TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TEPPO KEKÄLÄINEN

ONTOLOGIALINJAUSPROSESSIT JA -MENETELMÄT

Diplomityö

Tarkastaja: dosentti Ossi Nykänen
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-
neuvoston kokouksessa 9.4.2014

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
KEKÄLÄINEN, TEPPO: Ontologialinjausprosessit ja -menetelmät
Diplomityö, 46 sivua, 2 liitesivua
Marraskuu 2014
Pääaine: Hypermedia
Tarkastaja: dosentti Ossi Nykänen
Avainsanat: semanttinen web, ontologia, ontologialinjaus, prosessi, linjausmenetelmä

Verkkopalveluiden kehittyessä tarve määritellä ontologioita on kasvanut. Ontologioiden avulla Internetin dokumenttien semanttisuutta voidaan kasvattaa ja luoda dokumentteja siten, että tietokone pystyy ymmärtämään niiden sisällön. Työn tavoitteena on tutkia ontologioiden linjaamista ja ymmärtää siihen liittyvää kokonaisprosessia. Linjausprosessissa keskitytään erityisesti asiantuntijan rooliin. Työn teoriatausta on toteutettu kirjallisuuskatsauksena ontologialinjaukseen liittyen ja eri ontologialinjaustyökaluja tutkimalla. Linjausprosessia on tutkittu käytännössä tapaustutkimuksena kahden eri tapauksen avulla. Tapaustutkimuksen pohjalta pystytään toteamaan, että asiantuntijan rooli on merkittävä ontologioiden ja erityisesti ontologialinjauksen suhteen. Tutkimusta ontologioiden hyödyntämiselle ja kehittämiselle tarvitaan, jotta ontologioita aletaan hyödyntämään entisestään verkkopalveluissa.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
KEKÄLÄINEN, TEPPO: Ontology alignment processes and methods
Master of Science Thesis, 46 pages, 2 Appendix pages
November 2014
Major: Hypermedia
Examiner: Adjunct Professor Ossi Nykänen
Keywords: semantic web, ontology, ontology alignment, process, aligment method

As web services evolve, the need for defining ontologies has risen. With ontologies, the semantics of documents in the Internet can be increased and we can create documents in a manner that a computer can understand the contents of them. The goal of this thesis is to study ontology alignment and understand the whole process around it. The main focus is on the role of the specialist in the process. The theory basis of this thesis has been done as a literature review consentrating on ontology alignment and studying different ontology alignment tools. Practical approach to alignment process has been done as a case study. As a result of the case study we can state the role of the scecialist in the alignment process is significant. Additional studies are needed on studying and developing the possibilities of utilizing ontologies in web services in the future.

ALKUSANAT

Diplomityön suorittaminen alkoi keväällä 2014 omakustanteisena projektina. Työn ohjaamisesta on vastannut alusta alkaen dosentti Ossi Nykänen Intelligent Information Systems Laboratorystä (IISLab).

Päätös ontologialinjausprosessien ja -menetelmien valitsemisesta diplomityön aiheeksi syntyi kiinnostuksesta semanttiseen webiin ja sen mahdollisuuksiin tulevaisuuden verkkopalveluissa. Suuri kiitos työn mahdollistamisesta kuuluu ohjaajalleni Ossi Nykäselle, joka aktiivisella toiminnallaan on motivoinut työtäni alusta alkaen. Kiitokset kuuluvat myös työtovereilleni, ystävilleni ja perheelleni kannustuksesta ja tuesta näiden kuukausien aikana. Kiitos kuuluu myös Merja Keräselle avusta työn oikoluvun suorittamisessa.

Teppo Kekäläinen, 13.10.2014

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

IRI	Internationalized Resource Identifier on merkkijono, joka yksilöi resurssin sijainnin. URI-tunniste pohjautuu UNICODE/ISO 10646-merkistöön.
Ontologia	Tietyn aihealueen jaetun käsitteistön eksplisiittinen ja formaali spesifikaatio.
Ontologialinjaus	Prosessi, jossa etsitään kahdesta ontologiasta semanttisesti vastaavia käsitteitä.
OWL	Web Ontology Language on W3C:n standardisoima semanttinen merkkauskieli kuvaamaan ontologioita Internetissä.
RDF	Resource Description Framework on W3C:n määrittelemä ja standardisoima viitekehys kuvaamaan tietoa Internetissä.
RDF-S	Resource Description Framework Schema on semanttinen laajennus RDF määritelmään.
Semanttinen web	Tim Berners-Leen esittelemä visio tulevaisuuden Internetistä, jossa verkossa esiintyvät dokumentit on luotu semanttisiksi tietokoneen ymmärrettävään muotoon.
URI	Uniform Resource Identifier on merkkijono, joka yksilöi resurssin sijainnin. URI-tunniste pohjautuu ISO 8859-1-merkistöön.
W3C	World Wide Web Consortium on vuonna 1994 Tim Berners-Leen perustama kansainvälinen yritysten ja yhteisöjen yhteenliittymä, joka ylläpitää ja kehittää WWW:n standardeja.

1 JOHDANTO

Verkkopalvelujen ja tietojärjestelmien kehittyessä entisestään on kasvanut tarve määritellä ontologioita, jotta käsiteltävät dokumentit olisivat enemmän semanttisia ja tietokoneen ymmärtämässä muodossa. Data ja tieto erityisesti Internetissä ovat usein heterogeenistä ja päällekkäisiä käsitteitä on paljon. Tällöin nousee tarve tunnistaa vastaavia käsitteitä ja karsia päällekkäisyyksiä. Ontologioita linjaamalla, eli löytämällä vastaavat käsitteet, voidaan ratkaista tätä ongelmaa.

Työ keskittyy tutkimaan ontologialinjauksen ongelmaa ja erityisesti siihen liittyvää kokonaisvaltaista prosessia. Työn keskeisimpänä tutkimuskysymys on määritelty tukemaan tätä tavoitetta:

- Kuinka suuri osuus asiantuntijalla on linjausprosessissa?

Työssä tutkitaan ontologialinjausta ja ontologialinjausmenetelmiä. Päätutkimuskysymyksen tueksi määritellään seuraavat apukysymykset:

- Mitä ontologialinjaus tarkoittaa?
- Mitä ontologialinjausmenetelmiä on?
- Minkälainen on ontologialinjausprosessi?

Työ on rajattu koskemaan ontologialinjausprosessia kokonaisuutena. Ontologialinjausmenetelmät esitellään pintapuolisesti ja pääpaino tutkimuksessa on ontologia-asiantuntijan roolissa ontologialinjausprosessikokonaisuudessa.

1.1 Tutkimusmenetelmät

Tutkimuksen teoriatausta koostetaan kirjallisuuskatsauksena. Kirjallisuuskatsauksen aineisto rajataan koskemaan ontologialinjausta tutkiviin artikkeleihin, kirjoihin ja ontologialinjaustyökalujen dokumentaatioihin. Kirjallisuuskatsauksessa tutkituista työkaluista valitaan tapaustutkimukseen sopivin, jotta voidaan tutkia linjausprosessia kokonaisuutena.

Ontologialinjausprosessia tutkitaan tässä työssä tapaustutkimuksena. Tapaustutkimuksessa toteutetaan vertailemalla kahta eri tapausta, joilla pyritään havainnollistamaan ontologialinjauksen mahdollisia käyttötilanteita. Tapaukset testataan ensisijaisessa testiympäristössä ja dokumentoidaan toissijaisessa testiympäristössä. Tapaustutkimuksessa keskitytään tarkastelemaan ontologialinjausprosessin eri vaiheita ja miten teoria toteutuu linjauksen lopputuloksissa.

Tapauksien testiaineisto on rajattu koskemaan aineistoja, jotka eivät vaadi erillistä asiantuntemusta käsitteistöön liittyen. Esimerkiksi biologiaan tai lääketieteeseen liittyvät ontologiat vaatisivat asiantuntemusta, mikä ei tukisi tutkimuksen tavoitteita. Tapaukset on generoitu kuvaamaan eri ääripäätilanteita, eli tapaukset jossa ontologiat ovat joko lähes samoja tai lähes täysin erilaiset, mutta liittyvät samaan aihealueeseen.

1.2 Tutkimuksen rakenne

Työ koostuu johdannon lisäksi teoriataustasta (luku 2), jossa esitellään tutkimuksen kannalta olennaisin teoria ja määritelmät. Kolmannessa luvussa esitellään työhön valitut ontologialinjaustyökalut ja niiden toteutukset yleisellä tasolla. Neljäs luku keskittyy lähdeaineistosta ja linjaustyökaluissa tunnistettuihin linjausmenetelmiin. Viidennessä luvussa esitellään ja määritellään tapaustutkimus, ja sen tapaustutkimuksen tulokset. Kuudennessa luvussa määritellään kirjallisuuskatsauksen ja tapaustutkimuksen pohjalta tunnistettu yleinen malli ontologialinjausprosessille esitellään yhteenveto.

2 TEOREETTINEN TAUSTA

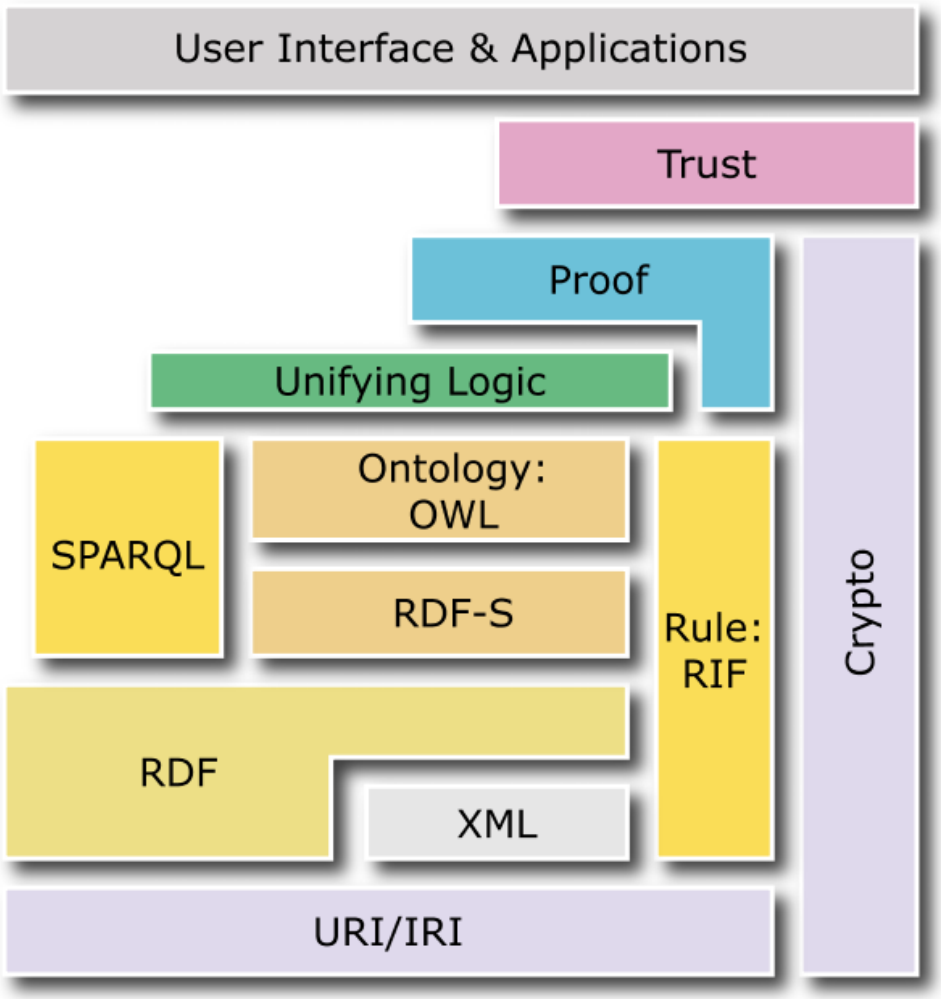
Tässä luvussa esitellään työn osalta keskeisimmät käsitteet. Keskeisimpien käsitteiden ymmärtäminen työn kokonaisuuden kannalta on olennaista. Tällöin on mahdollista ymmärtää suurempi kokonaisuus johon työn sisältö kohdistuu käsitteidensä osalta.

2.1 Semanttinen web

World Wide Webin (WWW) keksijänä ja kehittäjänä tunnettu Tim Berners-Lee toi vuonna 2001 esille idean semanttisesta webistä (semantic web) [1]. Tieto Internetissä on ensisijaisesti suunniteltu ihmiselle tulkittavaksi, ei tietokoneen ymmärrettävässä muodossa. Semanttisessa web ei ole erillinen toteutus, vaan enemmänkin laajennus nykyiseen. Sen toteuttamiseksi on aloitettu monia eri toimenpiteitä, kuten linkitetty data (linked data) [2; 3], jossa ajatuksena on saada raaka data avoimeksi Internetiin kaikkien saavutettavaksi ja käytettäväksi.

Semanttisen webin topologia on kuvattu kuvassa 2.1 pinona. Topologiapinosta vain osa on valmiina ja standardisoituna:

- IRI (Internationalized Resource Identifier): Yksiselitteitten nimeäminen
- RDF (Resource Description Framework): Datan kuvaaminen ja linkittäminen
- RDF-S (Resource Description Framework Schema): Luokat ja hierarkiat
- OWL (Web Ontology Language): Ontologiat
- SPARQL (SPARQL Protocol and RDF Query Language): Kyselykieli
- RIF (Rule Interchange Format): Sääntöjärjestelmä.



Kuva 2.1 Semanttinen web pino [4]

Pinon keskeneräinen osuus koostuu kryptografiasta, eli miten suojata ja salata datalähteet siten, että niihin voidaan luottaa. Luottamuksen mahdollistamiseksi pitää löytyä yhdistävä logiikka, jolla todistaa lähteen luotettavuus. Topologian varaan rakentuu lopulta käyttöliittymä- ja sovelluskerros, jossa tuotetaan sovellutukset semanttisen webin käyttäjille.

2.1.1 RDF

Resource Description Framework (RDF) on World Wide Web Consortiumin (W3C) [5] määrittelemä ja standardisoima viitekehys kuvaamaan tietoa Internetissä. RDF-graafin ydinrakenne on lajitelma triplettejä, jotka koostuvat subjektista, predikaatista ja objektista. Subjekti voi olla IRI-tunniste (Internationalized Resource Identifier) tai tyhjä solmu, predikaatti on IRI-tunniste, ja objekti voi olla IRI-tunniste, literaali tai tyhjä solmu. [6]

IRI-tunniste on yleistys URI-tunnisteesta (Uniform Resource Identifier), jossa voidaan käyttää laajempaa Unicode-merkistökokoelmaa. Jokainen absoluuttinen URI ja URL

(Uniform Resource Locator) on IRI, mutta jokainen IRI-tunniste ei ole URI. Jotta IRI-tunniste voidaan käyttää URI-tunnisteena, se pitää ensin muuttaa kapeamman merkistön mukaiseksi. Literaaleja käytetään lukuarvoina, kuten merkkijonoina, numeroina ja päivämäärinä. Literaali RDF-graafissa sisältää kahdesta kolmeen elementtiä: leksikaalisen muodon, datatyyppi IRI:n tai kielitunnisteen, jos ja vain jos datatyyppi IRI on ”http://www.w3.org/1999/02/22-rdf-syntax-ns#langString”. Tyhjät solmut ovat erillään IRI-tunnisteista ja literaaleista. Muilta osin kokoelma mahdollisista tyhjistä solmuista on mielivaltainen. RDF ei määrittele erikseen tyhjän solmun rakennetta. [6]

Alla on kuvattu esimerkki RDF syntaksista, jossa kuvataan juoma olut ja sen eri juomatyypit:

```
<?xml version="1.0"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
     xmlns:juoma="http://esimerkki.com/juomat#">
     <juoma:olut>
          <juoma:tyyppi>sahti</juoma:tyyppi>
          <juoma:tyyppi>ale</juoma:tyyppi>
          <juoma:tyyppi>lager</juoma:tyyppi>
          <juoma:tyyppi>lambic</juoma:tyyppi>
     </juoma:olut>
</rdf:RDF>
```

Esimerkissä on esitelty neljä RDF-triplettiä, joissa subjektina on ”juoma:olut”, predikaattina on ”juoma:tyyppi” ja objektina ”sahti”, ”ale”, ”lager” ja ”lambic”.

2.1.2 RDF-S

Resource Description Framework Schema (RDF-S) on semanttinen laajennus RDF-määritelmään. Se tuo mukanaan datamallinnussanaston RDF-dataan. RDF-S:n avulla voidaan kuvailla resurssien välisiä suhteita ja resurssien ominaisuuksia. Resurssit voidaan jakaa luokkiin. Luokkien jäseniä kuvataan instansseina, lisäksi luokille voidaan määritellä aliluokkia. [7]

Alla on kuvattu esimerkki RDF-S ominaisuuksista:

```
<?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#"
xml:base="http://esimerkki.com/juomat#">

<rdfs:Class rdf:ID="olut" />
<rdfs:Class rdf:ID="sahti">
  <rdfs:subClassOf rdf:resource="#olut"/>
</rdfs:Class>
<rdfs:Class rdf:ID="ale">
  <rdfs:subClassOf rdf:resource="#olut"/>
</rdfs:Class>
<rdfs:Class rdf:ID="lager">
  <rdfs:subClassOf rdf:resource="#olut"/>
</rdfs:Class>
<rdfs:Class rdf:ID="lambic">
  <rdfs:subClassOf rdf:resource="#olut"/>
</rdfs:Class>
</rdf:RDF>
```

Esimerkissä on kuvattu luokat ”olut” ja sen alaluokat ”sahti”, ”ale”, ”lager” ja ”lambic”.

2.1.3 OWL

Web Ontology Language (OWL) on W3C:n standardisoima semanttinen merkkauskieli, jolla kuvataan ontologioita Internetissä. OWL:n avulla pystytään kuvaamaan laajempaa ja monipuolisempaa semantiikkaa kuin sen edeltäjillä RDF ja RDF-S. OWL on kehitetty DAML+OIL ontologiakielten pohjalta. [8]

Alla on kuvattu esimerkki OWL-syntaksista ja sen ominaisuuksista:

```
<owl:Class rdf:ID="olut">
  <rdfs:subClassOf rdf:resource="juoma"/>
  <owl:disjointWith rdf:resource="viini"/>
  <owl:equilaventClass rdf:resource="sahti"/>
  <owl:equilaventClass rdf:resource="ale"/>
  <owl:equilaventClass rdf:resource="lager"/>
  <owl:equilaventClass rdf:resource="lambic"/>
</owl:Class>
```

Esimerkissä on kuvattu luokka ”olut”, joka on alaluokka luokalle ”juoma”. Lisäksi on määritelty, että ”olut” ei voi olla ”viini”, mutta on vastaava kuin ”sahti”, ”ale”, ”lager” ja ”lambic”.

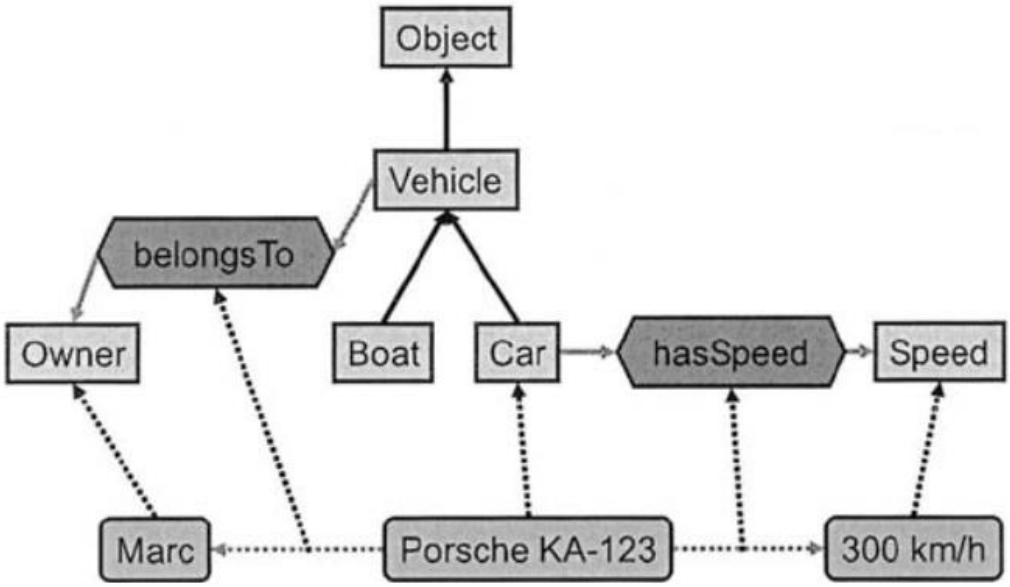
OWL 2 on OWL:n uudempi ja kehittyneempi versio, joka parantaa entisestään ontologioiden kuvaamista. OWL 2 tuo mukanaan uusia ominaisuuksia, kuten avainarvoja (keys), omaisuusketjuja (property chains) ja monipuolisempia datatyyppejä. [9]

2.2 Ontologia

Ontologian määritelmästä on useita versioita, mutta viitatuin määritelmä on Thomas R. Gruberin [10; 11] esittelemä määritelmä, jonka mukaan ontologia on ”käsitteistön eksplisiittinen spesifikaatio”. Marc Ehrig [12] määrittelee ontologian laajemmin ”tietyn aihealueen jaetun käsitteistön eksplisiittisenä ja formaalina spesifikaationa”. Formaalius mahdollistaa koneluettavan muodon ontologialle, ja jaettavuus korostaa jaettua ja yhteistä tietoa, esimerkiksi jonkin ryhmän sisällä jaettua tietoa. Tiettyyn aihealueeseen rajautuminen korostaa sitä, että ontologia ei pyri määrittelemään kaikkea mahdollista, vaan rajautuu pienempään kokonaisuuteen, joka on käyttötarpeeseen soveltuva.

Tarve ontologioille on syntynyt nykyajan lukuisten tietojärjestelmien dataa kuvaavien skeemojen kautta. Semanttinen ontologioiden linkittäminen on tullut tarpeelliseksi järjestelmien yhteensopivuuden mahdollistamiseksi. On helppo havaita, että järjestelmiä, ja sitä kautta ontologioita, on lukematon määrä. Ontologialinjausmenetelmien automaattisten ja puoliautomaattisten menetelmien kehittäminen on välttämätöntä manuaalisen työn vähentämiseksi. [12]

Esimerkkiontologia on havainnollistettu kuvassa 2.2. Konseptit on kuvattu suorakulmioina, relaatiot kuusikulmioina ja instanssit pyöreäreunaisina suorakulmioina. Kiinteäviivaiset nuolet kuvaavat luokiteltuja relaatioita ja katkoviivaiset nuolet konseptin sekä relaation instansseja.



Kuva 2.2 Esimerkki ontologiasta [12]

Esimerkkikuvasta voidaan tunnistaa kuusi konseptia: esine (object), ajoneuvo (vehicle), omistaja (owner), vene (boat), nopeus (speed) ja auto (car); kaksi relaatiota onNopeus (hasSpeed), kuuluuJollekin (belongsTo) sekä kolme instanssia: Marc, Porsche KA-123 ja 300 km/h. Kuvasta voidaan muun muassa johtaa, että auto ja vene ovat ajoneuvoja, jotka ovat esineitä. Autolla on nopeus. Porsche KA-123 on auto ja sen nopeus on 300 km / h. [12]

2.2.1 Ontologian formaali määritelmä

Ontologian formaalista määritelmästä on useampia versioita. Xingsi Xue et al. [13] esittelevät Gruberin [10; 11] määritelmän ontologiasta, jossa ontologia on tripletti:

$$O = (C, P, I), \tag{1}$$

missä *O* on joukko, joka sisältää konsepteja *C*, ominaisuuksia *P* ja instansseja *I*. Juanzi Li et al. [14] ovat määritelleet ontologian formaalisti kuusituplana, jossa

$$O = \{C, P, H^C, H^P, A^O, I\}, \tag{2}$$

missä C ja P ovat konseptien (concepts) ja ominaisuuksien (properties) joukkoja. Joukko H^C sisältää hierarkkiset suhteet konsepteille niiden joukkojen karteesisena tulona $H^C \subset C \times C$. Vastaavasti joukko H^P sisältää hierarkkiset suhteet ominaisuuksille niiden joukkojen karteesisena tulona $H^P \subset P \times P$. Joukko A^O sisältää ontologian aksioomat ja *I* konseptien ja ominaisuuksien instanssit. [14]

Tarkasteltaessa annettuja formaaleja määritelmiä (1) ja (2), voidaan havaita määritelmien ytimessä olevan konseptit, joihin viitataan myös luokkina [13], ominaisuudet ja instanssit. Yhdessä nämä muodostavat ontologian entiteetit.

2.3 Ontologialinjaus

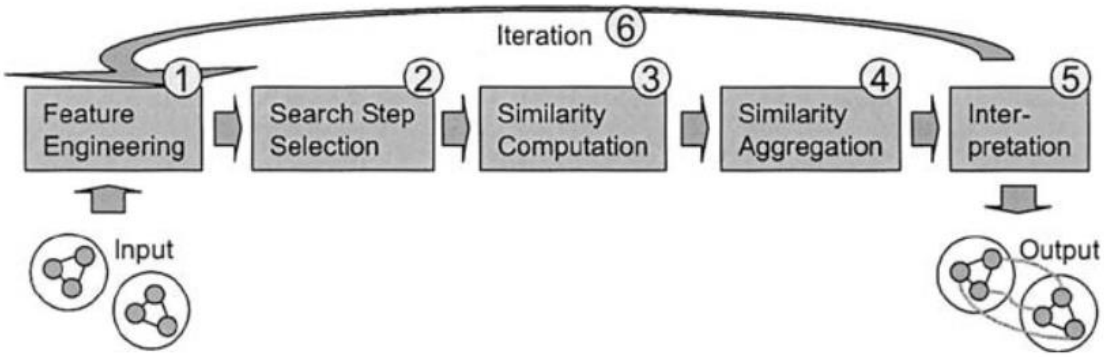
Jérôme Euzenat ja Pavel Shvaiko esittelevät kirjassaan ”Ontology Matching” [15] kuusi tunnettua sovellusta, missä ontologialinjaus on havaittu mahdolliseksi pitkäaikaisratkaisuksi:

- Ontologioiden mallintaminen (ontology engineering).
- Tiedon integrointi (information integration).
- Peer-to-peer tiedonjakaminen (peer-to-peer information sharing).
- Teknisten verkkopalveluiden koostaminen (web service composition).
- Autonomiset kommunikaatiojärjestelmät (autonomous communication systems).
- Navigointi- ja kyselyjärjestelmät (navigation and query answering on the web).

Ontologioiden mallintaminen sisältää toimenpiteet, joissa suunnitellaan, implementoidaan ja hallitaan ontologiapohjaisia järjestelmiä. Tiedon integrointi sisältää skeemojen integroinnin ja datan integroinnin toteuttavat järjestelmät. Peer-to-peer tiedonjakamisen haaste on vertaisverkossa toimivien päätepisteiden (peer) käsittelevien tiedon monikirjaisuudessa. Vastaavasti verkkopalvelut ja kommunikointijärjestelmät sisältävät ja käsittelevät tietoa, joka on heterogeenistä, eli monimuotoista. Navigoinnin parantaminen ja siirtyminen hakupohjaisuudesta kyselypohjaisuuteen uskotaan parantavan käyttökokemusta semanttisessa webissä. [15]

Yhteenvetona voidaan todeta, että ontologialinjauksen tarve on heterogeenisen tiedon ymmärtämisessä ja muokkaamisessa siten, että järjestelmät pystyvät kommunikoimaan autonomisesti ja tehokkaasti. Vastaavat käsitteet eri ontologioista löytämällä ja linjaamalla pystytään välttämään raskaita logiikkaratkaisuja ja parantamaan tiedon saavutettavuutta.

Ontologialinjauksen toteuttamiseksi on oltava prosessi, jonka tuloksena saavutetaan linjattu ontologia. Marc Ehrig esittelee [12] kuvassa 2.3 havainnollistetun iteratiivinen ontologialinjausprosessin.

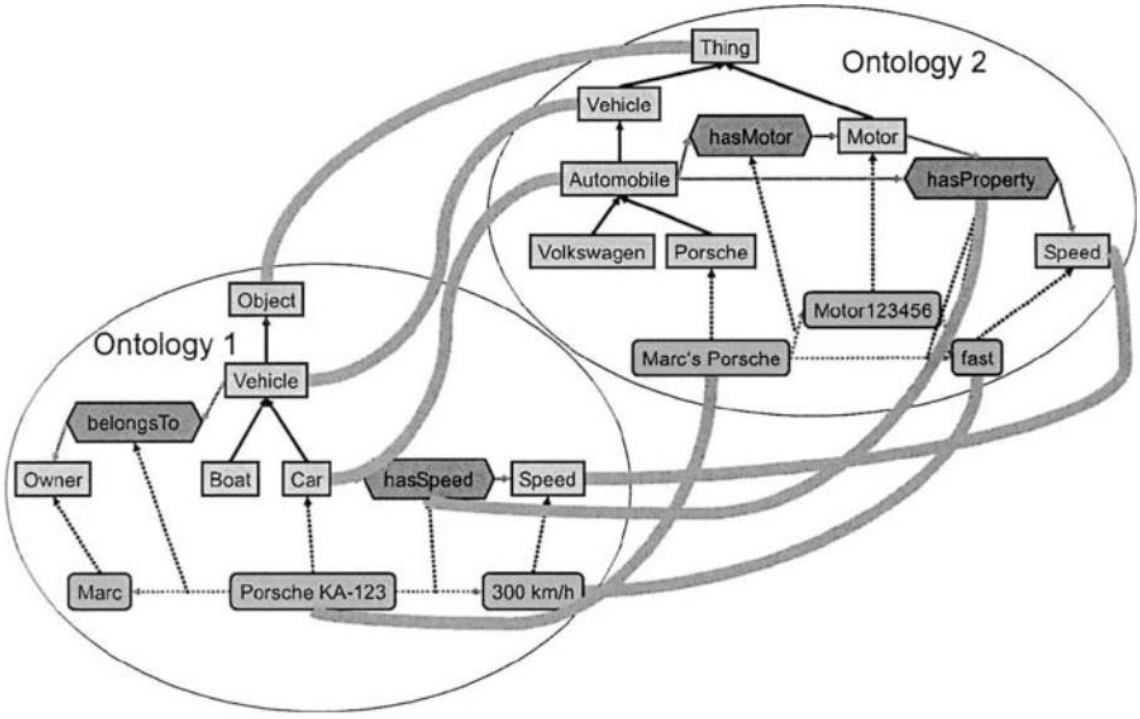


Kuva 2.3 Esimerkki ontologialinjausprosessista [12]

Ontologialinjausprosessi koostuu kuudesta eri vaiheesta, jotka ovat:

1. Ominaisuuksien järjestelyssä (feature engineering) valitaan pieniä otteita yleisestä ontologiasta kuvatakseen tiettyjä entiteettejä. Otteilla on tietty merkitys ontologian sisällä, ne kuvaavat tiettyä semantiikkaa. Esimerkiksi nimilappu ”car” valitaan kuvaamaan käsitettä o1:car.
2. Etsintävaiheen valinta (search step selection) johtaa alustavia linjauksia valittujen entiteettipariehdokkaiden joukosta. Esimerkiksi vertailu suoritetaan vain o1:car ja o2:automobile, mutta ei o2:hasMotor.
3. Vastaavuuden laskenta (similarity computation) vaiheessa lasketaan valittujen entiteettipariehdokkaiden vastaavuus laskennallisesti.
4. Vastaavuuksien yhdistäminen (similarity aggregation) yhdistää eri entiteettejä yhdistävät tekijät yhdeksi arviotavaksi arvoksi.
5. Tulkintavaiheessa (interpration) johtaa aiemmin määriteltyjen vastaavuusarvioiden pohjalta linjaustuloksena ehdotuksen entiteettiparista, jos ollenkaan.
6. Iteraatio (iteration) mahdollistaa laskettujen entiteettiparien vaikutuksen toisiinsa. Iteraatio päättyy kun uusia linjausehdotuksia ei nouse esille. [12]

Linjausprosessin lopputuloksena syntyvä linjaus on havainnollistettu kuvassa 2.4. Kuvassa vastaavat entiteetit on yhdistetty paksulla viivalla.



Kuva 2.4 Esimerkki ontologialinjauksesta [12]

Kuvasta voidaan tunnistaa linjauksen tuloksena muun muassa, että relaatiot asia (thing) ja esine (object) ovat linjattu, kuten auto (car) ja automobiilikin (automobile).

2.3.1 Ontologialinjauksen formaali määritelmä

Juanzi Li et al. [14] ovat määritelleet ontologialinjauksen formaalisti mukaillen OAEI:n vuoden 2006 määrityksiä [16]. Otettaessa kaksi ontologiaa O_1ja O_2, ontologialinjausprosessi tai -menetelmä löytää jokaiselle entiteetille ontologiassa O_1 vastaavan entiteetin ontologiassa O_2. Ontologia O_1 on lähdeontologia ja O_2 kohdeontologia. Tällöin ontologialinjaus määritellään formaalisti relaationa, jossa

$$Align(O_1, O_2) = \left\{ \begin{matrix} (e_{i1}, e_{i2}, con_i, relation_i) | \\ e_{i1} \in O_1, e_{i2} \in O_2, con_i \in [0,1], \\ relation_i \in \{exact, narrower, broader, overlap\} \end{matrix} \right\}.$$

Joukon $Align(O_1, O_2)$ jokainen neljäpaikkainen alkio $(e_{i1}, e_{i2}, con_i, relation_i)$ esittävät, että entiteetti e_{i1} ontologiasta O_1 on linjattuna entiteetin e_{i2} kanssa ontologiasta O_2 luotettavuudella (confidence) con_i ja linjaustyypillä $relation_i$. Linjaustyyppi kuvastaa linjauksen tyypin, jossa *exact* tarkoittaa entiteettien olevan täsmälleen samat, *narrower* tarkoittaa entiteetin e_{i1} olevan alakäsite entiteetille e_{i2} ja *broader* päinvastaisesti laajempi yläkäsite. Osittain vastaavat käsitteet kuvataan tyypillä *overlap*. Luotettavuusarvo con_i on laskennallinen arvo, joka kuvaa linjauksen luotettavuutta. Mitä suurempi luotettavuusarvo on, sitä luotettavampi linjaustulos on. [14]

2.3.2 Linjauksen laadukkuuden arviointi

Ontologialinjauksen laadukkuuden arviointi suoritetaan yleensä hyödyntäen hollantilaisen C.J. Van Rijsbergenin [17] määrittelemien saanti- (recall) ja tarkkuuskäsitteiden (precision) avulla. Saanti laskee löydettyjen linjauksien lukumäärän linjatusta ontologiasta A suhteessa oikeiden linjauksien lukumäärään referenssilinjauksessa R. Saantiarvon ollessa 1 voidaan todeta, että kaikki linjaukset löydettiin.

$$recall = \frac{|R \cap A|}{|R|}.$$

Saantiarvo ei kerro virheellisten linjauksien lukumäärää, joten saantiarvon vastapainoksi lasketaan tarkkuus. Tarkkuus kertoo mitkä löydetyistä linjauksista ovat oikeita linjauksia. Tarkkuuden ollessa 1 voidaan todeta, että kaikki löydetyt linjaukset ovat oikeita.

$$precision = \frac{|R \cap A|}{|A|}.$$

Tarkkuus ei kuitenkaan kerro onko linjauksia ollenkaan löydetty. Tällöin saanti ja tarkkuus on tasapainotettava toisiaan kohden niin sanotun F-mitan (f-measure) avulla.

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

Kahden linjauksen F-mitan ollessa sama on mahdotonta arvioida kumman laadukkuus on parempi. Lisäksi on vaikea arvioida kumpi linjauksista on vähemmän jäävi tarkkuutensa tai saantinsa suhteen. [13]

3 LINJAUSTYÖKALUT

Linjaustyökaluja tutkimalla saadaan ymmärrystä toteutuksien yhtäläisyyksistä ja eroavaisuuksista. Havaintoja hyödynnetään yleisen linjausprosessimallin toteutuksessa. Linjaustyökalujen valintaperusteena oli helppo saatavuus ja sopivuus tutkimuksen tavoitteisiin. Linjaustyökalut, jotka oli suunniteltu tiettyjä ontologioita varten, kuten biologisiin ontologioihin erikoistunut Sambo-työkalu [18], rajattiin valintojen ulkopuolelle. Saatavuus määrittyi sen perusteella kuinka usein ja milloin viimeksi kyseistä työkalua oli tutkittu.

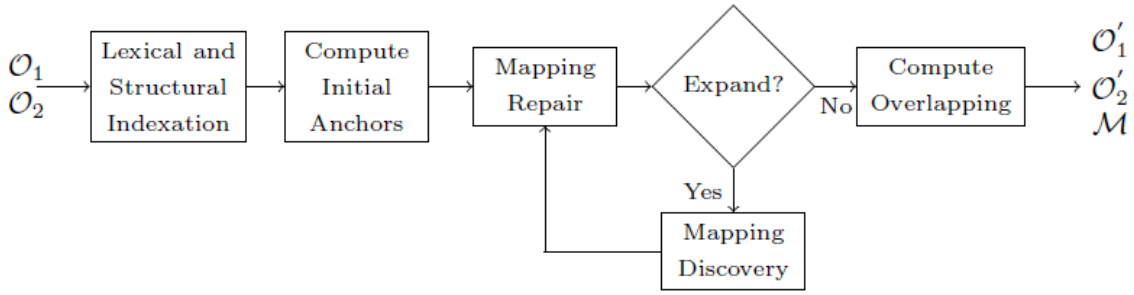
Työhön valitut linjaustyökalut on valittu OAEI:n (Ontology Alignment Evaluation Iniative) [19] vuoden 2013 ontologialinjauskilpailun osallistujista [20]. Valitut linjaustyökalut ovat LogMap, LilyIOM, RiMOM ja SLINT+. OAEI on järjestänyt vuosittaiset ontologialinjauskilpailut vuodesta 2004 lähtien. Kilpailujen tavoitteena on testata ja vertailla eri linjausmenetelmiä. Kaikkiaan kilpailuun osallistui 23 eri linjausprosessitoteutusta, joista valittiin instanssien linjaukseen (instance matching) osallistuneet toteutukset. [19]

Linjaustyökaluista LogMap osoittautui sopivimmaksi työn toteutuksen kannalta. Muiden työkalujen toimivuus oli epävarmaa ja niiden lisäksi työkalujen dokumentointi sekä ohjeistus olivat vajavaisia.

3.1 LogMap

LogMap työkalun ovat kehittäneet Ernesto Jimenez-Ruiz, Bernarno Cuenca Gra ja Ian Horrocks. LogMap sisältää sisäänrakennettuja päättely- ja epäjohdonmukaisuuden korjausominaisuuksia. LogMap etsii linjauksia luokkien, ominaisuuksien ja instanssien joukosta. [20]

LogMap-työkalun linjausprosessi on havainnollistettu vuokaaviona kuvassa 3.1. Linjausprosessissa on neljä vaihetta: sanastollinen ja rakenteellinen indeksointi, alustavien ankkurilinjauksien laskenta, linjausten etsintä ja korjaus, ja päällekkäisyyksien arviointi. Prosessi ottaa vastaan kaksi linjattavaa ontologiaa O_1 ja O_2. Prosessin ulostulona on M, joka sisältää linjatut käsitteet. [21]



Kuva 3.1 LogMapin linjausprosessi [21]

Sanastollisessa indeksoinnissa (lexical indexation) LogMap indeksoi molempien ontologioiden nimilaput ja myös niiden sanastolliset variaatiot. Tämä mahdollistaa ulkopuolisen sanaston, kuten WordNetin, hyödyntämisen tulevissa LogMap-versioissa. [21]

Rakenteellinen indeksointi toteutetaan hyödyntämällä intervallinimiöintiskeemaa (interval labeling scheme), jonka avulla pystytään esittämään laajennettu luokkahierarkia molemmista syötetyistä ontologioista. Linjausprosessin intervallinimiöintiskeema on esitelty tarkemmin kappaleessa 4.2.1. [21]

Kolmannessa vaiheessa suoritetaan alustavien ankkurilinjauksien laskenta (computation of initial anchor mappings), eli lasketaan alustava joukko toisiaan vastaavia ankkurilinjauksia risteyttämällä kieliopilliset indeksit jokaisesta syötetystä ontologiasta. Nämä linjaukset voidaan tulkita olevan toisiinsa nähden vastaavia ja ne toimivat myöhemmin aloituspisteenä mahdollisien uusien linjauksien havaitsemisessa. [21]

Neljäntenä vaiheena suoritetaan linjausten korjaus- ja havaitsemisvaihe (mapping repair and discovery). Se on iteratiivinen prosessi, joka vaihtelee korjaus- ja havaitsemisvaiheen välillä. Korjausvaiheessa LogMap käyttää päättelyalgoritmia tunnistamaan luokat, jotka ovat tyydyttämättömiä molempien syötettyjen ontologioiden osalta ja tähän vaiheeseen mennessä havaitut linjaukset. Tämän jälkeen jokainen epämieluisa looginen seuraus on automaattisesti korjattu käyttäen ahnetta diagnostiikka-algoritmia. [21]

Uusien linjauksien havaitseminen on toteutettu ylläpitämällä kahta kontekstia, eli semanttisesti vastaavaa luokkaa, jokaista ankkurilinjausta kohden. Kontekstit vastaaville ankkureille on laajennettu rinnakkain käyttäen luokkahierarkioita syötetyistä ontologioista. Uudet linjaukset on tämän jälkeen linjattu laskemalla luokkien vastaavuus käyttäen ISUB-työkalua. ISUB-työkalu on esitelty tarkemmin kappaleessa 4.1.4. Uusien linjauksien havaitseminen pohjautuu lokaalisuuden periaatteeseen (principle of locality). Jos luokat C_1 ja C_2 ovat oikein linjattuja, niin luokat, jotka ovat semanttisesti vastaavia (semantically related) luokkaan C_1 ontologiassa O_1, ovat todennäköisesti linjattuna niihin, jotka ovat semanttisesti liittyviä luokkaan C_2 ontologiassa O_2. Iteratiivinen prosessi jatkuu niin kauan kunnes yksikään konteksti ei ole laajennettavissa havaitsemisvaiheessa. Prosessin ulostulona on lajitelma linjauksia, jotka ovat mitä todennäköisimmin ”puhtaita”, eli linjauksia jotka eivät johda loogisiin virheisiin liitettäessä syötettyihin ontologioin. [21]

Viidennessä vaiheessa suoritetaan ontologioiden päällekkäisyyden arviointi. Linjaustuloksen lisäksi LogMap palauttaa myös kaksi muuta ontologiaa, jotka kuvaavat ontologioiden päällekkäisyyksiä. Päällekkäisyysontologiat O_1' ja O_2' kuvaavat päällekkäisyyksiä ontologioiden O_1 ja O_2 välillä. Päällekkäisyysontologiat tuotetaan erityisesti asiantuntijoita varten, jotka haluavat tarkastella LogMapin tuloksia kun jokin linjaus puuttuu. [21]

3.1.1 LogMap muissa tutkimuksissa

LogMap-työkalun laadukkuutta on tutkittu OAEI:n vuosittaisissa ontologialinjauskilpailuissa vuosina 2011 – 2013. Vuosina 2012 ja 2012 LogMap osallistui kaikkiin testattaviin testiskenaarioihin. Vuonna 2011 LogMap osallistui kolmeen kaikista viidestä mahdollisesta skenaariosta. Jokaisena vuotenaan LogMap on pärjännyt hyvin verrattuna muihin kilpailijoihin niin tehokkuuden kuin luotettavuuden ja tarkkuudenkim suhteen. LogMap:sta toimitettu kevyempi LogMapLite versio pärjäsi erityisesti hyvin tehokkuudessa ja nopeudessa. Kevyemmässä versiossa on käytetty vain tarvittavia komponentteja tehokkuuden kasvattamiseksi. [20; 22; 23]

Antón Morant tutki LogMapia [24] ja esitteli parannuksia sen vastaavuuksien löytämiseen käytettyihin tekniikoihin. Parannukset koskivat LogMap-prosessissa kieliopillista ja rakenteellista indeksointia sekä korjausalgoritmia. Kieliopillista indeksointia Morant paransi WordNet-synonyymeillä ja karsimalla sanaliitteitä (stemming). Erityisesti sanojen sanaliitteiden karsiminen, eli sanojen palauttaminen tiettyihin kantasanoihin kasvatti tehokkuutta merkittävästi. Rakenteellisessa indeksoinnissa saavutettiin jopa 97 % lasku suoritusajassa toteutetulla kirjastolla LogMap:n intervallinimiöintiskeemaan. Linjauksen korjaus- ja havaitsemisvaiheessa käytettyä korjausalgoritmia Morant optimoi kahdella tavalla saavuttaen 17 % laskun indeksointivaiheiden suoritusajoissa. Morantin esittämät parannukset huomiotiin LogMap:ssa saman tien ja paranneltu versio oli jo käytössä vuoden 2011 OAEI:n ontologialinjauskilpailussa. [24]

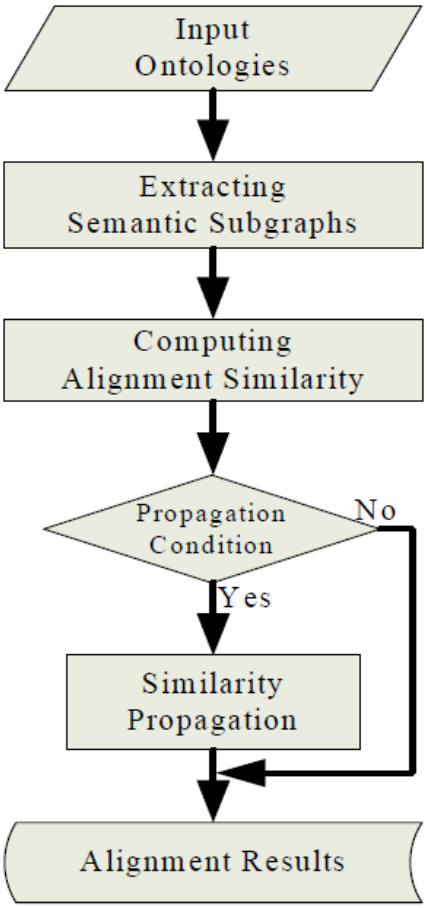
3.2 Muut työkalut

Muut linjaustyökalut esitellään pintapuolisesti keskittyen niiden linjausprosessitoteutukseen. Järjestelmien testaaminen osoittautui hankalaksi. LilyIOM:n asentaminen ei ollut mahdollista ensisijaiseen testiympäristöön ja toissijaisessa testiympäristössä työkalu ei toiminut luotettavasti. Työkalu kaatui ja tuotti epäluotettavia tuloksia. RiMOM-työkalun asentaminen ei onnistunut, eikä asennuksen ohjeistus ollut tarpeeksi kattavaa. SLINT+:n käyttö ei onnistunut, koska sen vaatimat ontologiat olisi pitänyt toteuttaa eri toteutuksella verrattuna valittuihin ontologioihin, jotka oli toteutettu OWL:lla.

3.2.1 LilyIOM

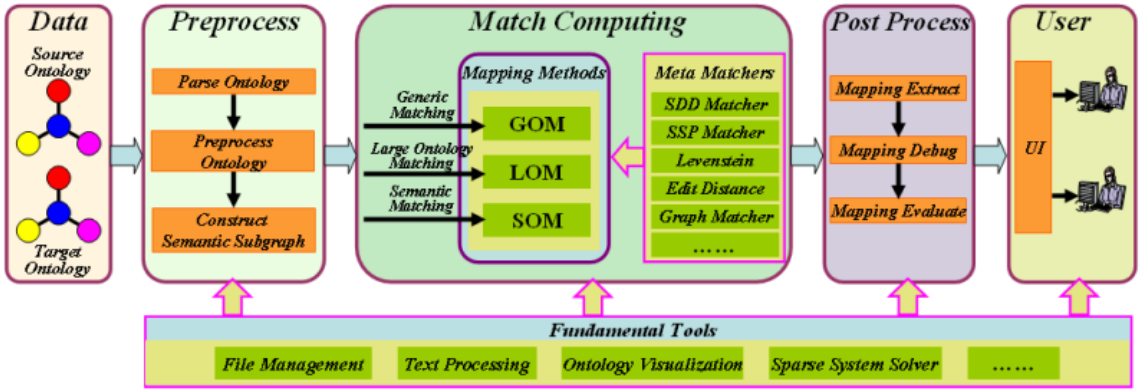
LilyIOM on yksi ontologialinjausjärjestelmä Lilyn komponenteista. Komponentti hyödyntää kieliopillista ja rakennepohjaista menetelmää, sekä samankaltaisuuden arviointia. Lily-järjestelmän on kehittänyt Peng Wang. [20]

Lily-järjestelmä on ohjelmoitu Java- ja C++-ohjelmointikielillä. Järjestelmän linjausprosessi on havainnollistettu kuvassa 3.2. Linjausprosessi pohjautuu järjestelmän versioon 1.2. Lilyn linjausprosessi jakautuu kolmeen vaiheeseen: semanttisen alagraafin poimimiseen (extracting semantic subgraph), linjauksen samankaltaisuuden laskentaan (computing alignment similarity) ja samankaltaisuuden päättelemiseen (similarity propagation). Semanttisen alagraafin poiminnassa tavoitteena on tuottaa jokaiselle entiteetille tarkka merkitys. Tämä on toteutettu hyödyntämällä Christos Faloutsosin, Kevin S. McCurleyn ja Andrew Tomkinsin [25] kehittelemää menetelmää poimia alagraafi ontologiagraafin solulle. Linjauksen samankaltaisuuden laskenta pohjautuu semanttisiin alagraafien tuottamaan informaatioon. Laskennassa huomioidaan sekä perustason kuvaustieto että semanttinen kuvaustieto. Perustason kuvaustieto sisältää identiteetin (identifier), nimilapun (label) ja kommentit (comments). Semanttinen kuvaustieto sisältää luokkahierarkiat, niihin liittyvät ominaisuudet ja instanssit. [26]



Kuva 3.2 Lily -järjestelmän linjausprosessi [26]

Lily-järjestelmän linjausprosessin arkkitehtuuri on havainnollistettu yksityiskohtaisemmin kuvassa 3.3. Arkkitehtuurikuva pohjautuu järjestelmän versioon 2.0. Arkkitehtuurissa on linjausprosessin kolme vaihetta: esivaihe (preprocess), linjauksen laskenta (match computing) ja jälkivaihe (post process).



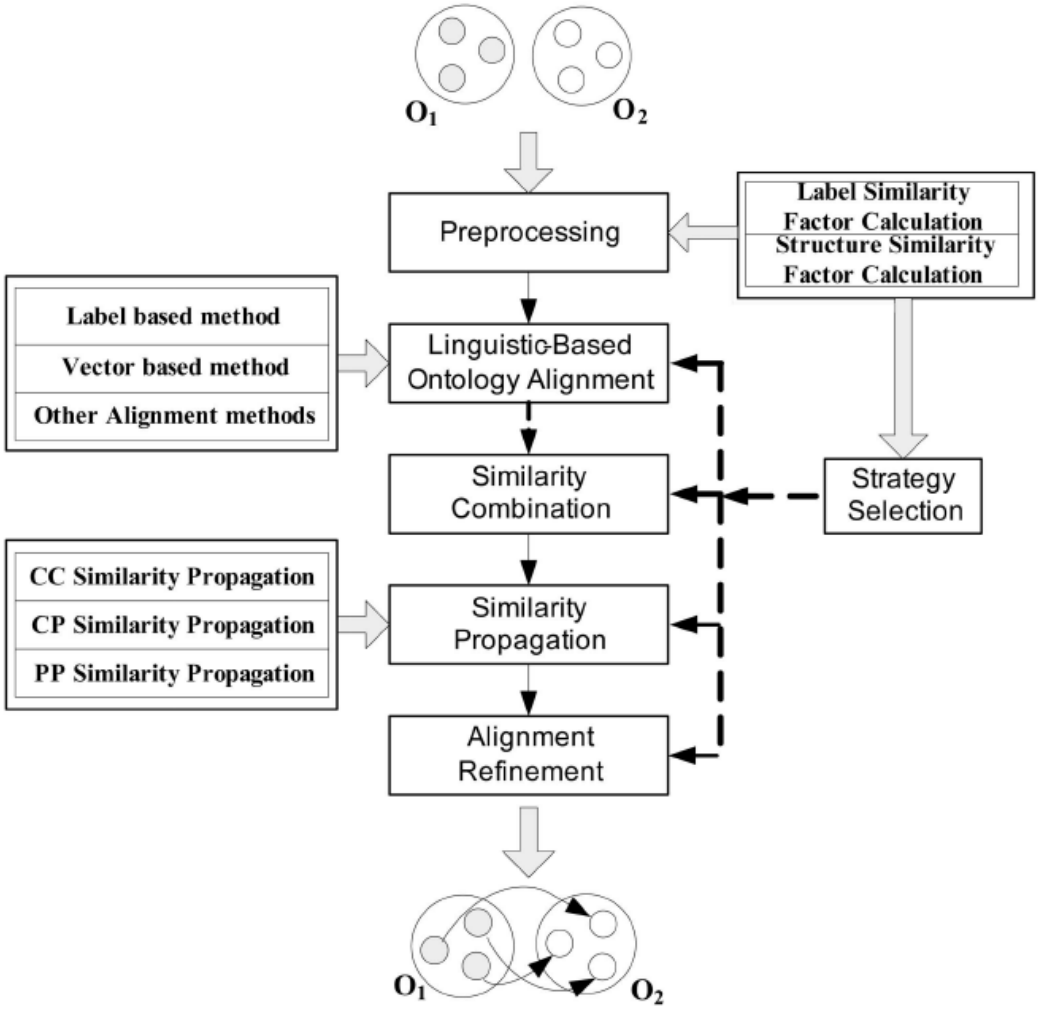
Kuva 3.3 Lily -järjestelmän arkkitehtuuri [27]

Esivaiheen tuloksena on semanttinen alagraafi. Linjauksen laskennassa hyödynnetään eri linjausmenetelmiä, kuten kieliopillisia ja rakenteellisia menetelmiä. Arkkitehtuurissa esitellään kieliopillisista menetelmistä muun muassa Levensteinin etäisyys, joka esitellään tarkemmin kappaleessa 4.1.1. [27]

3.2.2 RiMOM

RiMOM-työkalun ovat kehittäneet Juanz Li, Chao Shao, Qian Zheng, Zhichun Wang ja Linmei Hu. RiMOM mahdollistaa eri linjausmenetelmien, eli strategioiden, yhdistelemisen tavoitteenaan löytää optimaalinen linjaustulos. [20]

RiMOM-työkalun linjausprosessi on havainnollistettu kuvassa 3.4. Prosessi jakautuu viiteen eri vaiheeseen: esikäsittelyyn, kieliopillinen linjaamiseen, samankaltaisuuksien yhdistelemiseen, samankaltaisuuden päättelemiseen ja linjauksen jalostamiseen.



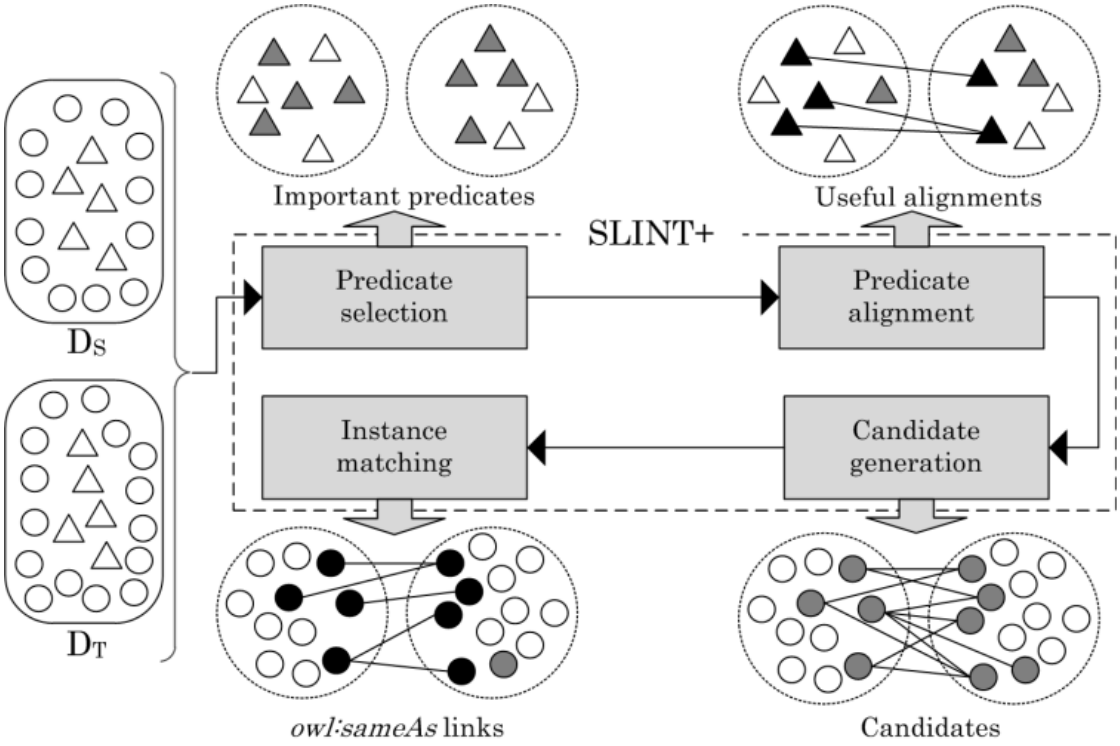
Kuva 3.4 RiMOM -työkalun linjausprosessi [14]

Esikäsittelyssä annettujen ontologioiden osalta järjestelmä tuottaa kuvauksen jokaiselle entiteetille ja laskee kaksi vastaavuustekijää, joita hyödynnetään myöhemmissä vaiheissa. Kieliopillisessa linjauksessa hyödynnetään useita kieliopillisia linjausmenetelmiä, joista vektorietäisyys esitellään tarkemmin kappaleessa 4.1.5. Jokainen menetelmä hyödyntää eri ontologista tietoa ja hankkii näin vastaavuustuloksia jokaiselle entiteettiparille. Samankaltaisuuksien yhdistelyssä yhdistetään valittujen menetelmien vastaavuustekijät. Yhdistelmien painoarvot määrittyvät esikäsittelyssä määriteltyjen kahden vastaavuustekijän perusteella. Samankaltaisuuden päättelyssä hyödynnetään rakenteellisia linjausmenetelmiä, pääasiassa käsite-käsite- (Concept-to-Concept), ominaisuus-ominaisuus- (Property-to-Property) ja käsite-ominaisuusmenetelmiä (Concept-to-Property). Rakenteellisista linjausmenetelmistä RiMOM hyödyntää vastaavuuden ylikuormittamista, joka esitellään tarkemmin kappaleessa 4.2.2. Viimeisessä, eli viidennessä, vaiheessa RiMOM luo ja hienosäätää linjauslopputuloksen. [14]

3.2.3 SLINT+

SLINT+ on Khai Nguyenin ja Ryutaro Ichisen kehittämä skeemariippumaton linjaustyökalu. Se tunnistaa kahden annetun linkitetyn datalähteen sisältämät vastaavat instanssit ja toteuttaa niille owl:sameAs-määritteen. [20]

SLINT+:n linjausprosessi on esitelty kuvassa 3.5. Linjausprosessissa on neljä vaihetta: predikaattien valinta, predikaattien linjaaminen, kandidaattien luominen ja instanssien linjaaminen.



Kuva 3.5 SLINT+ -linjausprosessi [28]

Predikaattien valinnassa järjestelmä kerää linjauksen kannalta olennaisimmat predikaatit datalähteistä D_S ja D_T. Predikaattien linjauksessa järjestelmä yhdistelee predikaatit ja luo näin alustavan linjauksen. Alustavista linjauksista valitaan niiden luotettavuusarvon perusteella hyödylliset linjaukset. Kandidaattien luonnissa järjestelmä etsii identiteettikandidaatit ryhmittelemällä vastaavat instanssit yhdeksi klusteriksi, eli joukoksi. Instanssien linjaus varmistaa kaikki kandidaatit ja luo lopulliset owl:sameAs-linjaukset.

4 LINJAUSMENETELMÄT

Tässä luvussa esitellään eri linjausmenetelmiä, jotka on tunnistettu luvussa kolme esitellyistä linjausprosesseista ja lähdeaineistosta. Linjausmenetelmät on jaettu kahteen kategoriaan: kieliopilliseen ja rakenteelliseen linjaamiseen.

4.1 Kieliopillisen linjausmenetelmät

Kieliopillisten linjausmenetelmien ajatuksena on tarkastella eri ontologioiden entiteettien vastaavuutta kieliopillisten menetelmien kautta. Entiteettien vastaavuutta voidaan tarkastella vertaamalla entiteettien muutosetäisyyttä toisiinsa tai vertaamalla niitä ulkoiseen sanastoon, kuten WordNet [29]. Ulkoiseen sanastoon vertaamalla voidaan tunnistaa onko tarkasteltava sana synonyymi, eli samaa tarkoittava, hyponyymi, eli yleisempi termi, vai antonyymi, eli vastakohta.

Luvussa kolme esitellyistä työkaluista tunnistettiin kieliopillisina menetelminä Levenshteinin etäisyys, vektorietäisyys ja ISUB-työkalu. Lisäksi ISUB-työkalussa viitattiin Jaro- ja Jaro-Winkler-etäisyyksiin.

4.1.1 Levenshteinin etäisyys

Vladimir Levenshteinin vuonna 1965 kehittämä kahden merkkijonon niin sanotun muutosetäisyyden (edit distance) laskentakaava on laajalti käytetty [12]. Laskentakaavan ideana on laskea kuinka monta muutosta tarvitaan kahden eri merkkijonon muuttamiseksi keskenään samaksi. Muutokset voivat olla merkin lisääminen, poistaminen tai korvaaminen. Levenshteinin etäisyyden, eli muutosetäisyyden, kaava on

$$Levenshtein\ (s_1, s_2) = max\left(0, \frac{\min(|s_1|, |s_2|) - d(s_1, s_2)}{\min(|s_1, s_2|)}\right),$$

missä s_1 ja $s_2$ovat vertailtavia merkkijonoja. $|s_1|$ ja $|s_2|$ määrittelevät merkkijonojen pituudet ja $d(s_1, s_2)$ on lukumäärä tarvittavista muutoksista. Levenshteinin etäisyys voi saada lukuarvot välillä [0,1]. Etäisyyden ollessa 1 ovat merkkijonot identtiset. Muussa tapauksessa mitä suurempi etäisyys, sitä samankaltaisemmat ne ovat. Esimerkiksi kun vertailemme sanoja ”daughter” ja ”daugther”, saamme etäisyydeksi $\frac{8-1}{8} = \frac{7}{8}$.

4.1.2 Jaro-etäisyys

Matthew A. Jaro kehitti vuonna 1989 [30] menetelmän, jonka avulla pystyttiin laskemaan kahden merkkijonon vastaavuutta niissä esiintyvien yhteisten merkkien avulla. Jaro-etäisyyden laskentakaava on:

$$Jaro(s_1, s_2) = \frac{1}{3}\left(\frac{com(s_1, s_2)}{|s_1|} + \frac{com(s_1, s_2)}{|s_2|} + \frac{com(s_1, s_2) - trans(s_1, s_2)}{com(s_1, s_2)}\right),$$

missä $com(s_1, s_2)$ laskee yhteisten merkkien määrän merkkijonoissa s_1 ja s_2, ja $trans(s_1, s_2)$ laskee lukumäärän pareista, jossa samat merkit esiintyvät eri kohdissa merkkijonoa. Jaro-etäisyys voi saada lukuarvot välillä [0,1].

4.1.3 Jaro-Winkler-etäisyys

William E. Winkler paranteli Jaro-etäisyyttä vuonna 1990 [31]. Parannuksien myötä vastaavuudenlaskennassa suosittiin merkkijonoja, joissa oli pidemmät yhteiset etuliitteet [15]. Jaro-Winkler-etäisyyden kaava on:

$$Jaro - Winkler(s_1, s_2) = Jaro(s_1, s_2) + P \cdot Q \cdot \frac{\left(1 - Jaro(s_1, s_2)\right)}{10},$$

missä P on yhteisen etuliitteen pituus ja Q on vakio. Jaro-Winkler-etäisyys voi saada lukuarvot välillä [0,1].

4.1.4 ISUB-työkalu

Giorgos Stoilosin, Giorgos Stamoun ja Stefanos Kolliasin kehittämä ”ISUB-työkalu” on ontologialinjausta varten kehitelty merkkijonojen vertailutyökalu. Muut merkkijonojen vertailutyökalut, kuten Levenshteinin, Jaron ja Jaro-Winklerin etäisyydet, ovat puutteellisia ontologialinjauksen tarpeisiin nähden. ISUB-työkalussa yhdistellään eri menetelmiä paremman lopputuloksen saavuttamiseksi. ISUB-työkalun laskentakaava on määritelty seuraavasti:

$$Sim(s_1, s_2) = Comm(s_1, s_2) - Diff(s_1, s_2) + winkler(s_1, s_2),$$

missä $comm(s_1, s_2)$ laskee kahden merkkijonon s_1 ja s_2 yhteneväisyyden ja $diff(s_1, s_2)$ niiden eroavaisuuden toisistaan. Lopputuloksen parantamiseksi hyödynnetään William E. Winklerin vuonna 1999 kehittämää menetelmää [32].

Yhteneväisyyden laskemisessa on hyödynnetty alamerkkijonojen laskentamenetelmää, jossa etsitään kahdesta merkkijonosta suurin yhteinen alamerkkijono. Yhteneväisyys on määritelty seuraavasti:

$$Comm(s_1, s_2) = \frac{2 * \sum_i length(maxComSubString_i)}{length(s_1) + length(s_2)}.$$

Eroavaisuuden laskennassa on hyödynnetty H. Hamacherin menetelmää:

$$Diff(s_1, s_2) = \frac{uLen_{s_1} \cdot uLen_{s_2}}{p + (1 - p) \cdot (uLen_{s_1} + uLen_{s_2} - uLen_{s_1} * uLen_{s_2})},$$

missä $p \in [0, \infty)$, ja $uLen_{s_1}$, $uLen_{s_2}$ kuvaavat vastaamattomien alamerkkijonojen pituutta annetuista merkkijonoista s_1 ja s_2. Parametria p voidaan hienosäätää tarvittaessa antamaan merkitystä eri tekijöille. [33]

4.1.5 Vektorietäisyys

Juanz Li et al. RiMOM-työkaluun kehitetty vektorietäisyys (vector-distance) hyödyntää muuta tietoa ontologiassa kuin pelkästään entiteetin nimeä, kuten muutosetäisyyden laskennassa. Erona muihin vektorietäisyyslaskentamenetelmissä RiMOM-työkalun vektorietäisyydessä vektorin sisältö rakennetaan dynaamisesti ontologian eri vastaavuusominaisuuksien avulla. [14]

Jokaiselle entiteetille *e* ontologiassa *O*, tietoa tarkastellaan dokumenttina *D(e)*. *D(e)*:n sisältämät sanat muokataan muuttamalla sanat perusmuotoon ja poistetaan turhat sanat, kuten propositiot (stop word removal). Konseptientiteeteille määritellään $e \in C$:

$$D(e) = \left\{ \left| w_i \in \begin{Bmatrix} (w_1, count(w_1)), \dots, (w_n, count(w_n)) \\ \bigcup \\ Meta(e) \cup_{e_j \in Hier(e)} Meta(e_j) \\ \bigcup \\ \cup_{e_j \in Rest(e)} Meta(e_j) \cup_{e_j \in Inst(e)} Meta(e_j) \end{Bmatrix} \right. \right\}.$$

Konseptientiteetit *D(e)*:ssa sisältävät kaiken meta-datan konseptissa *e*, eli sen ominaisuudet, alakonseptit ja instanssit. Vastaavasti määritellään ominaisuusentiteetille $f \in C$:

$$D(f) = \left\{ \left| w_i \in \begin{Bmatrix} (w_1, count(w_1)), \dots, (w_n, count(w_n)) \\ \bigcup \\ Meta(f) \cup_{f_j \in Hier(f)} Meta(f_j) \cup_{f_j \in Inst(f)} Meta(f_j) \\ \bigcup \\ \cup_{f_j \in Doma(f)} Meta(f_j) \cup_{f_j \in Rang(f)} Meta(f_j) \end{Bmatrix} \right. \right\}.$$

Ominaisuusentiteetit D(f):ssa sisältävät kaiken meta-datan ominaisuudessa *f*, eli konseptit jotka liittyvät ominaisuuteen *f* ja *f*:n instanssit. [14]

Määritelmissä *Meta(e)* tai *Meta(f)* on lista sanoja kuvaamaan metadataa entiteetissä *e* tai *f*, kuten nimi, nimilappu ja kommentti. *Hier(e)* kuvaa alakonseptien joukkoa konseptissa *e* ja tai *Hier(f)* joukkoa alaominaisuuksia ominaisuudessa *f*. *Inst(e)* on joukko instansseja konseptissa *e*, *Inst(f)* ja joukko instansseja ominaisuudessa *f*. *Rest(e)* on joukko ominaisuuksia ja konsepteja, missä jokainen ominaisuus on ominaisuus konseptissa *e*, ja jokainen konsepti on käytetty kuvaamaan konseptia *e*. *Doma(f)* on joukko konsepteja joilla on ominaisuus *f*. *Rang(f)* on joukko konsepteja, joiden instansseilla voi olla ominaisuus *f*. [14]

Vastaavuuden laskemiseksi rakennetaan painotettu ominaisuusvektori (weighted feature vector) käyttäen kaavaa $tf \cdot idf$, missä tf_i on sanan w_i toistuvuus *D(e)*:ssa, mikä on kuvattu ($count(w_i)$:nä, missä idf on käänteinen w_i sanojen lukumäärälle. Tällöin jokainen entiteetti lähdeontologiassa O_1 ja kohdeontologiassa O_2 on muunnettu vastaaviksi painotetuiksi ominaisuusvektoreiksi $V(e_1)$ ja $V(e_2)$. Entiteettien e_1 ja e_2 vastaavuuden on laskettu kahden vektorin kosinilauseella (cosine). [14]

4.2 Rakennepohjainen linjaus

Rakennepohjaisilla linjausmenetelmillä etsitään vastaavuuksia ontologioiden rakenteista. Menetelmän pohja-ajatuksena on: jos kaksi entiteettiä ontologioista O_1 ja O_2 ovat vastaavat toisiinsa nähden, kasvaa niihin liittyvien entiteettien vastaavuus entisestään. Eli ontologioiden entiteettien vastaavuus vaikuttaa myös entiteettien naapureiden vastaavuuteen. [14]

Rakennepohjaisena linjausmenetelmänä LogMap-työkalusta tunnistettiin intervalli-nimiöintiskeema. Vastaavuuden ylikuormittaminen on rakennepohjainen menetelmä, jota käytetään RiMOM-työkalussa.

4.2.1 Intervallinimiöintiskeema

Intervallinimiöintiskeeman (interval labeling scheme) on havaittu olevan tehokas laskettaessa tyypillisiä kyselyitä suurissa luokkahierarkioissa [34; 35]. Sitä käytetään *is-a* suhteiden indeksoimiseen ontologian graafirakenteessa.

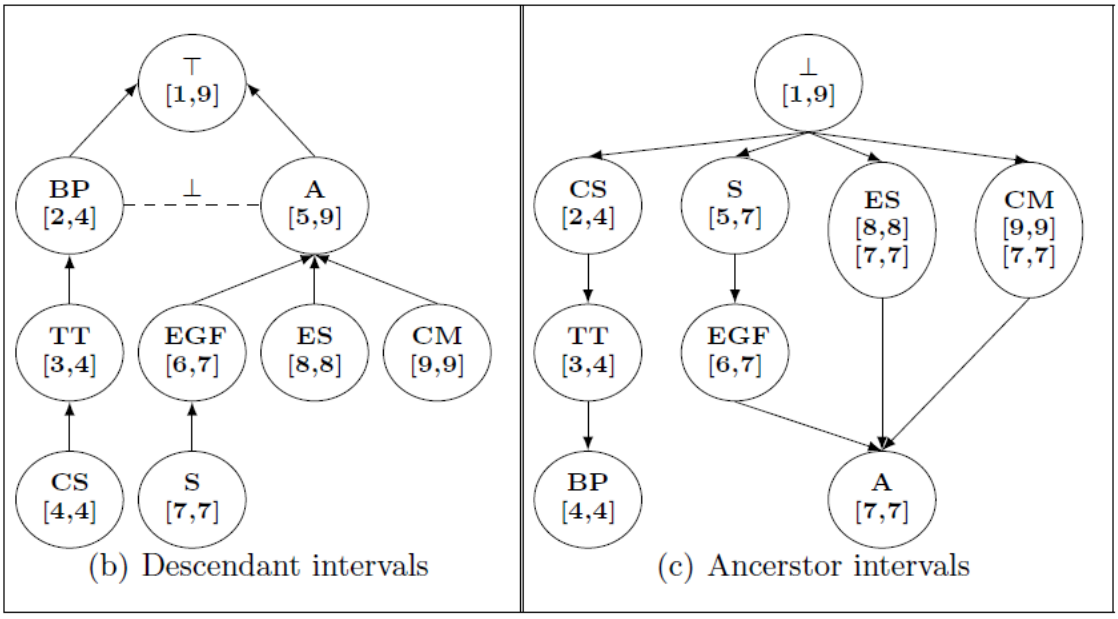
Intervallinimiöintiskeemassa ontologiahierarkiaa käsitellään kahtena suunnattuna asyklisena graafina, eli DAG:na (directed acyclic graph): jälkeläisten (descendants) suhteita kuvaava DAG ja vastaavasti vanhempien (ancestors) suhteita kuvaava DAG. Jokainen ontologiassa nimetty luokka *C* on kuvattu solmuna molemmissa DAG:ssa ja siihen on liitetty seuraavat tiedot:

- Jälkeläisten esijärjestysnumero (descendants preorder number): *predesc(C)* on järjestys missä C:ssä vieraillaan käyttäen syvyys-ensin hakualgoritmia jälkeläisten DAG:ssa
- Vanhempien esijärjestysnumero (ancestors preorder number): *peranc(C)*
- Topologinen järjestys (topolocigal order): syvin jälkeläisten DAG:hen liittyvä taso
- Jälkeläisten intervalli (descendants interval): tieto C:n jälkeläisistä on mallinnettu käyttäen intervallia [*predesc(C), maxpredesc(C)*], missä *maxpredesc(C)* on korkein C:n lasten esijärjestysnumero jälkeläisten DAG:ssa
- Vanhempien intervalli (ancestors interval): tieto C:n vanhemmista on mallinnettuna käyttäen intervallia [*predesc(C), maxpreanc(C)*], missä *maxpreanc(C)* on korkein C:n vanhempien esijärjestysnumero vanhempien DAG:ssa. [21]

Kuvassa 4.1 on esitelty esimerkki intervallinimiöintiskeeman toiminnasta. Kuvan a-kohdassa esitellään ote NCI-ontologiasta (The National Cancer Institute Thesaurus). Kuvan b-kohdassa on jälkeläisten intervallit ja c-kohdassa vanhempien intervallit. B- ja c-kohdan graafeissa solmujen arvot ovat lyhenteitä seuraavasti: BP=BiologicalProcess, A=Anatomy, TT-TransmembraneTransport, CM=CellularMembrane, EGF=ExocrineGlandFluid, CS=CellularScretion, ES=ExocrineSystem ja S=Smegma.

$$\text{Anatomy} \sqsubseteq \neg \text{BiologicalProcess}$$
$$\text{TransmembraneTransport} \sqsubseteq \exists \text{BP\_hasLocation.CellularMembrane}$$
$$\exists \text{BP\_hasLocation.} \top \sqsubseteq \text{BiologicalProcess}$$
$$\top \sqsubseteq \forall \text{BP\_hasLocation.Anatomy}$$
$$\text{CellularSecretion} \sqsubseteq \text{TransmembraneTransport}$$
$$\text{ExocrineGlandFluid} \sqsubseteq \exists \text{AS\_hasLocation.ExocrineSystem}$$
$$\top \sqsubseteq \forall \text{AS\_hasLocation.Anatomy}$$
$$\exists \text{AS\_hasLocation.} \top \sqsubseteq \text{Anatomy}$$
$$\text{Smegma} \sqsubseteq \text{ExocrineGlandFluid}$$
$$\text{ExocrineGlandFluid} \sqcap \text{ExfoliatedCells} \sqsubseteq \text{Smegma}$$

(a) NCI ontology fragment



Kuva 4.1 Esimerkki intervallinimiöintiskeeman toiminnasta [21]

Intervallinimiöintiskeeman avulla on mahdollista suorittaa tehokkaita kyselyitä taksonomisista suhteista. Esimerkiksi kysely, jossa selvitetään onko ”Smegma” luokan ”Anatomy” alaluokka toteutuu tarkistamalla jos *predesc(S)* sisältyy jälkeläisten intervalliin *[predesc(A), maxpredesc(A)]*. Havaitaan, että *predesc(S)* on 7 ja jälkeläisten intervalli on [5,9]. [21]

4.2.2 Vastaavuuden ylivuotaminen

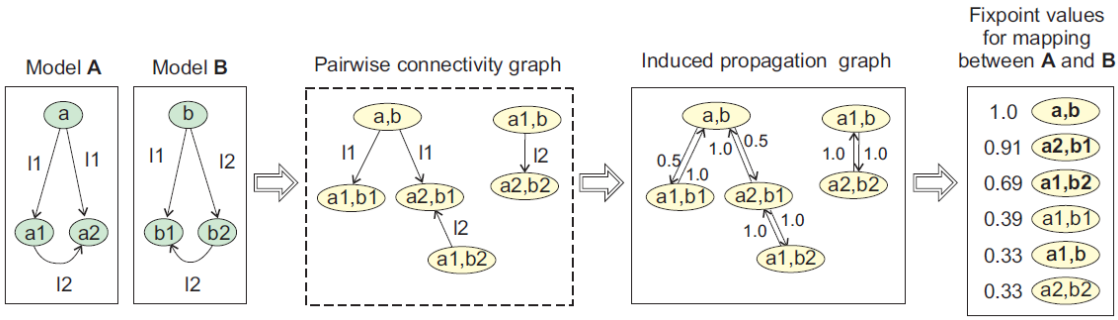
Vastaavuuden ylivuotamisen (similarity flooding) algoritmi jakautuu kahteen prosessiin: parittaisen liitettävyysgraafin, eli PCG:n (pairwise connectivity graph), rakentamiseen ja vastaavuuden johtamiseen (similarity propagation). Jokainen linjattava ontologia esitetään suunnattuna nimettynä graafina eli DLG:nä (directed labeled graph). Jokainen reuna DLG:ssä on kuvattu triplettinä (*s*,*p*,*o*), jossa *s* ja *o* ovat lähde- ja kohdesolmuja, ja *p* on reunan nimilappu. DLG:n muuttaminen PCG:ksi määritellään:

$$\big((x,y),p,(x',y')\big) \in PCG(A,B) \Leftrightarrow (x,p,y') \in A,$$

ja

$$(x,p,y') \in B.$$

Kuvassa 4.2 on kuvattu datamallit A ja B, joiden vastaavuuksia lasketaan kiintopistelaskennalla (fixpoint computation). Jokainen solmu graafissa on elementti A:n ja B:n karteesisesta tulosta. Näitä solmuja kutsutaan linjauspareiksi.



Kuva 4.2 Esimerkki vastaavuuden ylivuotamisalgoritmin toiminnasta [36]

Esimerkiksi voimme tarkastella pareja (a,b) ja (a_1,b_1). Jos a on vastaava kuin b, niin todennäköisesti a_1 on jotakuinkin vastaava kuin b_1. Todisteet päätelmälle löytyvät l_1 nuolilta (edge), jotka yhdistävät solmut a ja a_1 graafissa A ja b ja b_1 graafissa B. Solmupareja (a,b) ja (a_1,b_1) kutsutaan naapureiksi ja niitä yhdistää l_1 PCG:ssä. [36]

Jokaista PCG:n nuolta kohden johdettu vastaavuusgraafi (induced propagation graph) malleille A ja B sisältää lisänuolen vastakkaiseen suuntaan. Nuolien vierellä kuvatut painoarvot, eli vastaavuuskertoimet, kuvaavat kuinka suuri vastaavuus linjauspareilla on suhteessa niiden naapureihin ja takaisin. Painoarvon lukuarvo saa arvoja välillä [0,1]. Kuvasta voidaan havaita, että parista (a_1,b) lähtee täsmälleen yksi l_2 nuoli. Tässä tilanteessa määrittelemme vastaavuuskertoimeksi $w((a_1,b),(a_2,b_2))$ arvon 1.0. Kertoimen ollessa 1.0 voimme todeta, että vastaavuus solmujen a_1 ja b välillä on täysin sama kuin mitä se on solmujen a_2 ja b_2 välillä. Vastaavasti voidaan havaita, että parista (a,b) lähtee kaksi l_1 nuolta. Tällöin painoarvo 1.0 jakautuu tasaisesti $w\big((a,b),(a_1,b_1)\big) = 0.5$ ja $w\big((a,b),(a_2,b_1)\big) = 0.5$. [36]

Vastaavuuden ylikuormittamisen algoritmi pohjautuu iteratiiviseen kiintopistelaskentaan, jossa σ kuvaa linjausta. Olkoon $\sigma(x, y) \geq 0$ vastaavuus solmuille $x \in A$ ja $y \in B$. σ^i kuvaa linjausta i iteraation jälkeen. Linjaus σ^0 kuvaa alustavaa vastaavuutta A:n ja B:n solmujen välillä. Esimerkissä oletetaan, että alustavaa linjausta A:n ja B:n välillä ei ole, jolloin $\sigma^0 = 1.0$ kaikille $(x, y) \in A \times B$. Jokaisella iteraatiolla linjausarvot σ linjausparille (x, y) on lisätty naapuriparien linjausarvolla kerrottuna niiden linjauskertoimella. Esimerkiksi ensimmäisen iteraation jälkeen $\sigma^1(a_1, b_1) = \sigma^0(a_1, b_1) + \sigma^0(a, b) \cdot 0.5 = 1.5$. Lopuksi kaikki arvot normalisoidaan, eli jaetaan jokaisen iteraation suurimmalla linjausarvolla. Linjaus σ^{i+1} lasketaan linjauksesta σ^i seuraavasti:

$$\sigma^{i+1}(x, y) = \sigma^i(x, y) +$$

$$\sum_{(a_u, p, x) \in A, (b_u, p, y) \in B} \sigma^i(a_u, b_u) \cdot w((a_u, b_u), (x, y) +$$

$$\sum_{(x, p, a_v) \in A, (y, p, b_v) \in B} \sigma^i(a_v, b_v) \cdot w((a_v, b_v), (x, y).$$

Iteraatioita jatketaan niin kauan kunnes jäljellä olevan vektorin euklidinen pituus $\Delta(\sigma^n, \sigma^{n-1})$ on vähemmän kuin ε tai erikseen määrätyn iteraatiomäärän jälkeen. [36]

5 TAPAUSTUTKIMUS

Tässä luvussa määritellään tapaustutkimus ja esitellään tutkimuksen tulokset. Tapaustutkimuksen tuloksien ja havaintojen pohjalta voidaan vahvistavat kirjallisuuskatsauksen tuloksia ja havaintoja.

5.1 Tapaustutkimuksen määrittely

Tutkimusmenetelmäksi on valittu tapaustutkimus, koska sen avulla saadaan kirjallisuuskatsauksen avulla muodostettuun teoriapohjaan laajempaa näkyvyyttä. Käytännön kokeiluilla voidaan simuloida linjausprosessia kokonaisuutena ja peilata havaintoja teoriapohjan havaintoihin. Tapaustutkimuksen tapaukset on valittu simuloimaan yksinkertaistetusti tunnistettuja skenaarioita, joissa ontologialinjausta voidaan hyödyntää. Tapauksien testiaineisto on valittu siten, että niiden ymmärtämiseksi ei vaadita suurempaa asiantuntemusta. Aineisto on myös valittu tukemaan muodostettuja simuloitavia skenaarioita. Tapaustutkimuksesta esitellään missä testiympäristössä tutkimus on toteutettu ja mitkä aineistot on valittu mihinkin tapaukseen.

5.1.1 Testiympäristö

Ensisijaiseksi testiympäristöksi valmisteltiin kannettava työasema, johon asennettiin käyttöjärjestelmäksi 32-bittinen Ubuntu 12.04.4 LTS. Työasemassa on 3072 Mt RAM muistia ja 1,86 GHz kahden ytimen prosessori.

Toissijaiseksi testiympäristöksi valmisteltiin pöytäkone. Pöytäkoneessa on 16,0 Gt RAM muistia, 3,50 GHz kuuden ytimen prosessori ja käyttöjärjestelmänä 64-bittinen Windows 7 Home Premium Service Pack 1.

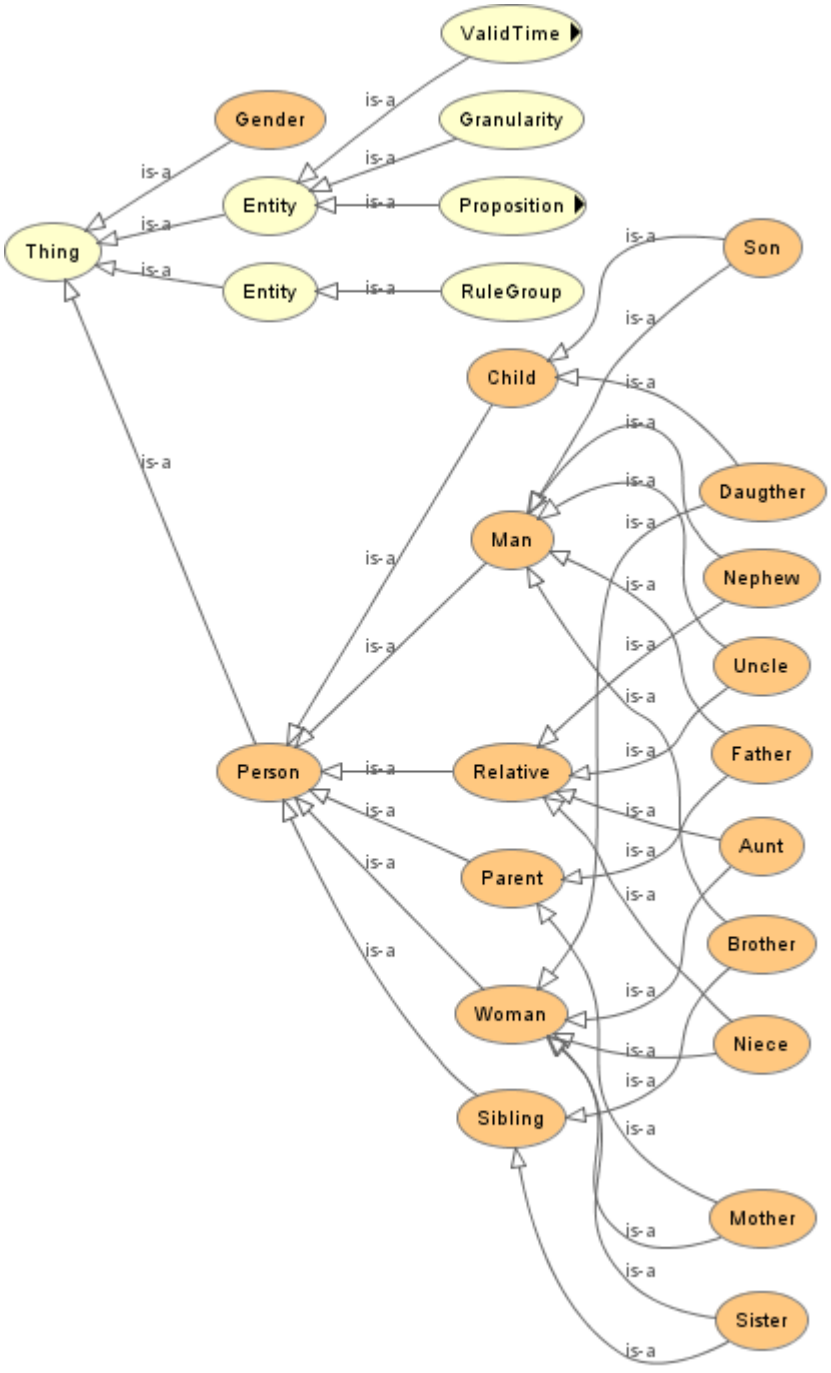
Toissijaiseen testiympäristöön asennettiin 64-bittinen Protégé Desktop 4.3 [37], jossa on OWL 2.0 tuki. Protégén visualisointiominaisuuksien hyödyntämiseksi testausympäristöön asennettiin myös Graphviz 2.38 [38]. Protégéympäristöä käytetään tulos- ja lähdeontologioiden visualisoimiseksi.

5.1.2 Tapaus 1: Eri ontologiat samasta aiheesta

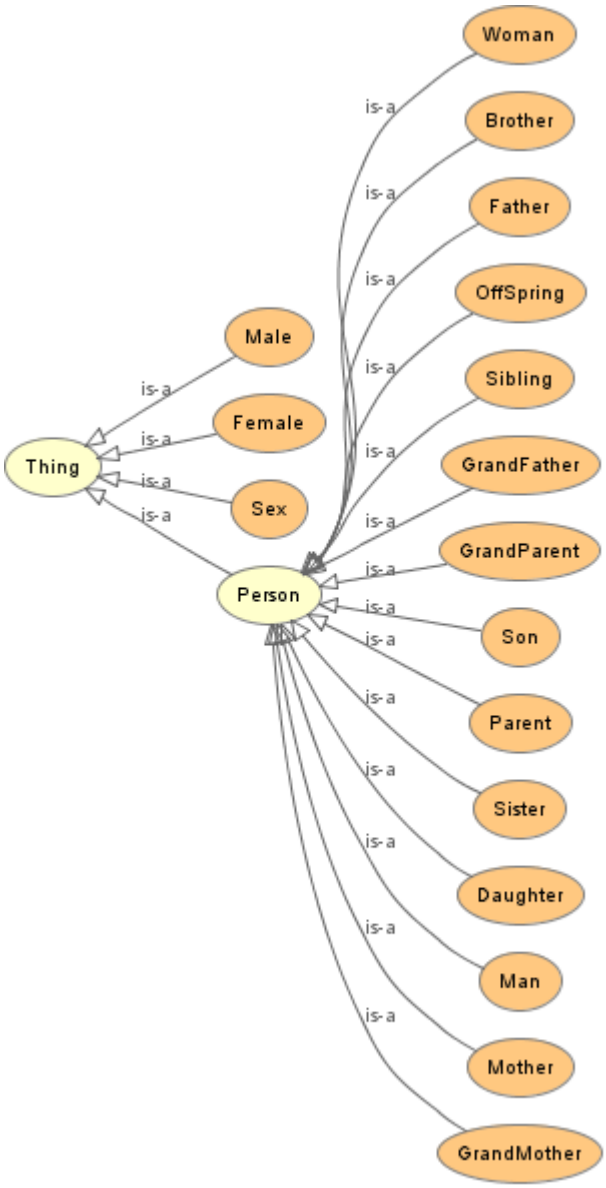
Tapaus, jossa eri ontologiat on valittu samasta aiheesta kuvaa tilannetta, jossa identtiset tai lähes identtiset käsitteistöt halutaan käyttöön samassa järjestelmässä. Esimerkiksi yritysmaailmassa samaa käsitettä voidaan kuvata eri sanoilla.

Tapauksen testiaineistoksi valittiin perhesanastoa kuvaavat ontologiat. Kuvassa 5.1 on kuvattu Christine Golbreichin [39] toteuttama perheontologia, jossa on kuvattu perheen

sisäisiä suhteita. Kuvassa 5.2 on kuvattu Matthew Horridgen [40] sukupolviontologia, jossa pääpaino on suvun rooleissa.



Kuva 5.1 Perheontologia



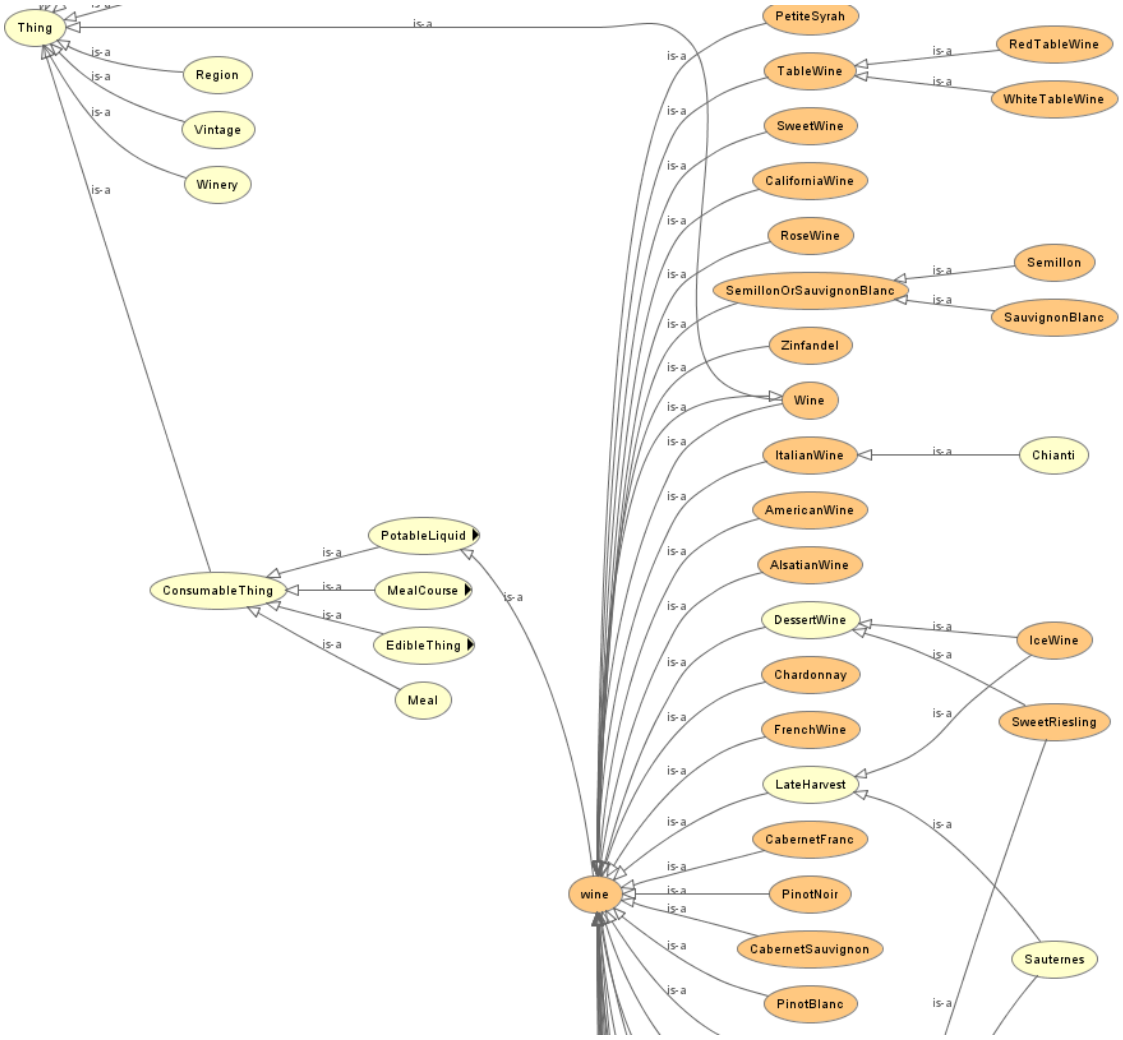
Kuva 5.2 Sukuontologia

5.1.3 Tapaus 2: Eri ontologiat samasta aihealueesta

Tapaus, jossa eri ontologiat on valittu samasta aihealueesta kuvaa tilannetta, jossa kaksi eri toisiaan lähellä olevaa ontologiaa halutaan yhteen samaan käsitteistöön. Tällainen tilanne voi olla esimerkiksi verkkopalvelussa, jossa käyttöönotetaan uusi ominaisuus, joka käyttää erilaista termistöä.

Tapauksen testiaineistoksi on valittu olut- ja viiniontologiat. Oluet ja viinit ovat molemmat alkoholipitoisia juomia, mutta niiden käsitteistö on pääsääntöisesti eri. Kuvassa 5.3 on ote olutontologiasta (kokonainen kuva liitteessä 2) [41]. Kuvassa 5.4 on ote havainnollistetusta viiniontologiasta (kokonainen kuva liitteessä 1) [42].

Kuva 5.3 Ote Olutontologiasta



Kuva 5.4 Ote viiniontologiasta

5.2 Tapaustutkimuksen tulokset

LogMap:n käyttö vaatii java-ympäristön toimiakseen. Ympäristön asennus ei vaadi muuta kuin asennuspaketin purkamisen hakemistoon. LogMap:n toimintaa voi hienosäätää muokkaamalla työkalun raja-arvoja tiedostossa parameters.txt. Testitapaukset on suoritettu oletusarvoilla.

Työkalun käynnistäminen tapahtuu ajamalla komento java –jar logmap2_standalone.jar ja syöttämällä viisi parametria työkalulle käsiteltäväksi:

- MATCHER-parametri käynnistää LogMap:n linjaustyökalun.
- Lähdeontologian IRI-tunniste (International Resource Identifier), eli sijainti
- Kohdeontologian IRI-tunniste.
- Tulostetiedostojen kohdehakemisto.
- Tarkistetaanko lopputulos HermiT-työkalulla, arvo true tai false. [43]

LogMap tulostaa mahdolliset virheilmoitukset ja lopettaa toimintansa kun linjaus on suoritettu. Määriteltyyn kohdehakemistoon tallennetaan lopputuloksena viisi eri tiedostoa:

- Logmap2_mappings.owl.
- Logmap2_mappings.rdf.
- Logmap2_mappings.txt.
- Module1_overlapping_logmap2.owl.
- Module2_overlapping_logmap2.owl.

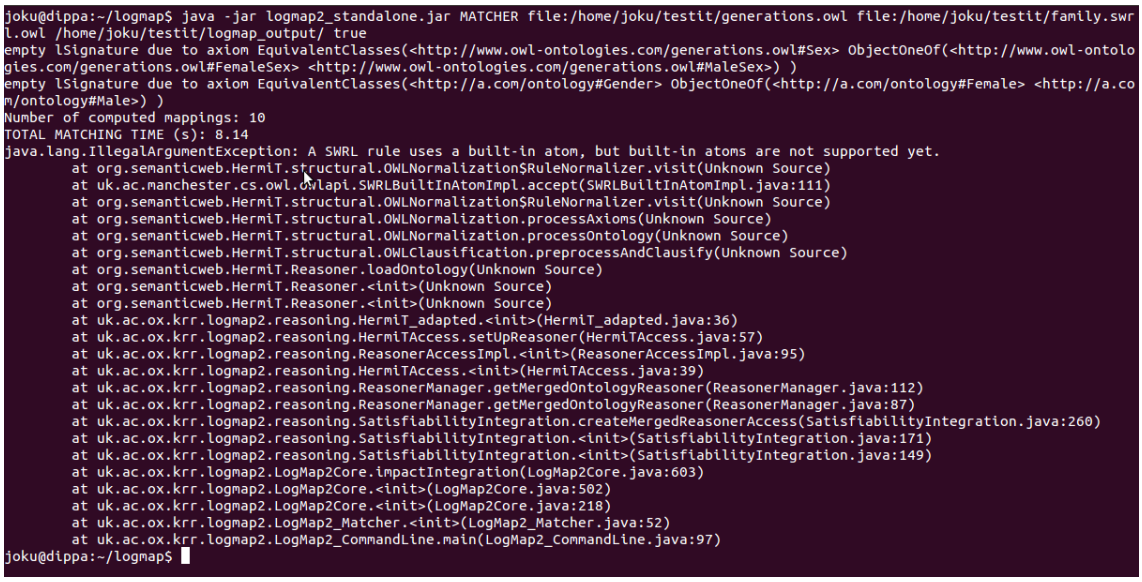
Ensimmäiset kaksi tiedostoa sisältävät linjaustuloksen OWL- ja RDF-formaatissa. Kolmas tiedosto sisältää tuloksen tekstimuodossa ja kaksi viimeistä sisältävät päällekkäiset käsitteet. Tärkeimmät tulokset tutkimuksen kannalta ovat tulosteiden ensimmäinen ja kolmas tiedosto. Kappaleessa 3.1 esiteltyä prosessia tarkasteltaessa voidaan havaita, että ulostulotiedostoista kolme ensimmäistä (logmap2_mappings.owl, logmap2_mappings.rdf ja logmap2_mappings.txt) ovat prosessin linjaustulos M ja vastaavasti tiedostot module1_overlapping_logmap2.owl ja module2_overlapping_logmap2.owl ovat O_1' ja O_2'.

Tekstimuotoinen tulostiedosto on ryhmitelty siten, että jokainen linjaustulos on omalla rivillään ja jokainen linjaustulos on eritelty viiteen eri arvoon ”|” merkillä eroteltuna. Ensimmäiset kaksi arvoa kuvaavat lähde- ja kohdeontologiasta löydettyjä vastaavia entiteettejä. Kolmas arvo kertoo entiteettien relaation, neljäs luotettavuusarvon ja viides suhteen tyypin. Luotettavuusarvo on laskettu ISUB-työkalulla, jonka toiminta on esitelty kappaleessa 4.1.4. Tyyppi voi olla arvoltaan:

- CLS: luokka
- OPROP: objektin ominaisuus
- DPROP: dataominaisuus
- INST: instanssi. [44]

5.2.1 Tapaus 1

Tapauksen testisuoritus suoritettiin valitsemalla lähdeontologiaksi sukuontologia generations.owl ja kohdeontologiaksi perheontologia family.swrl.owl. Kuvassa 5.5 voidaan nähdä testisuorituksen ajo komentorivillä. Testisuoritus tuotti virheilmoituksia, jotka viittaavat SWRL-sääntöihin, jotka eivät ole tuettuja. Virheilmoitukset eivät vaikuttaneet testituloksien tarkasteluun, eivätkä lopputulokseen. Testisuoritus toteutui noin kahdeksassa (8) sekunnissa.



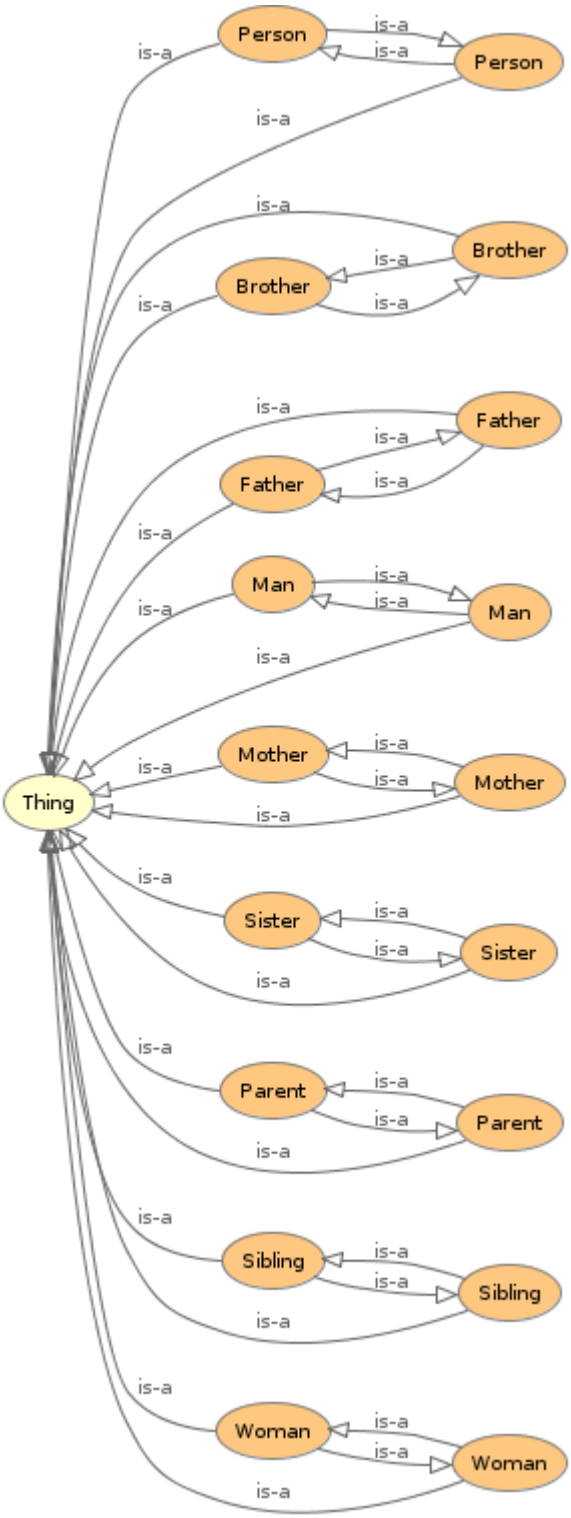
Kuva 5.5 Kuvakaappaus testisuorituksesta

Lopputuloksena syntyneen logmap2_mappings.txt tiedosto sisälsi kymmenen (10) riviä:

```
http://www.owl-
ontologies.com/generations.owl#Parent|http://a.com/ont
ology#Parent|=|0.7|CLS
http://www.owl-
ontologies.com/generations.owl#Mother|http://a.com/ont
ology#Mother|=|0.7|CLS
http://www.owl-
ontologies.com/generations.owl#Woman|http://a.com/onto
logy#Woman|=|0.7|CLS
http://www.owl-
ontologies.com/generations.owl#Brother|http://a.com/on
tology#Brother|=|0.7|CLS
http://www.owl-
ontologies.com/generations.owl#Father|http://a.com/ont
ology#Father|=|0.7|CLS
http://www.owl-
ontologies.com/generations.owl#Person|http://a.com/ont
ology#Person|=|0.7|CLS
http://www.owl-
ontologies.com/generations.owl#Man|http://a.com/ontolo
gy#Man|=|0.7|CLS
http://www.owl-
ontologies.com/generations.owl#Sibling|http://a.com/on
tology#Sibling|=|0.7|CLS
```

```
http://www.owl-
ontologies.com/generations.owl#Sister|http://a.com/ont
ology#Sister|=|0.7|CLS
http://www.owl-
ontologies.com/generations.owl#FemaleSex|http://a.com/
ontology#Female|=|0.88|INST
```

Riveistä on havaittavissa, että yhdeksän kymmenestä vastaavasta entiteetistä on tyypiltään luokka ja luotettavuusarvoltaan 0.7. Viimeinen entiteettipari on tyypiltään instanssi ja luotettavuusarvoltaan 0.88. Jokainen entiteettipari on relaatioarvoltaan kuvattu yhtäsuuruudella ”=”. Tuloksista voidaan havaita vastaavuuksia ontologialinjauksen formaaliin määritelmään, joka on esitelty kappaleessa 2.3.1. Määritelmän nelituplan entiteetit e_{i1} ja e_{i2} ovat rivin kaksi ensimmäistä URI-tunnistetta ”|” merkillä eroteltuina. Linjaustyyppi $relation_i$ on rivin kolmas alkio ja luotettavuusarvo con_i rivin neljäs alkio. Teoriasta poiketen rivillä on viides tekijä, entiteettiparin tyyppi. Tuloksen ontologia on havainnollistettu kuvassa 5.6.



Kuva 5.6 Linjaustulos perhe- ja sukuontologiasta

5.2.2 Tapaus 2

Tapauksen testisuoritus suoritettiin valitsemalla lähdeontologioiksi olutontologia beer.owl ja wine.rdf. Kuvassa 5.7 voidaan nähdä testisuorituksen ajo komentorivillä. Testisuoritus ei tuottanut virheilmoituksia ja suoritus toteutui noin 15 sekunnissa.

Kuva 5.7 Kuvakaappaus testisuorituksesta

Lopputuloksena syntyneen logmap2_mappings.txt tiedosto sisälsi yhden rivin:

```
file:/home/joku/testit/beer.owl#Region|http://www.w3.o
rg/TR/2003/PR-owl-guide-20031209/wine#Region|=|0.7|CLS
```

Rivistä on havaittavissa, että annetuissa olut- ja viiniontologioissa vastaava entiteetti on alue (region), jossa juomaa valmistetaan. Linjauksen tyyppi on luokka ja luotettavuus arvo 0.7. Muodostunut entiteettipari on relaatioarvoltaan kuvattu yhtäsuuruudella ”=”. Tuloksista voidaan havaita vastaavuuksia ontologialinjauksen formaaliin määritelmään, joka on esitelty kappaleessa 2.3.1. Määritelmän nelituplan entiteetit e_{i1} ja e_{i2} ovat rivin kaksi ensimmäistä URI-tunnistetta ”|” merkillä eroteltuina. Linjaustyyppi $relation_i$ on rivin kolmas alkio ja luotettavuusarvo con_i rivin neljäs alkio. Teoriasta poiketen rivillä on viides tekijä, entiteettiparin tyyppi. Tuloksen ontologia on havainnollistettu kuvassa 5.8.

Kuva 5.8 Linjaustulos olut- ja viiniontologiasta

5.3 Tapaustutkimuksen havainnot

Ensimmäisen testisuorituksen tuloksesta voidaan havaita kuinka monta vastaavaa entiteettiä löydetään lähes identtisissä ontologioissa. Tuloksen hyödyntäminen voidaan nähdä yrityssovellutuksissa, joissa integroidaan vastaavia sanastoja hyödyntämiä järjestelmiä ja voidaan tunnistaa päällekkäisiä käsitteitä. Turhan päällekkäisyyden tunnistaminen parantaa järjestelmän optimointia. Mielenkiintoisin havainto ensimmäisessä testissä on se, että LogMap-työkalun linjaus ei tunnistanut toisessa ontologiassa esiintyvää kirjoitusvirhettä ”Daugther”, joka olisi pitänyt linjautua kieliopillisen menetelmän avulla käsitteen

”Daughter” kanssa. Havaittu virhe johtunee parametrien valinnasta. Virheen korjaamiseksi pitäisi iteroida eri parametrivalintoja ja löytää sopiva tasapaino lopputuloksen saavuttamiseksi. Tämä osoittaa, että tuloksien tarkistaminen jälkikäteen on tärkeää virheiden tunnistamiseksi.

Toisen testisuorituksen tuloksesta voidaan havaita hyödyntämismahdollisuus verkkopalvelussa. Oletetaan, että käytössä on verkkopalvelu, joka sisältää tietoa oluista. Verkkopalveluun halutaan tuoda mukaan tietämystä viineistä, joten linjaustuloksen avulla on mahdollista löytää sijainnin perusteella siihen liittyvät olut- ja viinitilat ilman erillistä hakua.

Molempien testisuorituksien tulosontologiasta havaittiin mielenkiintoinen ominaisuus. Tulosontologioissa viitataan kehittäjien omaan standardisoimattomaan sanastoon:

```
<owl:Ontology
rdf:about="http://www.cs.ox.ac.uk/isg/projects/LogMap/
mappings.owl"/>
```

Esitellyn owl:Ontology viittavaa mappings.owl tiedostoa ei enää löydy esitellystä osoitteesta, joten sen käyttötarkoituksen arviointi ei ole mahdollista. Muutoin tuloksissa on käytetty W3C:n standardisoiman OWL:n ominaisuuksia kattavasti, esimerkkiksi:

```
<owl:Class rdf:about="http://a.com/ontology#Brother">
    <owl:equivalentClass rdf:resource="http://www.owl-
ontologies.com/generations.owl#Brother"/>
</owl:Class>
<owl:Axiom>
     <measure
rdf:datatype="http://www.w3.org/2001/XMLSchema#double"
>0.7030000000000001</measure>
     <relation
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>=</relation>
     <entity2
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>http://a.com/ontology#Brother</entity2>
     <entity1
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>http://www.owl-
ontologies.com/generations.owl#Brother</entity1>
     <owl:annotatedSource
rdf:resource="http://a.com/ontology#Brother"/>
```

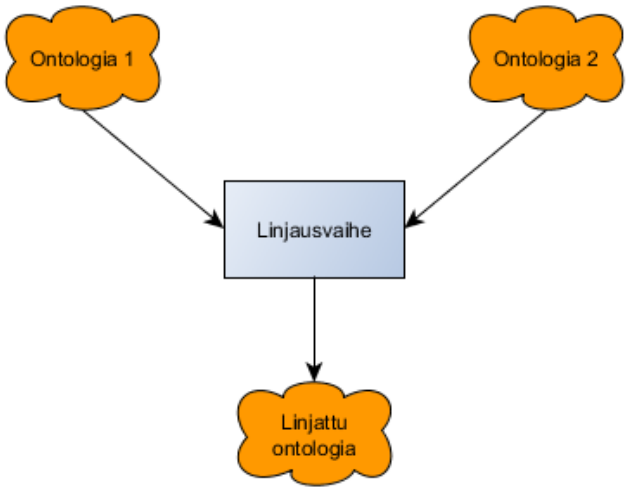
```
      <owl:annotatedTarget
rdf:resource="http://www.owl-
ontologies.com/generations.owl#Brother"/>
      <owl:annotatedProperty
rdf:resource="http://www.w3.org/2002/07/owl#equivalent
Class"/>
</owl:Axiom>
```

Linjaustuloksesta voidaan havaita, että kappaleessa 2.3.1 esitelty ontologialinjauksen formaali määritelmä toteutuu linjaustuloksessa. Määritelmän nelituplan entiteetti e_{i1} on tuloksessa <entity1> ja vastaavasti e_{i2} on <entity2>. Luotettavuusarvo con_i on <measure> ja linjaustyyppi $relation_i$ on <relation>. Linjaustyyppi on kuvattu yhtäsuuruudella ”=”, joten siitä voidaan päätellä, että linjaustyyppi on teorian mukaisesti *exact*. Dokumentaatiossa ei ole tarkennettu miten muut linjaustyypit *narrower*, *broader* ja *overlap* ovat kuvattu.

6 YHTEENVETO

Luvussa suoritetaan tutkimuksessa toteutettujen kirjallisuuskatsauksen ja tapaustutkimuksen tuloksien ja havaintojen pohjalta yhteenveto. Yhteenvedossa määritellään yleinen linjausprosessimalli. Luvussa toteutetut vuokaaviot on toteutettu yEd graafityökalulla (versio 3.13) [45].

Tarkastelemalla luvussa neljä esiteltyjä linjaustyökaluja ja niiden linjausprosessien (luku 3) vuokaavioita, voidaan havaita yhtäläisyyksiä ja eroavaisuuksia. Kaikille yhteistä on se, että linjaustyökalu ottaa vastaan kaksi ontologiaa ja tuottaa ulostulona linjatun ontologian. Lopputuloksen saavuttamiseksi jokainen työkalu esittelee oman toteutusmallinsa, jotka ovat itsessään automaattisia ja työkalupohjaisia. Jokaisella työkalulla on omat etunsa ja heikkoutensa. Kuvassa 6.1 on yksinkertaistettu yhteenveto näistä prosesseista. Linjausvaiheen yksityiskohdat eivät ole tässä vaiheessa merkittäviä, koska niiden vaikutus kokonaisprosessissa on valitun linjaustyökalun suoriutumisessa linjaustyössä.



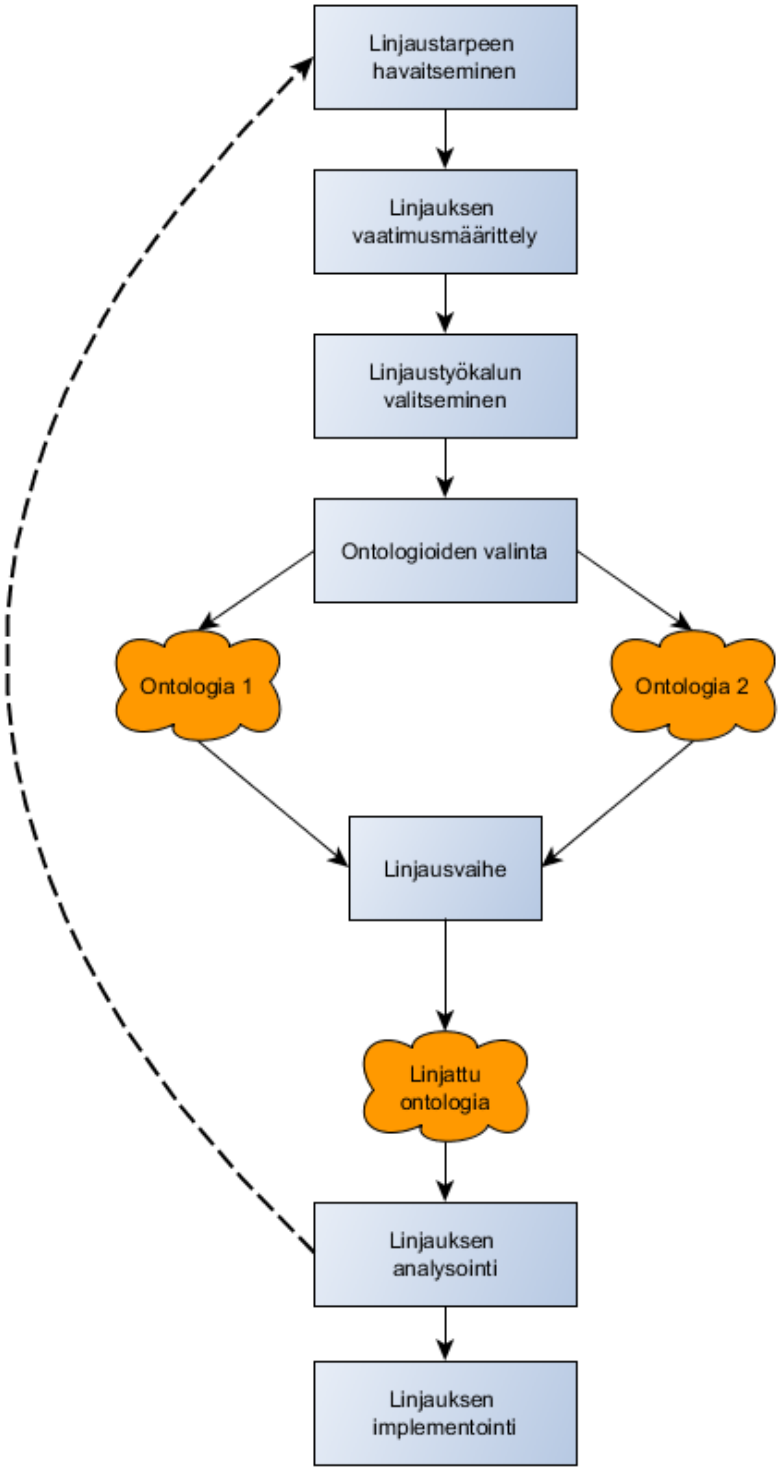
Kuva 6.1 Linjausprosessi yksinkertaistettuna

Esitellyt linjaustyökalujen linjausprosessit keskittyivät linjauslopputuloksen tuottamisen eri vaiheisiin. Kokonaisvaltaista linjausprosessia, joka alkaa linjaustarpeen havaitsemisesta ja jatkuu aina linjaustuloksen implementointiin saakka, ei prosesseissa käsitelty lainkaan. Määriteltäessä yleistä linjausprosessimallia voidaan havaita jo yksinkertaistetusta linjausprosessiversiosta tarpeita ja vaatimuksia yleistä mallia ajatellen:

- Mitkä ovat ontologioiden valintaperusteet?
- Mikä on linjaustyökalun valintaperuste?
- Miten voidaan arvioida linjatun ontologian onnistumista ja laadukkuutta?

6.1 Yleinen linjausprosessimalli

Yleistä linjausprosessimallia määriteltäessä on tarkasteltava kokonaisuutta laajemmalti kuin eri toteutuksien yksityiskohtia. Edellä mainittujen nostojen lisäksi tapaustutkimusta toteuttaessa havaittiin tarpeita yleistä prosessimallia varten. Kuvassa 6.2 on esitelty vuokaaviona yleinen linjausprosessimalli. Linjausprosessimalli on iteratiivinen toteutus.



Kuva 6.2 Yleinen linjausprosessimalli

Ontologialinjauksen toteuttaminen alkaa aina tarpeen havaitsemisesta. Tätä tutkimustyötä toteuttaessa havaitseminen ilmeni tarpeesta tutkia linjausmenetelmiä ja -työkaluja, mutta esimerkiksi verkkopalvelujen kohdalla tarve voi nousta laajennuksesta, joka tuo mukanaan tarpeen laajentaa olemassa olevaa ontologiaa linjaamalla se toiseen. Linjaamistarpeen havaitsemisen jälkeen on suoritettava vaatimusmäärittely linjausprosessille. On pystyttävä tarkastelemaan mitä linjauksella tavoitellaan, jotta myöhemmin voidaan arvioida onnistuttiinko tavoitteessa. Vaatimusmäärittelyn lisäksi on valittava vaatimuksiin vastaava linjaustyökalu, jolla varsinainen linjaus aiotaan suorittaa, sekä on valittava linjattavat ontologiat. Linjauksen lopputulosta on analysoitava suhteessa vaatimusmäärittelyssä luotuihin tarpeisiin ja arvioitava oliko linjaus onnistunut vai nouseeko uusi tarve linjaukselle. Linjauksen onnistumisen arviointi perustuu käytännössä vertaamalla aikaisempiin käyttökokemuksiin saantiarvon (recall) ja tarkkuuden (precision) avulla. Prosessimallin iteratiivisuus jatkuu niin kauan kunnes lopputuloksena saatu linjattu ontologia vastaa määrittelyn vaatimuksia. Tämän jälkeen linjattu ontologia on valmis implementoitavaksi tarpeen mukaisesti.

6.2 Johtopäätökset

Ontologialinjauksella tarkoitetaan prosessia, jonka yhteydessä kahdesta ontologiasta etsitään toisiaan vastaavat käsitteet. Prosessissa otetaan vastaan kaksi linjattavaa ontologiaa ja lopputuloksena saadaan näiden pohjalta muodostetun linjatun ontologian. Linjauksen lopputuloksena muodostettu linjattu ontologia sisältää ainoastaan annetuista ontologioista löydetyt vastaavuudet.

Ontologialinjauksessa hyödynnettävät menetelmät jakautuvat kahteen kategoriaan: kieliopillisiin ja rakenteellisiin linjausmenetelmiin. Kieliopilliset linjausmenetelmät tarkastelevat luokkien (käsitteiden) nimiä ja päättelevät niiden vastaavuutta toisiinsa. Rakenteellisilla menetelmillä voidaan tukea kieliopillisien menetelmien tuloksia antamalla havaintoja käsitteiden rakenteellisista suhteista. Rakenteellisilla menetelmillä voidaan havaita käsitteille ylä- tai alakäsitteitä.

Tässä työssä määritellyssä yleisessä linjausprosessimallissa on huomioitu linjaukseen vaikuttavia tekijöitä linjaustyökalun yksityiskohtien ulkopuolelta. Havaittiin, että varsinainen linjausvaihe on oma yksittäinen vaiheensa, jonka rooli on kokonaisuudessa hyvin pieni. Tärkeämmäksi havaittiin linjauksen vaatimusmäärittely, joka määrittelee kuinka hyvin linjausvaihe onnistui. Linjaustarpeen havaitseminen mahdollistaa vaatimusmäärittelyn käynnistämisen, joka itsessään johtaa linjaustyökalun ja linjattavien ontologioiden valitsemiseen. Lopputuloksena saatavan linjatun ontologian analysointi vaatimusmäärittelyyn peilaten on tärkeää, jotta voidaan olla varmoja, että linjattua ontologiaa voidaan implementoida havaittuun tarpeeseen. Muuten joudutaan aloittamaan linjausprosessi alusta ja tekemään kaikki uudestaan. Lisäksi linjauksen analysoinnissa on pystyttävä ar-

vioimaan linjauksen laadukkuutta, mutta se ei onnistu laskennallisesti ilman referenssiontologiaa. Referenssiontologian puuttuessa asiantuntija rooli analyysivaiheessa korostuu entisestään.

Tässä työssä muodostetussa yleisestä linjausprosessimallista voidaan helposti tunnistaa kuinka suuri ihmisen rooli kokonaisuudessaan on linjausprosessissa. Kaikista prosessimallin vaiheista ainoastaan linjausvaihe on sellainen, joka voidaan toteuttaa automaationa ohjelmallisesti. Kaikki muut vaiheet vaativat ihmisen tekemää valintaa ja määrittelyä. Ontologioiden aihealueen osaaminen on välttämätöntä, jotta voidaan arvioida saavutettua lopputulosta. Ontologialinjauksen onnistuminen on suurilta osin kiinni linjausta suorittavan asiantuntijan valinnoista ja asiantuntemuksesta. Tapauksissa havaittiin tilanne, jossa työkalu ei tunnistanut kirjoitusvirhettä, joka aiheutti puutteellisen lopputuloksen. Tämä korostaa asiantuntijan roolia. Ratkaisuna ongelmatilanteeseen olisi korjata kirjoitusvirhe alkuperäisessä ontologiassa tai parantaa työkaluun määriteltyjä parametreja, jotta se hyväksyisi vastaavat tilanteet. Tämä tosin saattaisi kasvattaa muiden virhetulkintojen määrää, joka entisestään korostaa asiantuntijan roolia linjausprosessissa.

Työssä suoritettujen testien ja tutkimuksien pohjalta voidaan todeta, että ontologialinjaus on vielä kypsymisvaiheessa. Työkaluja on rajoitetusti ja niiden hyödyntäminen on rajallista ilman asiantuntemusta. Ontologioiden yleistyminen on välttämätöntä ontologialinjauksen tarpeen tunnistamiseksi. Tutkimusta ontologioiden hyödyntämiselle ja kehittämiselle tarvitaan, jotta ontologioita aletaan hyödyntämään entisestään verkkopalveluissa. Ontologioiden linjaamisen logiikkaa ja laadukkuuden automaattista analysointia on pystyttävä kehittämään. Yhtenä vaihtoehtona on määritellä avoimia, universaaleja ja standardisoituja ontologioita, joihin ontologialinjaustuloksia vertaamalla voidaan tutkia eri menetelmien ja linjauslogiikoiden toimivuutta. Tulevaisuudessa ontologioiden hyödyntämisen haasteena on varmistaa niihin kohdistuva luotettavuus.

LÄHTEET

[1] Berners-Lee, T., Hendler, J. & Lassila, O. The Semantic Web. [viitattu 1.7.2014]. Saatavissa: http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html.

[2] Berners-Lee, T. The next web. [viitattu 1.7.2014]. Saatavissa: http://www.ted.com/talks/tim_berners_lee_on_the_next_web.

[3] Berners-Lee, T. The year open data went worldwide. [viitattu 1.7.2014]. Saatavissa: http://www.ted.com/talks/tim_berners_lee_the_year_open_data_went_worldwide.

[4] Semantic Web: Linked Data on the Web. [viitattu 1.7.2014]. Saatavissa: http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#%2824%29.

[5] The World Wide Web Consortium (W3C). [viitattu 5.10.2014]. Saatavissa: http://www.w3.org/.

[6] RDF 1.1 Concepts and Abstract Syntax. [viitattu 27.9.2014]. Saatavissa: http://www.w3.org/TR/rdf11-concepts.

[7] RDF Schema 1.1. [viitattu 27.9.2014]. Saatavissa: http://www.w3.org/TR/rdf-schema/.

[8] OWL Web Ontology Language Overview. [viitattu 1.7.2014]. Saatavissa: http://www.w3.org/TR/owl-features/.

[9] OWL 2 Web Ontology Language Document Overview (Second Edition). [viitattu 1.7.2014]. Saatavissa: http://www.w3.org/TR/owl2-overview/.

[10] Gruber, T.,R. A Translation Approach to Portable Ontology Specifications. [viitattu 30.6.2014]. Saatavissa: http://www.dbis.informatik.hu-berlin.de/dbisold/lehre/WS0203/SemWeb/lit/KSL-92-17.pdf.

[11] Gruber, T.R. Toward principles for the design of ontologies used for knowledge sharing? International Journal of Human-Computer Studies 43(1995)5–6, pp. 907-928.

[12] Ontology Alignment : Bridging the Semantic Gap. In: Semantic Web and Beyond, Computing for Human Experience, ISSN 1559-7474 ; 4.; Springer eBooks. Boston, MA 2007, Springer Science+Business Media, LLC. Computer scienceInformation systemsDatabase managementMultimedia systemsArtificial intelligenceElectronic commerceInformation Systems Applications (incl.Internet)Artificial Intelligence (incl. Robotics)Information Systems and Communication ServiceMultimedia Information SystemsElectronic Commerce/e-business.

[13] Xue, X., Wang, Y. & Ren, A. Optimizing ontology alignment through Memetic Algorithm based on Partial Reference Alignment. Expert Systems with Applications 41(2014)7, pp. 3213-3222.

[14] Li, J., Tang, J. & Luo, Q. RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. [viitattu 15.6.2014]. Saatavissa: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4633358.

[15] Euzenat, J. & Shvaiko, P. Ontology Matching. [viitattu 21.9.2014]. Saatavissa: http://link.springer.com/book/10.1007%2F978-3-540-49612-0.

[16] Results of the Ontology Alignment Evaluation Initiative 2006. [viitattu 4.10.2014]. Saatavissa: http://oaei.ontologymatching.org/2006/results/oaei2006.pdf.

[17] Van Rijsberge, C.J. Information retrieval. [viitattu 7.9.2014]. Saatavissa: http://www.dcs.gla.ac.uk/Keith/Preface.html.

[18] Lambrix, P. SAMBO—A system for aligning and merging biomedical ontologies. Web Semantics: Science, Services and Agents on the World Wide Web, 2006, Vol.4(3), pp.196-206 4(2006)3, pp. 196-206.

[19] Ontology Alignment Evaluation Initiative. [viitattu 15.6.2014]. Saatavissa: http://oaei.ontologymatching.org/.

[20] Results of the Ontology Alignment Evaluation Initiative 2013. [viitattu 28.9.2014]. Saatavissa: http://oaei.ontologymatching.org/2013/results/oaei2013.pdf.

[21] Jiménez-Ruiz, E. & Grau, B.,Cuenca LogMap: Logic-based and Scalable Ontology Matching. [viitattu 15.6.2014]. Saatavissa: http://www.cs.ox.ac.uk/isg/projects/LogMap/papers/paper_ISWC2011.pdf.

[22] Final results of the Ontology Alignment Evaluation Initiative 2011. [viitattu 28.9.2014]. Saatavissa: http://oaei.ontologymatching.org/2011/results/oaei2011.pdf.

[23] Results of the Ontology Alignment Evaluation Initiative 2012. [viitattu 28.9.2014]. Saatavissa: http://oaei.ontologymatching.org/2012/results/oaei2012.pdf.

[24] Morant, A. Extending and Optimizing an Ontology Matching System. [viitattu 7.9.2014]. Saatavissa: http://www.cs.ox.ac.uk/isg/projects/LogMap/papers/Master_thesis_anton_morant.pdf.

[25] Faloutsos, C. Fast discovery of connection subgraphs. Conference on Knowledge Discovery in Data: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining; 22-25 Aug.2004 (2004) pp. 118.

[26] Wang, P. & Xu, B. LILY: The Results for the Ontology Alignment Contest OAEI 2007. [viitattu 15.6.2014]. Saatavissa: http://disi.unitn.it/~p2p/OM-2007/6-oLILY_SEU_OAEI07.pdf.

[27] Wang, P. Lily — An Ontology Mapping System. [viitattu 15.6.2014]. Saatavissa: http://cse.seu.edu.cn/people/pwang/lily.htm.

[28] Nguyen, K., Ichise, R. & Le, B. Interlinking Linked Data Sources Using a Domain-Independent System. [viitattu 15.6.2014]. Saatavissa: http://ri-www.nii.ac.jp/SLINT/JIST2012.pdf.

[29] Miller, G.A. WordNet: A lexical database for English. Association for Computing Machinery.Communications of the ACM 38(1995)11, pp. 39.

[30] Jaro, M.A. Advances in Record-Linkage Methodology as Applied to matching the 1985 Census of Tampa, Florida. Journal of the American Statistical Association 84(1989)406, .

[31] Winkler, W.E. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. 1990, .

[32] Winkler, W.E. The State of Record Linkage and Current Research Problems. [viitattu 7.9.2014]. Saatavissa: http://www.census.gov/srd/papers/pdf/rr99-04.pdf.

[33] Stoilos, G. A string metric for ontology alignment. Semantic Web - Iswc 2005, Proceedings, 2005, Vol.3729, pp.624-637 3729(2005) pp. 624-Iswc.

[34] Christophides, V. On labeling schemes for the semantic web. World Wide Web: Proceedings of the 12th international conference, (WWW '03), 2003, pp.544-555 (2003) pp. 544-555.

[35] Nebot, V. Efficient retrieval of ontology fragments using an interval labeling scheme. Information Sciences 179(2009)24, pp. 4151.

[36] Melnik, S. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. 18th International Conference on Data Engineering; San Jose, CA; United States; 26 Feb.-01 Mar.2002 (2002) pp. 117.

[37] Protégé. [viitattu 15.6.2014]. Saatavissa: http://protege.stanford.edu/.

[38] Graphviz. [viitattu 15.6.2014]. Saatavissa: http://www.graphviz.org/.

[39] Golbreich, C. Esimerkki perheontologiasta. [viitattu 15.6.2014]. Saatavissa: http://protege.cim3.net/file/pub/ontologies/family.swrl.owl/family.swrl.owl.

[40] Horridge, M. Esimerkki sukupolviontologiasta. [viitattu 15.6.2014]. Saatavissa: http://protege.cim3.net/file/pub/ontologies/generations/generations.owl.

[41] Esimerkki olutontologiasta. [viitattu 15.6.2014]. Saatavissa: http://dbs.uni-leipzig.de/files/coma/sources/fd/beer.owl.

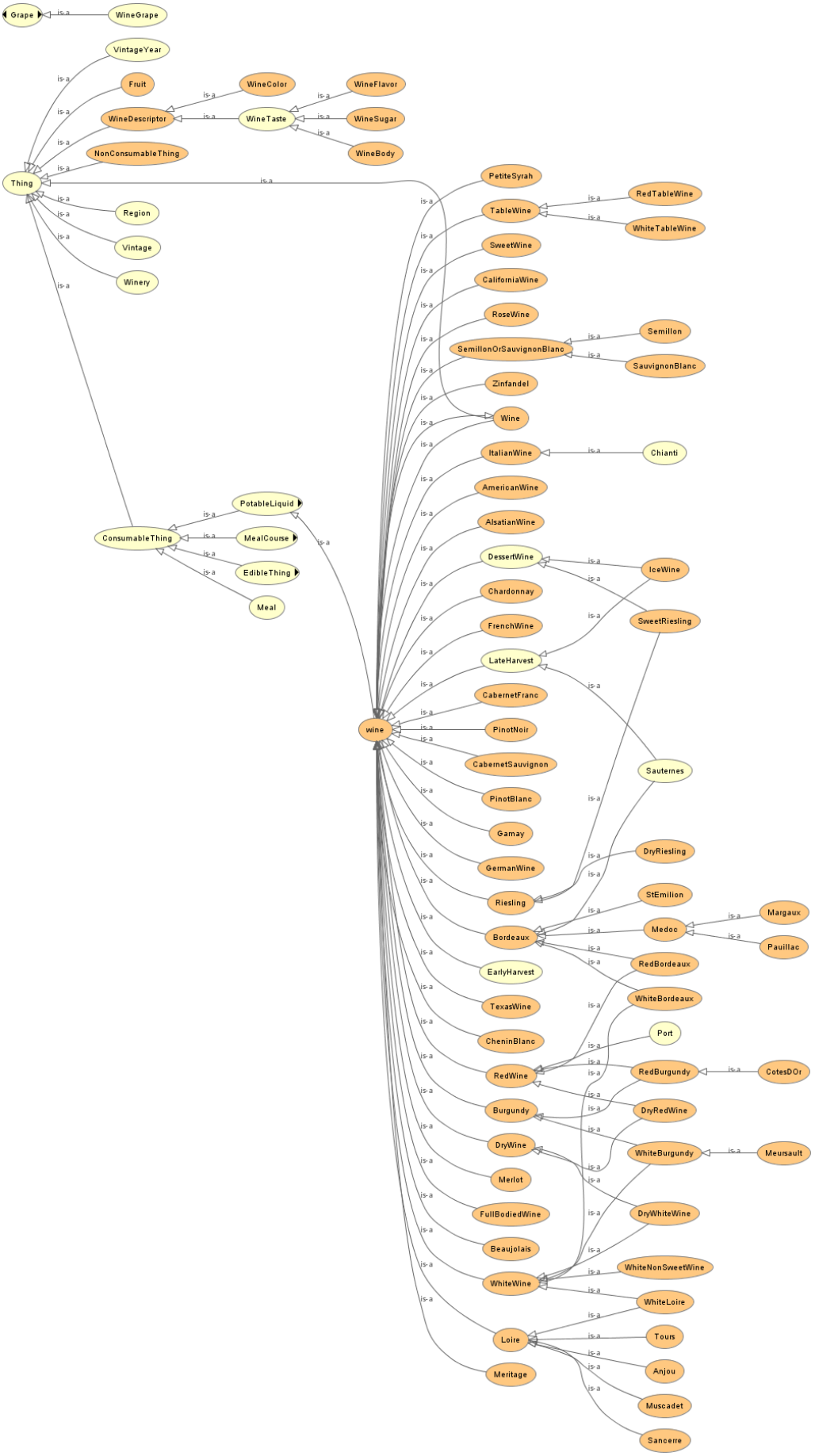
[42] Smith, M.,K., Welty, C. & McGuinness, D.,L. Esimerkki viiniontologiasta. [viitattu 15.6.2014]. Saatavissa: http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf.

[43] Ruiz, E.,Jiménez Using LogMap from the command line. [viitattu 2.8.2014]. Saatavissa: https://code.google.com/p/logmap-matcher/wiki/LogMapCommandLine.

[44] Jiménez-Ruiz, E. What does "CLS" mean in the TXT mapping format? [viitattu 2.8.2014]. Saatavissa: https://groups.google.com/forum/#!topic/logmap-matcher-discussion/-eVjbHZXmWI.

[45] yEd Graph Editor. [viitattu 15.6.2014]. Saatavissa: http://www.yworks.com/en/products_yed_about.html.

LIITE 1: VIINIONTOLOGIA VISUALISOITUNA



LIITE 2: OLUTONTOLOGIA VISUALISOITUNA