XML2RDF muunnosprosessi:

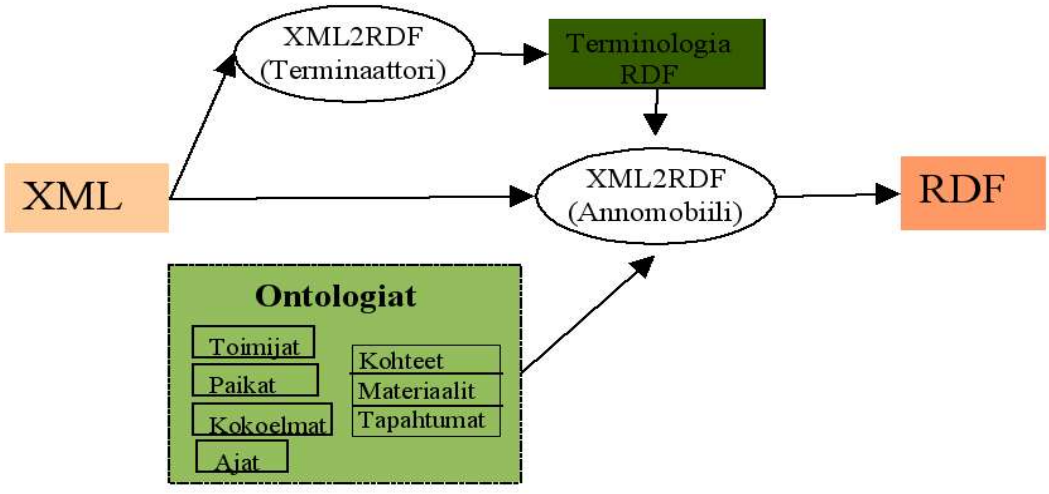
Terminaattori- ja Annomobiili-ohjelmistot

Eero Hyvönen, Mirva Salminen 19.03.2004

Yksittäisten museoiden kokoelmien polku MuseoSuomi-järjestelmän käyttämään muotoon vaatii aineiston yhtenäistämisen sekä syntaktisesti että semanttisesti. XML2RDF muunnos toteuttaa semanttisen yhdistämisen. Semanttisessa yhdistämisessä saavutetaan neljä etua:

1. Eri termistöt saadaan yhteismitallisiksi. Esimerkiksi pikkupöydät tunnistetaan pöydiksi, lenkkarit lenkkitossuiksi ja lyhenne HKI Helsingiksi.
2. Termien semanttinen epävarmuus ja monimerkityksellisyys ratkaistaan. Esimerkiksi homonymisen termin viitat – tienviitat ja päällysvaatteet – kohdalla voidaan eksplisiittisesti kertoa, kumpaa viittaa tarkoitetaan.
3. Yhteisiin resursseihin voidaan viitata yksikäsitteisesti. Esimerkiksi Suomessa on kaksi eri Helsinkiä, jotka ontologian avulla voidaan erottaa toisistaan. Samoin voidaan yksikäsitteisesti viitata muun muassa muihinkin paikkoihin sekä toimijoihin.
4. Semanttisen yhteensopivuuden kautta voidaan löytää ei vain termien, vaan myös käsitteiden ja esineiden väliset erilaiset yhteydet.

Semanttinen yhdistäminen suoritetaan puoliautomaattisesti XML-korttien, MuseoSuomen ontologioiden ja ns. termiontologian eli termikorttien avulla. Termiontologia käsittää museokokoelmien kuvailussa käytettyjä termejä, joille on tehty ontologinen ripustus. Ontologinen ripustus tarkoittaa tässä sitä, että on formaalisti ilmaistu, mihin ontologian käsitteeseen/käsitteisiin termi liittyy. MuseoSuomessa semanttisen yhdistämisen suorittamiseksi on tuotettu kaksi ohjelmistoa: Terminaattori ja Annomobiili. Prosessikuva (Kuva 1) näyttää, miten ohjelmat, syötteet ja tulosteet liittyvät toisiinsa.



Kuva 1: Kuva prosessista, jossa semanttisen yhdistäminen suoritetaan.

Terminaattori-ohjelma

Terminaattoritiedosto tuottaa uusia termikortteja museokokoelmien tietoja sisältävistä XML-korteista. Semanttisessa yhdistämisessä kaikkia museoesineen kokelmatietoja ei ripusteta ontologioihin, vaan vain valitut tapaukset. Ripustettavat tiedot, ns. ontologiset ominaisuudet, ripustetaan aina tiettyyn ontologiaan. Näistä ontologisista ominaisuuksista tulee tehdä termikortit, jotta ontologinen ripustaminen onnistuu. Ontologiset ominaisuudet ja ontologiat, joihin niiden tiedot ripustetaan, näkyvät taulukossa 1. Samassa taulukossa näkyvät myös MuseoSuomen yhteisessä käytössä olevat termikortit ja museoiden omat paikalliset termikorttitiedostojen nimet.

Ontologia	***Ontologinen ominaisuus (XML elementti)***	***Paikalliset termikortit***	***MuseoSuomen termikortit***	***Asetus-tiedostot***
mao	asiasana	maoTermit_omaMuseo.rdf	maoTermit.rdf	konfig_ mao.txt
	materiaali			
	kohde			
toimijat	tekija	paikkaTermit_omaMuseo.rdf	paikkaTermit.rdf	konfig_ paikat.txt
	käyttäjä			
	vastuumuseo			
paikat	valmistuspaikka	toimijaTermit_omaMuseo.rdf	toimijaTermit.rdf	konfig_ toimijat.txt
	käyttöpaikka			
kokoelmat	kokoelmat	kokoelmaTermit_omaMuseo.rdf	kokoelmaTermit.rdf	konfig_ kokoelmat.txt

Taulukko 1: Ontologiset ominaisuudet ja niitä vastaavat ontologiat ja termikortistot.

Terminaattorilla on asetustiedosto *konfig_X.txt*, jossa X =mao, toimijat, paikat, kokoelmat. Asetustiedostossa kerrotaan viisi asiaa:

1. XML-kortin tutkittavien ominaisuuksien nimet (esim. valmistaja ja kayttaja)
2. MuseoSuomen globaali termikorttitiedosto *X*termit.rdf
3. Museon paikallinen termikorttitiedosto *X*termit_*omaMuseo*.rdf
4. Termikorttitiedoston nimiavaruus, esimerkiksi http://www.cs.helsinki.fi/group/seco/ns/2004/03/18-terms#
5. Selite, jonka haluaa uusiin termikortteihin, esimerkiksi oman museon nimi ja päivämäärä.

Lisäksi Terminaattorilla on setup_terminator, joka pitää ajaa ennen Terminaattorin ajamista esimerkiksi komennolla:

%source setup_terminator

Terminaattoriohjelma käynnistetään komentoriviltä ja sille annetaan parametrina XML-tiedosto sekä asetustiedosto. Käynnistys tapahtuu esimerkiksi näin:

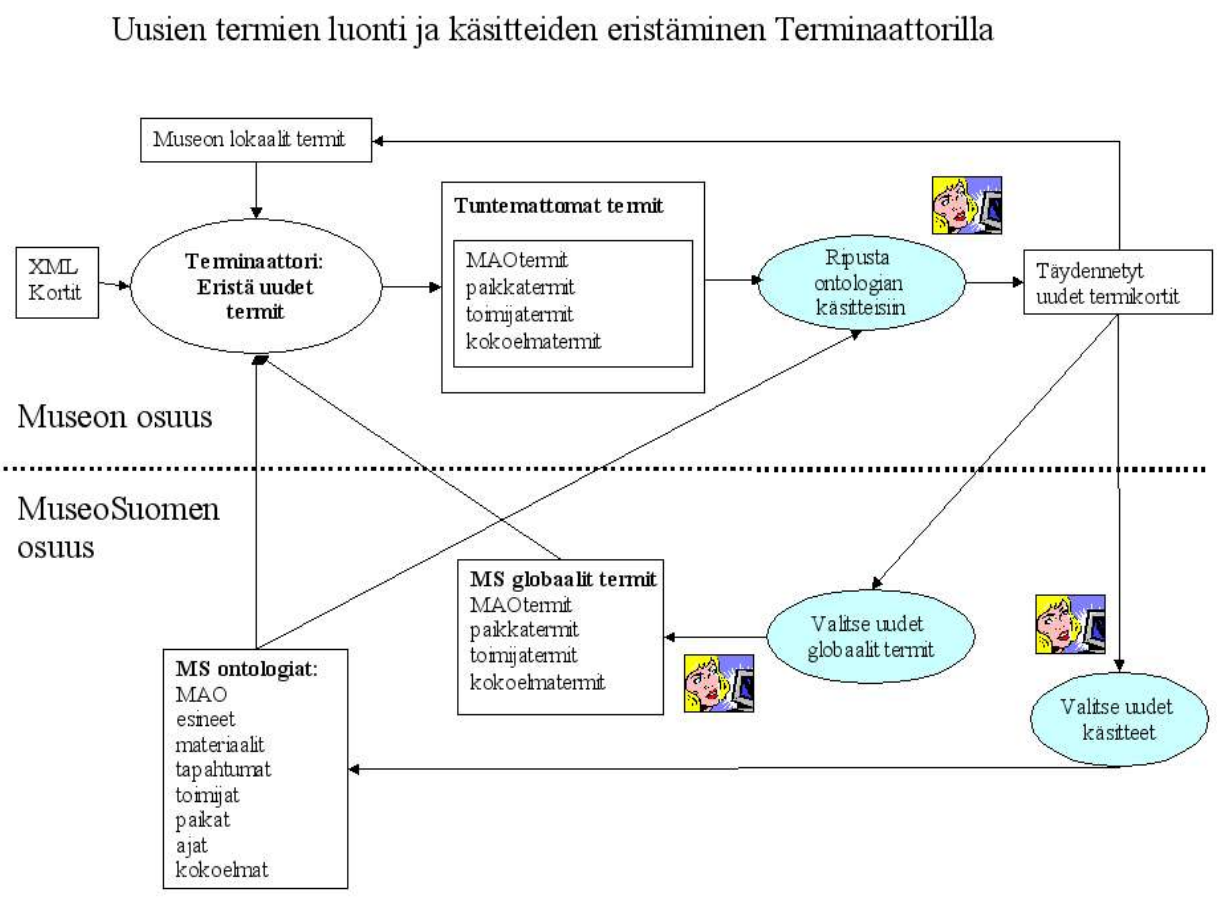
%java Terminaattori xmlkortit.xml konfig_toimijat.txt

Terminaattoriohjelma erottaa XML-tiedostosta uudet termikandidaatit, luo niistä tyhjät termikortit, ja esitäyttää termikortit. Terminaattoriohjelma tunnistaa tuntemattomat termit vastaavissa XML-elementeissä eli toisin sanoen sellaiset termit, joista ei ole jo termikorttia olemassa. Tunnistuksen jälkeen uusia termejä vastaavat puoliksi täytetyt termikortit lisätään museon omiin termikortteihin luokan ”tuntematon” instansseiksi. Esitäytössä termille luodaan uri, alkuperäismuotoinen ilmaisu talletetaan ja kommentiksi asetetaan tieto siitä, mistä kentästä termi on erotettu.

Terminaattorin tulosteena on kyseiset yllä mainityt esitäytetyt uudet termikortit. Nämä uudet lisätään vanhojen museon omien eli lokaalien termien seuraksi tiedostoon *X*Termit_*omaMuseo*.rdf ihmisen edelleen toimitettavaksi ohjetiedoston avulla.

Esitäytettyjen termikorttien niissä olevien kommenttien perusteella ihmistoimittaja osaa tehdä tarvittavat korjaukset ja päättää, mitkä termit asetetaan etusijalle. Ajatuksena on, että ihmiseditori täydentään vajavaisten termikorttien viittaukset vastaavan skeeman mukaiseen ontologiaan ja siirtää valmiin kortti-instanssin lopuksi oikeaan paikkaan luokan ”Terms” alle.

Kuva 2 alla esittää uusien termien ja käsitteiden tuottamisprosessia Terminaattorin avulla. Ohjelman ajo voidaan suorittaa uudelleen uusilla parametreilla tai uudella XML-tiedostolla, kunnes ei synny uusia instansseja, eli kaikki termit on tunnistettu.



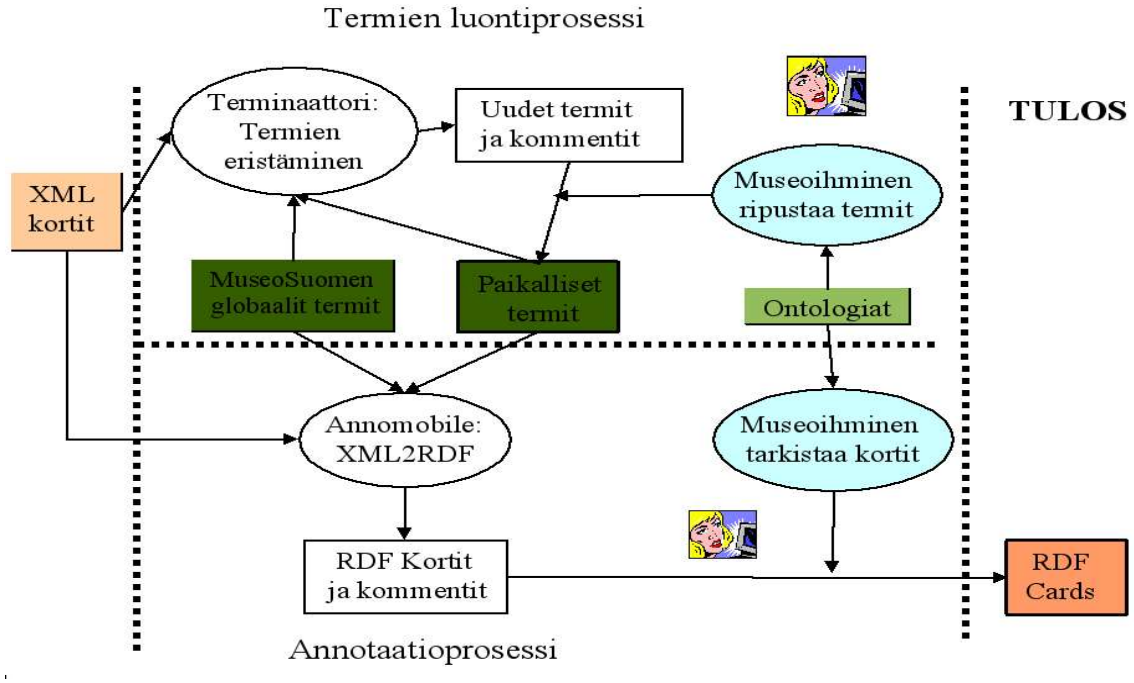
Kuva 2: Uusien termien ja käsitteiden tuottaminen.

Annomobiili-ohjelma eli XML2RDF-muunnos

Kuva 3 alla esittää koko XML2RDF muunnosprosessia. Prosessi on kaksivaiheinen:

1. Ensin tuotetaan uusista XML korteista tarvittavat uudet termit ja yksilöt Terminaattorin avulla.
2. Tämän jälkeen Annomobiiililla voidaan suorittaa varsinainen XML2RDF muunnos.

Molemmissa vaiheissa ihmisen tulee tarkistaa ja täydentää epäselvien tapauksien kohdalla koneen tekemää työtä. Tämä työn avuksi Terminaattori ja Annomobiili lisäävät tuotoksiinsa kommentteja, joiden avulla epätäydelliset tai epäilyttävät termi- ja RDF-kortit voidaan löytää ja tarkistaa.



Kuva 3: XML2RDF prosessien suoritus.

Alla oleva taulukko (taulukko 2.) kuvaa XML-kokoelmatietojen kuvautumisen RDF-ominaisuuksille (ontologiset ja literaalit), näkymiksi ja termeiksi.

	XML-kortti: elementit	**RDF-kortti: ontologinen ominaisuus**	**Ontologia**	**Ontologian juuriresurssi**	**RDF-kortti: literaali ominaisuus**	**Käyttö-liitymän näkymä**
Kohdetiedot	kohde	kohde	mao	esineet	www_kohde	otsikko
	materiaali	materiaali	mao	materiaalit ja aineet	www_ materiaali	materiaali
	kuva	-	-	-	www_kuva	(kuva)
	asiasana	asiasana	mao	mao-käsitteet	www_ asiasanat	asiasana
	mitat	-	-	-	www_mitat	mitat
	kuvailu	-	-	-	www_ kuvailu	kuvailu
Tekotiedot	tekija	tekija	toimijat	toimijat	www_tekija	valmistaja

	XML-kortti: elementit	RDF-kortti: ontologinen ominaisuus	Ontologia	Ontologian juuriresurssi	RDF-kortti: literaali ominaisuus	Käyttö-liitymän näkymä
	valmistuspaikka	paikka	paikat	paikka	www_valmistuspaikka	valmistus-paikka
	valmistusaika	valmistusaika_alku	ajat	valmistusaika	www_valmistusaika	valmistusaika
	valmistusaika	valmistusaika_loppu	ajat	valmistusaika	www_valmistusaika	valmistusaika
Käyttötiedot	kayttaja	kayttaja	toimijat	toimijat	www_kayttaja	käyttäjä
	kayttopaikka	kayttopaikka	paikka	paikka	www_kayttopaikka	käyttöpaikka
	-	kayttotilanne	-	-	-	-
Kokoelma tiedot	museo	museo	toimijat	toimijat	www_vastuumuseo	vastuumuseo
	kokoelma	kokoelma	kokoelmat	kokoelmat	www_museo kokoelma	museo-kokoelma
	numero	-	-	-	www_numero	esineen numero
	esineid	-	-	-	www_esineid	id

Taulukko 2: XML-kokoelmatietojen kuvautuminen.

Annomobiilla on kolme asetustiedostoa:

1. *konfig_range.txt*, jossa määritellään juuriresurssit ontologisille ominaisuuksille
2. *konfig_tiedosto.txt*, jossa annetaan ontologioiden ja termitiedostojen hakemistot, sekä tieto, mihin tiedostoon tulos kirjoitetaan
3. konfig.txt tiedostossa kerrotaan ontologisten ominaisuuksien nimiavaruudet.

Lisäksi Annomobiililla on setup_annomobile, joka pitää ajaa ennen annomobiilin ajamista esimerkiksi komennolla:

%source setup_annomobile

Annomobiili käynnistetään komentoriviltä ja sille annetaan parametrina XML-tiedosto. Käynnistys tapahtuu esimerkiksi näin:

%java Annomobiili xmlkortit.xml annolog.txt

Annomobiili lukee sisäänsä ensin ontologiat. Sitten se alkaa parsia XML-tiedostoa käyden sen läpi esineen kerrallaan. Joka esineestä se käy läpi joka elementin ja muodostaa niistä literaaliominaisuuden. Ontologisten ominaisuuksien kohdalla Annomobiili etsii termitiedostoista kaikki termiä vastaavat mahdolliset ontologiaripustukset. Löydettyjen ontologiaripustusten kohdalla tarkistetaan täyttääkö kyseinen ontologiaripustus konf_range-asetustiedostossa määritellyt ripustusrajoitukset. Rajoitusten lisäksi tarkistetaan, ettei ripustuksiin lisätä samaan haaraan kuuluvia ontologiaripustuksia, esimerkiksi ettei lisätä ripustusta kenkiin, jos mukana on myös ripustus

korkokenkiin. Mikäli nämä molemmat ehdot – range-rajoitus ja ei-sama-haara – täyttyvät, Annomobiili suorittaa ripustuksen. Kun kaikki esinekortit on käyty läpi, Annomobiili kirjoittaa ne tiedostoon, joka on määritelty konf_tiedosto.txt-tiedostossa. Esinekortteihin Annomobiili kirjoittaa lisätietoja annotoinnin helpottamiseksi. Lisätiedoitoihin on kerrottu kaksi asiaa:

1. Monimerkitykselliset termit, eli missä esinekortissa esiintyy monimerkityksellinen sana ja mikä kyseinen sana on. Monimerkityksellisiä tietoja sisältävät kortit kannattaa käydä läpi ja poistaa toinen ripustuksista, koska ne molemmat eivät todennäköisesti ole haluttuja.
2. Termikortittomat termit, eli missä kortissa esiintyy termi, jota vastaavaa termikorttia ei ole tehty. Tällaisia termejä ei pitäisi esiintyä, mikäli Terminaattori on ajettu. Kyseisille termeille on hyvät tehdä vastaava termikortti museon omaan termikorttitiedostoon.