtoehto RDF-skeemojen luomiseen tai RDF-kuvailujen validoimiseen RDF-skeeman mukaiseksi. Se on ontologiaeditori ja tietämyksenhankintajärjestelmä (knowledge acquisition system), joka on kehitetty Stanfordin lääketieteellisessä yliopistossa (Stanford University School of Medicine).

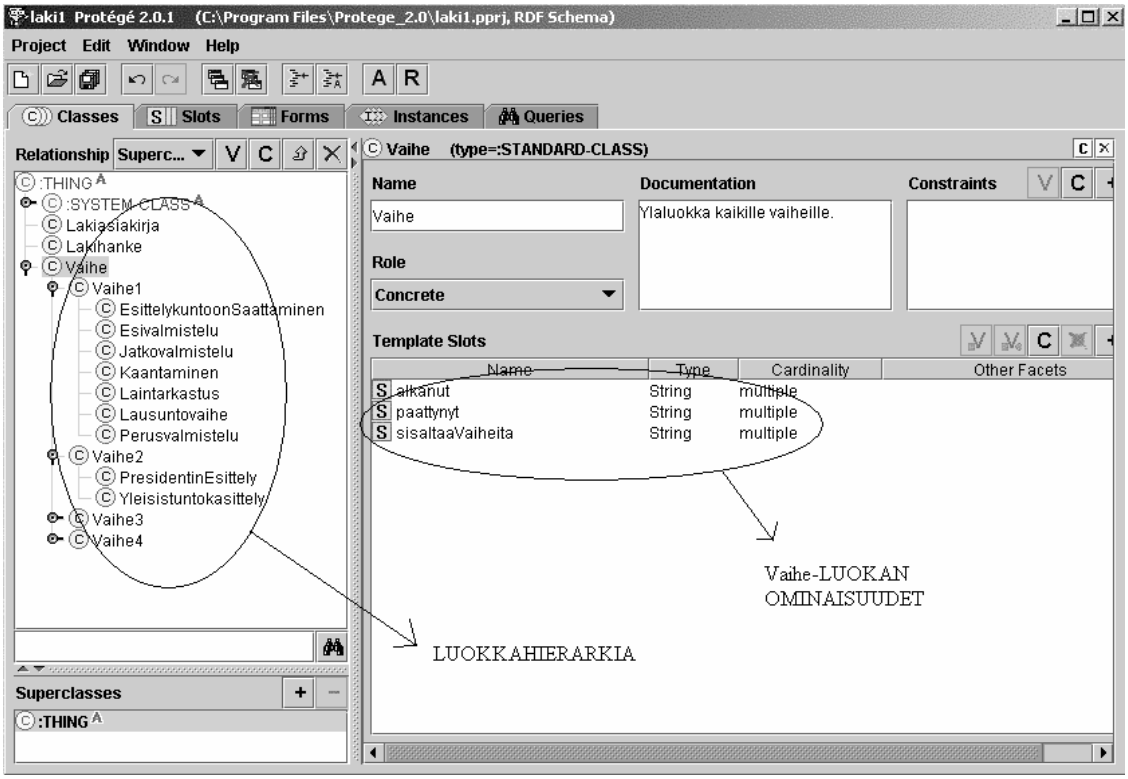
Protégé valittiin tämän tutkielman esimerkkityökaluksi, koska sitä on kehitelty pitkään (1990-luvun alusta asti), lukuisat projektit muun muassa Helsingin, Washingtonin, Amsterdamin ja Heidelbergin yliopistoista käyttävät tai ovat käyttäneet sitä (Protégé-2000 2004; Hyvönen, Saarela, Viljanen ym. 2004) ja se mainitaan RDF:stä kertovassa kirjallisuudessa (mm. Hjelm 2001, 27; Ahmed, Ayers, Birbeck ym. 2001, 466).

Protégé-2000:ta ei ole kehitetty alun perin RDF-editoriksi, mutta työkaluun on saatavissa RDF-lisäosa (plugin). Skeeman suunnittelu alkaa Protégéssa luokkien ja ominaisuuksien nimien määrittämisellä ja vasta sitten skeema tallennetaan RDF/XML-muotoon. Toinen vaihtoehto Protégé-2000-työkalun käyttämiseen on valmiiden RDF-skeemojen ja -kuvausten tuominen (import) Protegeen ja kuvausten validointi skeeman mukaisesti.

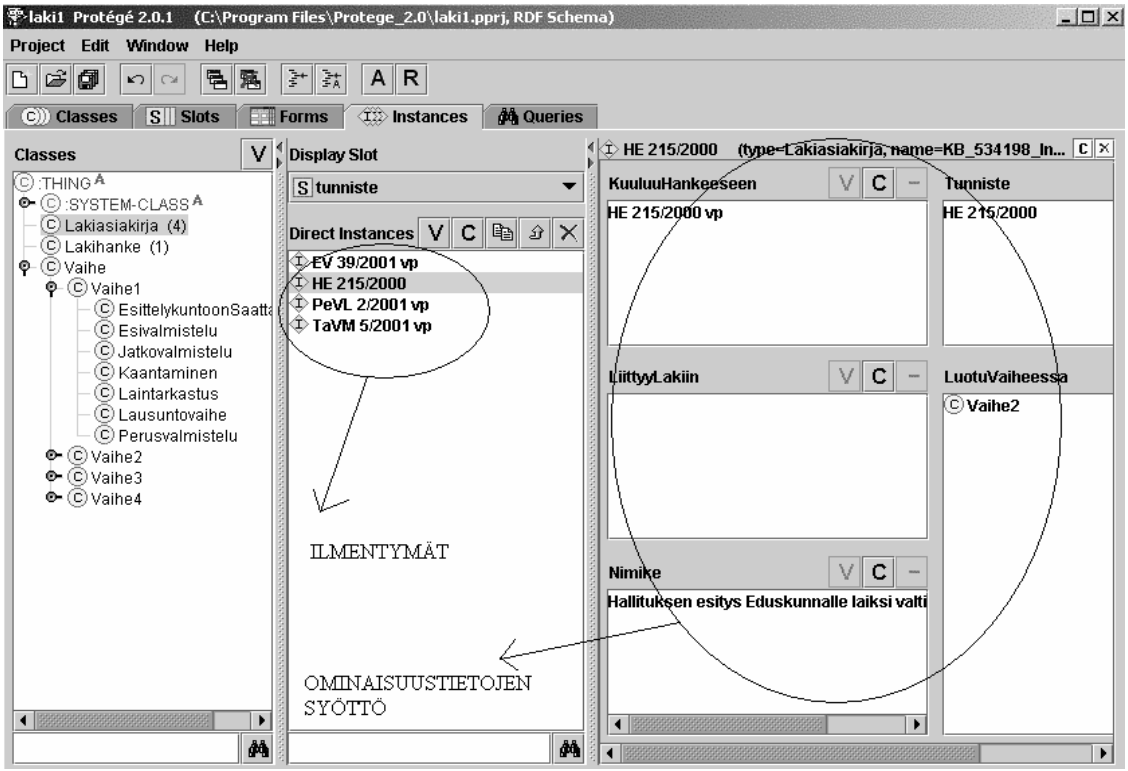
Kun projekti tallennetaan RDF/XML-muotoon, Protégé muodostaa haluttuun hakemistoon kaksi eri tiedostoa: RDF-skeeman mallinnettujen luokkien ja ominaisuuksien mukaan (tiedostopääte ”`.rdfs`”) sekä RDF-kuvauksen skeeman ilmentymistä (tiedostopääte ”`.rdf`”). Protégén avulla skeeman voi tallentaa myös muiden semanttisen webin ontologiakielten muodossa. Protégé tukee muun muassa kieliä: RDF, DAML+OIL, OWL ja XML Schema. (Noy, Sintek, Decker ym. 2001.)

Kuviot 18 ja 19 ovat Protégé-2000-käyttöliittymästä. Käyttöliittymässä on viisi eri välilehteä, jotka on lueteltu alla tehtävineen.

- Luokat, Classes (kuvio 18): Luokkien, luokkahierarkian ja ominaisuuksien määrittely.
- Ominaisuudet, Slots: Ominaisuuksien määrittely.
- Lomakkeet, Forms: Ilmentymät-välilehden lomakkeiden ulkoasun muokkaus.
- Ilmentymät, Instances (kuvio 19): Ilmentymien tietojen syöttö luokkien ja ominaisuuksien mukaisesti.
- Kyselyt, Queries: Kyselyjen tekeminen ja tallentaminen ilmentymätiedoista.



KUVIO 18. Protégé-2000 Luokat-välilehti: Lainsäädäntöprosessia kuvaavan RDF-skeeman *luokkahierarkiaa*.



KUVIO 19. Protégé-2000 Ilmentymät-välilehti: Lainsäädäntöprosessia kuvaavan RDF-skeeman ilmentymiä.

Kuviossa 18 on luokat-välilehden vasemmasta reunasta ympyröity skeeman luokkahierarkian luokat. Oikealla puolella näkyvät valitun luokan ominaisuudet eli tässä tapauksessa luokan `Vaihe` ominaisuudet `alkanut`, `päättynyt` ja `sisältääVaiheita`. Kuviossa 19 on ilmentymät-välilehden keskiosasta ympyröity valitun luokan (`Lakiasiakirja`) ilmentymät. Käyttöliittymän oikeassa reunassa näkyvät valitun ilmentymän (HE 215/2000) ominaisuudet.

Työkalut, jotka on kehitetty nimenomaan RDF:lle, saattavat joissain tapauksissa olla parempia ja tarkempia sanaston kehittämiseen. Protégé-2000-työkalun erityinen etu on kuitenkin se, että sanaston kehittäjä voi muodostaa sanaston ensin ja miettiä vasta jälkeenpäin, minkä semanttisen webin kielen mukaan sanasto halutaan esittää. (Noy, Sintek, Decker ym. 2001.)

Protégén generoima RDF-koodi on virheetöntä W3C:n jäsentäjän mukaan. Kokeilujen perusteella havaitsin kuitenkin muutamia ongelmia Protégé-2000:n RDF/XML-syntaksissa, jotka Protégén käyttöoppaassakin mainitaan. Ne liittyvät muun muassa nimiavaruuksiin (rdfs-nimiavaruus vanha, omia nimiavaruuksia voi määritellä vain yhden skeemaa kohti), URI-tunnisteisiin (ei voida määrittää haluamakseen) ja XML Schema –määrityksiin (eivät mahdollisia) (Protégé-2000 User Guide 2004). Nämä Protégén tuottaman skeeman virheet ja puutteet voidaan kuitenkin korjata käsin jälkeenpäin, joten Protégén tuottama RDF/XML-muoto nopeuttaa mielestäni huomattavasti RDF-skeeman ja RDF-kuvailujen luomista verrattuna skeeman luomiseen alusta alkaen käsin.

7 LAINSÄÄDÄNTÖPROSESSIN TIEDONHALLINNAN TUKEMINEN RDF:N AVULLA

Tässä luvussa kerrotaan, miten RDF:ää voidaan käyttää suomalaisen lainsäädäntöprosessin tiedonhallinnan tukemiseen. Ensin esitellään asiaa tutkiva RASKE2-projekti, jonka osana tämä tutkielma tehdään. Lainsäädäntöprosessista esitellään prosessin vaiheet ja keskeisimmät tietojärjestelmät. Sen jälkeen kartoitetaan eduskunnan ja valtioneuvoston tiedonhallinnon haasteita ja ongelmia. Lisäksi esitellään tutkielman tuloksena syntynyt RDF-prosessiskeemaehdotus, skeeman luokkia ja ominaisuuksia sekä hyödyntämismahdollisuuksia ja käyttöideoita. Lopuksi arvioidaan prosessiskeeman hyötyjä ja ongelmia.

7.1 RASKE2-projekti

Tämä tutkielma tehdään osana RASKE2-projektin tutkimusta. RASKE2-projektin muu tutkimus ja kartoitustyö liittyvät tämän tutkimuksen taustoihin ja jatkotutkimusaiheisiin, joten projekti kuvataan lyhyesti tässä yhteydessä.

RASKE2-projekti on eduskunnan, valtiovarainministeriön, oikeusministeriön ja Jyväskylän kolmivuotinen yhteistyöprojekti. Projekti on jatkoa RASKE- ja EULEGIS- projekteille, joissa kehitettiin uusia ratkaisuja sähköisten asiakirjojen hallintaan ja rakenteistamiseen sekä eurooppalaisen lakitiedon hakuun. RASKE-projekti alkoi vuonna 1994, ja sen tuloksena syntyivät SGML-muotoiset dokumenttityyppimääritykset eduskunnalle ja ministeriöille. RASKE2-projektissa hyödynnetään edellisten projektien tuloksia ja ratkaisuissa pyritään käyttämään semanttisen webin tekniikoita. RASKE2-projektin tavoitteena on suomalaisen lainsäädäntöprosessin keskeisten metatietojen standardoiminen ja tiedonhallinnan parantaminen tätä kautta. (Lehtinen, Salminen & Huhtanen 2004, 1-2.)

Keskeisten metatietojen kartoittamiseksi on kesän 2004 aikana toteutettu asiantuntijahaastatteluita. Tavoitteena on tuottaa metatietoanalyysi lainsäädäntöprosessille haastattelujen, asiakirja-analyysin, järjestelmäanalyysin ja prosessikuvausten pohjalta.

7.2 Lainsäädäntöprosessin vaiheet ja keskeisimmät tietojärjestelmät

Lainsäädäntöprosessin tiedonhallinta on hyvin monimutkaista. Prosessiin osallistuu monia eri organisaatioita ja henkilöitä, prosessissa saattaa syntyä erittäin laaja dokumentaatio ja toimijat käyttävät apunaan kymmeniä eri tietojärjestelmiä. Prosessi toistuu harvoin täsmälleen samanlaisena, sen kesto voi olla useita vuosia ja sen aikana tarvitaan paljon aikaisemmissa lainsäädäntöhankkeissa tuotettua aineistoa. (Lehtinen, Salminen & Huhtanen 2004, 1.) Tässä luvussa esitellään lainsäädäntöprosessin vaiheet ja sen tiedonhallinnassa käytettäviä tietojärjestelmiä.

Lainsäädäntöprosessin vaiheet

RASKE2-projektin väliraportissa (Lehtinen, Salminen & Huhtanen 2004, 21-22) lainsäädäntöprosessi on jaettu neljään eri vaiheeseen, jotka tässä esitellään lyhyesti väliraportin tietoihin perustuen.

Vaiheessa 1 *lainsäädännön valmistelu* tehdään siinä ministeriössä, jonka toimialaan asia kuuluu. Valmistelun voi tehdä yksi virkamies tai lainsäädäntöhanketta varten perustettu työryhmä tai komitea. Vaihe voidaan jakaa edelleen esivalmisteluun, perusvalmisteluun, lausuntovaiheeseen, jatkovalmisteluun, kääntämiseen, laintarkastukseen ja esittelykuntoon saattamiseen. Keskeisenä tuloksena vaiheesta syntyy valtioneuvoston käsittelyyn lähetettävä hallituksen esitys.

Vaihe 2, *hallituksen esityksen valtioneuvostokäsittely*, voidaan jakaa yleisistuntokäsittelyyn ja presidentin esittelyyn. Valtioneuvoston yleisistunnossa esittelijänä toimii valtioneuvoston esittelijä ja presidentin esittelyssä ministeri. Vaiheen tuloksena on hallituksen esitys, joka voidaan lähettää eduskuntakäsittelyyn.

Vaihe 3, *lakiesityksen eduskuntakäsittely*, voidaan jakaa viiteen alavaiheeseen: vireille tulo eduskunnan täysistunnossa, lähetekeskustelu, valiokuntakäsittely sekä eduskunnan täysistunnon ensimmäinen ja toinen käsittely. Valiokunta valmistelee mietinnön, joka käsitellään

täysistunnossa. Keskeisimpänä tuloksena tästä vaiheesta syntyy eduskunnan vastaus.

Vaiheessa 4 *hyväksytty laki käsitellään valtioneuvostossa*. Alavaiheina ovat presidentin tekemä lain vahvistaminen, lain julkaiseminen, lain voimaantuleminen sekä lain täytäntöönpano ja seuranta. Vaiheen tuloksena syntynyt laki julkaistaan Suomen säädöskokoelmassa sekä paperimuodossa että sähköisenä Internetissä FINLEX-säädöstietopankissa.

Lainsäädäntöprosessin keskeisimmät tietojärjestelmät

Lainsäädäntöprosessiin liittyy monia tietojärjestelmiä ja tässä esitellään lyhyesti niistä keskeisimmät. *PTJ* on valtioneuvoston päätöksenteon tukijärjestelmä, jolla tehdään valtioneuvoston istuntojen esittelylistat ja niiden liitteet sekä hallituksen esitysten taittaminen. PTJ:n kautta asiakirjoja jaellaan valtioneuvoston istuntoihin, tasavallan presidentin esittelyyn, eduskuntaan ja painotalo Editaan.

Ministeriön omilta web-sivuilta saa tietoa valmisteilla olevista lainsäädäntöhankkeista ja muusta lainsäädäntötyöstä, mitä kyseisessä ministeriössä on tehty. *Valtioneuvoston web-sivuilta* (http://www.valtioneuvosto.fi) saa tiedot valtioneuvoston yleisistunnossa tehdyistä ratkaisuehdotuksista ja tasavallan presidentin esittelystä.

HARE (http://www.hare.vn.fi/) on eduskunnan ja ministeriöiden yhteinen hankerekisteri, johon ministeriöt ovat velvoitettuja ylläpitämään tietoja lainsäädäntö-, kehittämis- ja valmisteluhankkeista. Eduskunnalla (http://www.eduskunta.fi) on monenlaisia *intranet ja internet –palveluja*. Kaikki lainsäädäntöön liittyvä aineisto erikoisvaliokuntien pöytäkirjoja lukuun ottamatta on saatavissa internetistä. Eduskunnan tuottamat valtiopäiväasiakirjat laaditaan rakenteissa SGML-muodossa ja internetissä asiakirjoja voidaan hakea monipuolisia hakupalveluja käyttäen pdf-, sgml- tai html-muodossa. *FINLEX*-säädöstietopankista (http://www.finlex.fi/) käsin pääsee lainsäädäntöön liittyen hakemaan sekä julkaistuja lakeja että hallituksen esityksiä.

7.3 Katsaus eduskunnan ja valtioneuvoston tiedonhallinnan haasteisiin ja ongelmiin

Lainsäädäntöprosessin tiedonhallinta koostuu sekä eduskunnan että valtioneuvoston järjestelmistä ja toimintatavoista. Tässä kappaleessa kerrotaan näiden yhteisöjen tiedonhallinnan haasteista ja ongelmista.

Asiakirjahallinnon kehittämiselle aiheuttavat vaatimuksia viranomaisen toiminnan julkisuudesta annetut säädökset sekä julkisuuslain edellyttämä hyvä tiedonhallintatapa. Ne edellyttävät julkishallinnossa mahdollisimman helposti ja taloudellisesti käytössä olevia ajantasaisia tietovarantoja. (Valtioneuvoston tietohallintostrategia 2003-2007.) SÄHKE-määrittelyn (Moisio & Leppänen 2003) mukaan muita lakeja tai ohjeita, jotka aiheuttavat vaatimuksia tiedonhallinnalle julkishallinnossa ovat henkilötietolaki, laki sähköisestä asioinnista viranomaistoiminnassa, valtiovarainministeriön tietoturvallisuusohje ja arkistolaki.

Koska tietotekniikan käyttö ulottuu yhä syvemmälle eduskunnan ja valtioneuvoston toiminnan ytimeen, tietohallinnon osaaminen ja koordinoiminen on tärkeää (Eduskunnan tietohallinnon linjat 2002-2004; Valtioneuvoston tietohallintostrategia 2003-2007). Valtioneuvostossa on useita yhteisiä tietojärjestelmiä, joiden metatiedot poikkeavat toisistaan hyvin paljon. Vain Senaattorissa (valtioneuvoston intranet-portaali) käytetään JHS 143-metatietosuositusta. Järjestelmien metatietokenttien yhdenmukaistaminen mahdollistaisi XML-tiedonsiirrot järjestelmien välillä. (Saatsi 2003.)

Muutenkin eduskunnan ja valtioneuvoston nykyisistä ydinjärjestelmistä puuttuu yhtenäinen arkkitehtuuri. Sovellusten sisältämien tietojen yhtenäisyys ja käyttö muista sovelluksista on vaikeaa, niiden tekninen perusta on erilainen ja integroitavuus muihin järjestelmiin usein vajavainen. (Eduskunnan tietohallinnon linjat 2002-2004; Valtioneuvoston tietohallintostrategia 2003-2007.) Huhtasen (2003, 84) mukaan hajanaisten järjestelmien seurauksena on, että prosessit katkeilevat ja laadullisia ongelmia esiintyy, kun asiakirjoja siirretään järjestelemästä ja esitystavasta toiseen.

Tärkeä tavoite eduskunnan tiedon- ja tietämyksenhallinnassa on tiedon tehokas ja käyttäjä-

lähtöinen hyödynnettävyys sekä yksilön tukeminen tietotulvan hallinnassa. Esimerkiksi kansanedustajilla on poliittisina toimijoina erilaiset tiedon hallintaan liittyvät tavoitteet kuin eduskunnan virkamiehistöllä. (Eduskunnan tiedon ja tietämyksen hallinta 2001.) Saatavilla olevan tiedon määrä on suuri, mutta laatu vaihteleva. Eduskunnassa tarvitaankin uusia keinoja tiedon suodattamiseen ja profilointiin. (Eduskunnan tietohallinnon linjat 2002-2004.)

RASKE2-projektin asiantuntijahaastatteluissa on tähän mennessä esille tullut muun muassa seuraavia erityisesti lainsäädäntöprosessin tiedonhallintaan liittyviä ongelmia. Ongelmia ja niiden seurauksia on kuvailtu laajemmin jäljempänä tulevissa kappaleissa.

1. Tiedotuksen puutteellisuus lakihankkeen alkamisesta ja etenemisestä.
2. Erilaisten järjestelmien suuri määrä ja hajanaisuus.
3. Eduskunnan ja valtioneuvoston asiakirjojen rakenteellinen ja ulkoasua koskeva erilaisuus.
4. Asiakirjojen ja asioiden tunnisteiden erilaisuus valtioneuvostossa ja eduskunnassa.
5. Epäyhtenäisyys ja erilainen navigointirakenne ministeriöiden web-sivuilla.
6. Samojen metatietojen tallentaminen moneen paikkaan.

Keskeinen ongelmakenttä lainsäädäntöprosessin aikana tapahtuvassa tiedonhaussa on saada tietää, mitä lakihankkeita on vireillä. Saattaa olla, että kaksi toisiaan koskevaa lakihanketta etenee jopa lakiesityksen eduskuntakäsittelyyn saakka ilman, että ne ovat tietoisia toistensa olemassa olosta. Jos näin käy, prosessissa saatetaan tehdä turhaa työtä, jos esimerkiksi kahden eri hankkeen tulokset ovat ristiriidassa keskenään. HARE-järjestelmästä pitäisi löytyä kaikki lakihankkeet, joita ministeriöissä valmistellaan. Järjestelmään kuitenkin tuotetaan tietoja epäyhtenäisellä tavalla eri ministeriöissä, eivätkä tiedot tästä syystä aina ole kattavia ja ajan tasaisia. HARE-järjestelmän käytössä on ongelmana sen erillisyys muista järjestelmistä ja se, ettei sitä sen takia ole aikaa ja halua käyttää.

Sekä eduskunnan, että valtioneuvoston nykyiset sovellusarkkitehtuurit perustuvat erillisiin tietojärjestelmiin ja sovellusten sisältämien tietojen yhtenäisyys ja integroitavuus muihin järjestelmiin on vaikeaa (Eduskunnan tietohallinnon linjat 2002-2004, Valtioneuvoston tietohallintostrategia 2003-2007). Huhtasen (2003, 84) mukaan hajanaisten järjestelmien seu-

rauksena on, että prosessit katkeilevat ja laadullisia ongelmia esiintyy, kun asiakirjoja siirretään järjestelemästä ja esitystavasta toiseen. Saatavilla olevan tiedon määrä on suuri ja ongelmana on oleellisen tiedon löytäminen. Tarvitaan keinoja tiedon suodattamiseen ja henkilökohtaiseen profilointiin. (Eduskunnan tietohallinnon linjat 2002-2004.)

Eduskunnan sisällä asiakirjojen identifiointi tuli RASKE-projektin kehittämien SGML-muotoisten dokumenttityyppimääritysten jälkeen yhtenäiseksi (Salminen, Lyytikäinen, Tiitinen ja Mustajärvi 2001). Yksikäsitteiset tunnisteet (esimerkiksi koko prosessin ajan pysyvä lakihanke- tai lakiasiakirjatunnus) puuttuvat kuitenkin valtioneuvoston ja eduskunnan väliltä. Se aiheuttaa ongelmia ja lisätyötä (Salminen, Tiitinen ja Lyytikäinen 1999). Tunnisteet ovat nimittäin tärkeitä, että prosessit ja niihin liittyvät asiakirjat voitaisiin paikantaa.

Saatsin (2003) mukaan valtioneuvostonkin sisällä on käytössä useita erilaisia identifiointitunnuksia sekä asialle että asiakirjoille. PTJ:ssä käytetään asian tunnistetta (ministeriö/pvm/juokseva numero). HARE antaa oman tunnistenumeronsa, josta näkyy asiaa valmistellut ministeriö, juoksevat numerot pää- ja alihankkeelle ja avausvuosi (esimerkiksi KTM035:00/2003). (Saatsi 2003.) Eduskunnassa asia tunnistetaan HE-numerolla (esimerkiksi HE 215/2000 vp), johon kaikki muut eduskunnassa tuotetut asiakirjat, kuten valiokuntien mietinnöt viittaavat.

Lainsäädäntöprosessin toimija joutuu tallentamaan samoja metatietoja useaan eri paikkaan, mikä aiheuttaa lisää vaivaa ja työtä. Esimerkiksi lain valmisteluvaiheessa alkanut lainsäädäntöhanke pitäisi tallentaa sekä hankerekisteriin (HARE) että PTJ-järjestelmään. Moneen kertaan saman tekstin kirjoittaminen aiheuttaa usein epäyhtenäisyyksiä asiakirjoissa (Salminen, Lyytikäinen ja Tiitinen 2000).

7.4 RDF-prosessiskeema lainsäädäntöprosessin tiedonhallinnan tukemiseksi

Tässä kappaleessa esitellään tutkielman tuloksena syntynyt RDF-prosessiskeema, sen tarkoitus sekä perusteita skeemassa olevien luokkien ja ominaisuuksien valinnoille. Koska prosessin vaiheiden kuvaaminen on skeeman keskeinen ominaisuus, vaiheiden kuvaaminen

skeemassa esitellään erikseen.

Prosessiskeeman tarkoitus ja esittely

Tärkeimpiä RDF-skeemojen luvussa 4.2. esitellyistä hyödyistä lainsäädäntöprosessin kannalta ovat tiedon esittäminen koneelle ymmärrettävässä muodossa, metatietojen hallinnan yhtenäistyminen eduskunnan ja ministeriöiden välillä, hakujen ja tiedon jakamisen tehostuminen sekä päättelysääntöjen ja älykkäiden palvelujen mahdollistuminen. Tässä tutkielmassa ehdotetaan lainsäädäntöprosessia tukevaa RDF-prosessiskeemaa, joka on tutkielman liitteessä 1.

Prosessiskeeman ehdotetaan olevan yhteinen eduskunnan ja ministeriöiden välillä. Sen tavoitteena on esittää sanastoa lakiasiakirjojen ja lakihankkeiden metatiedoista niin yleisellä tasolla, että sen pohjalta tehtäviä RDF-kuvailuja on mahdollista käyttää mahdollisimman laajasti lainsäädäntöprosessiin liittyvissä toiminnoissa ja järjestelmissä. Metatiedot, kuten asiakirjan tunniste ja nimike, lakihankkeessa toimiva esittelijä ja lakihankkeeseen tai lakiasiakirjaan liittyvä vaihe ovat keskeisimpiä metatietoja, jotka liittyvät kaikkiin lainsäädäntöprosessin vaiheisiin ja asiakirjoihin.

Mallinnettujen luokkien ja ominaisuuksien valinnassa on jo tässä vaiheessa hyödynnetty RASKE2-projektin haastattelujen materiaalia, vaikka haastattelut ja niiden analysointi on tutkielman kirjoitushetkellä vielä kesken. Tällaisenaan skeema tuskin on vielä valmis, mutta se antaa kuvan siitä, millainen koko lainsäädäntöprosessia tukeva skeema voisi olla. Skeemaa voidaan myöhemmin korjata ja kehittää RASKE2-projektissa toteutettavan metatietoanalyysin tiedoilla sekä asiantuntijoiden kommenteilla.

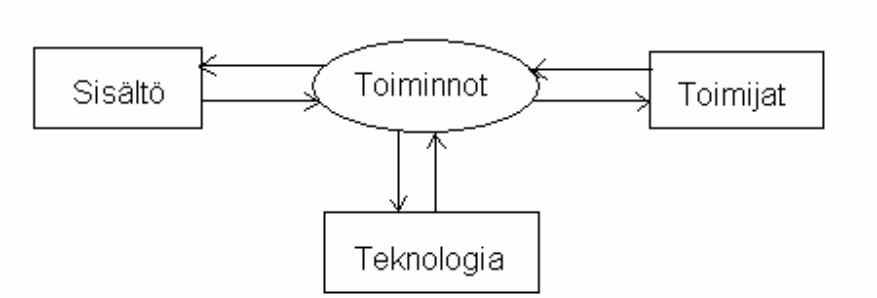
Taulussa 3 on lyhyesti esitetty skeemassa olevat luokat ja ominaisuudet. Koska joillakin luokilla on lukuisia aliluokkia, on niistä merkitty tauluun selkeyden vuoksi vain niiden lukumäärä.

TAULU 3. Lainsäädäntöprosessia tukevan RDF-prosessiskeeman luokat, aliluokat ja ominaisuudet.

LUOKKA	**ALILUOKKA**	**OMINAISUUS**
Vaihe	Vaihe1 (aliluokkia 7)	sisaltaaVaiheita
	Vaihe2 (aluluokkia 2)	
	Vaihe3 (aliluokkia 5)	
	Vaihe4 (aliluokkia 4)	
Lakihanke		esittelija
		hankeNimi
		hankeTunniste
		liittyyHankkeeseen
		onVaiheessa
		valmisteluMinisteriossa
		kasittelyValiokunnassa
Lakiasiakirja		kuuluuHankeeseen
		nimike
		tunniste
		pvm
		luotuVaiheessa
Henkilö		etunimi
		sukunimi
		osoite
		puhnro
Ministerio (aliluokkia 13)		
Valiokunta (aliluokkia 14)		

Perusteita prosessiskeemassa olevien luokkien ja ominaisuuksien valinnoille

Kuviossa 20 on esitetty RASKE-projektissa syntynyt malli sisällönhallintaympäristöstä (Salminen 2003b). Prosessiskeemaan on mallinnettu mallin mukaisesti *sisältöä* (Lakiasiakirjat, Lakihankkeet), *toimintoja* (Vaihe1-Vaihe4 ja niiden aliluokat) ja *toimijoita* (Henkilö, Ministeriö, Valiokunta, esittelijä, valmisteluMinisteriossa, kasittelyValiokunnassa). Sisältö ja toimijat ovat keskeisimmässä asemassa skeemassa. *Teknologia* taas on jätetty tarkoituksella kokonaan pois. Prosessiskeemaa voivat hyödyntää erilaiset sovellukset ja järjestelmät, mutta teknologiasta itsestään ei tarvitse tallentaa metatietoa, joten niitä ei ole mallinnettu skeemaankaan.



KUVIO 20. Sisällönhallintaympäristö RASKE-mallin mukaisesti (Salminen 2003b).

Velardi, Fabriani. & Missikoff (2001) jaottelevat RASKE-mallin tavoin ontologiaan tai skeemaan tulevat käsitteet toimijoihin (actor), toimintoihin (process, activity) ja entiteetteihin (object, entity). He jaottelevat semanttisia suhteita ilmaisevat ominaisuudet *vertikaalisiin* ja *horisontaalisiin* suhteisiin. Vertikaaliset suhteet kertovat käsitteiden hierarkiasta ja horisontaaliset suhteet käsitteiden samanlaisuudesta. Prosessiskeemaan ei ole mallinnettu horisontaalisia samanlaisuussuhteita, koska prosessiskeeman käsitteet ovat sen verran yleisiä, ettei siihen ole tarkoituksenmukaista määritellä tarkasti yksittäisten käsitteiden samanlaisuusasteita tai synonyymiutta. Vertikaalisia suhteita ovat prosessiskeemassa esimerkiksi luokan `Vaihe` ominaisuus `sisältääVaiheita`.

Dokumenttien ensisijainen piirre on ISO 15489-1 standardin (2001) mukaan *dynaamisuus*. Lainsäädäntöprosessissa tuotettavat asiakirjat ovat erityisen dynaamisia ja muuttuvia, koska prosessi on pitkä ja samalla asiakirjalla saattaa olla monia eri luojia ja versioita (esimerkiksi hallituksen esityksestä saatetaan jo valmisteluvaiheessa tehdä enemmän kuin kaksi eri versiota). Dokumentin dynaamisuuden takia on tärkeää, että *kontekstuaalista metatietoa* liitetään dokumenttiin kuvaamaan sen asiayhteys ja tausta (ISO 15489-1 2001, 11). Kontekstuaalista metatietoa on mallinnettu myös prosessiskeemaan. Alla olevassa listassa on lueteltu ISO-standardin mukaisia kontekstimetatietoja sekä näitä kuvaavat luokat ja ominaisuudet prosessiskeemassa.

- Mihin tehtävä tai toiminto liittyy. [Prosessiskeemassa: Vaihe, Lakiasiakirja]
- Mitä toiminnon aikana on päätetty. [Prosessiskeemassa: Vaihe (toiminto), Lakiasiakirja (sisältää päätökset)]

- Millainen on ollut päätöksentekoprosessi. [Prosessiskeemassa: Vaiheet ja niiden järjestys]
- Milloin organisaatio tai henkilö on suorittanut toiminnon. [Prosessiskeemassa: Lakiasiakirja-luokan ominaisuus pvm]

(ISO 15489-1 2001, 11.)

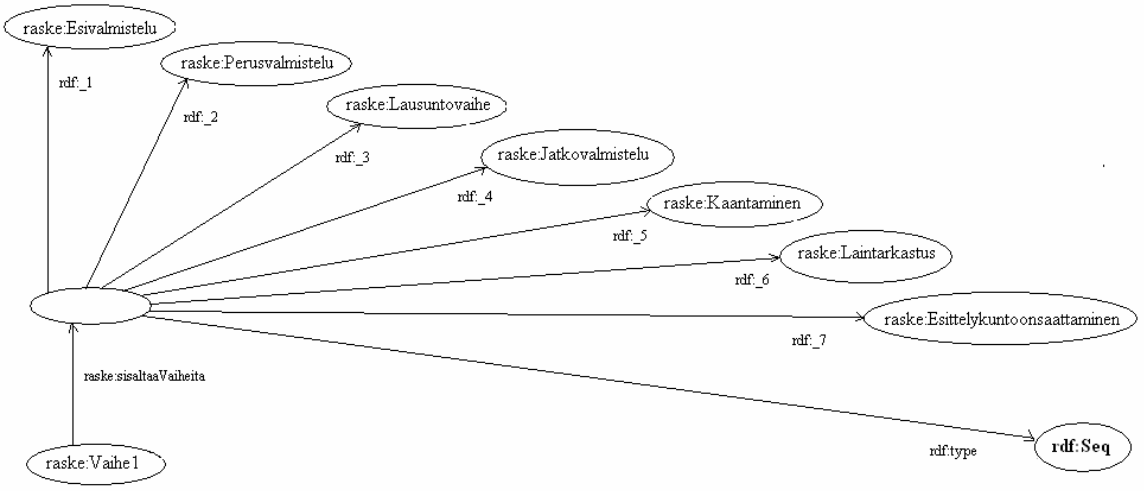
Edellä mainituista kontekstimetatiedoista toiminnon aikana tehdyt päätökset (2. kohta) eivät ole ekplisiittisinä metatietoina skeemassa, vaan ne sisältyvät lakiasiakirjaan. Prosesssiskeemassa mallinnetut metatiedot kuitenkin auttavat tässäkin tapauksessa yhdistämään tietyn lakiasiakirjan ja sen sisältävän päätöksen tiettyyn prosessin vaiheeseen.

Vaiheiden kuvaaminen skeemassa

Keskeisin ominaisuus prosessiskeemassa on vaiheiden kuvaaminen. Prosessiskeeman ensimmäisen vaiheen vaihekuvaus on esitetty RDF-graafina kuviossa 21 (RDF/XML-muoto liitteessä 1). Vaiheet ja niiden keskinäinen järjestys on kuvattu `rdf:Seq`-säiliöluokan avulla (säiliöluokat kuvattu kappaleessa 4.4.1.). Luokan `Vaihe` ominaisuutta `sisaltaaVaiheita` käytetään tässä apuominaisuutena, jolla saadaan kokoelma-luokan jäsenet kuvattua yhdeksi ryhmäksi.

Vaiheet olisi voitu kuvata myös esimerkiksi kokoelmaluokan avulla (kuvattu kappaleessa 4.4.2.). Tässä päädyttiin kuitenkin sarja-rakenteeseen (Sequence), koska se ei kokoelmaluokan tavoin sulje sitä mahdollisuutta, että vaiheita olisi olemassa muitakin kuin skeemassa mainitut. Sarja-rakenne sallii myös jäsenten toiston rakenteessa ja sen avulla voidaan eksplisiittisesti ilmaista vaiheiden tietty järjestys. (Manola & Miller 2004) Se, että rakenne ei ole suljettu ja toistot ovat mahdollisia, sopii lainsäädäntöprosessiin, koska prosessissa voidaan palata edellisiin vaiheisiin ja prosessin kulku ei muutenkaan aina ole ennalta arvattavissa. Esimerkiksi eduskunnan täysistunnon ensimmäisessä käsittelyssä asia voidaan lähettää uudestaan jonkin valiokunnan käsiteltäväksi. Prosessi-ilmentymissä saattaa myös syntyä välivaiheita, joita ei skeemassa ole mallinnettu. Joskus esimerkiksi täysistunto ei hyväksy valiokunnan mietintöä sellaisenaan ja silloin asia lähetetään suureen valiokuntaan.

Tätä vaihetta ei ole mallinnettu prosessiskeemaan vaiheen harvinaisuuden takia.



KUVIO 21. Lainsäädäntöprosessin ensimmäinen vaihe sarja-rakenteella (rdf:Seq) kuvattuna.

Sellaiset vaiheet, joista ei ole syntynyt asiakirjaa, eivät tule kuvatuksi RDF-kuvauksessa. Luokassa `Lakihanke` on nimittäin suora viittaus vain siihen vaiheeseen, missä lakihanke on menossa (ominaisuus `liittyyVaiheeseen`). Muut prosessin vaiheet tulevat ilmi lakihankkeeseen liittyvien asiakirjojen kautta. Yleensä kuitenkin voidaan sanoa, että jos prosessin vaihe on olemassa, siitä on myös syntynyt asiakirja.

Prosessiskeemaa voidaan käyttää asiakirjojen löytämiseen, asiakirjojen järjestyksen ja paikan osoittamiseen prosessissa, näkymien esittämiseen tiedosta sekä tiedon *profiloimiseen* erilaisille käyttäjille. Lakiasiakirjojen keskinäinen järjestys ja paikka prosessissa saadaan kuvattua liittämällä asiakirja oikeaan vaiheeseen ominaisuudella `luotuVaiheessa`. Tietylle käyttäjälle voidaan profiloiden esittää esimerkiksi tietyssä prosessin vaiheessa tuotetut asiakirjat tai antaa käyttäjän selata asiakirjoja prosessin mukaisessa järjestyksessä eteenpäin ja taaksepäin.

Tämäntyyppistä järjestykseen liittyvää tietoa on saatavilla eduskunnan asiakirja-arkistosta jo nyt. Asiakirjoja voidaan etsiä esimerkiksi käsittelyvaiheen perusteella. RDF-

skeemasanastolla tehdyt kuvaukset antaisivat kuitenkin lisäarvoa vaihetiedon esittämiselle. Vaihetieto voidaan näin esittää koneelle ymmärrettävässä yhtenäisessä ja formaalissa RDF-muodossa, mikä mahdollistaa vaihetiedon käyttämisen laajemmin ja joustavammin kuin pelkästään eduskunnan kyseisellä hakusivulla. RDF-kuvaukset voitaisiin jakaa monen järjestelmän kesken, eikä samaa hakumahdollisuutta tarvitsisi kehittää alusta alkaen jokaiseen järjestelmään. Jos samat RDF-kuvaukset pystyttäisiin tällä tavalla jakamaan, tärkeimmät metatiedot olisi mahdollista päivittää keskitetysti. RDF-kuvausten käyttö sovelluksissa tietysti edellyttää sovelluksien laajentamisen niin, että ne osaavat tulkita RDF-muotoista tietoa.

7.5 Prosessiskeeman hyödyntäminen: Semanttinen samoilu ja suosittelu

Kappaleissa 4.2.9. ja 4.2.10. käsiteltiin RDF-tiedon esittämistä käyttäjälle, päättelysääntöjä ja ontologian pohjalta rakennettavia älykkäitä palveluja. Lainsäädäntöprosessin tietojärjestelmissäkin olisi mahdollista RDF-skeeman pohjalta soveltaa esimerkiksi MuseoSuomi-portaalin (MuseoSuomi 2004; Hyvönen, Saarela, Viljanen ym. 2004) mallia näkymistä, semanttisesta samoilusta ja semanttisesta suosittelusta, joissa skeeman tietokategorioita (luokkia) esitetään käyttäjälle haun helpottamiseksi ja varsinaisen haun rinnalla esitetään muita aihealueeseen läheisesti liittyviä linkkejä.

Semanttinen samoilu on hyödyksi esitettäessä käyttöliittymässä metatietoa siitä, mitä tietoa järjestelmän kautta on mahdollista saada (Hyvönen, Saarela, Viljanen ym. 2004). Semanttista suosittelua voisi käyttää lainsäädäntöprosessin järjestelmissä esimerkiksi niin, että lakihankkeeseen liittyviä muita hankkeita esitettäisiin varsinaisen lakihankkeen tietojen rinnalla. Suosittelu voitaisiin tehdä esimerkiksi ministeriöiden lakihankkeita koskevilla web-sivuilla tai HARE-järjestelmässä. Tämän palvelun avulla voitaisiin ainakin osittain välttää sitä, että toisiinsa liittyvät lakihankkeet etenevät toisistaan tietämättä. Semanttisen suosittelun avulla tietoa muista lakihankkeista olisi mahdollista esittää esimerkiksi lainvalmistelijalle ilman, että sitä tarvitsisi erikseen etsiä. Hankkeiden liittymät voitaisiin päivittää keskitetysti samoihin RDF-kuvailuihin prosessiskeeman mukaisesti, joten linkit web-sivuilla päivittyisivät tätä kautta automaattisesti.

Semanttisen suosittelun ideaa lainsäädäntöprosessissa voisi jatkaa luokittelemalla rinnalla näytettävät linkit kategorioihin: esimerkiksi liitynnät muihin lakihankkeisiin ja varsinaisen lakihankkeen asiakirjat vaiheisiin. Tietokategorioiden hakuosumien näyttäminen jo ennalta helpottaisi käyttäjää ennakoimaan hakutuloksen suuruutta (Hyvönen, Saarela, Viljanen ym. 2004).

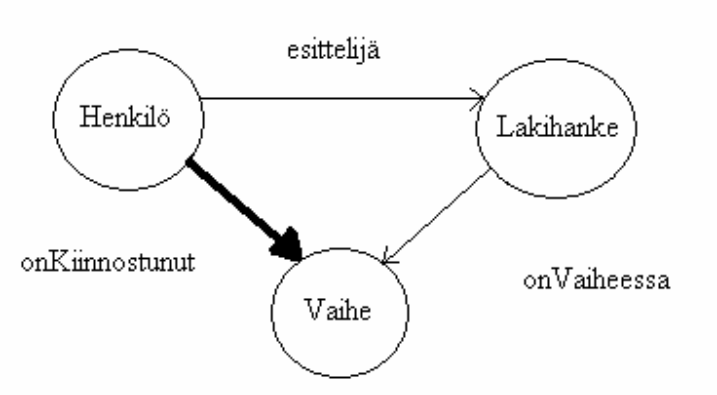
Semanttinen suosittelu perustuu RDF-kuvailutiedosta tehtävään päättelyyn ja päättelysääntöihin (Hyvönen, Saarela, Viljanen ym. 2004). Seuraavassa kappaleessa esitetään joitain esimerkkejä päättelystä ja päättelysäännöistä. Osa säännöistä on mahdollista toteuttaa prosessiskeeman pohjalta, osa on prosessiskeeman ulkopuolisia ideoita.

7.6 Päättelysäännöt

Jos halutaan esittää pelkkiä metatietoja, ne voidaan esittää myös ilman RDF-mallia. RDF-mallin todellinen hyöty tulee esille päättelysäännöissä ja logiikassa. Kuten Noy ja McGuinness (2001) toteavat, useinkaan skeeman tai ontologian kehittäminen ei ole päätavoite, vaan se tehdään jotain sovellusohjelmaa varten. RDF-skeemasta ja RDF-kuvailuista on hyötyä vasta sovelluksen tulkitessa ja prosessoidessa niitä. Suoraan RDF:n avulla ei päättelysääntöjä ole mahdollista esittää, mutta RDF-muotosen tiedon pohjalta on mahdollista rakentaa sovelluksia, jotka osaavat päättelysääntöjä muodostaa. Tässä luvussa esitetään muutamia esimerkkejä siitä, millaisia lainsäädäntöprosessiin liittyvät päättelysäännöt sekä luvussa 5 esitellyt Quanin ja Kargerin (2004) *virtuaaliset ominaisuudet* (Hyvönen, Saarela, Viljanen ym. 2004: *implisiittiset linkit*) voisivat olla.

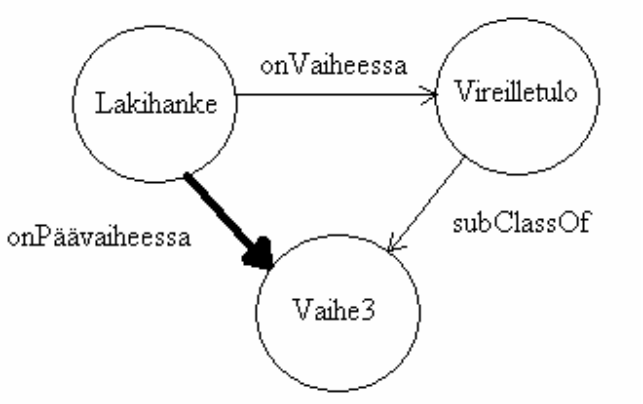
Päättelysäännöt esitetään RDF-mallin mukaisena resurssi/ominaisuus-kuviona. Kahdessa ensimmäisessä kuviossa (kuviot 22-23) käytetään prosessiskeeman käsitteitä, kuvioissa 24-26 käytetään käsitteitä, jotka on mahdollista lisätä prosessiskeemaan. Kuvion alla esitetään päättelyketjuesimerkki, eli millainen voisi olla RDF-skeeman ilmentymä asiasta. Päättelyketjuesimerkin jälkeen esitetään käyttöesimerkki kyseiselle päättelysäännölle. Virtuaaliset ominaisuudet, jotka on päätelty muiden ominaisuuksien avulla, on esitetty tummemmalla

nuolella kuin tavalliset ominaisuudet, joita Hyvönen, Saarela, Viljanen ym. (2004) nimittävät *eksplisiittisiksi linkeiksi*. Esimerkiksi kuvion 22 virtuaalinen ominaisuus on ”onKiinnostunut” ja eksplisiittiset linkit ”esittelijä” ja ”onVaiheessa”.



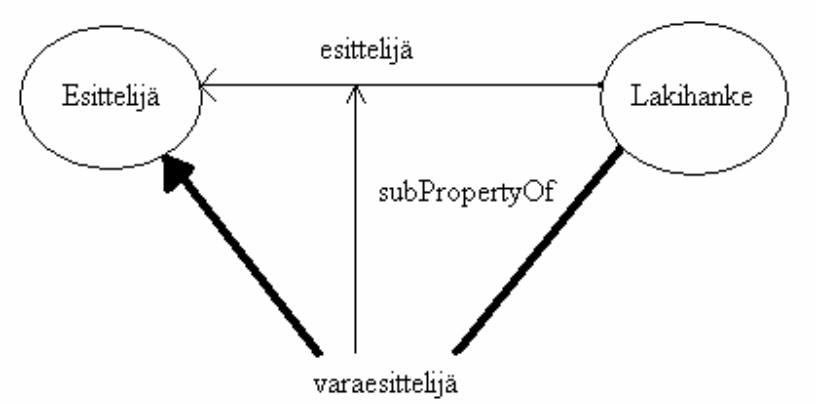
KUVIO 22. Virtuaalinen ominaisuus ”onKiinnostunut”.

Päättelyketjuesimerkki: Esittelijä Leena on esittelijänä lakihankkeessa maakaasumarkkinalain muuttamisesta. → Leena on mitä todennäköisimmin kiinnostunut siitä, että kyseisen hankkeen hallituksen esitystä käsitellään parhaillaan talousvaliokunnassa.
Käyttö: Tiedon profilointi lakihankkeen esittelijälle. Esittelijälle voidaan näyttää heidän vastuullaan olevien lakihankkeiden vaihe ja eteneminen prosessissa.



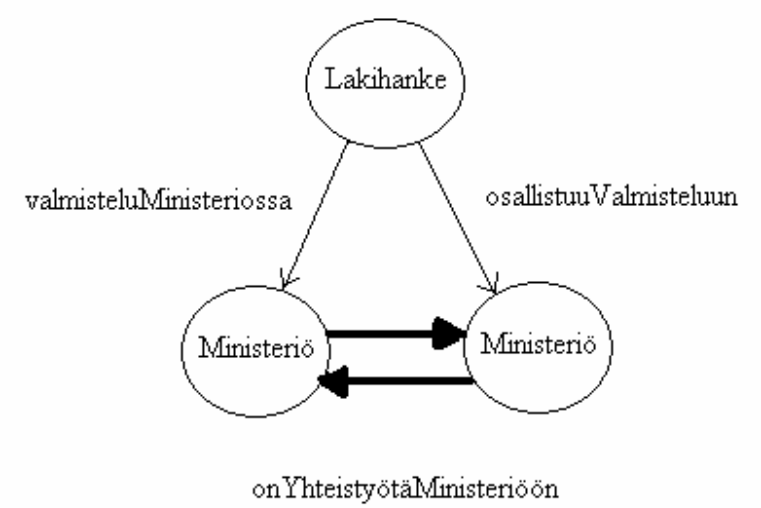
KUVIO 23. Virtuaalinen ominaisuus ”onPäävaiheessa”.

Päättelyketjuesimerkki: Lakihanke maa-aineslain muuttamisesta on vaiheessa vireilletulo. → Lakihanke on luokkahierarkian mukaan päävaiheessa eduskuntakäsittely.
Käyttö: Erilaiset näkymät tietoon. Voidaan esittää asiakirjoja alavaiheiden sekä päävaiheiden mukaan.



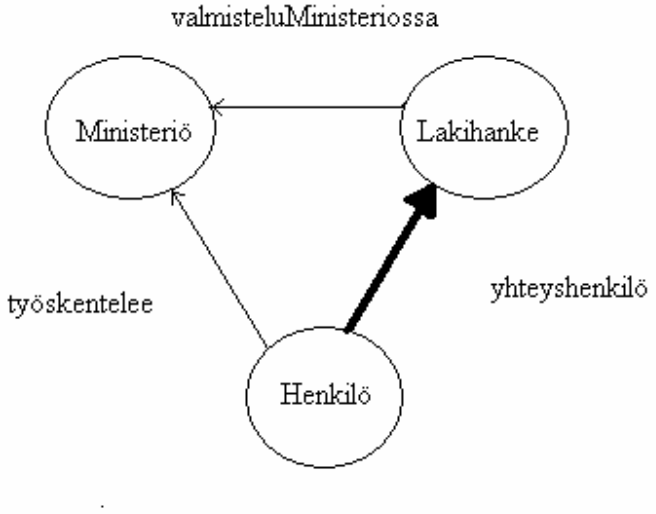
KUVIO 24. Virtuaalinen ominaisuus ”varaesittelijä”.

Päättelyketjuesimerkki: Henkilö Kalle on esittelijänä lakihankeessa aravalain muuttamisesta. Hän on pyytänyt osastoltaan henkilön Pekka varaesittelijäksi siltä varalta, että esittelijän pitää olla paikalla valtioneuvostokäsittelyn presidentille esittelyssä ja koska hän itse on ulkomaanmatkan vuoksi estynyt menemästä paikalle. → Varaesittelijä Pekka on myös esittelijä.
Käyttö: Varaesittelijän tiedot päivittyvät järjestelmässä kohtaan esittelijä, jos virallinen esittelijä ei pääse paikalle.



KUVIO 25. Virtuaalinen ominaisuus ”onYhteistyötäMinisteriöön”.

Päättelyketjuesimerkki: Maa- ja Metsätalousministeriö on vastuussa vesistöjen suojeluun liittyvän lakihankkeen valmistelusta. Ympäristöministeriö on osallistunut valmistelutyöhön antamalla asiantuntijalausunnon. → Ministeriöt tekevät yhteistyötä keskenään.
Käyttö: Tiedon profilointi lainvalmistelijalle. Esimerkiksi esittelijälle voidaan esittää kunkin hankkeen osalta yhteistyössä toimineet ministeriöt.



KUVIO 26. Virtuaalinen ominaisuus ”yhteyshenkilö”.

Päättelyketjuesimerkki: Henkilö Matti työskentelee kauppa- ja teollisuusministeriössä. Kauppa- ja teollisuusministeriö valmistelee lakihanketta korkeakoulujen keksintölaista. → Matti on eräs mahdollinen yhteyshenkilö kyseiseen lakihankkeeseen.
Käyttö: Semanttinen suosittelu: Yhteyshenkilön näyttäminen hankkeen yhteydessä lisätietojen kysymistä varten.

Viimeisessä esimerkissä (kuvio 26) päättelysääntö ei ole hyvä, jos ministeriö on laaja, kuten yleensä onkin. Jos esimerkiksi henkilöitä on yhteensä 300, kaikki työntekijät eivät millään voi tietää kaikesta ministeriön sisällä tapahtuvasta toiminnasta. Kyseinen päättelysääntö sopisi siis paremmin jollekin pienemmälle yhteisölle.

Jos sen sijaan ministeriö on pienempi kooltaan, voidaan päätellä, että kyseisen yhteisön kaikilla jäsenillä on tietoa yhteisön sisällä tapahtuvista toiminnoista.

Kuvioissa 23 ja 24 on päättelyssä hyödynnetty RDF:n subClassOf- ja subPropertyOf-määritteitä eli periytymistä. Periytymisen hyväksikäyttö on yksi keino päättelysääntöjen muodostamiseen. Periytymisen avulla saadaan esitetty eri *näkymiä* tietoon: laajempia näkymiä yliluokkien ja yliominaisuuksien mukaisesti ja yksityiskohtaisempia näkymiä aliluokkien ja aliominaisuuksien mukaisesti.

Päättelysääntöjä lainsäädäntöprosessin alueelta voi keksiä rajattomasti lisää. Tässä esiteltiin vain muutamia mahdollisia esimerkkejä ja niiden hyödyllisyys täytyy analysoida erikseen. Esimerkeistä on tarkoitus käydä ilmi päättelysääntöjen idea ja kannustaa sovellusalan asiantuntijoita keksimään päättelysääntöjä, joissa olisi todellista hyötyä. Konkreettista sovellusta rakennettaessa päättelysäännöt täytyy suunnitella aina huolellisesti erikseen käyttötarkoitusten mukaan.

7.7 Ehdotetun prosessiskeeman arviointi

Tutkielmassa esitettyyn RDF-prosessiskeemaan on mallinnettu prosessin toimintoja, toimijoita ja sisältöä. Skeemassa kuvattujen prosessivaiheiden avulla pystytään osoittamaan vaiheiden sekä asiakirjojen järjestys koneen ymmärtämässä muodossa. Prosessiskeemaa voidaan käyttää erilaisten näkymien esittämiseen tiedosta, semanttiseen samoiluun ja päättelysääntöjen avulla tapahtuvaan semanttiseen suositteluun.

Prosessiskeeman keskeisenä hyötynä on mahdollisuus kuvata lainsäädäntöprosessin metatietoja eksplisiittisesti ja formaalisti. Tällaisen tiedon avulla ihmisten lisäksi myös sovellukset voivat suorittaa ontologisia päätelmiä.

Mitä tarkemmin tieto on kuvattu, sitä tehokkaammin sovellukset voivat käsitellä tietoa automaattisesti. Tiedon hyvin yksityiskohtainen kuvaaminen ja yksityiskohtaisten ontologioiden rakentaminen on kuitenkin työlästä ja aikaa vievää (Wieling, Schreiber, Wielemaker & Sandberg 2001; Gruninger & Lee 2002). Skeeman tarkkuuden ja toteutuksen monimutkaisuuden välille pitää löytää jonkinlainen tasapaino tai kompromissi. Prosessiskeemassa tämä kompromissi on toteutettu mallintamalla skeemaan vain oleellisimmat lainsäädäntöprosessin metatiedot. Kompromissin takia täysi ontologinen päättely prosessiskeeman pohjalta ei ole mahdollista. Jotkut ominaisuudet, kuten lakihankkeiden liittymät (ominaisuus `liittyyHankeeseen`), pitää päivittää RDF-kuvauksissa käsin. Jotta lakihankkeiden liittymät voitaisiin automaattisesti päätellä, tarvittaisiin yksityiskohtaisempia lisäontologioita. Hyvin yksityiskohtaisia juridisia lakitekstien liittymäkohtia olisi lähes mahdotonta ylläpitää skeemassa, mutta lisäontologiat voisivat koostua esimerkiksi lakihankkeiden yleisen tason luokittelusta ministeriöittäin ja aihealueittain (esimerkiksi asuminen, ympäristönsuojelu, verotus jne.). Tällaisten luokittelujen avulla voitaisiin esimerkiksi näyttää asumiseen liittyvän hankkeen rinnalla muita asumiseen liittyviä hankkeita. Prosessiskeeman täydentäminen tällaisilla sovellusala- tai ministeriökohtaisilla ontologioilla onnistuisi joustavasti XML-nimiavaruuksien avulla.

Vaikka täysi ontologinen päättely ei olekaan prosessiskeeman avulla mahdollista, voidaan

jo sen tiedoilla tehdä yksinkertaista päättelyä. Esimerkkejä tällaisesta päättelystä esitettiin edellisessä luvussa (7.6.). Päättely tukee lainsäädäntöprosessin tiedonhallintaa semanttisen suosittelun, profiloinnin ja näkymien muodossa.

RDF käyttää identifioimisvälineenä URI-tunnisteita. Ainakin eduskunnan järjestelmien kautta on monet lainsäädäntöprosessin asiakirjoille saatavissa URL-osoite, joten näitä osoitteita olisi mahdollista suoraan käyttää URI-tunnisteina. Sen sijaan esimerkiksi henkilöiden URI-tunnisteista pitäisi yhteisesti erikseen sopia.

Prosessiskeema voidaan jakaa kaikkien sellaisten järjestelmien kesken, jotka osaavat tulkita RDF-muotoista tietoa. RDF-tuki on mahdollista toteuttaa järjestelmiin, sillä semanttisen webin standardin kaltaisena esitystapana RDF:lle löytyy omia jäsennin- ja muita työkaluja. Samoja RDF-muodossa olevia metatietoja voisivat siis hyödyntää monet eri järjestelmät.

RDF-muodon edut tulevat esille hierarkkisessa ja rikkaassa tiedon kuvailussa ja metatietojen esittämisessä yksitasoisiin metatietosanastoihin verrattuna (Hunter & Lagoze 2001). RDF-mallin avulla voidaan paremmin esittää asioiden välisiä suhteita RDF-luokkien ja -ominaisuuksien avulla. RDF-malli on yksinkertainen graafi, johon on helpompi tehdä kyselyjä kuin syviin XML-dokumentteihin (Berners-Lee 1998). Lainsäädäntöprosessin keskeisiä tietoresursseja ovat asiantuntijat (Lehtinen, Salminen & Huhtanen 2004), joista on mahdollista tallentaa RDF-muotoista tietoa web-resurssien ohella (Manola & Miller 2004).

XML-muotoisen tiedon kuvailu RDF:n avulla saattaa olla haastavaa, sillä XML:n ja RDF:n tietomallit ovat erilaiset (Klein 2002). Koska RDF on suhteellisen uusi tekniikka, kunnollisia ja monipuolisia työkaluja RDF-muotoisen tiedon luomiseen ja työstämiseen voi olla vaikea saada (Candan, Liu & Suvarna 2001).

Prosessiskeeman vaihemallinnuksen voidaan sanoa olevan hieman jäykkä lainsäädäntöprosessin luonnetta ajatellen. Skeemaan ei ole voitu mallintaa kaikkia mahdollisia prosessi-ilmentymien muotoja: esimerkiksi mahdollista suuren valiokunnan käsittelyä tai muita ideaalitilanteeseen verrattuna osin suorastaan todennäköisiä poikkeuksia prosessin kulussa.

Vaiheiden mallintaminen sarja-rakenteen (Sequence) avulla antaa kuitenkin joustavuutta ja sallii esimerkiksi prosessin mallinnuksessa palaamisen edelliseen vaiheeseen. Tällöin mallinnetaan sama vaihe kaksi kertaa ja vaiheiden järjestys osoitetaan sarja-rakenteen ominaisuuksilla `rdf:_1`, `rdf:_2`, `rdf:_3` jne. Muutenkaan sarja-rakennetta käytettäessä ei oleteta, että kaikki sarja-rakenteen kokoamat jäsenet olisivat ainoita jäseniä ryhmässä. Sarja-rakenteella voidaan siis osoittaa, että kaikki mahdollisen lainsäädäntöprosessin vaiheet eivät ole skeemassa sarja-rakenteen sisällä.

Muuten kritiikkiä prosessiskeemaa kohtaan voidaan osoittaa ainakin sen suhteen, että prosessiskeeman todellinen hyöty on mahdollista osoittaa vasta konkreettisessa sovelluksessa. Prosessiskeemaa ei ole muokattu asiantuntijoiden kanssa, joten siihen mallinnetut tiedot eivät välttämättä ole parhaita mahdollisia lainsäädäntöprosessin kannalta. Esimerkiksi tarkkoja alkamis- ja päättymispäivämääriä ei löydy yksittäisen prosessi-ilmentymän eri vaiheisiin.

8 YHTEENVETO

Nykyisen webin teknologioita täytyy kehittää tiedon haun ja tiedonhallinnan ongelmien ratkaisemiseksi. Jotta hakukoneiden ja muiden sovellusten avulla pystyttäisiin tietoa löytämään tehokkaammin ja täsmällisemmin, pitäisi sovellusten osata tulkita webissä olevaa tietoa. Semanttisessa webissä tieto esitetään niin formaalissa muodossa, että sovelluksetkin pystyvät sitä ”ymmärtämään” ja tekemään sen pohjalta automaattisia päätelmiä (Berners-Lee, Hendler & Lassila 2001). Formaali muoto saadaan aikaan esittämällä sovellusalan käsitteet ja suhteet ontologian avulla (Berners-Lee, Hendler & Lassila 2001; Gruninger & Lee 2002). Tämän tutkielman tutkimuskohteena on RDF (Resource Description Framework), jolla voidaan esittää kevyitä ontologioita RDF Schema -kielellä sekä metakuvauksia RDF-skeemojen mukaisesti (Manola & Miller 2004; Volz, Oberle & Studer 2003).

Tutkielman tavoitteena oli selvittää, kuinka RDF-tietomallia voidaan hyödyntää toimintaprosessien tiedonhallinnassa. Tavoitteeseen sisältyy RDF-mallin käytettävyyden arvioiminen ja RDF-skeemojen suunnittelun kuvaaminen. Tutkielma on käsitteellis-teoreettinen, joten tutkimusmenetelmänä käytettiin tieteellisen ja muun lähdeaineiston analysointia aihepiiriin liittyen. Tutkielmaan sisältyy konstruktiivinen osuus RDF-skeeman muodossa.

Tutkielman tuloksena saatiin laaja ja syvällinen käsitys RDF-tekniikasta, sen ominaisuuksista, mahdollisuuksista ja ongelmista. RDF-skeemojen suunnittelu esitettiin tutkielmassa seitsemän vaiheen mukaisesti. Skeemasuunnitteluun annettiin yleisiä ohjeita ja esiteltiin esimerkkityökaluja skeemojen suunnittelun tueksi. Toimintaprosesseista käytettiin esimerkkinä suomalaista lainsäädäntöprosessia. RDF:n hyötyjä ja mahdollisuuksia lainsäädäntöprosessissa kartoitettiin viimeisessä luvussa tutkielmassa syntyneen prosessiskeeman kautta.

Suuressa osassa aiemmista tutkimuksista RDF Schema –luokkia on käytetty kuvaamaan perinteisiä hierarkioita asioiden välillä, kuten museoesineitä tai muita konkreettisia ja pysyviä asioita (Quan & Karger 2004; Hyvönen, Saarela, Viljanen ym. 2004). Toimintaprosesseissa dokumentteihin, toimintoihin ja toimijoihin liittyvät tilanteet ovat usein dynaami-

sempia. Esimerkiksi lainsäädäntöprosessissa samaa lakitekstiä voivat muokata ja kommentoida monet eri toimijat ja asiakirjoista saattaa olla monia eri versioita. Tutkielmassa hyödynnettiin aiemman tutkimuksen ideoita ontologisesta päättelystä, semanttisesta suosittelusta ja semanttisesta samoilusta, mutta RDF-skeema rakennettiin kuvaamaan nimenomaan prosesseja. RDF-skeemaan ei kannata tai ole mahdollistakaan mallintaa kovin yksityiskohtaisesti esimerkiksi eri lakien juridisia liittymiä. Prosessiskeema soveltuu kuitenkin yleisyydestään huolimatta suomalaisen lainsäädäntöprosessin tiedonhallinnan tukemiseen ja yksinkertaisten päättelyjen tekemiseen. Skeemaa on mahdollista käyttää semanttisen suosittelun ja samoilun sekä profiloinnin toteuttamiseen ja erilaisten tietonäkymien esittämiseen käyttöliittymissä.

Tutkielmassa on keskitytty vahvasti suomalaisen lainsäädäntöprosessin tiedonhallinnan tarpeisiin. Muunlaiset toimintaprosessit on jätetty analysoimatta. Muiden toimintaprosessien, esimerkiksi alihankintaketjun, tiedonhallinnassa saattaa kuitenkin olla monenlaisia erityispiirteitä, joihin ei tässä esitettyjä prosessiskeemaa sekä ideoita voida suoraan soveltaa. Vaikka tutkielmaan sisältyy konstruktiivinen osuus, tutkielma on vahvasti käsitteellis-teoreettinen. Tämän tyyppisessä käytännönläheisessä aiheessa käsitteellis-teoreettisuus saattaa johtaa osittain harhaan, koska ideoita ei ole käytännössä konkreettisesti kokeiltu. Prosessiskeeman hyödyt olisi mahdollista todeta varmasti vasta sellaisella sovelluksella, joka osaisi käyttää hyväkseen RDF-muotoista tietoa.

Jatkotutkimuksessa olisi erityisen hyödyllistä demonstroida RDF:n ja prosessiskeeman hyötyjä käytännössä konkreettisten sovellusten avulla. Tällaisten sovellusten pitäisi pystyä prosessoimaan ja hallitsemaan RDF-muotoista tietoa, joten jatkotutkimusaiheiksi sopivat tähän liittyen RDF-kyselykielet, RDF:n hallinta tietokannassa ja toteutetut kokonaisarkkitehtuurit RDF-tiedon hallitsemiseen. Eräs tässä tutkielmassa ulkopuolelle jätetty aihe on metakuvausten tuottaminen RDF-skeemojen mukaisesti. Mitä automaattisemmin metakuvausten luonti pystytään toteuttamaan, sitä käytettävämpänä RDF-mallia voidaan pitää. Metakuvausten generoinnin mahdollisuus esimerkiksi suoraan tietokannassa olevien tietojen tai eri formaatissa (esimerkiksi XML, pdf, HTML) olevien dokumenttien pohjalta kannattaisi tässä yhteydessä kartoittaa.

LÄHTEET

Ahmed K., Ayers D., Birbeck M., Cousins J., Dodds D., Lubell J., Nic M., Rivers-Moore D., Watt A., Worden R. & Wrightson A. 2001. Professional XML Meta Data. UK: Wrox Press Ltd.

Beckett D. (toim.) 2004. RDF/XML Syntax Specification (Revised). W3C Recommendation [online], [viitattu 18.3.2004]. Saatavilla www-muodossa <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>.

Berners-Lee T. 1998. Why RDF model is different from the XML model [online], [viitattu 19.7.2004]. Saatavilla www-muodossa <http://www.w3.org/DesignIssues/RDF-XML.html>.

Berners-Lee T., Fielding R. & Masinter L. 1998. RFC 2396 – Uniform Resource Identifiers (URI): Generic Syntax [online], [viitattu 4.8.2004]. Saatavilla www-muodossa <http://www.isi.edu/in-notes/rfc2396.txt>.

Berners-Lee T. 2000. Semantic Web on XML. Kalvot XML2000-tapahtumasta (Washington DC) [online], [viitattu 5.8.2004]. Saatavilla www-muodossa <http://www.w3.org/2000/Talks/1206-xml2k-tbl/>.

Berners-Lee T., Hendler J. & Lassila O. 2001. The Semantic Web. Scientific American 284(5), 34-43.

Boulos K., Roudsari A. & Carson E. 2002. Towards a Semantic medical Web: HealthCyberMap's tool for building an RDF metadata base of health information resources based on the Qualified Dublin Core Metadata Set. Medical Science Monitor 8(7).

Brickley D. & Guha R.V. (toim.) 2004. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation [online], [viitattu 2.4.2004]. Saatavilla www-muodossa

<http://www.w3.org/TR/rdf-schema/>.

Broekstra J., Klein M., Decker S., Fensel D., Harmelen F. & Horrocks I. 2001. Enabling knowledge representation on the Web by extending RDF schema. Proceedings of the Tenth International Conference on World Wide Web, Hong Kong. Computer Networks 39, 609-634.

Candan K., Liu H. & Suvarna R. 2001. Resource Description Framework: Metadata and its applications. ACM SIGKDD Explorations Newsletter 3(1), 6-19.

Connolly D., Harmelen F., Horrocks I., McGuinness D., Patel-Schneider P. & Stein L. 2001. DAML+OIL Reference Description. W3C Note 18.12.2001 [online], [viitattu 23.08.2004]. Saatavilla www-muodossa <http://www.w3.org/TR/daml+oil-reference>.

Decker S., Mitra P. & Melnik S. 2000. Framework for the Semantic Web: An RDF Tutorial. IEEE, Internet computing, 68-73.

Decker S., Melnik S., Harmelen F., Fensel D., Klein M., Broekstra J., Erdmann M. & Horrocks I. 2000. The Semantic Web: The roles of XML and RDF. IEEE Internet Computing 4(5), 63-74.

DCMI. 2004. Dublin Core Metadata Initiative, Dublin Core -metatietoformaatin sivusto [online], [viitattu 18.6.2004]. Saatavilla www-muodossa <http://dublincore.org/>.

Ebenhoch P. 2001. Legal knowledge representation using the Resource Description Framework (RDF). 12th International Workshop on Databases and Expert Systems Applications, Munich, Germany, 369-373.

Eduskunnan tiedon ja tietämyksen hallinta, loppuraportti. Eduskunnan kanslian julkaisu 6/2001.

Eduskunnan tietohallinnon linjat 2002-2004. Eduskunnan kanslian julkaisu 3/2002.

Ensel C. & Keller A. 2002. An approach for managing service dependencies with XML and the Resource Description Framework. Journal of Network and Systems Management 10(2), 147-170.

Fernandes, Moura & Porto. 2003. An ontology-based approach for organizing, sharing and querying knowledge objects on the Web. Proceedings of the 14th International Workshop on Database and Expert System Applications (DEXA'03), Prague, Czech Republic, 604-609.

FINLEX-säädöstietopankki [online], Oikeusministeriö [viitattu 06.07.2004]. Saatavilla www-muodossa <http://www.finlex.fi/>.

Gilliland-Swetland A. 2000. Setting the stage [online], [viitattu 09.07.2004]. Saatavilla www-muodossa <http://www.getty.edu/research/conducting_research/standards/intrometadata/pdf/swetland.pdf>.

Grant J. & Beckett D. (toim.) 2004. RDF Test Cases. W3C Recommendation [online], [viitattu 24.4.2004]. Saatavilla www-muodossa <http://www.w3.org/TR/rdf-testcases/>.

Gruber T. 1993. A Translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199-220.

Gruninger M. & Fox M.S. 1995. Methodology for the design and evaluation of ontologies. Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada.

Gruninger M. & Lee J. 2002. Ontology applications and design. Communications of the ACM 45(2), 39-41.

Guha R., McCool R. & Miller E. 2003. Semantic search. Proceedings of the Twelfth International Conference on World Wide Web, Budapest, Hungary, 700-709.

Heinisuo R. & Ekholm K. 1997. Elektronisen viittaamisen opas. Jyväskylä: Jyväskylän yliopistopaino.

Helsingin yliopiston kirjasto. 1999. YSA – Yleinen suomalainen asiasanasto [online], [viitattu 15.07.2004]. Saatavilla www-muodossa <http://vesa.lib.helsinki.fi/ysa/>.

Hayes P. (toim.) 2004. RDF Semantics. W3C Recommendation [online], [viitattu 24.4.2004]. Saatavilla www-muodossa <http://www.w3.org/TR/rdf-mt/>.

Hjelm J. 2001. Creating the Semantic Web with RDF. Canada: John Wiley & Sons, Inc.

Horrocks I. & Patel-Schneider P. 2003. Three thesis of representation in the Semantic Web. Proceedings of the Twelfth International World Wide Web Conference, Budapest, Hungary, 39-47.

Hovy E. 2003. Using an ontology to simplify data access. Communications of the ACM, 46(1), 47-49.

Huhtanen K. 2003. Ontologioiden kehittäminen lainsäädäntötyötä varten. Jyväskylän yliopisto, Tietojärjestelmätieteen pro gradu –tutkielma.

Hunter J. & Lagoze C. 2001. Combining RDF and XML schemas to enhance interoperability between metadata application profiles. Proceedings of the Tenth International Conference on World Wide Web, Hong Kong, 457-466.

Hyvönen E., Saarela S., Viljanen K., Mäkelä E., Valo A., Salminen M., Kettula S. & Junnila M. 2004. A portal for publishing museum collections on the Web. Proceedings of ECAI/PAIS 2004, Valencia, Spain (tulossa).

Hyvönen E., Salminen M., Junnila M. & Kettula S. 2004. A content creation process for the Semantic Web. Proceedings of OntoLex 2004, Lisbon, Portugal (tulossa).

ISO 15489-1. 2001. International Standard. Information and documentation – Records management. Part 1: General.

ISO/PDTS 23081. 2003. International Standard / Proposed Draft Technical Specification. Information and documentation – Records Management Processes – Metadata Records. Part1: Principles.

JUHTA. 2004. Julkisen hallinnon tietohallinnon neuvottelukunnan sivusto: JHS 143. Asiakirjojen kuvailun ja hallinnan metatiedot. Saatavilla www-muodossa <http://www.intermin.fi/juhta/suomi/jhs143>

Karvounarakis G. 1999. RDF Query Languages: A state-of-the-art [online], [viitattu 03.09.2004]. Saatavilla www-muodossa <http://139.91.183.30:9090/RDF/publications/state.html >.

Klein M. 2002. Interpreting documents via an RDF Schema ontology. Proceedings of the 13th International Workshop on Database and Expert Systems Applications, Aix-en-Provece, France, 1-5.

Klyne G. & Carroll J. (toim.) 2004. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation [online], [viitattu 24.4.2004]. Saatavilla www-muodossa <http://www.w3.org/TR/rdf-concepts/>.

Koivunen M. & Miller E. 2001. W3C Semantic Web Activity [online], [viitattu 29.07.2004]. Proceedings of the Semantic Web Kick-off Seminar in Finland. Saatavilla www-muodossa <http://www.w3.org/2001/12/semweb-fin/w3csw>.

Lehtinen A., Salminen A. & Huhtanen K. 2004. Tiedonhallinta suomalaisessa lainsäädäntöprosessissa. RASKE2-projektin väliraportti.

Magkanaraki A., Alexaki S., Christophides V. & Plexousakis D. 2002. Benchmarking RDF schemas for the Semantic Web. Lecture Notes in Computer Science 2342. Springer-Verlag, 132-146.

Manola F. & Miller E. (toim.) 2004. RDF Primer. W3C Recommendation [online], [viitattu 18.3.2004]. Saatavilla www-muodossa <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.

McGuinness D. & Harmelen F. (toim.) 2004. OWL Web ontology language. Overview. W3C Recommendation [online], [viitattu 23.08.2004]. Saatavilla www-muodossa <http://www.w3.org/TR/owl-features/>.

Middleton S., Alani H., Shadbolt N. & De Roure D. 2002. Exploiting synergy between ontologies and recommender systems. In Proceedings of Semantic Web Workshop 2002 at the eleventh International World Wide Web Conference Hawaii, USA.

Moisio R. & Leppänen T. 2003. SÄHKE-määrittely 1. Abstrakti mallintaminen. Kansallisarkiston SÄHKE-hankeen mallintamisen osahanke.

MuseoSuomi. 2004. MuseoSuomi-portaali. Suomen museot semanttisessa webissä. [Viitattu 20.08.2004]. <http://museosuomi.cs.helsinki.fi/>.

Niemijärvi V. 2002. Metatieto tietovarastoympäristössä. Jyväskylän yliopisto, Tietojärjestelmätieteen pro gradu –tutkielma.

Noy N. & McGuinness D. 2001. Ontology development 101: A guide to creating your first ontology, [online], [viitattu 14.06.2004]. KSL (Knowledge Systems Laboratory) Report of Stanford University. Saatavilla www-muodossa

<http://protege.stanford.edu/publications/ontology_development/ontology101.pd>

Noy N. & Rector A. 2004. Defining N-ary relations on the Semantic Web: Use With Individuals. W3C Working Draft [online], [viitattu 03.08.2004]. Saatavilla www-muodossa <http://www.w3.org/TR/2004/WD-swbp-n-aryRelations-20040721/ >.

Noy N., Sintek M., Decker S., Crubezy M., Fergerson R. & Musen M. 2001. Creating Semantic Web contents with Protege-2000. IEEE Intelligent Systems 16(2), 60-71.

Protégé-2000. 2004. Protégé-projektin web-sivut. [Viitattu 25.8.2004]. <http://protege.stanford.edu/>.

Protégé-2000 User Guide. 2004. [Viitattu 17.06.2004]. Saatavilla www-muodossa <http://protege.stanford.edu/doc/users_guide/index.html >.

Quan D., Huynh D. & Karger R. 2003 Haystack: A platform for authoring end user Semantic Web applications. Proceedings of the 2nd International Semantic Web Conference 2003, Budapest, Hungary, 1-16.

Quan D. & Karger R. 2004. How to make a Semantic Web browser. Proceedings of the 13th Conference on World Wide Web, New York, USA, 255-265.

RASKE2 2004. RASKE2-projektin kotisivut [online], [viitattu 05.06.2004]. Saatavilla www-muodossa < http://www.it.jyu.fi/raske/>.

Ribiére M. & Charlton P. 2000. Ontology overview [online]. Motorola Labs, Networking and Application Lab, Paris. [Viitattu 03.09.2004]. Saatavilla www-muodossa <http://www.fipa.org/docs/input/f-in-00045/f-in-00045.pdf >.

Saatsi P. 2003. Valtioneuvoston yhteisten tietojärjestelmien metatietojen kartoitus. Luonnos, lokakuu 2003. RASKE2-projektin saamaa aineistoa.

Salminen A., Tiitinen P. & Lyytikäinen V. 1999. Usability evaluation structured document archive. Proceedings of the Thirty-Second Hawaii International Conference on System Sciences. Los Alamitos, CA, (file ddhfu06.pdf at CD-ROM).

Salminen A., Lyytikäinen V. & Tiitinen P. 2000. Putting documents into their work context in document analysis. Information Processing & Management 36 (4), 623-641.

Salminen A., Lyytikäinen V., Tiitinen P. & Mustajärvi O. 2001. Experiences of SGML standardization: The case of the Finnish legislative documents. Proceedings of the Thirty-Fourth Hawaii International Conference on System Sciences. Los Alamitos, CA, 72-81.

Salminen A. 2003a. Towards digital government by XML standardization: Methods and Experiences. In Proceedings of the XML Finland 2003, Kuopio, Finland, 5-15.

Salminen A. 2003b. Document analysis methods. In C.L. Bernie (Ed.), Encyclopedia of Library and Information Science, Second Edition, Revised and Expanded. New York: Marcel Dekker, 916-927.

Shen J. & Yang Y. 2004. Extending RDF in distributed knowledge-intensive applications. Future Generation Computer Science Systems 20(1), 27-46.

Staab S. Erdmann M. Maedche A. & Decker S. 2002. An extensible approach for modeling ontologies in RDF(S). Knowledge Media in Healthcare: Opportunities and Challenges, 234-254.

Stoffel K., Taylor M. & Hendler J. 1997. Efficient management of very large ontologies. Proceedings of Fourteenth American Association for Artificial Intelligence Conference (AAAI-97), Providence, Rhode Island, AAAI/MIT Press.

Stuckenschmidt H. & Harmelen F. 2001. Ontology-based metadata generation from semi-

structured information. Proceedings of the International Conference on Knowledge Capture, Victoria, British Colimbia, Canada, 163-170.

Swick R. 2000. W3C Metadata Activity Statement [online], [viitattu 29.7.2004]. Saatavilla www-muodossa <http://www.w3.org/Metadata/Activity>.

Suomen eduskunta. 2004. Suomen eduskunnan internet-palvelut. [Viitattu 10.08.2004]. < http://www.eduskunta.fi/>.

Valtioneuvoston tietohallintostrategia 2003-2007. Valtiovarainministeriö, valtioneuvoston tietohallintoyksikön julkaisuja 5/2003.

Valtioneuvosto. 2004. Valtioneuvoston verkkopalvelu. [Viitattu 10.08.2004]. <http://www.valtioneuvosto.fi>.

Velardi P., Fabriani P. & Missikoff M. 2001. Using text processing techniques to automatically enrich a domain ontology. Proceedings of the International Conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA, 270-284.

Volz R., Oberle D. & Studer R. 2003. Views for light-weight Web ontologies. Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), Melbourne, FL, USA, 1168-1173.

W3C. 2004. World Wide Web Consortiumin web-sivut, [viitattu 31.8.2004]. Saatavilla www-muodossa <http://www.w3.org/>.

W3C RDF Validation Service. 2004. W3C:n ylläpitämä RDF-jäsennin [online], [viitattu 05.06.2004]. <http://www.w3.org/RDF/Validator/>.

Wieling B., Schreiber A., Wielemaker J. & Sandberg J. 2001. From thesaurus to ontology. Proceedings of the International Conference on Knowledge Capture, Victoria, Canada, 194-201.

LIITE 1: Prosessiskeemaehdotus lainsäädäntöprosessille

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY raske 'http://www.it.jyu.fi/raske/rdf#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
        xmlns:raske="&raske;"
        xmlns:rdfs="&rdfs;">


<rdfs:Class rdf:about="&raske;Esivalmistelu">
<rdfs:subClassOf rdf:resource="&raske;Vaihe1"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Perusvalmistelu">
<rdfs:subClassOf rdf:resource="&raske;Vaihe1"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Lausuntovaihe">
<rdfs:subClassOf rdf:resource="&raske;Vaihe1"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Jatkovalmistelu">
<rdfs:subClassOf rdf:resource="&raske;Vaihe1"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Kaantaminen">
<rdfs:subClassOf rdf:resource="&raske;Vaihe1"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Laintarkastus">
<rdfs:subClassOf rdf:resource="&raske;Vaihe1"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;EsittelykuntoonSaattaminen">
<rdfs:subClassOf rdf:resource="&raske;Vaihe1"/>
</rdfs:Class>
rdfs:Class rdf:about="&raske;Yleisistuntokasittely">
<rdfs:subClassOf rdf:resource="&raske;Vaihe2"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;PresidentinEsittely">
<rdfs:subClassOf rdf:resource="&raske;Vaihe2"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Vireilletulo">
<rdfs:subClassOf rdf:resource="&raske;Vaihe3"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Lahetekeskustelu">
<rdfs:subClassOf rdf:resource="&raske;Vaihe3"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Valiokuntakasittely">
<rdfs:subClassOf rdf:resource="&raske;Vaihe3"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;EnsimmainenKasittely">
<rdfs:subClassOf rdf:resource="&raske;Vaihe3"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;ToinenKasittely">
<rdfs:subClassOf rdf:resource="&raske;Vaihe3"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;LainVahvistaminen">
<rdfs:subClassOf rdf:resource="&raske;Vaihe4"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;LainJulkaiseminen">
<rdfs:subClassOf rdf:resource="&raske;Vaihe4"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;LainVoimaantulo">
<rdfs:subClassOf rdf:resource="&raske;Vaihe4"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;LainTaytantoonpanoJaSeuranta">
<rdfs:subClassOf rdf:resource="&raske;Vaihe4"/>
</rdfs:Class>


<rdf:Description rdf:about="&raske;Vaihe">
       <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
       <rdfs:comment>Ylaluokka kaikille vaiheille.<rdfs:comment/>
       <rdfs:label>Vaihe<rdfs:label/>
       <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
      <raske:sisaltaaVaiheita>

       <rdf:Description rdf:about="&raske;Vaihe1">
```

```
                <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
                rdfs:comment="Hallituksen esityksen valmistelu ministeriossa."
                rdfs:label="Vaihe1">
                <rdfs:subClassOf rdf:resource="&raske;Vaihe"/>
              <raske:sisaltaavaiheita>
                  <rdf:Seq>
                       <rdf:li rdf:resource="&raske;Esivalmistelu"/>
                       <rdf:li rdf:resource="&raske;Perusvalmistelu"/>
                       <rdf:li rdf:resource="&raske;Lausuntovaihe"/>
                       <rdf:li rdf:resource="&raske;Jatkovalmistelu"/>
                       <rdf:li rdf:resource="&raske;Kaantaminen"/>
                       <rdf:li rdf:resource="&raske;Laintarkastus"/>
                       <rdf:li rdf:resource="&raske;EsittelykuntoonSaattaminen"/>
                  </rdf:Seq>
              </raske:sisaltaavaiheita>
        </rdf:Description>

        <rdf:Description rdf:about="&raske;Vaihe2">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
                rdfs:comment="Hallituksen esityksen valtioneuvostokasittely."
                rdfs:label="Vaihe2">
                <rdfs:subClassOf rdf:resource="&raske;Vaihe"/>
              <raske:sisaltaavaiheita>
                  <rdf:Seq>
                       <rdf:li rdf:resource="&raske;Yleisistuntokasittely"/>
                       <rdf:li rdf:resource="&raske;PresidentinEsittely"/>
                  </rdf:Seq>
              </raske:sisaltaavaiheita>
        </rdf:Description>

        <rdf:Description rdf:about="&raske;Vaihe3">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
                rdfs:comment="Lakiesityksen eduskuntakasittely."
                rdfs:label="Vaihe3">
                <rdfs:subClassOf rdf:resource="&raske;Vaihe"/>
              <raske:sisaltaavaiheita>
                  <rdf:Seq>
                       <rdf:li rdf:resource="&raske;Vireilletulo"/>
                       <rdf:li rdf:resource="&raske;Lahetekeskustelu"/>
                       <rdf:li rdf:resource="&raske;Valiokuntakasittely"/>
                       <rdf:li rdf:resource="&raske;EnsimmainenKasittely"/>
                       <rdf:li rdf:resource="&raske;ToinenKasittely"/>
                  </rdf:Seq>
              </raske:sisaltaavaiheita>
        </rdf:Description>

        <rdf:Description rdf:about="&raske;Vaihe4">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
                rdfs:comment="Hyvaksytyn lain kasittely valtioneuvostossa."
                rdfs:label="Vaihe4">
                <rdfs:subClassOf rdf:resource="&raske;Vaihe"/>
              <raske:sisaltaavaiheita>
                  <rdf:Seq>
                       <rdf:li rdf:resource="&raske;LainVahvistaminen"/>
                       <rdf:li rdf:resource="&raske;LainJulkaiseminen"/>
                       <rdf:li rdf:resource="&raske;LainVoimaantulo"/>
                       <rdf:li rdf:resource="&raske;LainTaytantoonpanoJaSeuranta"/>
                  </rdf:Seq>
              </raske:sisaltaavaiheita>
        </rdf:Description>

</raske:sisaltaaVaiheita>
</rdf:Description>


<rdfs:Class rdf:about="&raske;Hallintovaliokunta_HaV"
         rdfs:label="Hallintovaliokunta_HaV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Henkilo"
         rdfs:label="Henkilo">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;KauppaJaTeollisuusministerio"
         rdfs:label="KauppaJaTeollisuusministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Lakiasiakirja"
         rdfs:label="Lakiasiakirja">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:about="&raske;Lakihanke"
         rdfs:label="Lakihanke">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Lakivaliokunta_LaV"
         rdfs:label="Lakivaliokunta_LaV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;LiikenneJaViestintaministerio"
         rdfs:label="LiikenneJaViestintaministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;LiikenneJaViestintavaliokunta_LiV"
         rdfs:label="LiikenneJaViestintavaliokunta_LiV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;MaaJaMetsatalousministerio"
         rdfs:label="MaaJaMetsatalousministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;MaaJaMetsatalousvaliokunta_MmV"
         rdfs:label="MaaJaMetsatalousvaliokunta_MmV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Ministerio"
         rdfs:label="Ministerio">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Oikeusministerio"
         rdfs:label="Oikeusministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Opetusministerio"
         rdfs:label="Opetusministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Perustuslakivaliokunta_PeV"
         rdfs:label="Perustuslakivaliokunta_PeV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Puolustusministerio"
         rdfs:label="Puolustusministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Puolustusvaliokunta_PuV"
         rdfs:label="Puolustusvaloikunta_PuV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Sisaasiainministerio"
         rdfs:label="Sisaasiainministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Sivistysvaliokunta_Siv"
         rdfs:label="Sivistysvaliokunta_Siv">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;SosiaaliJaTerveysministerio"
         rdfs:label="SosiaaliJaTerveysministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;SuuriValiokunta_SuV"
         rdfs:label="SuuriValiokunta_SuV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Talousvaliokunta_TaV"
         rdfs:label="Talousvaliokunta_TaV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Tulevaisuusvaliokunta_TuV"
         rdfs:label="Tulevaisuusvaliokunta_TuV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;TyoelamaJaTasaarvovaliokunta_TyV"
         rdfs:label="TyoelamaJaTasaarvovaliokunta_TyV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Tyoministerio"
         rdfs:label="Tyoministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Ulkoasiainministerio"
```

```
         rdfs:label="Ulkoasiainministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Ulkoasiainvaliokunta_UaV"
         rdfs:label="Ulkoasiainvaliokunta_UaV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;ValtioneuvostonKanslia"
         rdfs:label="ValtioneuvostonKanslia">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>

<rdfs:Class rdf:about="&raske;Valiokunta"
         rdfs:label="Valiokunta">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Valiokuntakasittely"
         rdfs:label="Valiokuntakasittely">
        <rdfs:subClassOf rdf:resource="&raske;Vaihe3"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Valtionvarainvaliokunta_VaV"
         rdfs:label="Valtionvarainvaliokunta_VaV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Valtiovarainministerio"
         rdfs:label="Valtiovarainministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Ymparistoministerio"
         rdfs:label="Ymparistoministerio">
        <rdfs:subClassOf rdf:resource="&raske;Ministerio"/>
</rdfs:Class>
<rdfs:Class rdf:about="&raske;Ymparistovaliokunta_YmV"
         rdfs:label="Ymparistovaliokunta_YmV">
        <rdfs:subClassOf rdf:resource="&raske;Valiokunta"/>
</rdfs:Class>


<rdf:Property rdf:about="&raske;esittelija"
         rdfs:label="esittelija">
        <rdfs:domain rdf:resource="&raske;Lakihanke"/>
        <rdfs:range rdf:resource="&rdfs;Class"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;etunimi"
         rdfs:label="etunimi">
        <rdfs:domain rdf:resource="&raske;Henkilo"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;hankeNimi"
         rdfs:label="hankeNimi">
        <rdfs:domain rdf:resource="&raske;Lakihanke"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;hankeTunniste"
         rdfs:comment="Yksikasitteinen tunniste vaiheelle (HARE-numero, HE-numero)"
         rdfs:label="hankeTunniste">
        <rdfs:domain rdf:resource="&raske;Lakihanke"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;kasittelyValiokunnassa"
         rdfs:comment="Mahdollisuus nakymiin valiokunnittain."
         rdfs:label="kasittelyValiokunnassa">
        <rdfs:domain rdf:resource="&raske;Lakihanke"/>
        <rdfs:range rdf:resource="&rdfs;Class"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;kuuluuHankeeseen"
         rdfs:label="kuuluuHankeeseen">
        <rdfs:domain rdf:resource="&raske;Lakiasiakirja"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;liittyyHankkeeseen"
         rdfs:comment="Liittymakohdat muihin saman aihealueen lakihankkeisiin."
         rdfs:label="liittyyHankkeeseen">
        <rdfs:domain rdf:resource="&raske;Lakihanke"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;luotuVaiheessa"
         rdfs:label="luotuVaiheessa">
        <rdfs:domain rdf:resource="&raske;Lakiasiakirja"/>
        <rdfs:range rdf:resource="&rdfs;Class"/>
</rdf:Property>
```

```
<rdf:Property rdf:about="&raske;nimi"
         rdfs:label="nimi">
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;nimike"
         rdfs:label="nimike">
        <rdfs:domain rdf:resource="&raske;Lakiasiakirja"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;numero"
         rdfs:label="numero">
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;onVaiheessa"
         rdfs:comment="Lakihankkeet kasittelyvaihe."
         rdfs:label="onVaiheessa">
        <rdfs:domain rdf:resource="&raske;Lakihanke"/>
        <rdfs:range rdf:resource="&rdfs;Class"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;osoite"
         rdfs:label="osoite">
        <rdfs:domain rdf:resource="&raske;Henkilo"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;puhnro"
         rdfs:label="puhnro">
        <rdfs:domain rdf:resource="&raske;Henkilo"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;sisaltaaVaiheita"
         rdfs:label="sisaltaaVaiheita">
        <rdfs:domain rdf:resource="&raske;Vaihe"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;sukunimi"
         rdfs:label="sukunimi">
        <rdfs:domain rdf:resource="&raske;Henkilo"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;tunniste"
         rdfs:label="tunniste">
        <rdfs:domain rdf:resource="&raske;Lakiasiakirja"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&raske;valmisteluMinisteriossa"
         rdfs:comment="Lakihankkeen valmistellut ministerio/ministeriot."
         rdfs:label="valmisteluMinisteriossa">
        <rdfs:domain rdf:resource="&raske;Lakihanke"/>
        <rdfs:range rdf:resource="&rdfs;Class"/>
</rdf:Property>
</rdf:RDF>
```

LIITE 2: RDF-kuvausesimerkki

(Laki valtion vientitakuista, HE 215/2000 vp. Prosessin keskeisimmät asiakirjat.)

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY raske 'http://www.it.jyu.fi/raske/rdf#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>

<rdf:RDF xmlns:rdf="&rdf;"
         xmlns:raske="&raske;"
         xmlns:rdfs="&rdfs;">

<raske:Lakiasiakirja rdf:about="http://www.it.jyu.fi/raske/rdf#HE215_2000"
         raske:kuuluuHankeeseen="HE 215/2000 vp"
         raske:pvm="2001-01-19"
         raske:tunniste="HE 215/2000"
         rdfs:label="HE 215/2000">
        <raske:nimike>Hallituksen   esitys   Eduskunnalle   laiksi   valtion   vientitakuis-
ta</raske:nimike>
        <raske:luotuVaiheessa rdf:resource="&raske;PresidentinEsittely"/>
</raske:Lakiasiakirja>
<raske:Lakiasiakirja rdf:about="http://www.eduskunta.fi/faktatmp/utatmp/pevl_2_2001_p.htm"
         raske:kuuluuHankeeseen="HE 215/2000 vp"
         raske:nimike="Perustuslakivaliokunnan lausunto"
         raske:pvm="2001-02-14"
         raske:tunniste="PeVL 2/2001 vp"
         rdfs:label="PeVL 2/2001 vp">
        <raske:luotuVaiheessa rdf:resource="&raske;Valiokuntakasittely"/>
</raske:Lakiasiakirja>
<raske:Lakiasiakirja rdf:about="http://www.eduskunta.fi/faktatmp/utatmp/ev_39_2001_p.htm"
         raske:kuuluuHankeeseen="HE 215/2000 vp"
         raske:pvm="2001-05-14"
         raske:tunniste="EV 39/2001 vp"
         rdfs:label="EV 39/2001 vp">
        <raske:nimike>Eduskunnan vastaus 39/2001 vp, Hallituksen esitys laiksi valtion vien-
titakuista</raske:nimike>
        <raske:luotuVaiheessa rdf:resource="&raske;ToinenKasittely"/>
</raske:Lakiasiakirja>
<raske:Lakihanke rdf:about="http://www.it.jyu.fi/raske/rdf#HE215_2000vp"
         raske:hankeNimi="Laki valtion vientitakuista"
         raske:hankeTunniste="HE 215/2000"
         rdfs:label="Laki valtion vientitakuista">
        <raske:valmisteluMinisteriossa rdf:resource="&raske;KauppaJaTeollisuusministerio"/>
        <raske:onVaiheessa rdf:resource="&raske;LainVoimaantulo"/>
        <raske:kasittelyValiokunnassa rdf:resource="&raske;Perustuslakivaliokunta_PeV"/>
        <raske:kasittelyValiokunnassa rdf:resource="&raske;Talousvaliokunta_TaV"/>
        <raske:esittelija rdf:resource=" http://www.ktm.fi/index.phtml?menu_id=931/365"/>
</raske:Lakihanke>
<raske:Henkilo rdf:about="http://www.ktm.fi/index.phtml?menu_id=931/365"
         raske:osoite="Esimerkkikatu 4"
         raske:puhnro="1234"
         raske:sukunimi="Immonen"
         rdfs:label="Immonen"/>
<raske:Lakiasiakirja rdf:about="http://www.eduskunta.fi/faktatmp/utatmp/tavm_5_2001_p.htm"
         raske:kuuluuHankeeseen="HE 215/2000"
         raske:pvm="2001-04-25"
         raske:tunniste="TaVM 5/2001 vp"
         rdfs:label="TaVM 5/2001 vp">
<raske:nimike>Talousvaliokunnan mietinto 5/2001 vp, Hallituksen esitys laiksi valtion vien-
titakuista</raske:nimike>
        <raske:luotuVaiheessa rdf:resource="&raske;Valiokuntakasittely"/>
</raske:Lakiasiakirja>
</rdf:RDF>
```