Rakenteiset sanakirjat

Outi Lehtinen

Helsinki 10.4.2008

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO 1 HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta/Osasto 1 Fakultet/Sektion –Faculty/Section	Laitos 1 Institution 1 Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä Författare 1 Author

Outi Lehtinen

Työn nimi Arbetets titel 1 Title

Rakenteiset sanakirjat

Oppiaine 1 Läroämne 1 Subject

Tietojenkäsittelytiede

Työn laji Arbetets art 1 Level	Aika Datum 1 Month and year	Sivumäärä Sidoantal 1 Number of pages
Pro gradu -tutkielma	10.4.2008	57 sivua + 13 liitesivua

Tiivistelmä Referat 1 Abstract

Sanakirjat ovat rakenteisia tekstejä. Sana-artikkeleissa on nähtävissä selvästi erotettavissa olevia rakenneosia. Painetun sanakirjan muuntaminen rakenteiseen muotoon ei kuitenkaan ole aivan suoraviivaista. Tässä tutkielmassa kerrotaan sanakirjan rakennekuvauksen määrittelyn ja painetun sanakirjan rakenteistuksen ongelmista. Esimerkkinä käytetään Kotimaisten kielten tutkimuskeskuksessa (Kotus) kirjoitettavan Suomen murteiden sanakirjan (SMS) rakenteistusprojektia ja sen yhteydessä määriteltyä rakennekuvausta. SMS:n rakennekuvausta verrataan kielitieteellisten aineistojen koodaamisessa yleisesti käytetyn Text Encoding Initiative -suosituksen sekä uuden ISO 1951:2007 -sanakirjastandardin määrittelemiin rakennekuvauksiin. Muuntamista testataan käytännössä kirjoittamalla XSL-muunnosskripti, joka muuntaa SMS:n aineiston ISO-standardimuotoon. Muunnosprosessin voi todeta olleen hyödyllinen, koska sen aikana paljastui ongelmia sekä SMS:n rakennekuvauksessa että itse standardin määrittelyssä. Tulosten avulla SMS:n koodausta voidaan kehittää edelleen. Lopuksi tarkastellaan standardien käytöstä yleisesti saatavia hyötyjä ja mahdollisuuksia soveltaa niitä Kotuksen sanakirjatyöhön.

H.3.1 Information storage and retrieval: Content analysis and indexing
I.7.2 Document and text processing: Document preparation
J.5 Arts and Humanities

Avainsanat – Nyckelord 1 Keywords

sanakirjat, rakennekuvaukset, standardit, XML

Säilytyspaikka 1 Förvaringställe 1 Where deposited

Kumpulan tiedekirjasto, sarjanumero C-

Muita tietoja 1 Övriga uppgifter 1 Additional information

Sisältö

1 Johdanto

Sanakirjat ovat hyvä esimerkki rakenteisesta tekstistä. Sanakirjojen sisältö on jaettu sana-artikkeleihin, jotka puolestaan koostuvat selkeästi erotettavissa olevista osista, kuten hakusanasta, määritelmistä, viittauksista tai käyttöesimerkeistä. Samat osat toistuvat artikkelista toiseen, vaikkakin niiden järjestys voi vaihdella. Perinteisesti sanakirjat on suunniteltu julkaistaviksi painettuina kirjoina. Tämän takia artikkelit on usein esitetty paperia säästäen, lyhenteitä ja erilaisia lyhennystapoja käyttäen. Tieto on pyritty ilmaisemaan mahdollisimman tiiviisti, mutta kuitenkin ilman että sen luettavuus ja ymmärrettävyys kärsivät.

Painetusta sanakirjasta voidaan tehdä sähköinen versio niin, että aineisto digitoidaan sähköiseen muotoon. Artikkelien sisältö ja ulkoasu säilytetään samanlaisena, jolloin tuloksena on sähköinen kopio sanakirjasta. Sähköinen sanakirja toimii lukijan kannalta siis lähes samalla tavoin kuin painettukin. Tämän lisäksi se antaa mahdollisuuden tehdä hakuja artikkelien tekstistä eli etsiä tietoa muidenkin sanojen kuin vain hakusanoiksi valittujen sanojen perusteella.

Sanakirja on oikeastaan eräänlainen tietokanta, jonka artikkeli voidaan ajatella tietueena ja hakusana avaimena. Sanakirjaahan ei yleensä lueta järjestyksessä vaan artikkeleita etsitään tarpeen mukaan avaimen eli hakusanan perusteella. Sanakirjojen kuvaaminen relaatiotietokantana on kuitenkin osoittautunut vaikeaksi, koska artikkelit ovat usein hyvin hierarkkisia [IdV95]. Tästä syystä rakenteisen tekstin koodaustavat, ensin SGML ja myöhemmin XML, ovat vallanneet alaa sanakirjojen kuvausmuotoina. Rakenteeltaan sanakirjoja yksinkertaisimpien leksikkotietokantojen kuvaamisessa sen sijaan on käytetty monia eri tapoja, mm. WordNet [Mil90] perustui alunperin täysin tekstipohjaiseen koodaukseen ja Open Source Lexical Information Network [Jan05] taas yksinkertaiseen relaatiotietokantarakenteeseen.

Sähköistämisen lisäksi tehty sanakirjan rakenteistus lisää aineiston käyttömahdollisuuksia, mm. hakuja voidaan kohdistaa nimettyihin osiin sanakirjaa, esimerkiksi selitteisiin tai esimerkkeihin. Rakenteistuksella tarkoitetaan tässä aineiston koodausta siten, että siinä esiintyvät, merkitykseltään erotettavissa olevat rakenteet tunnistetaan, nimetään ja merkitään yhtenäisellä tavalla. Yleensä on tarpeen määritellä myös rakennekuvaus, joka

kertoo, mitä rakenteita sanakirjassa on ja miten ne voivat sijoittua suhteessa toisiinsa. Kun aineisto on kuvattu systemaattisesti tietokoneen ymmärtämällä tavalla, se mahdollistaa myös sanakirjojen hyödyntämisen erilaisissa sovelluksissa. Riippuen sanakirjan laajuudesta, kirjoittamisessa käytettyjen toimitusohjeiden laadusta ja halutusta rakenteistuksen tarkkuuden tasosta, voi rakenteistus olla hyvinkin aikaa vievä projekti. Rakenteistuksessa eteen tulevien ongelmien ratkaiseminen saattaa edellyttää myös artikkelien sisältöön puuttumista. Jotta rakenne olisi looginen, pitää purkaa ja kirjoittaa auki sanakirjassa käytetyt lyhennystavat ja luettavuuden lisäämiseksi tehdyt poikkeukset artikkelien rakenteessa. Tämänkin jälkeen artikkeleissa saattaa olla vielä monitulkintaisuutta, joka ihmislukijan on helppo tulkita oikein mutta koneen ei.

Tässä tutkielmassa kuvataan erilaisia tapoja esittää sanakirja-aineistoa rakenteisessa muodossa ja kerrotaan rakenteiseen muotoon muuntamisen ongelmista. Painettujen sanakirjojen rakenteistusta ja artikkelien sisältämän tiedon avaamista tietokoneen ymmärtämään muotoon käsitellään yleisesti luvussa 2. Luvussa 3 esitellään tarkemmin kaksi yleiskäyttöistä standardia sähköisten sanakirjojen koodaamiseen: TEI sanakirjasuositus [TEI07] ja uusi standardi ISO 1951:2007 [ISO07]. Näiden vertailukohdaksi on valittu Kotimaisten kielten tutkimuskeskuksessa (Kotus) suunniteltu rakennekuvaus Suomen murteiden sanakirjaa (SMS) varten. SMS:n rakenteistusprojektia ja sen tuloksena syntynyttä rakennekuvausta esitellään luvussa 4. TEI-suositusta ja ISO-standardia verrataan SMS:n rakennekuvaukseen luvussa 5 etsimällä kummastakin SMS:n rakenneosia vastaavat elementit ja tutkimalla, miten ne soveltuvat SMS:n koodaamiseen. Muunnos standardimuotoon toteutetaan XSL-muunnosskriptinä. Prosessissa vastaan tulleet ongelmat kuvataan ja niiden ratkaisemiseksi esitetään vaihtoehtoja. Lopuksi muunnoksen tulosta arvioidaan ja testataan muuntamalla standardimuotoinen aineisto takaisin alkuperäiseen muotoon.

Luvussa 6 kerrotaan yleisesti standardoinnin hyödyistä ja sovelluksista sekä standardimuotojen muunnettavuudesta. Lisäksi esitellään muutamia tapoja, joilla hyvin määriteltyä koodausta käyttäviä sanakirja-aineistoja voidaan jatkohyödyntää mm. erilaisia sanakirjoja yhdistelemällä. Lopuksi pohditaan, miten standardeja voitaisiin hyödyntää Kotuksen sanakirjatyössä. Tutkielman yhteenveto on luvussa 7.

2 Sanakirjojen rakenteistus

Sanakirja-artikkelien rakennetta on tutkittu laajasti (mm. artikkeleissa [AmT88], [IdV95] ja [BeB00]). Tarkoituksena on ollut sellaisen mallin löytäminen, jonka avulla sanakirjat voitaisiin koodata yhtenäiseen rakenteiseen (tietokanta)muotoon. Malleja sekä sähköisten sanakirjojen että leksikoiden koodaamiseen on vuosien aikana kehitetty Text Encoding Initiative -hankkeen[1] (TEI) ja useiden EU-hankkeiden työryhmissä. EU-projekteissa on panostettu erityisesti kieliteknologisissa sovelluksissa tarvittavien leksikkojen (sanastojen) määrittelyyn ja kokoamiseen. Projekteja ja suosituksia useita (mm. GENELEX [Gen94], SIMPLE [Len00], IMDI [WPB02], OLIF[2]).

International Standards Organization (ISO) on perustanut komitean TC37 (Terminology and other language and content resources) kehittämään terminologisia ja muita kieliresursseja koskevia standardeja. Sen uusimpana saavutuksena on vuonna 2007 ilmestynyt standardi *ISO 1951:2007 Presentation/representation of entries in dictionaries* [ISO07], joka määrittelee formaalisti sähköisen sanakirjan rakenteen. Ensimmäisenä standardikokoelmasta valmistui termistöstandardi ISO 16642:2003 (Terminological Markup Framework). Vastaava leksikkostandardi (Lexical Markup Framework, LMF) on vielä tekeillä [Fra06].

Tässä luvussa määritellään ensin lyhyesti tutkielmassa käytettävät termit ja käsitteet ja kerrotaan painetun sanakirjan rakenteistukseen liittyvistä ongelmista. Luvun lopussa tarkastellaan yhtä formaalia tapaa purkaa sana-artikkelien sisältämää implisiittistä tietoa. Käsittely keskittyy enimmäkseen yksikielisiin sanakirjoihin, koska ne ovat usein rakenteeltaan rikkaampia ja mutkikkaampia kuin kaksikieliset.

2.1 Sanastoa

Kirjallisuudessa ([Lit06], [HaI06]) käytetään usein termiä *sanakirjatietokanta* (dictionary database, lexical database, computational lexicon), jolla tarkoitetaan yleisesti kaikkia sellaisia sanastotietokantoja, joiden pohjalta voidaan tuottaa leksikografisia tuotteita. Sanakirjatietokantoihin sisältyvät sekä *sähköiset sanakirjat* (konelukuinen sanakirja, machine-readable dictionary) että *leksikot* (sanasto, lexicon). Sähköisellä sanakirjalla tarkoitetaan laajasti mitä tahansa koneellisesti käsiteltävässä muodossa ole-

1 http://www.tei-c.org
2 http://www.olif.net

vaa sanakirjaa, joka on tarkoitettu ihmisten luettavaksi. Nykyään se on usein myös koodattu rakenteiseen muotoon, jolloin sitä voidaan kutsua *rakenteiseksi sanakirjaksi*. Leksikoilla tarkoitetaan tässä koneellisesti käytettäväksi tarkoitettuja sanastoja. Ne voivat vaihdella erilaisista sanalistoista sanastotietokantoihin, wordnetiin ja jopa ontologioihin. *Termistöt* eroavat sanakirjoista ja leksikoista siinä, että ne eivät perustu hakusanojen vaan käsitteiden hierarkkiseen kuvaamiseen.

Jatkossa oletetaan, että lukijalle ovat tuttuja yleisimmät W3C:n XML-standardeihin[3] liittyvät termit, joista tässä selitetään vain tärkeimmät. Termillä *elementti* tarkoitetaan erityisesti XML-koodauksen elementtiä (element), jonka ilmaisemiseen käytetään *tunnisteita* (tags). *Rakenne* (myös rakenneosa) tarkoittaa jotain määrättyä tekstinosaa, joka voidaan tunnisteilla merkata elementiksi. *Rakennekuvauksella* tarkoitetaan mitä tahansa tapaa kuvata rakenneosien tai elementtien sisältöä, järjestystä ja hierarkiaa. Tämä tapa voi olla DTD, skeema tai jokin muu formaali tapa kuvata rakennetta.

Tekstissä käytetään myös joitakin leksikografisia eli sanakirjan kirjoittamiseen liittyviä termejä. Seuraavat termit on selitetty hieman yksinkertaistaen, mutta ne vastaavat määritelmiä lähteissä [BCF97], [Aap01] ja [Suo88]. *Homonyymi* on eri alkuperää oleva, erimerkityksinen mutta äänne- tai kirjoitusasultaan identtinen (haku)sana. *Merkitysryhmät* ovat ryhmiä, joihin sana-artikkeli on jaettu hakusanan erilaisten merkitysten tai käyttötapojen mukaan. *Alihakusana* on sana-artikkelissa omana alaryhmänään tai muuten korostettuna esitetty hakusanan muoto, sanonta, sanaliitto tms. SMS:ssä alihakusana on erityisesti selitysosassa oleva sitaattimuotoinen kiteymän tai sanonnan sisältävä hakusana, joka valaisee jotain varsinaisen hakusanan käyttöyhteyttä ja joka selitetään kokonaisuutena. Termillä *vartalo* tarkoitetaan SMS:ssä hakusanan vokaalivartaloa (esim. *lakeinen*, vartalo: *lakehitte-*). *Tyvi* tarkoittaa esitystavan kannalta tarkoituksenmukaisesti katkaistua hakusanan alkuosaa.

2.2 Painetun sanakirjan rakenteistuksesta

Painetussa sanakirjassa artikkelin eri osat on erotettu lukijaa varten erilaisilla muotoiluilla: hakusanat on lihavoitu, esimerkit kursivoitu jne. Ulkoasu vaihtelee sanakirjasta toiseen sen mukaan, mikä on ollut sanakirjan tarkoitus ja millaiset toimitusohjeet kirjoittajilla on ollut käytössään. Osa sanakirja-artikkelin visuaalisesta ulkoasusta on

3 http://www.w3.org/XML/

sisällön ymmärtämisen kannalta tärkeää, ja ulkoasun perusteella voidaan tehdä päätelmiä artikkelin rakenteesta [BeB00]. Muotoiluja on toki useasti käytetty pelkästään ulkoasun kaunisteluun ja luettavuuden parantamiseen. Käytettyjen muotoilujen ja kirjoituskäytäntöjen perusteella voidaan löytää ja tunnistaa suurin osa artikkelin rakenneosista. Syvempi informaatiorakenne, tietojen liittyminen toisiinsa, periytyvyys ja hierarkia ovat usein implisiittisesti ihmislukijan pääteltävissä.

Nykyään on tarve tuottaa sanakirjatietokantoja, joita voidaan käyttää hyväksi erilaisissa kieliteknologisissa sovelluksissa. Tämä asettaa lisävaatimuksia koodaustavalle ja -tarkkuudelle. Sanakirjatietokantaan voidaan sisällyttää paljon sellaista leksikaalista tietoa, jota ei ole eksplisiittisesti kirjoitettu näkyviin painetussa sanakirjassa. Tällaisia tietoja ovat esimerkiksi hakusanojen taivutusmuotosarjat, joita ei kannata painaa kirjaan, koska lukija pystyy yleensä generoimaan ne kielitajunsa perusteella.

Eri käyttäjäryhmien tarpeiden perusteella on määritelty erilaisia tarpeita yhteiselle sanakirjamuodolle [IdV95].

1. Sanakirjojen tekijät ja julkaisijat haluaisivat uudelleenkäyttää aineistoa erilaisten sanakirjaversioiden ja -tuotteiden tehokkaassa tuottamisessa.
2. Kieliteknologit voisivat käyttää erilaisia sanakirjoja leksikkojen koostamisessa sekä kieliteknologisten sovellusten tuottamisessa. Näin voitaisiin myös tuottaa yhteensopivia erikielisiä leksikoita.
3. Kielen- ja historiantutkijat voisivat helpommin vertailla ja jakaa sanakirja-aineistoa sekä käyttää yhtenäisiä tutkimussovelluksia.
4. Tavalliset sanakirjankäyttäjät voisivat yhdistellä ja käyttää tiedonhaussa useita sanakirjoja yhden käyttöliittymän kautta. Esimerkiksi Kotuksen sanakirjavalikoimasta löytyisivät mm. vanhan kirjasuomen sanamuodot, suomen ja saamen etymologiatiedot, sanojen murrevariantit ja nykykielen sanat.

Rakennemallin pitäisi siis tyydyttää monenlaisten käyttäjäryhmien tarpeita ja vaatimuksia. Ulkoasu on tärkeä sekä painettujen sanakirjojen julkaisijalle että sanakirjan käyttäjälle. Myös tutkijoille on usein tärkeää nähdä sanakirja (lähes) alkuperäisessä muodossaan, kun taas rakenteiden avulla kuvatun informaation tarkkuus on tärkeää aineiston jatkojalostuksen, tuotteistamisen ja kieliteknologisen hyödyntämisen takia.

Ensimmäinen vaihe sanakirjarakenteen luomisessa on pienimpien rakenneosasten tunnistaminen ja nimeäminen. Tätä on tutkittu paljon, ja jonkinlainen konsensus siitä, mitä rakenteita sanakirja-artikkeli voi sisältää oli olemassa jo ennen TEI:n sanakirjamäärittelyä [AmT88]. Toinen vaihe on vaikeampi, ja siihen liittyy löydettyjen osien ryhmittely sekä niiden välisen järjestyksen ja hierarkian määrittely. Ongelmia aiheuttaa sanakirjojen välillä ja jopa saman sanakirjan sisällä esiintyvä variaatio. Rakenteet voivat esiintyä eri järjestyksessä jopa saman sanakirjan eri artikkeleissa. Tämä voi johtua vaikka siitä, että toimittaja on muuttanut järjestystä tehdäkseen tekstin luettavammaksi tai toimitusohjeet ovat muuttuneet jossain vaiheessa laajan sanakirjan työstämistä. Eri sanakirjojen välillä variaatio ilmenee mm. erilaisena tapana jakaa tietoa artikkeleihin. Joissain sanakirjoissa homonyymiset hakusanat (kirjoitusasultaan samanlaiset mutta eri alkuperää olevat ja eri merkityksiset sanat, esim. *kuusi*[1]*: havupuu*, *kuusi*[2]*: lukusana*) ovat samassa artikkelissa, toisessa taas omissa artikkeleissaan. Tästä seuraa se ongelma, että soveltuakseen erilaisille sanakirjoille pitää rakennekuvauksen olla järjestyksen osalta hyvin vapaa. Hyvin yleistä kuvausta käytettäessä ei taas voida rajoittaa rakenteiden järjestystä juuri sellaiseksi, kuin kyseisen sanankirjan toimitusohjeissa on sovittu.

Sanakirjan rakennetta koodatessa täytyy usein tasapainotella typografisen ja tekstuaalisen ulkomuodon säilyttämisen ja informaatiosisällön tarkan mallintamisen välillä. Nancy Ide ja Jean Veronis [IdV95] määrittelevät kaksi tapaa koodata sanakirja-aineisto: *tekstuaalisen näkymän* (textual view) ja *leksikaalisen näkymän* (lexical view) koodaus.

Tekstuaalista näkymää koodattaessa pyritään säilyttämään sanakirjan tekstisisältö muuttumattomana.

- Mitään tekstiä ei saa poistaa tai muuttaa (poikkeuksena ulkoasuun liittyvä teksti).
- Alkuperäinen teksti pitää säilyttää elementtien sisältönä.
- Lisättyä tietoa ei saa laittaa elementin sisällöksi.
- Tekstin alkuperäistä järjestystä ei saa muuttaa.

Esimerkiksi seuraava artikkeli

> **kärvi|kissa** s. kiimainen kissa. | *Häriillänsä kun kärvikissa.* AHär

voitaisiin koodata (rivitystä on kaunisteltu luettavuuden parantamiseksi):

```
<artikkeli>
        <hakusana>kärvi|kissa</hakusana><sanaluokka> s.</sanaluokka>
        <selite> kiimainen kissa.</selite>
        <esim>| Häriillänsä kuin kärvikissa.</esim><pitäjä> AHär<pitäjä>
</artikkeli>
```

Tunnisteet poistamalla koodatusta versiosta saadaan täsmälleen sama merkkijono (rivitystä lukuunottamatta) kuin alkuperäisessä artikkelissa. Ulkoasuun liittyvä teksti voi olla välimerkkejä, sulkeita, lyhenteitä tai apusanoja. Edellisessä esimerkissä sellaisiksi voidaan laskea selitteen lopussa oleva piste sekä selitteen ja esimerkin erottava pystyviiva. Mikäli voidaan sanoa, että nämä ovat varmasti automaattisesti generoitavissa, ne voidaan jättää pois ja säilyttää erillisessä ulkoasusäännöstössä (esim. XSL). Kannattaa olla huolellinen sen suhteen, mitä voi generoida automaattisesti ja mitä ei. Esimerkiksi esimerkkirakenteen lopussa oleva piste kuuluu kiinteästi esimerkkiin, voisihan esimerkki päättyä myös huuto- tai kysymysmerkkiin. Myös rakenteiden välisten välilyöntien olemassaolon ennustaminen, varsinkin sisäkkäisissä rakenteissa, saattaa olla mahdotonta.

Leksikaalisessa näkymässä pyritään koodaamaan varsinainen tietosisältö tarkasti ja yksikäsitteisesti. Sanakirjasta riippuen se voi vaatia

- normalisointia, esimerkiksi lyhennysmerkintöjen avaamista (slg. => slangi tai a(a)men => amen, aamen),
- täydentämistä, esimerkiksi puuttuvan määriteosan lisääminen (aamukahvi, -puuro => aamukahvi, aamupuuro),
- elementtien uudelleenjärjestämistä, jotta yhteenkuuluvat tiedot saadaan ryhmiteltyä yhteen ja
- artikkelin jakamista tarvittaessa.

Esimerkkiartikkelissa on niputettu kolme hakusanaa yhteen tilan säästämiseksi.

> **eil|leen, -len, -äin** adv. -> eilen

Leksikaalisen näkymän mukaan tämä voitaisiin koodata täydentämällä hakusanat ja jakamalla artikkeli kolmeen osaan.

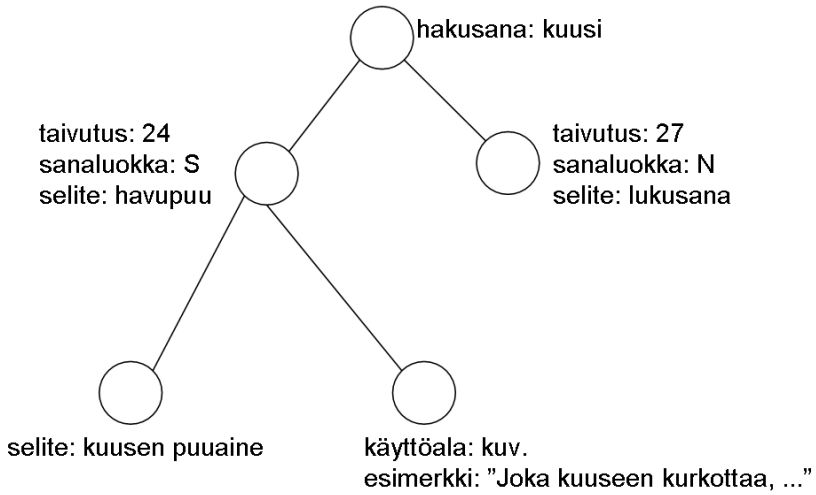
```
<artikkeli>
        <hakusana>eilleen</hakusana>
        <sanaluokka>adv.<sanaluokka>
        <viittaus>eilen</viittaus>
</artikkeli>
<artikkeli>
        <hakusana>eillen</hakusana>
        <sanaluokka>adv.<sanaluokka>
        <viittaus>eilen</viittaus>
</artikkeli>
<artikkeli>
        <hakusana>eiläin</hakusana>
        <sanaluokka>adv.<sanaluokka>
        <viittaus>eilen</viittaus>
</artikkeli>
```

On ilmeistä, että nämä näkymät ovat monessa kohtaa ristiriidassa keskenään, ja tällöin molempien mahdollistaminen ja koodaus samaan aineistoon on hankalaa ja aikavievää. Jos elementtien järjestystä ei tarvitse muuttaa, voidaan valita toinen näkymistä päänäkymäksi ja koodata toissijaisen näkymän tiedot attribuutteina. Jos järjestys muuttuu, voidaan käyttää erityistä viittausjärjestelmää säilyttämään siirrettyjen elementtien alkuperäinen sijainti tekstissä. Artikkelin jakamisen suhteen on vain pakko valita jompikumpi koodaustapa. Päätökset pitää tehdä sen mukaan, mikä on kyseiselle aineistolle tärkeää. Monessa tapauksessa kannattaa mieluummin tehdä jonkinlainen kompromissi näiden kahden näkymän välillä kuin tehdä koodaamisesta turhan monimutkaista.

2.3 Implisiittinen tieto sana-artikkeleissa

Rakennemallit ja niiden mukainen koodaus eivät sinänsä vielä ratkaise kaikkia ongelmia. Jotta sähköisiä sanakirjoja voitaisiin muuntaa sellaiseen yhtenäiseen muotoon, josta on mahdollista generoida mm. leksikkotietokantoja, tarvitaan muutakin kuin rakenteiden nimeämistä sovitulla tavalla. Sanakirja-artikkeleihin sisältyvä implisiittinen tieto pitää saada muutettua näkyväksi, jolloin se on vasta koneellisesti hyödynnettävissä. Tämän aikaansaamiseksi pitää tutkia, miten sanakirjan sisältämät tiedot liittyvät toisiinsa ja millaisia periytymissääntöjä ne noudattavat.

Artikkelissa [IKR00] tätä on tutkittu kuvaamalla sanakirja-artikkeli yksinkertaisena puurakenteena, joka koostuu solmuista ja niihin liittyvistä ominaisuuksista. Hakusana ja siihen suoraan liittyvät tiedot (artikkelin alusta) tulevat juurisolmun ominaisuuksiksi. Puuhun syntyy uusi solmu, kun artikkeli jakautuu osiin homonymian perusteella tai artikkelissa on useampia merkityksiä tai artikkeli sisältää aliartikkelin (esim. alihakusana, josta ei ole omaa erillistä artikkelia). Esimerkkikuvassa 1 on Kielitoimiston sanakirjasta poimittu sanan *kuusi* kaksi homonyymistä merkitystä, joista toinen (*havupuu*) jakautuu vielä kahteen alimerkitykseen. Kuhunkin solmuun liittyvät ominaisuudet on esitetty muodossa ”ominaisuuden nimi: arvo”.



Kuva 1: Kuusi-artikkeli puurakenteena.

Ominaisuuksilla on seuraavanlaisia tapoja periytyä puussa ylhäältä alas:

- *Kumulatiivinen:* ominaisuudella voi olla useimpia arvoja ja se periytyy lisäämällä itsensä alempana oleviin solmuihin. Tästä esimerkkinä voisi mainita käyttöalan, joka periytyy ja kumuloituu puussa alempana olevien merkitysryhmäsolmujen käyttöalamääritysten kanssa.
- *Yliajava (overwriting):* ominaisuudella voi olla vain yksi arvo. Se periytyy puussa alaspäin, kunnes törmään solmuun, jossa ominaisuus on jo määritelty. Esimerkiksi sanaluokka voi olla merkitysryhmäsolmussa eri kuin juurisolmussa.
- *Paikallinen:* ominaisuus ei periydy vaan koskee vain sitä solmua, jossa on määrittely. Tällaisia ovat mm. esimerkit ja paikalliset viittaukset.

Näiden lisäksi tarvitaan vielä riippuvuussääntöjä joidenkin yliajavien ominaisuuksien suhteen. Esimerkiksi sanaluokan muuttuminen vaikuttaa myös mm. sanan taivuttami-

seen (verbit taipuvat eri tavalla kuin substantiivit). Erillisellä säännöllä voidaan myös määritellä, että yliajavan ominaisuuden muuttuessa tietyt, siitä riippuvat ominaisuudet, lopettavat periytymisen.

Kun puuta käydään läpi juuresta alaspäin aina yhden lapsisolmun verran eteenpäin siirtyen ja samalla periytetään tiedot ylemmästä solmusta edellä esitettyjen sääntöjen mukaan, niin jokaisen solmun kohdalla on tiedossa sekä juuri sillä tasolla määritellyt tiedot että läpikäynnin yhteydessä muista solmuista periytyneet tiedot. Tämän perusteella puurakennetta voidaan koneellisesti täydentää, niin että kukin solmu sisältää eksplisiittisesti sen tiedon, joka alkuperäisessä rakenteessa oli nähtävissä vain hierarkian kautta. Esimerkkipuun läpikäynti tuottaisi vasemmanpuoleiseen haaraan seuraavanlaiset lehtisolmut:

hakusana:	kuusi
taivutus:	24
sanaluokka:	S
selite:	havupuu, kuusen puuaine

hakusana:	kuusi
taivutus:	24
sanaluokka:	S
selite:	havupuu
käyttöala:	kuv.
esimerkki:	”Joka kuuseen kurkottaa...”

Samassa artikkelissa on esitetty myös tapa kuvata puurakenne XML-muotoisena, CONCEDE-projektissa [EEI00] suunniteltua yksinkertaista rakennekuvausta noudattaen. Kuvauksen mukaan puun solmut koodataan elementtinä <struct>, ominaisuuksien vaihtoehtoisuus ilmaistaan elementillä <alt> ja ryhmittely elementillä <brack>. Ominaisuuksien koodaamiseen käytetään TEI-suosituksesta poimittua elementtijoukkoa. Tällöin edellä esitetty *kuusi*-artikkelia kuvaava puu voitaisiin koodata sisäkkäisinä struct-rakenteina seuraavassa esimerkissä esitetyllä tavalla. Juurisolmuna käytetään elementtiä <entry>.

```
<entry>
  <hw>kuusi</hw>
        <struct>
                <itype>24</itype>
                <pos>S</pos>
                <struct>
                        <def>kuusen puuaine</def>
                </struct>
```

```
                <struct>
                        <usg>kuv.</usg><eg>Joka kuuseen kurkottaa...</eg>
                </struct>
        </struct>
        <struct>
                <itype>27</itype>
                <pos>N</pos>
                <def>lukusana</def>
        </struct>
</entry>
```

Näin muodostetulle rakenteelle on mahdollista kirjoittaa XSL-muunnos, joka toteuttaa edellä kuvatut periytymissäännöt. Kullakin sanakirjalla on oma tapansa esittää implisiittistä tietoa. Yllä kuvatun menetelmän avulla voidaan kuitenkin saada tiedot avattua, mutta se edellyttää että aineisto on koodattavissa edellä esitetyn rakennekuvauksen mukaisesti. Tähän antaa toivoa uusi ISO:n sanakirjastandardi, jota esitellään luvussa 3.2. Sen suunnittelun yhtenä periaatteena on ollut, että rakenneosien merkityksen pitää olla yksiselitteisesti tulkittavissa hierarkian perusteella, eikä se saa riippua elementtien järjestyksestä. Edellä esitetty puumalli pyrkii toimimaan juuri niin, että periytymisen kannalta merkitystä on vain solmujen välisellä hierarkialla eikä sillä, missä järjestyksessä ominaisuudet on solmuihin liitetty.

3 Olemassa olevat rakennesuositukset

Standardit ja suositukset ovat olennainen osa tutkimusinfrastruktuuria. Niiden avulla voidaan varmistaa, että aineistot ovat käytettävissä sekä eri tutkimuslaitosten kesken että myös pidemmän ajan kuluttua. Myös infrastruktuuriin kuuluvat aineistojen hyödyntämiseen tarvittavat sovellukset voidaan suunnitella yleiskäyttöisemmiksi. Standardien käyttö helpottaa osaltaan tutkimusyhteisöjen välistä yhteistyötä ja hyvien käytäntöjen leviämistä [FDM06].

Seuraavissa luvuissa esitellään kaksi eri lähtökohdista suunniteltua suositusta sanakirjojen koodaamista varten. Koska sanakirjastandardien kehittäminen on perustunut vahvasti TEI:ssä tehtyyn pohjatyöhön, esitellään ensin TEI-suosituksen sanakirjojen koodaukseen tarkoitettu rakennekuvaus. TEI:n pohjana on ollut tarve digitoida, säilyttää ja jakaa alunperin painetussa muodossa olevaa tekstiaineistoa yhtenäisessä sähköisessä muodossa. Sen jälkeen esiteltävä ISO-standardi taas perustuu nykyajan vaatimuksiin

hyödyntää sanakirja-aineistoja tuottamalla niistä erilaisia sähköisiä julkaisuja ja käyttämällä niitä kieliteknologisten sovellusten materiaalina.

3.1 TEI-suositus sanakirjoille

Text Encoding Initiative (TEI) on saanut alkunsa humanistisen tutkimuksen tarpeista, ja sen perusperiaatteisiin kuuluvat ymmärrettävyys, joustavuus ja laajennettavuus. Monimuotoisia tekstimateriaaleja varten tarvittiin yhtenäinen koodaustapa, jotta sähköisessä muodossa olevien aineistojen jakaminen ja yhteiskäyttö helpottuisi ja jotta niiden koneelliseen käsittelyyn voitaisiin kehittää yleiskäyttöisiä sovelluksia. TEI-suosituksen tarkoituksena oli tarjota yleiskäyttöinen malli, joka mahdollistaa kaikenlaisten tekstien koodaamisen eri näkökulmista ja erilaisia tarpeita varten. Ensimmäinen suositus TEI P1 julkaistiin jo vuonna 1990 ja uusin versio P5 [TEI07] vuonna 2007.

TEI koostuu osista, joista käyttäjä voi koota kuhunkin tarpeeseensa sopivan rakennekuvauksen. Ydinosa (core fragment) sisältää kaikille TEI-aineistoille yhteiset elementit. Perusosia (base fragments) on useita eri tekstityyppejä varten, mm. proosaa, näytelmiä, runoja, sanakirjoja ja puhuttua tekstiä varten. Lisäosat (additional fragments) tarjoavat lisäominaisuuksia esimerkiksi linkitykseen tai korpusaineiston koodaamiseen. Tämän lisäksi TEI tarjoaa mekanismin, jolla rakennekuvausta voi muokata ja laajentaa tarvittaessa siten, että koodaus säilyy yhteensopivana muiden TEI-koodattujen tekstien kanssa.

TEI:n sanakirjaosa on suunniteltu alun perin painettuna julkaistujen sanakirjojen sähköisten versioiden koodaamista varten. Artikkelien monimuotoisuuden ja variaation ongelma on ratkaistu TEI:ssä tarjoamalla kaksi eri artikkelirakennetta. Elementillä <entry> voidaan koodata artikkelin sisältö sellaisessa määrätyssä muodossa ja järjestyksessä, joka sopii useimmille perinteisille sanakirjoille. Elementillä <entryFree> merkattu artikkeli taas mahdollistaa vapaamuotoisemman sisällön järjestämisen. Aivan epätoivoisia tilanteita varten on vielä elementti <dictScrap>, jota voi käyttää artikkelin sisällä kokoamaan yhteen sekalaista tietoa, jota ei muulla keinoin onnistuta koodaamaan.

Homonyymiset artikkelit voidaan merkata kahdella tavalla, joko ryhmittelemällä ne yhteen käyttäen elementtiä <superEntry> tai koodaamalla ne saman artikkelin sisään, mutta elementin <hom> avulla erotettuna.

```
<superEntry>
        <entry n="1" type="hom"> ... </entry>
        <entry n=”2” type=”hom”> ... </entry>
<superEntry>

<entry>
        <hom n="1"> ... </hom>
        <hom n="2"> ... </hom>
</entry>
```

Esimerkki 1.

Tässä esitellään lyhyesti tärkeimmät artikkelitasolla esiintyvät elementit ja niiden käyttö esimerkkien avulla. Verkossa saatavilla oleva TEI Guidelines[4] tarjoaa helppokäyttöisen tavan selata rakennekuvausta HTML-muodossa. Seuraavissa esimerkeissä on esitetty ensin artikkeli siinä muodossa, kuin se on painetussa sanakirjassa, ja sen jälkeen sen TEI-koodattu versio. Esimerkkiartikkelit on poimittu Kielitoimiston sanakirjasta.

Artikkelin alussa on yleensä hakusana ja siihen liittyvät tiedot. TEI:ssä elementillä <form> ryhmitellään yhteen hakusanan eri kirjoitusasut (myös taivutusmuodot ja variantit), ääntämis- ja tavutusohjeet. Hakusanaan liittyvät kieliopilliset tiedot ryhmitellään elementin <gramGrp> alle.

Esimerkissä 2 on kaksi form-lohkoa, joista ensimmäinen sisältää varsinaisen hakusanan ja jälkimmäinen sen taivutusmuotoja. Kieliopillisista tiedoista on merkattu taivutusindeksi (<iType>) ja kaksi vaihtoehtoista ääntämisohjetta (<pron>). Hakusanan käyttöyhteyttä kuvaava lyhenne ”ruok.” on koodattu elementillä <usg>. Koska kyseessä on yksinkertainen viittausartikkeli, lopussa on pelkkä viittaus selittävään hakusanaan (<xr type=”synonym”>). Varsinainen viittaus on koodattu elementillä <ref>, jonka target-attribuutissa on osoitus viitattavan artikkelin id-attribuuttiin (vertaa esimerkki 2).

fondant[5] [fondant t. fõdã´] ***subst.*** (taivutus: fondanti/n, -a, -in) ***ruok.*** = fondantti.

```
<entry xml:id=”fondant”>
        <form type=”lemma”>
                <orth>fondant<orth>
        </form>
```

4 http://www.tei-c.org/P5/Guidelines/DI.html

```
        <gramGrp>
                <iType>5</iType>
                <pron>fondant</pron>
                <pron>fõdã´</pron>
                <pos>subst.</pos>
        </gramGrp>
        <form type="inflected">
                <orth>fondantin</orth>
                <orth>fondantia</orth>
                <orth>fondantiin</orth>
        </form>
        <usg type=”dom”>ruok.</usg>
        <xr type=”synonym”>= <ref target=”#fondantti”>fondantti</ref></xr>
</entry>
```

Esimerkki 2.

Esimerkin 3 artikkelissa on kaksi merkitysryhmää, jotka on koodattu elementillä <sense>. Hakusanalle on annettu selitteitä (<def>) ja käyttöesimerkki (<cit>). Koska elementti <cit> voi sisältää myös muuta esimerkkiin liittyvää tietoa, koodataan varsinainen esimerkkiteksti elementillä <quote>. Toisessa merkitysryhmässä oleva selite ei kuvaakaan varsinaisesti hakusanan toista merkitystä vaan selittää esimerkissä esiintyvän sanan.

fondantti^{5*C} ***subst.*** (***rinn.*** fondant) ***ruok.***
1. sulaa suklaata t. muuta täytettä sisältävä lämmin leivonnainen.
2. *Perunafondantti* liemessä kypsytetty muotoiltu peruna.

```
<entry xml:id=”fondantti”>
        <form>
                <orth>fondantti</orth>
        </form>
        <gramGrp>
                <gram type="inflectional type">5*C</gram>
                <gram type=”pos”>subst.</gram>
        </gramGrp>
        <xr type=”synonym”>rinn. <ref target=”#fondant”>fondant</ref></xr>
        <usg type=”dom”>ruok.</usg>
        <sense n="1">
                <def>sulaa suklaata t. muuta täytettä sisältävä lämmin leivonnainen.</def>
        </sense>
        <sense n="2">
                <cit type="example">
                        <quote>Perunafondantti</quote>
                        <def>liemessä kypsytetty muotoiltu peruna.</def>
                </cit>
        </sense>
</entry>
```

Esimerkki 3.

TEI-suositus antaa mahdollisuuden toteuttaa koodauksen eri tavoin mm. hakusanaan liittyvät kieliopilliset ominaisuudet voidaan koodata joko erillisillä nimetyillä elementeillä tai yleisemmällä elementillä <gram>, jota tarkennetaan attribuuteilla. Asian havainnollistamiseksi on esimerkissä 3 käytetty jälkimmäistä tapaa. Esimerkissä 4 on kuvitteellinen etymologia edellisen esimerkin hakusanalle *fondantti*. Etymologinen tieto voidaan esittää joko vapaamuotoisesti, tai sitten voidaan koodata erikseen mm. hakusanan alkuperään liittyvät sanat (mentioned), käytetty kieli (lang) ja lyhyet selitteet (gloss).

ransk. *fondant*, sulava.

```
<etym>
        <lang>ransk.</lang>
        <mentioned>fondant</mentioned>, <gloss>sulava</gloss>.
</etym>
```

Esimerkki 4.

Elementillä <re> voidaan koodata artikkelin sisäinen ”miniartikkeli”, hakusana, jolle jostain syystä ei ole ollut aiheellista kirjoittaa omaa artikkelia. Kyseessä on usein ns. alihakusana, joka sisältää johdoksen, kiteytyneen fraasin tai muuten päähakusanaan liittyvän sanamuodon. Artikkelin loppuun on mahdollista lisätä vapaamuotoisia kommentteja käyttämällä elementtiä <note>. Kommentit voidaan kohdistaa mihin tahansa kohtaan artikkelissa käyttämällä samanlaista viittaussysteemiä kuin artikkelin varsinaisissa viittauksissa.

TEI on levinnyt erityisesti akateemisessa maailmassa, ja sitä on käytetty eniten vanhojen kulttuurisesti arvokkaiden tekstiaineistojen, mukaan lukien sanakirjojen, tallentamisessa sähköiseen muotoon. TEI:n sanakirjakoodausta käyttää myös mm. avoimen koodin sanakirjaprojekti FreeDict[5]. Muita projekteja on lueteltu TEI:n sivuilta löytyvässä projektilistassa[6].

3.2 ISO 1951:2007 -sanakirjastandardi

Vuonna 2007 hyväksytty standardi *ISO 1951:2007 Presentation/representation of entries in dictionaries* [ISO07] on tarkoitettu kaikenlaisille sähköisille sanakirjoille. Itse standardin lisäksi mallia on esitelty mm. artikkelissa [LeM06]. Standardin luvataan ole-

5 http://freedict.org/
6 http://www.tei-c.org/Activities/Projects/

van yhteensopiva mm. ISO:n leksikkostandardin *Lexical Markup Framework* (ISO CD 24613, ei vielä hyväksytty), avoimen standardin *Open Lexicon Interchange Format* (OLIF) sekä termistöstandardin *Terminological Markup Framework* (ISO 16642:2003) kanssa.

Aikaisemmat ehdotukset sanakirjojen koodaamiseksi ovat pohjautuneet joko painettuihin sanakirjoihin (TEI) tai tiukasti määriteltyihin tietokantamaisiin leksikoihin (mm. SIMPLE). Uusi standardi pyrkii ottamaan huomioon molemmat näkökulmat ja tarjoamaan rakennemallin, joka sopii sekä leksikografin että kieliteknologin käyttöön. Tarkoituksena on ollut löytää tasapaino tiukan formaalin rakenteen ja käyttäjäystävällisyyden sekä perinteisten leksikografisten käytäntöjen välillä.

Periaatteena olevat neljä sääntöä takaavat, että malli on riippumaton sekä käytetyistä työvälineistä että käyttötavasta.

1. Looginen rakenne pidetään erillään typografisesta ulkoasusta. Välimerkkejä ja muuta ulkoasuun liittyvää ei koodata aineistoon, mikäli ne voidaan tuottaa jälkeenpäin automaattisesti rakenteen perusteella.
2. Rakenneosien merkitys on yksiselitteisesti tulkittavissa hierarkian perusteella, eikä se riipu elementtien järjestyksestä.
3. Rakenne on joustava ja modulaarinen, ja siitä voidaan ottaa käyttöön vain tarvittavat osat.
4. Rakenne voidaan kuvata XML-rakennekuvauksena (DTD tai skeema), jolloin aineiston käsittelyyn voidaan käyttää yleisesti saatavilla olevia XML-työkaluja.

Rakennekuvaus on esitetty formaalisti (eBNF-muodossa), ja siinä on kuvattu, mitä elementtejä sanakirjassa voi esiintyä ja miten niitä voidaan ryhmitellä. Rakenneosien nimeämisessä on pyritty soveltuvin osin noudattamaan standardia ISO 12620. Annettua rakennekuvausta kutsutaan nimellä *XmLex*. Standardin liitteessä C on useita esimerkkejä sen soveltamisesta XML-muotoon. Verkosta on myös saatavilla esimerkkitoteutus siitä, miten standardi voitaisiin esittää XML DTD:nä [XmL07].

Artikkelitasolla olevat elementit ovat saman tyyppisiä kuin TEI:ssä. Sana-artikkeli merkataan elementillä <DictionaryEntry>, ja useita artikkeleita voidaan koota yhteen elementin <NestEntry> avulla. Homonyymit koodataan joko omina artikkeleinaan tai ryhmittelemällä ne artikkelin sisällä käyttäen elementtiä <HomographGroup>.

```
<NestEntry>
        <DictionaryEntry homographNumber=”1”> ... </DictionaryEntry>
        <DictionaryEntry homographNumber=”2”> ... </DictionaryEntry>
</NestEntry>
<DictionaryEntry>
        <Headword> ... </Headword>
        <HomographGroup homographNumber=”1”> ... </HomographGroup>
        <HomographGroup homographNumber=”2”> ... </HomographGroup>
</DictionaryEntry>
```

Esimerkki 5.

Standardissa määritellyt rakenneosat voidaan luokitella tietoelementteihin, ryhmitteleviin elementteihin, muihin elementteihin. Tietoelementit sisältävät leksikaalista tietoa, kuten hakusanan, sen eri muodot ja niihin liittyvät tiedot, selitteen, esimerkit, käyttöalatiedot ja viittaukset. Ryhmittelevillä elementeillä voidaan koota ja ryhmitellä toisia elementtejä. Muilla elementeillä koodataan leksikaalista tietoa tarkentavaa muuta tietoa, ja niitä voi käyttää melkein missä vain muiden elementtien sisällä.

Artikkeli koostuu yhdestä tai useammasta hakusanasta ja niihin liittyvistä tiedoista sekä mahdollisista homonyymi- tai merkitysryhmistä. Hakusanan ja siihen suoraan liittyvien tietojen on oltava artikkelin alussa ennen merkitysryhmiä, mutta muuten järjestys ryhmittelevien elementtien sisällä on vapaa (vrt. periaate 2). Rakenteen havainnollistamiseksi esitetään seuraavassa aikaisemmin luvun 3.1 esimerkissä 3 TEI-koodattu artikkeli XmLexin mukaisesti koodattuna.

```
<DictionaryEntry identifier=”fondant”>
        <HeadwordCtn>
                <Headword>fondantti</Headword>
                <GrammaticalNote>5*C</GrammaticalNote>
                <PartOfSpeech value=”noun” freeValue=”subst.”/>
                <SeeAlso>rinn. <Ptr xlink:href=”fondant”>fondant</Ptr></SeeAlso>
                <RangeOfApplication>ruok.</RangeOfApplication>
        </HeadwordCtn>
        <SenseGroup>
                <Definition>
                        sulaa suklaata t. muuta täytettä sisältävä lämmin leivonnainen.
                </Definition>
        </SenseGroup>
        <SenseGroup>
                <ExampleCtn>
                        <Example>Perunafondantti</Example>
                        <Definition>liemessä kypsytetty muotoiltu peruna.</Definition>
                </ExampleCtn>
        </SenseGroup>
</DictionaryEntry>
```

Esimerkki 6.

Artikkelin tiedot on ryhmitelty hieman eri tavalla kuin TEI-koodatussa esimerkissä. Hakusanaan suoraan liittyvät tiedot on koottu yhteen elementillä <HeadwordCtn>. Tämän lisäksi päätasolla on vain merkitysryhmäelementtejä <SenseGroup>. Standardista ja liitteen esimerkeistä jää hieman epäselväksi, miten pitäisi koodata artikkelit, joissa ei ole erillisiä merkitysryhmiä. Tällöin on ilmeisesti mahdollista joko koota kaikki tieto elementin <HeadwordCtn> sisään tai luoda ylimääräinen <SenseGroup>, johon kootaan hakusanan merkitykseen liittyvät tiedot (selite, esimerkit, viittaukset yms.).

Sanaluokka on koodattu sijoittamalla sanaluokkatieto attribuutin arvoksi. Standardin mukaan arvon voisi sijoittaa myös elementin sisällöksi, mutta silloin pitäisi käyttää valmiiksi määriteltyjä vaihtoehtoja (noun, adjective, verb, jne.). Yleensä ottaen attribuuttien arvot voidaan antaa joko standardin määrittelemien arvojen mukaisena (attribuutit type ja value) tai käyttää sanakirjalle tyypillisiä itse määriteltyjä arvoja (attribuutit freeType ja freeValue). Tämä mahdollistaa sekä tietojen yhteensopivan koodauksen että omien luokittelujen säilyttämisen.

Koska rinnakkaisviittaus on synonyyminen, sen koodaamiseen olisi voinut käyttää elementtiä <See>. Jostain syystä tämä ei kuitenkaan ole rakenteessa sallittua. Elementtiä <See> voidaan käyttää vain merkitysryhmien sisällä, joten tässä esimerkissä on käytetty toista viittauselementtiä <SeeAlso>, jolla voidaan koodata kaikentyyppiset viittaukset. Standardin mukaan siis synonyymisiä viittauksia voi olla vain merkitysryhmissä. Toisaalta on ymmärrettävää, että synonyymisiä viittauksia käytetään vain tarkkojen merkitysten välillä, mutta tuntuu kuin tässä kohtaa ei olisi otettu huomioon sitä, että useasti hakusanalla on vain yksi merkitys, jonka ilmaisemiseen ei välttämättä tarvita yhtään merkitysryhmää. Muuten huomattavaa on, että osoitinelementti <Ptr> käyttää W3C:n XLink-suosituksen[7] mukaista osoitustapaa.

Yleensä ottaen kaikkien tietoelementtien sisältönä voi olla, rakennekuvauksen määrittelemien muiden tietoelementtien lisäksi, tekstiä, osoittimia (Ptr) tai joitakin XHTML:stä peräisin olevia elementtejä, mm. otsikkoja, kappaleita, taulukoita, kuvia ja korostuksia.

Standardin periaatteiden mukaan peräkkäisten rakenteiden merkitys ei riipu niiden järjestyksestä. Ryhmittelevät elementit ovat juuri tämän takia tärkeitä, koska niillä voidaan ilmaista hierarkian kautta peräkkäisten elementtien yhteyksiä toisiinsa. Ryhmitteleviä elementtejä on kolmea eri tyyppiä: Ctn-, Block- ja Group-loppuisia. Ctn-loppuista ele-

7 http://www.w3.org/TR/xlink/

menttiä käytetään kun jonkun elementin sisältämää tietoa halutaan tarkentaa. Sen avulla voidaan liittää hakusanaan sanaluokka ja ääntämisohje, sitaattiesimerkkiin lähdeviite, viittaukseen käyttöalamaininta jne. Yleisesti ottaen elementti <XyzCtn> sisältää aina vähintään yhden elementin <Xyz> ja mahdollisesti muita sitä tarkentavia elementtejä. Esimerkissä 7 on liitetty hakusanaan sitä tarkentavia tietoja (sanaluokka ja ääntämisohje) kokoamalla ne yhteen elementillä <HeadwordCtn>.

```
<HeadwordCtn>
        <Headword>brownie</Headword>
        <PartOfSpeech freeValue=”subst.”>
        <Pronunciation>brauni</Pronunciation>
</HeadwordCtn>
```

Esimerkki 7.

Painetut sanakirjat käyttävät usein tiettyjä välimerkkejä (yleensä kaksois- tai puolipisteitä) ilmaisemaan peräkkäin lueteltujen tietojen välistä ryhmittelyä. Esimerkissä 8 on esitetty hakusanalle ensin kaksi preesenstaivutusta ja sitten kolme imperfektitaivutusta. Tekstissä eri aikamuodossa olevat taivutukset on erotettu toisistaan puolipisteellä. Tälläista loogista ryhmittelyä voidaan kuvata Block-loppuisilla elementeillä. Yleisesti elementti <XyzBlock> sisältää vähintään yhden elementeistä <Xyz>, <XyzCtn> tai <XyzBlock> ja voi sen lisäksi sisältää muita tarkentavia elementtejä.

käydä v. Taivutus: prees. *käyy*, *kävöö*; imperf. *käi, käy*, *käävö*.

```
<DictionaryEntry>
        <Headword>käydä</Headword><PartOfSpeech freeValue=”v.”>
        <InflectionBlock>
                <GrammaticalNote>prees.</GrammaticalNote>
                <Inflection>käyy</Inflection>
                <Inflection>kävöö</Inflection>
        </InflectionBlock>
        <InflectionBlock>
                <GrammaticalNote>imperf.</GrammaticalNote>
                <Inflection>käi</Inflection>
                <Inflection>käy</Inflection>
                <Inflection>käävö</Inflection>
        </InflectionBlock>
</DictionaryEntry>
```

Esimerkki 8.

Group-loppuisilla elementeillä ryhmitellään joukko itsenäisiä elementtejä omiksi kokonaisuuksikseen. Tällaisia ovat merkitysryhmät (SenseGroup) ja homonyymin tiedot kokoava ryhmä (HomographGroup).

Kaikille elementeille yhteisiä attribuutteja ovat elementin kielen määrittävä *xml:lang*, yksikäsitteinen tunniste *identifier* sekä XHTML:stä periytyvät *style* ja *class*. Lisäksi voidaan käyttää attribuuttia *documentSize*, jolla voidaan kertoa kuinka laajaan (suppea, laaja, tms.) sanakirjaan elementti kuuluu, ja *display*, jolla voidaan säädellä elementin näkyvyyttä sanakirjajulkaisussa. Näiden avulla voidaan vaikuttaa useiden erilaisten sanakirjatuotteiden luomiseen yhdestä aineistosta.

Standardin luettavuus on paikka paikoin kehno kirjoitusvirheiden takia. Rakenneosien selitykset ovat monesti turhan suppeita ja annetut esimerkit joissain kohdin jopa ristiriidassa rakennemäärittelyn kanssa. Näiden lisäksi vaikeuksia tulkinnassa ja esimerkkien validoinnissa aiheutti DTD-esimerkkitoteutus, joka ei kaikilta osin ollut aivan yhtäpitävä standarditekstin kanssa.

4 Suomen murteiden sanakirjan rakenteistus

Sanakirjaprojektit ovat Kotuksessa perinteisesti pitkiä. Sanakirjan valmistuminen saattaa kestää useita vuosikymmeniä. Isoilla sanakirjahankkeilla on usein hyvinkin tarkat toimitusperiaatteet ja ohjeistus, joiden mukaan toimittajat sanakirjaa kirjoittavat. Toimitusohjeet keskittyvät kuitenkin enimmäkseen kuvamaan sisällön valintaa ja tiedon esitystapaa. Toimittajalle jää melko vapaat kädet muotoilla artikkeli oman mielensä mukaan.

Suomen murteiden sanakirja (SMS) on Kotuksenkin mittakaavassa erityisen pitkä hanke, koska se pantiin alulle jo 1800-luvun puolella [Str04]. Sanakirjan toimitus on myös iso. Tällä hetkellä siihen kuuluu yhdeksän toimittajaa. Valmistuessaan SMS on laaja kielitieteellinen lähdeteos, joka pyrkii kuvaamaan suomen kaikkien murteiden sanaston [Tuo89]. Sanakirja perustuu Suomen murteiden sana-arkiston sanalippuaineistoon, joka sisältää noin kahdeksan miljoonaa, enimmäkseen 1920-, 1930- ja 1960-luvuilla kerättyä sanalippua. Lipuissa sanan levikki on kerrottu keruupitäjän tarkkuudella. Pitäjä on myös sanakirjassa käytetty pienin havainnointialue, jonka lisäksi levikkejä on ilmaistu myös laajemman murrealuejaon avulla.

Suunnitelmien mukaan sanakirja on valmistuessaan 20-osainen ja sisältää noin 350 000 hakusanaa. Ensimmäinen osa *a–elää* ilmestyi vuonna 1985 ja kahdeksas osa *konkeri–kurvottaa* alkuvuodesta 2008. Viimeinen osa ilmestyy arvion mukaan 2030-luvun lopul-

la. Sanakirjaa on alusta alkaen kirjoitettu tietokoneella. Kolmannen osan kohdalla siirryttiin käyttämään Vax-keskuskoneella toimivaa WordPerfect-tekstikäsittelyohjelmaa. Vuonna 2008 julkaistu kahdeksas osa on vielä kirjoitettu loppuun asti WordPerfectillä. Ensimmäinen kokonaan rakenteisesti XML-muodossa toimitettu osa on yhdeksäs, joka ilmestyy suunnitelmien mukaan vuonna 2010 tai 2011.

4.1 Rakenteistusprojekti

Kotuksessa aloitettiin sanakirjojen rakenteistus jo 1990-luvun puolivälissä. Ensimmäisiä rakenteiseen toimitusjärjestelmään siirrettyjä sanakirjoja olivat Vanhan kirjasuomen sanakirja ja Suomen kielen perussanakirja (nykyään nimeltään Kielitoimiston sanakirja). Rakenteistamisen tarkoituksena oli valmistautua sanakirjojen verkkojulkaisemiseen, parantaa niiden käytettävyyttä sekä mahdollistaa systemaattisia tarkistuksia sanakirja-aineiston laadun parantamiseksi.

Suomen murteiden sanakirjan rakenteistusprojekti aloitettiin jo 1999, jolloin alettiin pohtia, millainen rakennekuvaus olisi sopiva tämän sanakirjan tarpeisiin. Silloin oli tärkeintä, että sanakirjaa voidaan jatkossakin julkaista painettuna kirjana ja että artikkelien muoto ja ulkoasu voidaan säilyttää mahdollisimman samanlaisena kuin aiemmissa osissa. Haluttiin myös valmistautua siihen mahdollisuuteen, että sanakirjaa voitaisiin jossain vaiheessa alkaa julkaista verkossa ja että sitä kautta voitaisiin tarjota erilaisia käyttö- ja hakumahdollisuuksia.

Rakenteistusprojekti pääsi varsinaisesti käyntiin vuonna 2000, jolloin aloitettiin rakennekuvauksen suunnittelu. Suunnitteluvaihe kesti reilun vuoden ja siihen osallistui kolme atk-suunnittelijaa, kolme SMS:n toimittajaa sekä silloisen Suomen kielen perussanakirjan päätoimittaja. Atk-yksikön resurssipula viivästytti rakenteisen toimitusohjelman käyttöönottoa niin, että testikäyttö päästiin aloittamaan vasta vuoden 2003 alussa. Seuraavana vuonna jo aloitetun yhdeksännen osan artikkelit muunnettiin rakenteiseen muotoon ja varsinainen rakenteinen toimitustyö alkoi. Vuonna 2005 muunnettiin rakenteisiksi osat 5–7 ja seuraavana vuonna loput osat, lukuunottamatta kahdeksatta, jota vielä kirjoitettiin vanhassa järjestelmässä.

Vanhojen WordPerfect-muotoisten osien muuntamisessa XML-muotoon oli suuresti apua WordPerfectin makrokielen hallitsevasta toimittajasta, joka makroja apuna käyttäen suoritti alustavan rakenteiden merkkaamisen aineistoon. Näin pystyttiin

varmistamaan, että muotoilut ja korostukset eivät hävinneet aineistoa järjestelmästä toiseen siirrettäessä.

Rakenteistusprosessi jaettiin karkeasti kolmeen osaan:

1. rakennekuvauksen suunnitteluun,
2. olemassa olevan aineiston muuntamiseen rakennekuvauksen mukaiseksi ja
3. rakenteistetun aineiston tietokantaistukseen.

4.2 Rakennekuvauksen suunnittelu

Rakennekuvauksen valinnassa oli vuosituhannen vaihteessa oikeastaan vain kaksi vaihtoehtoa: joko TEI tai oman rakennekuvauksen määrittely. TEI-suosituksen mukainen sanakirjarakenne oli houkutteleva, koska se oli tehty erityisesti painettuja sanakirjoja varten ja sitä kehittämässä oli ollut leksikografian asiantuntijoita ympäri maailmaa. TEI:n mukaisesti koodattu sanakirja olisi myös ollut yhteensopiva muiden TEI-muotoisten sanakirja-aineistojen kanssa. Myös oman rakennekuvauksen kehittämisestä oli jonkin verran kokemuksia. Kotuksessa oli aiemmin suunniteltu rakenne jo kahdelle sanakirjalle: Vanhan kirjasuomen sanakirjalle ja Suomen kielen perussanakirjalle. Kokemukset oman rakenteen suunnittelusta eivät kuitenkaan olleet pelkästään positiivisia. Rakenteista tuli helposti joko liian yksityiskohtaisia tai liian sallivia, ja kummallakin ääripäällä oli huonot puolensa.

Tärkeimmät vaatimukset voitiin kiteyttää seuraaviin kahteen ehtoon:

1. Kirjoittamisen on oltava suhteellisen helppoa. Toimittajilla on täysi työ jo uuden järjestelmän opettelussa eikä rakenteen pitäisi vaikeuttaa kirjoittamista suhteettomasti.
2. Mitään tietoa ei saa hävittää. Myös ulkoasuun liittyvä tieto pitää säilyttää, ja toisaalta kaikki relevantti tieto pitää voida koodata.

TEI-koodaus osoittautui ongelmalliseksi kirjoittamisen helppouden suhteen. Rakennetta pidettiin liian monimutkaisena, vaihtoehtoja oli kirjoittaessa liikaa ja runsas attribuuttien käyttö herätti hämmennystä. XML-rakenteisiin tottumattoman on yleensä helppo ymmärtää elementit ikään kuin täytettävinä lomakkeen kenttinä, mutta ”näkymättömiä” attribuutteja arvoineen oli vaikea ymmärtää. Koodauksen riittävyyden suhteen ei

TEI:ssä olisi tullut ongelmia, koska tarvittaessa rakennekuvausta olisi voinut vielä laajentaa omilla määrittelyillä.

Oman rakennekuvauksen avulla oli helpompi toteuttaa ensimmäinen ehto. Rakennetta kuvattaessa voitiin sopia toimittajien kanssa yhteinen sanasto, jolla elementit nimettiin. Tällöin voitiin puhua rakenteista toimittajien kanssa samalla kielellä ja kummatkin osapuolet tiesivät, mistä puhuttiin. Rakenteesta voitiin myös suunnitella kirjoittamisen kannalta mahdollisimman helppokäyttöinen, eli pyrittiin välttämään sekä syviä hierarkkisia rakenteita että attribuuttien käyttöä artikkelin tekstisisällön koodaamisessa. Monimutkaisen sanakirjarakenteen mallinnus ei ole helppoa, joten toisen ehdon täyttämiseksi täytyi rakenteen suunnitteluun varata runsaasti aikaa. Hankalaa oli myös tasapainoilu ”tiukan” ja ”löysän” rakennekuvauksen välillä. Rakenteen pitää toisaalta olla tarpeeksi tiukka, jotta se voisi ohjata toimittajaa tuottamaan oikean muotoisia artikkeleita, toisaalta taas toimittajalla pitää olla jonkin verran vapautta muokata artikkeli lukijan kannalta helppolukuiseen muotoon.

Rakenteen määrittely tehtiin tiiviissä yhteistyössä sanakirjan toimituksen kanssa. Suunnittelun apuna ja rakenneosien identifioinnissa käytettiin hyväksi kirjallisia toimitusohjeita [Suo88] sekä paljon ns. hiljaista tietoa toimittajilta, jotka tunsivat syvällisesti sanakirjan sisällön ja toimitushistorian. Tämän tuloksena tuotettiin listaus, jossa nimettiin sanakirjan keskeiset rakenneosat, se mitä niillä rakennekuvauksen yhteydessä tarkoitetaan sekä millaiselta ne näyttävät painetussa sanakirjassa. Listauksen perusteella suunniteltiin rakennekuvaus XML DTD -muodossa. Kuvausta tarkennettiin hierarkian ja järjestyksen osalta mm. tekemällä toimitukselle lisäkysymyksiä, kuten ”Seuraako esimerkkiä *aina* pitäjälyhenne?” tai ”Voiko selitteessä olla viittauksia?”. Koska sanakirjan osia tullaan julkaisemaan jatkossakin painetussa muodossa, koodauksen tavoitteena oli säilyttää tekstuaalinen näkymä. Tästä tingittiin joidenkin automaattisesti lisättävissä olevien tietojen kohdalta. Esimerkiksi merkitysryhmien numerointi oli järkevämpää tuottaa automaattisesti kuin koodata kirjoitusvaiheessa elementin tai attribuutin arvoksi.

Prosessin lopputuloksena päätettiin, että tavallinen SMS:n vakioartikkeli <VakioArt> voi sisältää (tässä järjestyksessä)

- yhden tai useampia hakusanoja (Hakus),
- yhden tai useampia sanaluokkalyhenteitä (Sanal),

- mahdollisesti yhden tai useampia tarkenteita (Tark),
- merkityksenselitteen (SelJakso),
- luetteloita hakusanan eri muodoista levikkitietoineen (MuotoJakso),
- varsinaisen hakusanan levikkitiedot (Lev),
- huomautuksia (Huom),
- viittauksia muihin artikkeleihin (ViitJakso),
- pitkissä artikkeleissa sisällysluettelon (Jasennys),
- merkitysryhmiä (MerkRyhma) ja
- esimerkkejä (EL).

Perusrakenne on melko samantyylinen kuin muissakin edellä esitetyissä kuvauksissa. Omanlaisensa lisän tuovat erityisesti murresanakirjaan kuuluvat levikkitiedot (Lev), runsas muotovarianttien esittäminen (MuotoJakso) ja viittausten kokoaminen artikkelin alkuun (ViitJakso). Jäsennys-elementti on tarkoitettu vain isoihin artikkeleihin automaattisesti generoitavia sisällysluetteloja varten eikä se siis ole varsinainen sisällöllinen elementti. Artikkelin päätasolla rakenne ohjaa hyvin vahvasti kirjoittajaa, koska asiat voidaan esittää vain tietyssä kiinnitetyssä järjestyksessä. Merkitysryhmien (MerkRyhma) perusrakenne on samanlainen kuin artikkelin päätason.

Lyhyitä viittausartikkeleita varten luotiin oma elementti <EtunArt>, joka hakusanan ja viittauksen lisäksi voi sisältää vain sanaluokan tarkenteineen sekä selitteen (katso esimerkki 9). Etunuoliartikkelin nimi viittaa siihen, että artikkelissa on vain ns. etunuoliviittaus, joka osoittaa että hakusanan tiedot löytyvät toisen artikkelin yhteydestä.

Vakio- ja etunuoliartikkelien lisäksi tarvittiin vielä hankalasti muotoiltavia poikkeusartikkeleja varten elementti <PoikkArt>, jota tosin saa käyttää vain ääritapauksissa päätoimittajan luvalla. Poikkeusartikkelit ovat melko harvinaisia. Niitä on osissa 1–7 noin 100, kun kaikkiaan artikkeleita näissä osissa on 112 000. Osa näistäkin sadasta olisi todennäköisesti mahdollista uudelleen muokkaamalla muuntaa vakioartikkeleiksi. Seuraavassa esittelen normaalin vakioartikkelin rakenteen osat sekä annan muutaman esimerkin koodaamisesta. SMS:n täydellinen rakennemäärittely DTD-muodossa on liitteessä 1.

Jokainen artikkeli alkaa yhdellä tai useammalla hakusanalla (Hakus). Hakusanoja voi olla useita, esimerkiksi jos artikkeliin on yhdistetty kaksi samanarvoista perusmuotoa (*ahkeroi|da, -ta*). Hakusanan attribuutilla Hom ilmaistaan, että kyseessä on homonyyminen hakusana, ja annetaan homonyymille järjestysnumero. Numero pitää olla toimittajan päätettävissä, joten sitä ei voi generoida automaattisesti. Attribuutilla Yhd merkitään yksittäiset yhdyssanajonojen ulkopuolella olevat yhdyssanat, ja attribuuttiin Norm generoidaan hakusanan normaalistettu hakumuoto, josta on poistettu pystyviivat ja mahdolliset tarkkeet (erikoismerkit, joilla kuvataan murretekstissä sanojen yleiskielestä poikkeavaa ääntämystä). Jokaisella artikkelilla on viittaamista varten yksikäsitteinen tunniste, joka tuotetaan automaattisesti attribuuttiin Id.

Sanaluokkalyhenne (Sanal) ja sitä seuraava tarkenne (Tark) voivat esiintyä artikkelin tai merkitysryhmän alussa. Tarkenteella ilmaistaan usein hakusanan monikollisuutta tai annetaan sanaluokkatiedolle joku muu tarkennus. Irrallisia tarkenteita voi lisäksi sijaita myös selitteissä. Niiden lisäksi merkityksen selite (SelJakso) voi sisältää tekstiä, käyttöalalyhenteitä (KayttoAla), tietynlaisia viittauksia (TakanViite) ja alihakusanoja (AliHakus).

Muotojakso (MuotoJakso) sisältää erityyppisiä hakusanan muotojen luetteloita (katso esimerkki 10). <Asu> koostuu hakusanan erilaisista paikallisista perusmuodoista, <Komp> komparaatiomuodoista, <Taiv> erilaisista muista taivutusmuodoista, <Vart> ja <Tyvi> hakusanan katkaistuista muodoista ja <AsuTaiv> perusmuodoista, joista joillekin on annettu samalla myös taivutus. Viimeksi mainittu on hyvä esimerkki tiedon tiivistämisestä painetuissa sanakirjoissa. Luettelot sisältävät hakusanan muodon (Muoto) ja siihen liittyvän levikkitiedon (Lev) sisältäviä ryhmiä (MuotoLev). Levikkitieto on sekoitus vapaamuotoista tekstiä, alue- (Alue) ja pitäjälyhenteitä (Pit) sekä käyttöalamerkintöjä. Muotojakson lopussa voi olla vielä vapaamuotoinen yleinen luonnehdinta (MuotoHuom).

Artikkelitasolla olevat viittaukset voivat esiintyä vain tietyssä järjestyksessä: ensin viittaukset hakusanan (yleensä esine tai laite) osiin (Osia), rinnakkaisviittaukset samaa tarkoittaviin hakusanoihin (Rinn), katso-viittaukset (Ks), vertaa-viittaukset (Vrt) ja viimeisenä yhdyssanaviittaukset (Yhd). Näiden ryhmittelevien elementtien sisällä olevat varsinaiset viittaukset koodataan elementillä <ViiteHakus>, jonka attribuutilla xref voidaan osoittaa joko hakusanan tai merkitysryhmän Id-attribuuttiin (katso esimerkki 9).

Viittauksia voi olla myös muualla artikkelissa. Erityisesti vertaa-viittauksia voi esiintyä myös esimerkeissä. Takanuoliviittaus (TakanViite) on vastaviittaus takaisin siihen artikkeliin, josta on viitattu rinnakkaisviittauksella. Takanuoliviittauksia voi esiintyä selitteissä, esimerkeissä sekä rinnakkais- että vertaa-viittauksien yhteydessä. Etunuoliviittauksia (EtunViite) esiintyy ainoastaan etunuoliartikkeleissa (EtunArt).

Esimerkistön kohdalla jouduttiin tekemään kompromisseja koodauksen tarkkuuden ja toimitustyön sujuvuuden suhteen. Esimerkkilohko (EL) koostuu jaksoista (EsimJakso), jotka puolestaan koostuvat varsinaisista esimerkeistä (Esim) ja lähteenä mainittavasta pitäjälyhenteestä (Pit). Rakenne <Esim> voi sisältää murreainesta (Mur), suoria lainauksia sanalippuaineistosta (Sit), lyhyitä sulkeissa olevia selitteitä (SuljeSel) sekä vapaata tekstiä. Vapaassa tekstissä voi olla joko yleiskielistettyjä osia murre-esimerkistä tai esimerkkiä selittävää tekstiä. Näiden luokittelu ja erottaminen toisistaan ei ole itsestään selvää, ja toimituksen toivomuksesta tässä kohtaa rakenne on tarkoituksellisen löysä.

Attribuutteja on pyritty käyttämään mahdollisimman vähän, ja osa niiden arvoista tuotetaan automaattisesti. Kirjoittajan vastuulla on lähinnä artikkelin ulkoasuun liittyvien attribuuttien arvojen lisääminen. Näihin kuuluvat artikkelin attribuutti mr-numerointi, jonka arvoilla voidaan säätää, miltä tasolta merkitysryhmänumerointi alkaa (numerot, roomalaiset numerot, suuraakkoset), hakusanan attribuutti Hom, joka sisältää hakusanan homonyyminumeron, ja esimerkkilohkon tyyppi-attribuutti, jonka avulla erotetaan poikkeavat esimerkit muista esimerkeistä. Hakusanojen ja merkitysryhmien Id-attribuuttien arvot generoidaan automaattisesti, samoin hakusanan normalisoitu muoto attribuuttiin Norm sekä levikkitietojen järjestämisessä tarvitut alueiden ja pitäjien järjestysnumerot. Viittausten xref-attribuutit pyritään generoimaan myös automaattisesti viitehakusanan perusteella, mutta järjestelmä on vielä kehittämisen tarpeessa, ja usein arvoja pitää korjailla käsin. Esimerkissä 9 on kuvattu lyhyiden viittausartikkelien ja homonyymisten hakusanojen koodaamista SMS:ssä.

aalu[1] s. → aaluva
aalu[2] s. → aatu

```
<EtunArt>
        <Hakus Id="sms1_1.aalu" Hom="1">aalu</Hakus>
        <Sanal>s.</Sanal>
        <EtunViite><ViiteHakus xref="smsosa_aaluva">aaluva</ViiteHakus></EtunViite>
</EtunArt>
<EtunArt>
```

```
        <Hakus Id="sms1_2.aalu" Hom="2">aalu</Hakus>
        <Sanal>s.</Sanal>
        <EtunViite><ViiteHakus xref="smsosa_aatu">aatu</ViiteHakus></EtunViite>
</EtunArt>
```

Esimerkki 9: Etunuoliartikkelien ja homonyymien koodaus.

Kuvassa 2 on SMS:stä poimittu esimerkkiartikkeli *elehursti*, josta näkee, millaiselta sana-artikkelit näyttävät painetussa sanakirjassa. Artikkeli on koodattu SMS:n rakenteen mukaisesti esimerkissä 10.

ele|hursti s. Asu: *ele(h)hursti* yleensä, *ele(h)vursti* Lep VehS, *eleshursti* Lep. Rinn. eleloimi, -vaate, elohursti, -loimi, -raanu, -vaate. Vrt. helehursti. **1.** iso vaate, jota käytetään jyvien talteenottovälineenä viljaa tuultaen puhdistettaessa sekä reen pohjalle, riihen oven eteen, auman viereen jne. levitettynä puimatonta viljaa siirrettäessä. KarP etelä- ja keskiosa, SavP (ei pohjoislaita eikä lounaiskulma), Pari Uuk Harlu SavR Ker Konn. | *elehhurstilòita tehtiin siitä ruohtimesta ja säkkijä.* Kite | *Kuv vieťtiin ahosta riihheen nir riihen oven ulukopuolellel làitettiin elehhursti siihen karis jyvvii.* PyhS | *neľpieťimiset on ne elehhurstit ja loajat ja piikoittain kuvoťtaan.* Pol | *Elehhurstille tippu päetä, lyhteem päetä.* Siil | *Viskoo lyhteet elehurstin piälle!* VehS **2.** suurikokoinen vaate, jota käytetään viljakuorman, puimakoneen, riihen kiukaan ym. peitteenä. | Tohm | *Kuv veś ja jàohot o hämmennetty* (viinankeittoa aloiteltaessa) *pèitetää tiinu elehhurstilla.* Kite | Kesä Kont Lep MuuV

Kuva 2: Esimerkkiartikkeli painetusta sanakirjasta.

```
<VakioArt>
 <Hakus Id="sms1_elehursti" Yhdys="on" Norm=”elehursti”>ele|hursti</Hakus>
 <Sanal>s.</Sanal>
 <MuotoJakso>
  <Asu>Asu:
   <MuotoLev><Muoto>ele(h)hursti</Muoto><Lev>yleensä</Lev></MuotoLev>,
   <MuotoLev><Muoto>ele(h)vursti</Muoto> <Lev><Pit jarjnro="313">Lep</Pit> <Pit
jarjnro="321">VehS</Pit>,</Lev></MuotoLev>
    <MuotoLev><Muoto>eleshursti</Muoto>
     <Lev><Pit jarjnro="313">Lep</Pit></Lev></MuotoLev>.
  </Asu>
```

```
 </MuotoJakso>
 <ViitJakso>
  <Rinn>
   <ViiteHakus xref="sms1_eleloimi">eleloimi</ViiteHakus>
   <ViiteHakus xref="sms1_elevaate">-vaate</ViiteHakus>
   <ViiteHakus xref="sms1_elohursti">elohursti</ViiteHakus>
   <ViiteHakus xref="sms1_elololoimi">-loimi</ViiteHakus>
   <ViiteHakus xref="sms1_eloraanu">-raanu</ViiteHakus>
   <ViiteHakus xref="sms1_elovaate">-vaate</ViiteHakus>
  </Rinn>
  <Vrt><ViiteHakus xref="sms3_helehursti">helehursti</ViiteHakus></Vrt>
 </ViitJakso>
 <MerkRyhma>
  <SelJakso>iso vaate, jota käytetään jyvien talteenottovälineenä viljaa tuultaen
puhdistettaessa sekä reen pohjalle, riihen oven eteen, auman viereen jne. levitettynä
puimatonta viljaa siirrettäessä.</SelJakso>
  <Lev>
   <Alue jarjnro="41">KarP</Alue> etelä- ja keskiosa,
   <Alue jarjnro="47">SavP</Alue> (ei pohjoislaita eikä lounaiskulma),
   <Pit jarjnro="258">Pari</Pit> <Pit jarjnro="262">Uuk</Pit>
   <Pit jarjnro="264">Harlu</Pit> <Pit jarjnro="293">SavR</Pit>
   <Pit jarjnro="295">Ker</Pit> <Pit jarjnro="355">Konn</Pit>.
  </Lev>
  <EL>
   <EsimJakso>
    <Esim><Mur>elehhurstilòita teht&amp;#x0306;tiin siitä ruohtimesta ja
säkkijä.</Mur></Esim>
    <Pit jarjnro="275">Kite</Pit>
   </EsimJakso>
   <EsimJakso>
    <Esim><Mur>Kuv viĕttiin ahosta riiĥheen
nir riihen oven ulukopuolellel làitettiin
elehhursti siihen karis jyv̆vii.</Mur></Esim>
    <Pit jarjnro="278">PyhS</Pit>
   </EsimJakso>
   <EsimJakso>
    <Esim><Mur>nelʹpieťimiset on ne
elehhurstit ja loajat ja piikoittain kuvo͡ttaan.</Mur></Esim>
    <Pit jarjnro="287">Pol</Pit>
   </EsimJakso>
   <EsimJakso>
    <Esim><Mur>Elehhurstille tippu päetä, lyhteem päetä.</Mur></Esim>
    <Pit jarjnro="319">Siil</Pit>
   </EsimJakso>
   <EsimJakso>
    <Esim><Mur>Viskoo lyhteet elehurstin piälle!</Mur></Esim>
    <Pit jarjnro="321">VehS</Pit>
   </EsimJakso>
  </EL>
 </MerkRyhma>
 <MerkRyhma>
  <SelJakso>suurikokoinen vaate, jota käytetään viljakuorman, puimakoneen, riihen kiukaan
ym. peitteenä.</SelJakso>
  <EL>
   <EsimJakso><Pit jarjnro="274">Tohm</Pit></EsimJakso>
   <EsimJakso>
    <Esim><Mur>Kuv ves ja jàohot o hämmennetty <SuljeSel>viinankeittoa
aloiteltaessa</SuljeSel> pèitetää tiinu elehhurstilla.</Mur></Esim>
```

```
      <Pit jarjnro="275">Kite</Pit>
    </EsimJakso>
    <EsimJakso><Pit jarjnro="276">Kesä</Pit></EsimJakso>
    <EsimJakso><Pit jarjnro="283">Kont</Pit></EsimJakso>
    <EsimJakso><Pit jarjnro="313">Lep</Pit></EsimJakso>
    <EsimJakso><Pit jarjnro="324">MuuV</Pit></EsimJakso>
   </EL>
  </MerkRyhma>
</VakioArt>
```

Esimerkki 10: Esimerkkiartikkeli koodattuna SMS-rakenteen mukaisesti.

Varsinaisten sisällöllisten elementtien lisäksi SMS:ssä käytetään myös muutamaa ulkoasullista korostustapaa: yläindeksiä (Yi), alaindeksiä (Ai), kursivointia (Korost) ja harvennusta (Harvennus). Ylä- ja alaindeksejä käytetään esimerkeissä murrepiirteiden merkitsemiseen. Suurin osa niistä voitaisiin nykyään muuntaa oikeiksi Unicode-merkeiksi, mutta sopivien fonttien vähäisyyden takia on parempi säilyttää ne helpommin visualisoitavassa muodossa. Kursivointia käytetään selitteissä ja huomautuksissa korostamaan kielenainesta. Esimerkiksi seuraavassa muotohuomautuksessa voidaan kursivoidut sanat koodata elementillä <Korost>: ”Sekaantumista *kytkyt*- ja *kytkin*-sanojen taivutukseen.” Jos esimerkissä on useampia kuin yksi hakusanan esiintymä, harvennusta käytetään osoittamaan, mikä niistä on tarkoitettu ensisijaiseksi.

Rakennekuvausta testattiin käytännössä antamalle toimituksen testihenkilölle käyttöön XML-editori. Testausta varten tehtiin myös CSS- ja XSL-tyylisivut, joiden avulla artikkeliteksti saatiin näytöllä muistuttamaan painetun artikkelin ulkoasua. Testivaiheen aikana rakennekuvauksesta korjattiin useita virheitä ja sitä muutettiin helppokäyttöisemmäksi. Tämänkään jälkeen rakenne ei ollut täydellinen, vaan sitä jouduttiin muokkaamaan vielä tuotantovaiheessa, kun vastaan tuli erilaisia poikkeustapauksia.

4.3 Muuntaminen rakenteiseen muotoon

Muunnosvaiheessa olemassa oleva aineisto, eli sanakirjan valmiit osat, muunnettiin rakenteiseen muotoon. Tämä aiheutti myös jonkin verran muutoksia rakennekuvaukseen, koska muunnettavasta aineistosta löytyi sellaisia tapauksia, joita ei vain ollut tultu ajatelleeksi. Myös toimitusohjeiden tarkentaminen ja se, että kaksi ensimmäistä osaa oli alunperin kirjoitettu eri järjestelmällä, hankaloitti jonkin verran koodausta.

Muunnosskriptit pyrkivät koodaamaan aineiston käyttämällä hyväksi lähinnä tekstin typografisia ominaisuuksia. Käytössä oli mm. seuraavanlaisia taktiikoita:

- muotoiluvihjeiden käyttö (kappalevaihdot, kursivointi, lihavointi ja alleviivaus); esim. kappaleenvaihto aloittaa uuden artikkelin, lihavoitu sana on hakusana ja alleviivattu aines on murre-esimerkkiä
- tekstivihjeiden käyttö; esim. merkkijono ”Vrt.” aloittaa vertaa-viittauksen, joka päättyy pisteeseen, ja tällä välillä viitehakusanat on erotettu toisistaan pilkuilla
- lueteltujen vakiomerkkijonojen tunnistaminen; esim. levikin alue- ja pitäjälyhenteet, käyttöalalyhenteet, sanaluokkalyhenteet
- päättely sijainnin perusteella; esim. hakusanaa seuraa aina sanaluokka ja esimerkin jälkeen tulee (lähes) aina pitäjälyhenne

Nämä taktiikat toimivat suuressa osasta tapauksista, mutta poikkeuksia ja kirjoitusvirheitä löytyy aina. Toimitusohjeista huolimatta painetuissa sanakirjoissa useasti myös muutetaan "rakenneosien" järjestystä tai muotoa, jotta lopputuloksena syntyvä artikkeli olisi luettavampi. Kaikkea ei siis voi koodata automaattisesti, ja täydelliseen lopputulokseen pääsemiseksi aineisto pitäisi käydä läpi käsin.

Mitä prosessista sitten opittiin? Mitä enemmän on toimittajia, sitä enemmän on vaihtelua artikkelien kirjoitustavoissa. Toimittajat uskovat usein myös liikaa siihen, että rakenne jotenkin ohjaisi kirjoittamaan sisällöltään oikeellisia artikkeleita. Elementtejä on aina mahdollista väärinkäyttää, ja toimitusohjeille on edelleen tarvetta. Attribuutit osoittautuivat hankaliksi käyttää, ja oikean arvon asettaminen saattaa jäädä tekemättä vahingossa. Joissakin kohtaa jouduttiin tekemään kompromisseja ja tinkimään koodauksen tarkkuudesta, jotta toimitustyötä ei turhaan hankaloitettaisi. Hakuliittymää tehdessä selvisi, että on ongelmallista, jos samaa asiaa kuvaavat rakenteet ovat eri nimisiä. SMS:ssä on esimerkiksi kolme erinimistä artikkelityyppiä: VakioArt, EtunArt ja PoikkArt. Hakujen kannalta olisi helpompaa, jos olisi vain yksi artikkelirakenne ja luokittelu olisi tehty attribuutilla. Se on kuitenkin mahdotonta, koska kullakin näistä artikkeleista on erilainen rakenne.

Näiden vaiheiden jälkeen sanakirja on periaatteessa rakenteistettu. Siitä ei kuitenkaan saada kaikkea hyötyä irti, ennen kuin aineisto on myös tietokantaistettu eli lyhennysmerkinnät on purettu ja täydennetty ja kaikki informaatio on koneellisesti käytettävissä.

4.4 Jatkokäsittely sanakirjatietokannaksi

Viimeisessä vaiheessa, joka on SMS:n osalta vielä kesken, aineistoa muokataan tietokantamaisempaan muotoon. Painetun sanakirjan osalta tämä helpottaa toimitustyön laadun varmistamista, ja sähköisen version suhteen on jopa välttämätöntä, jotta haut toimisivat järkevästi. Tästä on esimerkkinä mm. yhdyssanahakusanojen lyhentäminen peräkkäisissä artikkeleissa:

> **eine|kahvi** s. [...] **-karja** s. [...] **-kuppi** s. [...]

Sähköisestä sanakirjasta haettaessa artikkelien peräkkäisyys ei säily ja määriteosan puuttuminen hakusanasta todennäköisesti herättää hakijassa hämmennystä. Ongelma ratkaistiin yhdyssanojen kohdalla poimimalla perus- ja määriteosat omiin attribuutteihinsa ja kokoamalla Norm-attribuuttiin kokonaiset normaalistetut yhdyssanat. Edellisen esimerkin *-karja*-hakusana koodattaisiin siis muotoon

> <Hakus maarite=”eine” perusosa=”karja” Norm=”einekarja”>-karja</Hakus>

Samantyyppisiä lyhennysmerkintöjä on saatettu käyttää, kun on yhdistetty kaksi tai useampia hakusanoja samaan artikkeliin, esimerkiksi *aall|ova, -uva* tai *kutei(n)tua*. Ensimmäisessä tapauksessa on kaksi erillistä hakusanaa, jotka kummatkin normalisoidaan ja jälkimmäinen myös täydennetään Norm-attribuuttiin. Toisessa tapauksessa tekstuaalisen näkymän säilyttämiseksi käytetään vain yhtä <Hakus>-elementtiä, mutta Norm-attribuuttiin lavennetaan molemmat hakusanan muodot:

> <Hakus Norm=”kuteitua kuteintua”>kutei(n)tua</Hakus>

Pitkissä viittausluetteloissa viitehakusanat on usein lyhennetty. Ne täytyy saada täydennettyä kokonaisiksi, jotta muunnetun aineiston viittauslinkitys saadaan kohdalleen. Erityisesti yhdyssanaviittauksissa on tapana, että vain listan viimeinen viitehakusana on kokonainen ja muissa on mainittu vain yhdyssanan alkuosa. Tällaisissa tapauksissa viitehakusanojen täydennys onnistuu helposti lisäämällä loppuun hakusana.

> **kuukkari** s. [...] Yhd. jalka-, kahvi-, keskikuukkari.

Automaattisesti hankalia täydentää ovat viittaukset, joissa täydennysosa vaihtelee ja on kyllä mainittu osaluettelon alussa mutta ilman yhdyssanarajan ilmaisevaa pystyä. Näissä tapauksissa joudutaan tekemään täydennykset käsin.

> **kuski|pukki** [...] Rinn. kuskan-, kuskapalli, kuskatuoli, kuski, kuskijakku, -lauta, kuskinistuin, -lauta, -palli, -pukki, kuskipalli, -penkki, -sääti, -tuoli.

Levikkitiedot on SMS:ssä tiivistetty niin pitkälle, että alkuperäistä täsmällistä tietoa ei voida enää niistä päätellä. Tiedot ovat kuitenkin tallessa sähköisinä levikkikarttoina, jotka voidaan myöhemmin linkittää sanakirja-aineistoon.

5 Rakennekuvausten vertailu ja muuntaminen

Tässä luvussa verrataan SMS:n elementtejä TEI:n ja XmLexin elementteihin ja yritetään kartoittaa niiden väliset vastaavuudet. Tarkoituksena on selvittää, miten hyvin standardimuodot soveltuvat SMS:n koodaamiseen ja millaisia eroja koodaustavoissa on.

Toisena tavoitteena on testata SMS:n muuntamista XmLex-muotoon. Elementtien vastaavuusselvityksen perusteella kirjoitetaan XSL-muunnosskripti ja tutkitaan, miten muuntaminen onnistuu käytännössä. Muunnoksessa pyritään säilyttämään kaikki koodauksen sisältämä informaatio. Mahdollisuuksien mukaan pyritään myös säilyttämään sellaista ulkoasuun liittyvää tietoa, jota muilla tavoin on mahdotonta päätellä koodauksesta. Tietyt ryhmittelevät rakenteet voidaan jättää muuntamatta, jos niiden pääasiallinen tarkoitus on kiinnittää joidenkin toisten rakenteiden järjestys toimitusohjeiden mukaiseksi eivätkä ne sisällä uutta tietoa. Tarkoitus ei ole siirtyä käyttämään standardin mukaista rakennekuvausta varsinaisessa toimitustyössä vaan tutkia yleistä muunnettavuutta mm. sanakirjojen yhdistämistä ja yhteiskäyttöä silmällä pitäen.

Oletettavaa on myös, että muunnosvaiheessa SMS:n rakenteesta löytyy ongelmallisia kohtia, joita voidaan jatkossa kehittää erilaisiin standardimuotoihin perehtymisestä saatujen kokemusten perusteella.

5.1 Rakenneosien vastaavuudet

Elementtien vastaavuudet kuvataan taulukkoina, joiden sarakkeina ovat SMS-elementti, sen lyhyt kuvaus, TEI-elementti ja XmLex-elementti. Tekstissä on pyritty selittämään yksityiskohdat ja ongelmat, joita taulukossa ei voi esittää. Attribuutit on mainittu kunkin elementin jälkeen suluissa käyttäen merkintätapaa *@nimi=arvo*. Elementtien sisäkkäisyyttä on kuvattu kauttaviivalla, esim. artikkeli/hakusana. Tässä vaihessa tavoitteena oli löytää kutakin SMS:n elementtiä parhaiten vastaava TEI- ja XmLex-elementti. Element-

tien hierarkian ja järjestyksen muuntamista ja sen ongelmia on kuvattu tarkemmin seuraavassa luvussa.

Taulukossa 1 on kuvattu eri artikkelielementtien vastaavuudet. SMS:ssä on kolme eri artikkelityyppiä ohjaamassa kirjoittajaa: vakioartikkeli on tavallisin, etunuoliartikkeli on lyhyt ja sisältää lähinnä vain viittauksen ja poikkeusartikkelilla voidaan koodata ne harvinaiset tapaukset, joissa artikkelin sisäinen rakenne on jotenkin poikkeava. Koska etunuoliartikkelia käytetään lähinnä toimitusteknisistä syistä, se voidaan koodata TEI:ssä ja XmLexissä tavallisena artikkelina. Poikkeavien artikkelien koodaamiseen TEI:ssä voidaan käyttää elementtiä <entryFree>, mutta XmLexissä niiden koodaaminen saattaa tuottaa hankaluuksia. Huomattavaa on että tieto homonymiasta täytyy poimia SMS:n hakusanan yhteydestä, koska muissa kohdemalleissa se sijoitetaan artikkelin attribuutiksi. Yhdyssanajonojen ryhmittelyyn on tässä ehdotettu varsinaisesti homonyymiartikkelien ryhmittelyyn tarkoitettuja elementtejä (superEntry ja NestEntry).

SMS-elementti	**Kuvaus**	**TEI-elementti**	**XmLex-elementti**
VakioArt	artikkeli	entry(@type=hom)	DictionaryEntry (@homographNumber)
EtunArt	artikkeli	entry(@type=hom)	DictionaryEntry (@homographNumber)
PoikkArt	vapaa artikkeli	entryFree(@type=hom)	DictionaryEntry (@homographNumber)
YhdysJono	ryhmittelee yhteen yhdyssana-artikkelit	superEntry	NestEntry

Taulukko 1: Sana-artikkelit

Hakusana ja siihen liittyvät kieliopilliset tiedot näkyvät taulukossa 2. TEI-koodauksessa muoto (form) koostuu kirjoitusasusta (orth) tai yhdyssanojen kohdalla yhdyssanarajat sisältävästä elementistä (hyph). SMS:n yhdyssanahakusanoissa raja on merkitty pystyviivalla (esim. aamu|kahvi) ja attribuutissa Norm on pystyviivaton normalisoitu muoto, joka voidaan siirtää toiseen orth-elementtiin. XmLexissä yhdyssanahakusana voidaan erottaa tavallisista hakusanoista elementin <Headword> attribuutilla freeType ja normalisoitu muoto voidaan sijoittaa elementtiin <SearchForm>. Kaikki artikkelin päätasolla oleva hakusanaan liittyvä tieto, myös tässä mainitut kieliopilliset tiedot, täytyy koota yhteen elementillä <HeadwordCtn>.

Tarkenne on ongelmallinen, koska sitä ei ole SMS:ssä luokiteltu niin tarkasti kuin muissa rakennekuvauksista. Käytettävä elementti pitää valita rakenteen sisällön perusteella.

SMS-elementti	Kuvaus	TEI-elementti	XmLex-elementti
Hakus(@Yhd=ei)	hakusana	form/orth	HeadwordCtn/Headword
Hakus(@Yhd=on, @Norm)	yhdyssanahakusana	form/hyph(@type=compound) form/orth(@type=lemma)	HeadwordCtn/Headword(@freeType=compound) HeadwordCtn/SearchForm
Sanal	sanaluokka	gramGrp/pos	HeadwordCtn/PartOfSpeech(@freeType)
Tark	tarkenne	sisällöstä riippuen: gramGrp/number tai gramGrp/subc	sisällöstä riippuen: HeadwordCtn/GrammaticalNumber tai HeadwordCtn/Subcategorisation

Taulukko 2: Hakusanan tiedot

SMS:n hakusanan selitteelle löytyy suora vastine kummastakin muusta rakenteesta, kuten taulukosta 3 näkyy. Ongelmia muunnoksessa tulee aiheuttamaan se, että SMS:n SelJakso voi sisältää elementtejä (mm. viittauksia, käyttöalamerkintöjä, tarkenteita), joita ei TEI:ssä sallita selitteen sisään. Yhtenä ratkaisuna voisi olla jättää ne selitetekstiin ilman koodausta. Selitteessä esiintyvällä alihakusanalla (AliHakus) tarkoitetaan sitaattimuotoista kiteymää tai sanontaa sisältävää ”hakusanaa”, joka valaisee varsinaisen hakusanan jotain käyttöyhteyttä ja selitetään kokonaisuutena [Suo88]. Sitä vastaava koodaus TEI:ssä voisi olla <re> (related entry), joka tarkoittaa varsinaisen artikkelin sisällä olevaa miniartikkelia (lähinnä selite) sellaisesta hakusanasta, josta ei ole omaa artikkelia. XmLexissä vastaava elementti on <HiddenEntry>.

SMS-elementti	Kuvaus	TEI-elementti	XmLex-elementti
SelJakso	Selite	def	Definition
AliHakus	sitaattimuotoinen ilmaus	re	HiddenEntry

Taulukko 3: Merkityksenselite

Varsinaisen ongelmapesäkkeen muodostavat muotojakson sisältämät luettelot hakusanan varianteista, taivutetuista muodoista ja erilaisista johdoksista. Ylin elementti MuotoJakso on tarpeeton, koska se vain kiinnittää kirjoitustilanteessa järjestyksen niin, että MuotoHuom on aina muotoluetteloista viimeisenä.

Rakenteet Asu, AsuTaiv, Komp, Taiv, Tyvi ja Vart sisältävät sanamuotoluetteloita, joissa muoto ja sen levikki sidotaan yhteen rakenteella MuotoLev. TEI:ssä kaikenlaisten hakusanan muotojen koodaamiseen käytetään elementtien <form> ja <orth> yhdistelmää. Näiden ryhmittelyynkin on tässä käytetty sisäkkäisiä form-elementtejä, vaikka se ei eh-

kä ole aivan suosituksen hengen mukainen käyttötapa. XmLexissä on omat elementtinsä vaihtoehtoisille kirjoitusasuille (Variant), taivutusmuodoille (Inflection) ja johdoksille (Derivation). Lisäksi näitä voidaan ryhmitellä käyttämällä vastaavia Ctn- ja Block-loppuisia elementtejä. Elementin <AsuTaiv> luokittelu tällä tavoin on ongelmallista, koska sen sisällä voi olla sekä taivutettuja muotoja että perusmuotoja. <MuotoHuom>, joka sisältää vapaamuotoisia huomioita, on luonteva koodata kommenttielementtinä (LinguisticNote).

SMS-elementti	**Kuvaus**	**TEI-elementti**	**XmLex-elementti**
MuotoJakso	ryhmittelevä elementti	ei tarvita	ei tarvita
Asu	ryhmittelee yhteen eri asut ja niiden levikkitiedot	form(@type=variant)	VariantBlock
AsuTaiv	ryhmittelee yhteen sekalaiset asut ja niiden taivutukset sekä levikkitiedot	form	VariantBlock(voi sisältää sekä Variant- että Inflection-elementtejä)
Komp	ryhmittelee yhteen eri komparaatiomuodot	form	InflectionBlock
Taiv	ryhmittelee yhteen eri taivutusmuodot ja niiden levikkitiedot	form	InflectionBlock
Tyvi	ryhmittelee yhteen eri tyvimuodot	form	DerivationBlock
Vart	ryhmittelee yhteen eri vartalomuodot	form	DerivationBlock
MuotoLev	ryhmittelee yhteen muodon ja sen levikin	ylemmästä elementistä riippuen: form(@type=inflected), form(@type=variant) tai form(@type=part)	ylemmästä elementistä riippuen: InflectionCtn, DerivationCtn tai VariantCtn
MuotoHuom	epäformaalissa muodossa olevaa tietoa muodoista	note	LinguisticNote
Muoto	hakusanan muoto, variantti	orth tai jos tyvi tai vartalo niin: orth(@extent=part)	ylemmästä elementistä riippuen: Inflection, Inflection(@type=comparative), Derivation, Variant

Taulukko 4: Muodot

Levikkitietojen koodaamista on käsitelty taulukossa 5. SMS:ssä levikkirakenne voi koostua vapaasta tekstistä, alueista, pitäjistä ja käyttöaloista. Sekä TEI:stä että XmLexistä on vaikea löytää vastaavaa elementtiä. Sopivimpia olisivat TEI:n <usg> ja

XmLexin <GeographicalUsage>, mutta ne eivät voi esiintyä sisäkkäisinä. XmLexin elementeistä lähes ainoaksi vaihtoehdoksi jää <FreeTopic>, jolla voidaan koodata mikä tahansa standardista (tällä hetkellä) puuttuva leksikaalinen elementti. TEI:ssä vaihtoehto on vielä huonompi, <dictScrap>, joka ei todellakaan kuvaa murresanakirjoille olennaista levikkitietoa.

Käyttöalojen luokittelu on erityisesti XmLexissä tarkempi, joten oikea elementti jouduttaisiin valitsemaan rakenteen sisällön perusteella. Elementti <RangeOfApplication> sopii useimpiin tilanteisiin. TEI:ssä kelpaa usg-elementti, mutta sitäkin on ehkä aiheellista tarkentaa sopivalla type-attribuutin arvolla. Alueet ja pitäjät täytynee pitää erillään, joten niissäkin on käytettävä tarkentavaa attribuuttia.

SMS-elementti	**Kuvaus**	**TEI-elementti**	**XmLex-elementti**
Lev	ryhmittelee yhteet levikkiä kuvailevat tiedot	dictScrap	FreeTopic(@type=levikki)
Alue	maantieteellinen käyttöalue, murrealue	usg(@type=alue)	GeographicalUsage (@freeType=alue)
Pit	maantieteellinen käyttöalue, keruupitäjä	usg(@type=pitäjä)	GeographicalUsage (@freeType=pitäjä)
KayttoAla	käyttöala	sisällöstä riippuen: usg(@type=dom|reg|style)	sisällöstä riippuen: Register, SubjectField tai RangeOfApplication

Taulukko 5: Levikkitiedot

Viittaukset olivat SMS:n elementeistä yksinkertaisinta muunnettavaa. Ryhmittelevä ViitJakso voidaan jättää pois, koska se vaikuttaa vain kirjoitusvaiheessa viittausten järjestykseen. XmLexissä erotellaan synonyymiset (See) ja muut viittaukset (SeeAlso). Rinnakkais- ja vertaa-viittaukset vaativat ryhmittelevän elementin käyttöä, koska ne voivat sisältää viitehakusanan lisäksi myös pitäjätiedon. Varsinaisessa viittauksessa XmLexissä käytetään hyväksi XLink-suosituksen linkityskäytäntöjä.

SMS-elementti	Kuvaus	TEI-elementti	XmLex-elementti
ViitJakso	ryhmittelee yhteen viittaukset	(ei tarvita)	(ei tarvita)
ViiteHakus	vars. viitehakusana	ref(@target)	Ptr(@xlink:href)
Rinn	rinnakkaisviittaukset	xr(@type=rinnakkaismuoto)	SeeCtn
Vrt	vertaa-viittaukset	xr(@type=vertaa)	SeeAlsoCtn
Ks	katso-viittaukset	xr(@type=katso)	SeeAlso
Yhd	yhdyssanaviittaukset	xr(@type=yhd)	SeeAlso
EtunViite	viittaus eteenpäin, hakusanan tiedot viitatussa artikkelissa	xr(@type=siirto)	SeeAlso
Osia	viittauksia hakusanan (yl. laite tai esine) osiin	xr(@type=osia)	SeeAlso
TakanViite	viittaus takaisin hakusanaan, josta rinnakkaisviittaus	xr(@type=takanuoli)	See

Taulukko 6: Viittaukset

Erilaisten huomautusten koodaustavaksi valittiin toistaiseksi note-elementti. Sekalaisista huomautuksista voisi ehkä saada enemmän tietoa tutkimalla tarkemmin niiden sisältöä.

SMS-elementti	Kuvaus	TEI-elementti	XmLex-elementti
Huom	huomautuksia	note	Note
TyyppiHuom	huomautuksia hakusanan (yl. laite) tyypeistä	note	Note

Taulukko 7: Huomautukset

Esimerkistön koodauksessa törmätään taas ongelmiin. Kummastakaan kohdemallista ei löydy esimerkkilohkolle (EL) sopivaa vastinetta, ehkä siksi, että merkintätapa on jo alunperin hiukan kyseenalainen. Esimerkkilohkojen käytöllä pyritään erottelemaan samassa merkitysryhmässä sijaitsevat, jollain tavalla (merkitys, käyttö, tms.) ryhmän muista esimerkeistä poikkeavat esimerkit. Loogista olisi, jos eri merkityksiin liittyvät esimerkit olisi sijoitettu eri merkitysryhmiin, mutta tämä on mahdotonta, jos toimittaja on päättänyt poikkeaman olevan niin pieni, ettei erillisiä ryhmiä kannata luoda.

Esimerkkien sisällä on myös runsaasti selittävää tekstiä, jota ei ole koodattu mitenkään. Tämä johtuu esimerkkien sirpaleisesta rakenteesta ja toimittajan työn helpottamisesta. Tosin ainoalle esimerkin keskellä koodatulle selitteelle (SuljeSel, sulkeissa oleva synonyyminen selite) oli vaikea löytää vastaavuutta. TEI:ssä esimerkkien koodaamiseen

käytettyjen quote- ja q-elementtien sisällä ei voi olla selitettä (def). XmLexissä sen sijaan voidaan käyttää vähemmän formaalia seliterakennetta <Gloss>.

SMS-elementti	Kuvaus	TEI-elementti	XmLex-elementti
MerkRyhma	ryhmittelee yhteen samaan merkitykseen kuuluvat tiedot	sense	SenseGroup
EL (@type=normaali|poikkeava)	ryhmittelee yhteen eri merkityksiin tai käyttötapoihin liittyvät esimerkit (silloin kun merkityksr. ei ole käytetty)	ei löydy vastaavaa	ei löydy vastaavaa
EsimJakso	ryhmittelee yhteen saman pitäjän esimerkit	cit(@type=example)	ExampleCtn
Esim	esimerkki	quote	Example
Mur	murretekstiä	quote(@type=dialect)	Example(@style=dialect)
Sit	sitaatti	q	Citation
SuljeSel	esimerkinselite, selittää sanan tai esimerkin osan	cit/def quote/? q/?	Gloss

Taulukko 8: Merkitysryhmä ja esimerkit

Murrepiirteiden merkitsemiseen käytetyt ala- ja yläindeksi voidaan TEI:ssä koodata käyttäen geneeristä korostuselementtiä <hi>. Elementillä <emph> voidaan merkata kursivointi ja harvennus, joita käytetään kielenaineksen korostamiseen. XmLexissä näiden merkkaukseen voidaan käyttää rakennekuvauksessa olevia HTML-lisäyksiä sup, sub, i ja span.

SMS-elementti	Kuvaus	TEI-elementti	XmLex-elementti
Yi	yläindeksi	hi(@rend=sup)	sup
Ai	alaindeksi	hi(@rend=sub)	sub
Korost	tekstin kursivointi	emph(@rend=italic)	i
Harvennus	tekstin harvennus	emph(@rend=sparse)	span(@style=sparse)

Taulukko 9: Korostukset

TEI-mallissa koodaamiseen riittää yleensä pieni määrä elementtejä, joita sitten tarkennetaan type-attribuuteilla. Attribuuttien arvoille annetaan kuitenkin vain suosituksia, tai sitten ne ovat täysin vapaasti valittavissa.

XmLexissä eri nimisiä elementtejä on enemmän ja attribuutteja käytetään luokitteluun tai arvon koodaamiseen. Attribuuttien arvot voidaan sekä antaa standardin määrittele-

mien arvojen mukaisena (type, value) että säilyttää sanakirjalle tyypilliset itse määritellyt arvot (freeType, freeValue). Attribuutissa style voidaan näppärästi säilyttää mm. ulkoasuun liittyvää tietoa alkuperäisestä elementistä.

5.2 SMS:n muuntaminen XmLex-muotoon

Tässä on tarkoitus selostaa XSL-muunnoskriptin tekemisessä vastaan tulleita ongelmia ja niiden ratkaisuja. Varsinainen XSL-skripti on liitteessä 2:. Muunnoksen tuloksen validointiin käytettiin standardin liitteessä C viitattua esimerkki-DTD:tä [XmL07]. Ikävä kyllä standarditeksti ja DTD eivät olleet kaikilta osin täysin yhtäpitäviä. Selvä systemaattinen ero oli Group-loppuisten elementtien nimissä. Ne oli DTD:ssä lyhennetty muotoon Grp. Standardissa mainittu elementti <HiddenEntry> puuttui kokonaan ja sen sijaan tarjolla oli vain <HiddenEntryGrp>, jonka oli pakko sisältää sekä hakusana (Headword) että sen käännös (Translation). Tämän muotoinen rakenne ei enää soveltunut alihakusanojen koodaamiseen, joten tässä kohtaa päätettiin luottaa ensisijaisesti standardiin ja jättää validoinnin virheet näiltä osin huomiotta. Esimerkki-DTD oli yhdessä suhteessa myös hieman sallivampi kuin itse standardi. Artikkelin päätasolla olevia hakusanaan liittyviä tietoja ei tarvinnut välttämättä koota elementtiin <HeadwordCtn>. Kokoamisen toteuttaminen olisikin monimutkaistanut XSL-muunnosskriptiä, joten se jätettiin tällä kertaa tekemättä. Samalla voitiin jättää tekemättä päätös siitä, miten elementit ryhmitellään silloin, kun artikkelissa ei ole merkitysryhmiä. Tämä päätason elementtien ryhmittely voidaan jälkeenpäin toteuttaa helposti toisella XSL-muunnoksella.

Artikkelielementtien yhteyteen poimittiin hakusanasta id-tunniste sekä mahdollinen homonyymin numero. Yhdyssana-artikkelit kokoavalle elementille <NestEntry> tarvittiin id-tunniste, joten sellainen generoitiin ryhmän ensimmäisen hakusanan id:n perusteella. Merkitysryhmänumeroinnin ulkoasun määrittelevän mr-numerointi-attribuutin arvo säilytettiin artikkelin style-attribuutissa.

Sanaluokkalyhenne kopioitiin sellaisenaan freeType-attribuuttiin, mutta myöhemmin muunnokseen voisi lisätä ominaisuuden, joka muuntaisi SMS:n sanaluokkamerkinnät standardinmukaisiksi type-attribuuttia varten. Tarkenne muunnettiin elementiksi <GrammaticalNumber>, mikäli se ilmaisi monikollisuutta sisältämällä merkkijonon ”mon.”. Muissa tapauksessa käytettiin sanaluokkaa tarkentavaa elementtiä <Subcatego-

risation>. Molemmat edellä mainitut ryhmiteltiin artikkelin alussa sanaluokan yhteyteen (PartOfSpeechCtn). Ongelmia tässä kohtaa aiheuttaa se, että tarkenne voi esiintyä joskus myös yksinään ilman sanaluokkaa mm. viittausartikkelin alussa. Tällaisten tapausten koodaaminen on XmLexin mukaisesti mahdotonta ja antaa aihetta jatkotutkimuksiin ja SMS:n rakenteen kehittämiseen.

Eri muototyypit saatiin koodattua tutkimalla niiden sijaintia hierarkiassa. Esimerkiksi elementin <Taiv> alla esiintyvät muodot koodattiin elementillä <Inflection>. Poikkeuksena tästä on rakenne <AsuTaiv>, joka voi sisältää sekä taivutettuja että taivuttamattomia muotoja. Näiden koodaus voisi onnistua tekstivihjeiden avulla, koska yleensä perusmuoto on erotettu taivutuksesta kaksoispisteellä. Tähän ei kuitenkaan ryhdytty, koska sen toteuttaminen jollain muulla skriptikielellä kuin XSL:llä olisi huomattavasti helpompaa. Muotohuomautuksen sisällä esiintyvät muodot koodattiin geneerisellä elementillä <Fragment>, koska DTD:n mukaan <LinguisticNote> ei voi sisältää elementtiä <Variant>.

Muodon ja levikin yhdistävät <MuotoLev>-elementit koodattiin muotoja vastaavilla Ctn-loppuisilla elementeillä ja kokoavat luettelot (Asu, Taiv, jne.) vastaavilla Block-loppuisilla elementeillä. Jokaisen muotoluettelon alussa oli sitä kuvaava otsikko, esim. ”Asu:”, ”Asu myös:”, ”Taivutus (yks. 3. pers.):”, ”Komparatiivi:”. Koska Block-loppuisten elementtien sisällä ei saanut olla irrallista tekstiä, säilytettiin otsikot koodaamalla ne elementillä <FreeComment>. Muotojen väliset pilkut poistettiin, koska ne voitaisiin tuottaa ulkoasumäärityksellä.

Taivutusmuotoluetteloissa esiintyi jonkin verran hierarkiaa, jota ei ollut aina eksplisiittisesti koodattu. Erityyppiset taivutukset pitäisi kukin koodata omiin <Taiv>-elementteihinsä, mutta joskus toimittaja on kirjoittanut ne painetun tyylisesti samaan ja erottanut toisistaan puolipisteellä (prees. *käyy*, *kävyy*; imperf. *käi*, *käyi*). Joskus kuviota on vielä sotkettu sulutuksella ja vaihtoehtojen tiiviillä esittämisellä. Tämä epäloogisuus aiheuttaa niin paljon ongelmia muunnoksessa, että tapaukset olisi syytä käydä läpi ja korjata käsin. Toimitusohjeistuksessa on siis näiltä osin vielä parantamisen varaa.

Viittausten muuntaminen oli muuten suoraviivaista, mutta jostain syystä standardin mukaan synonyymisiä viittauksia (See) olisi saanut esiintyä vain merkitysryhmien sisällä. Tämän takia kaikki viittaukset jouduttiin koodaamaan elementillä <SeeAlso>. Viittaus-

ten alkuperäinen nimi, jonka avulla generoidaan viittauksille otsikot (vrt., ks., jne.), säilytettiin attribuutissa style.

Esimerkkilohkolle (EL) ei löytynyt vastaavaa elementtiä, koska standardi ei määrittele rakennetta <ExampleBlock>. Ikävä kyllä lohkojen koodaus on tärkeää, koska poikkeava esimerkkilohko pitää osata painetussa sanakirjassa erottaa tavallisesta lohkosta kaksoispystyllä (||). Yhtenä ratkaisuna voisi olla XmLexin DTD:n muokkaaminen, niin että voitaisiin käyttää <ExampleBlock>-elementtiä. Esimerkkijakso, joka yhdistää esimerkin ja lähdepitäjän, voi koostua joissain tapauksissa myös pelkästä pitäjälyhenteestä. Elementin <ExampleCtn> pitää kuitenkin aina sisältää vähintään yksi <Example>, joten näissä tapauksissa on hiukan huijattu ja generoitu pitäjän seuraksi tyhjä esimerkki.

5.3 Johtopäätökset muunnoksen onnistumisesta

SMS:n aineiston muuntaminen XmLex-muotoon onnistui melko hyvin. Vastaan tuli yllättävän vähän ongelmia. Seuraavassa on esitetty luvun 4.2 esimerkin 10 *elehursti*-artikkeli muunnettuna XmLex-muotoon. Artikkelin koodaus noudattaa DTD:tä mutta ei ole täysin standardin mukainen, koska siinä ei ole koottu merkitysryhmiä edeltäviä päätason elementtejä <HeadwordCtn>-elementin sisään.

```
<DictionaryEntry identifier="sms1_elehursti" style="normaali">
 <HeadwordCtn>
  <Headword freeType="compound">ele|hursti</Headword>
  <SearchForm>elehursti</SearchForm>
 </HeadwordCtn>
 <PartOfSpeechCtn><PartOfSpeech freeValue="s."/></PartOfSpeechCtn>
 <VariantBlock>
  <FreeComment type="heading">Asu: </FreeComment>
  <VariantCtn>
   <Variant>ele(h)hursti</Variant><FreeTopic type="levikki">yleensä</FreeTopic>
  </VariantCtn>
  <VariantCtn>
   <Variant>ele(h)vursti</Variant> <FreeTopic type="levikki">
    <GeographicalUsage type="used" freeType="pitäjä">Lep</GeographicalUsage>
    <GeographicalUsage type="used" freeType="pitäjä">VehS</GeographicalUsage>,
    </FreeTopic>
  </VariantCtn>
  <VariantCtn>
   <Variant>eleshursti</Variant> <FreeTopic type="levikki">
    <GeographicalUsage type="used" freeType="pitäjä">Lep</GeographicalUsage>
    </FreeTopic>
  </VariantCtn>
 </VariantBlock>
 <SeeAlso style="Rinn">
  <Ptr xlink:href="sms1_eleloimi">eleloimi</Ptr>
  <Ptr xlink:href="sms1_elevaate">-vaate</Ptr>
```

```
   <Ptr xlink:href="sms1_elohursti">elohursti</Ptr>
   <Ptr xlink:href="sms1_eloloimi">-loimi</Ptr>
   <Ptr xlink:href="sms1_eloraanu">-raanu</Ptr>
   <Ptr xlink:href="sms1_elovaate">-vaate</Ptr>
  </SeeAlso>
  <SeeAlso style="Vrt">
   <Ptr xlink:href="sms3_helehursti">helehursti</Ptr>
  </SeeAlso>
  <SenseGrp>
   <Definition>iso vaate, jota käytetään jyvien talteenottovälineenä viljaa tuultaen
puhdistettaessa sekä reen pohjalle, riihen oven eteen, auman viereen jne. levitettynä
puimatonta viljaa siirrettäessä.</Definition>
   <FreeTopic type="levikki">
    <GeographicalUsage type="used" freeType="alue">KarP</GeographicalUsage>
    etelä- ja keskiosa,
    <GeographicalUsage type="used" freeType="alue">SavP</GeographicalUsage>
    (ei pohjoislaita eikä lounaiskulma),
    <GeographicalUsage type="used" freeType="pitäjä">Pari</GeographicalUsage>
    <GeographicalUsage type="used" freeType="pitäjä">Uuk</GeographicalUsage>
    <GeographicalUsage type="used" freeType="pitäjä">Harlu</GeographicalUsage>
    <GeographicalUsage type="used" freeType="pitäjä">SavR</GeographicalUsage>
    <GeographicalUsage type="used" freeType="pitäjä">Ker</GeographicalUsage>
    <GeographicalUsage type="used" freeType="pitäjä">Konn</GeographicalUsage>.
   </FreeTopic>
   <ExampleCtn>
    <Example><Example
style="dialect">elehhurstilòita tehŏttiin siitä ruohtimesta ja
säkkijä.</Example></Example>
    <GeographicalUsage type="used" freeType="pitäjä">Kite</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example><Example
style="dialect">Kuv vieŏttiin ahosta riiĥheen nir riihen
oven ulukopuolellel làitettiin elehhursti siihen
karis jyv̆vii.</Example></Example>
    <GeographicalUsage type="used" freeType="pitäjä">PyhS</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example><Example
style="dialect">neľpieťimiset on ne elehhurstit ja loajat
ja piikoittain kuvoŏttaan.</Example></Example>
    <GeographicalUsage type="used" freeType="pitäjä">Pol</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example><Example style="dialect">Elehhurstille tippu päetä, lyhteem
päetä.</Example></Example>
    <GeographicalUsage type="used" freeType="pitäjä">Siil</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example><Example style="dialect">Viskoo lyhteet elehurstin
piälle!</Example></Example>
    <GeographicalUsage type="used" freeType="pitäjä">VehS</GeographicalUsage>
   </ExampleCtn>
  </SenseGrp>
  <SenseGrp>
   <Definition>suurikokoinen vaate, jota käytetään viljakuorman, puimakoneen, riihen
kiukaan ym. peitteenä.</Definition>
   <ExampleCtn>
```

```
    <Example/>
    <GeographicalUsage type="used" freeType="pitäjä">Tohm</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example><Example style="dialect">Kuv ves ja jàohot o
hämmennetty<Gloss>viinankeittoa aloiteltaessa</Gloss> pèitetää tiinu
elehhurstilla.</Example></Example>
    <GeographicalUsage type="used" freeType="pitäjä">Kite</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example/>
    <GeographicalUsage type="used" freeType="pitäjä">Kesä</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example/>
    <GeographicalUsage type="used" freeType="pitäjä">Kont</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example/>
    <GeographicalUsage type="used" freeType="pitäjä">Lep</GeographicalUsage>
   </ExampleCtn>
   <ExampleCtn>
    <Example/>
    <GeographicalUsage type="used" freeType="pitäjä">MuuV</GeographicalUsage>
   </ExampleCtn>
  </SenseGrp>
</DictionaryEntry>
```

Esimerkki 11. *Elehursti*-artikkeli XmLexin mukaisesti koodattuna.

Selkeimmän ongelmakohdan muodosti muotoluetteloiden (Asu, Taiv, jne.) muuntaminen, ja siinä havaittiinkin tarvetta sekä rakennekuvauksen että toimittajien ohjeistuksen kehittämiselle. Muotoluetteloiden sisällä oli sallittu vapaan tekstin käyttö, ja se antoi toimittajalle mahdollisuuden sijoittaa tekstiä vääriin paikkoihin. Tällaisissa tapauksissa yleensä levikkiin kuuluvia täsmenteitä oli kirjoitettu levikki-rakenteen ulkopuolelle. Mahdollinen ratkaisu voisi olla rakenteen tiukentaminen niin, että vapaata tekstiä ei sallittaisi missään muotoluettelon sisällä. Tällöin myös luettelon alussa olevaa otsikkoa varten pitäisi luoda uusi rakenneosa.

Toisen ongelman aiheutti se, että rakenne ei antanut mahdollisuutta jakaa muotoluetteloita osiin. Tämän ongelman kirjoittajat olivat kiertäneet käyttämällä sanakirjoissa yleistä tapaa erottaa osat toisistaan puolipisteellä (katso esimerkki 8 luvussa 3.2). Tässäkin ratkaisuna olisi joko lisätä uusi rakenneosa jaottelua varten tai ohjeistaa kirjoittajat käyttämään peräkkäisiä muotoluetteloita. Ensimmäinen vaihtoehto toisi taas yhden tason lisää hierarkiaan eikä siten ole kovin käyttäjäystävällinen. Toinen vaihtoehto on parempi ja vaatii lähinnä vain toimitusohjeiden ja ulkoasukuvauksen muokkaamista.

Tiedon säilymistä muunnoksessa tutkittiin kirjoittamalla XSL-muunnos, joka palauttaa XmLex-muotoisen aineiston takaisin SMS-muotoon. Havaittiin, että muunnoksessa katoaa tieto artikkelien (VakioArt, EtunArt vai PoikkArt) ja huomautusten (Huom vai TyyppiHuom) tyypeistä. Samoin häviävät eri muotojaksojen erot, koska ne on koodattu samalla elementillä. Esimerkiksi sekä <Komp> että <Taiv> on koodattu <InflectionBlock>-elementillä. Nämä kaikki voidaan haluttaessa säilyttää tallentamalla tieto alkuperäisestä elementistä sopivaan attribuuttiin (esim. style). Myöskään pitäjien järjestysnumerot eivät säily muunnoksessa, mutta ne voidaan helposti generoida uudelleen. Näiden lisäksi muunnoksessa häviää välilyöntejä ja pilkkuja, jotka useimmiten voidaan tuottaa jälkikäteen automaattisesti.

6 Standardimuodon hyödyntäminen

Uusien sanakirjojen luominen helpottuu, kun pohjaksi voidaan poimia aineistoa jostain jo valmistuneesta sanakirjasta. Erilaisia sanakirja-aineistoja voidaan käsitellä samoilla ohjelmistoilla ja hakujen tekeminen yksinkertaistuu, kun käsitteet ovat yhteisiä. Periaatteessa tämä kuulostaa hienolta. Käytännössä joudutaan kuitenkin usein tekemään kompromisseja ja vähintäänkin luomaan toimitustyötä varten oma tiukemmin ohjaava rakennekuvauksensa. Tärkeää onkin että omien rakennekuvausten suunnittelussa otetaan huomioon standardeissa määritellyt käytännöt ja varmistetaan, että muunnos standardimuotoon onnistuu tietoa hävittämättä. Standardin käytäntöihin perehtymisellä ja niiden hyväksikäytöllä voitaisiin sanakirjatyössä vahvistaa myös leksikografista yhteistyötä eri sanakirjaprojektien välillä.

6.1 Yhteensopivuus

ISO pyrkii varmistamaan tarjoamiensa standardien yhtenäisyyden ja yhteensopivuuden perustamalla ne alemman tason standardien päälle. Näillä perusstandardeilla määritellään mm. merkistökoodaus (Unicode), kielitunnukset (ISO-639) ja päiväysmuodot (ISO-8601). Tärkeän osan perustasta muodostaa valmistumassa oleva ISO/DIS 12620, joka esittää formaalin tavan määritellä rakennekuvauksissa ja koodauksessa käytettävä sanasto. Standardin mukainen sanasto koostuu *käsitteistä* (data category), jotka voivat olla joko lingvistisiä *ominaisuuksia*, kuten esimerkiksi sanaluokka, tai niiden *arvoja*, kuten verbi. ISO TC37/SC4 on tehnyt standardin pohjalta sanastosta toteutuksen, jota

kutsutaan nimellä *Data Category Registry* (DCR) [IdR04]. DCR sisältää joukon edellä mainitun kaltaisia käsitteitä. Kuhunkin käsitteeseen liittyy nimi, selite, mahdollisia käyttöesimerkkejä, kielikohtaiset tiedot sekä joukko ylläpidon kannalta olennaista tietoa. Käsitteet voivat liittyä toisiinsa kahdella tavalla: käsitteellä voi olla joukko mahdollisia arvokäsitteitä (sanaluokan arvo voi olla verbi, substantiivi jne.) tai käsitteellä voi olla yläkäsite (substantiivit kuuluvat nomineihin). Nämä suhteet muodostavat sanastoon kevyen hierarkian. Kielikohtaisissa tiedoissa voidaan määritellä mm. käsitteen nimi kyseisellä kielellä. DCR:n käsitteet ovat siis periaatteessa nimettävissä eri kielissä eri tavalla. Koska sanaston koostaminen on vielä kesken, kaikki käsitteet on nimetty englanninkielisillä nimillä. Näitä elementtien nimiä on käytetty myös XmLexin rakennekuvauksessa. DCR ei ole valmis, vaan sitä täydennetään tarpeen mukaan standardissa määritellyin menettelytavoin. Sanasto on kuitenkin koko ajan käytettävissä verkossa, ja siihen on tarjolla kaksikin käyttöliittymää. Syntax[8] on ensimmäinen toteutusversio, joka tarjoaa myös REST-pohjaisen liittymän tietokantaan. Uusin liittymä on nimeltään ISOcat[9], joka on käytettävissä myös www-sovelluspalveluna.

6.2 Muunnettavuus

XmLexin ensimmäinen testisovellus käsitteli kaksikielisen sanakirjan kielen kääntämistä niin, että kohdekielestä tulee lähdekieli [LeM06]. Tämä onnistuu, koska standardin mukaisten rakenteiden merkitys riippuu vain niiden asemasta hierarkiassa, ei niiden lineaarisesta järjestyksestä. Tämän periaatteen mukaiset rakenteet ovat koneellisesti käsiteltävissä ja uudelleen järjestettävissä.

XmLex-muotoisen sanakirjan muuntaminen leksikkostandardin LMF määrittelemään muotoon pitäisi olla teknisesti melko suoraviivaista, koska molemmat käyttävät samoja DCR:ssä määriteltyjä käsitteitä. Tämä ei kuitenkaan ole kovin hedelmällistä sinänsä, koska tuloksena olisi monessa tapauksessa tietosisällöltään melko köyhä leksikko. Mielenkiintoinen ajatus sen sijaan olisi ”rikastaa” sanakirjatietokantoja leksikoista saatavilla hyvin strukturoiduilla tiedoilla, esimerkiksi sanojen taivutusparadigmoilla ja syntaktisilla tiedoilla. Sanakirja-artikkelin sisältöön kannattaa tuskin lisätä ylettömästi tietoa, mutta linkittämällä leksikon tiedot artikkeliin niitä voisi käyttää hyväksi vaikka sähköisissä sanakirjoissa lisätietoa antavana osana. Tällöin käyttäjälle voitaisiin tarjota

8 http://syntax.inist.fr
9 http://www.isocat.org/

esimerkiksi hakusanasta mikä tahansa taivutusmuoto, kun nykyisellään mm. Kielitoimiston sanakirjan sähköinen versio tarjoaa vain muutaman esimerkkitaivutuksen.

Sanakirjat kuvaavat hakusanoja ja niiden eri merkityksiä, termistöt taas toisiinsa liittyviä käsitteitä. Saksalais-ranskalaisessa sanakirjatuottajien ja termistösovellusten (kääntäjien apuvälineiden) tarjoajien yhteisessä projektissa testattiin sanakirja-aineiston muunnosta termistömuotoon [LeD06]. Tarkoituksena oli tuottaa automaattisesti erikoisalojen sanakirjoista standardimuotoisia termistöjä kääntämisen apuvälineisiin liitettäväksi. Lähtömuotona oli XmLex ja tulos haluttiin termistöstandardin ISO 16642 mukaisena, jolloin se olisi helposti muunnettavissa ohjelmistotoimittajien käyttämiin binäärimuotoihin. Ensin tunnistettiin ja etsittiin kummankin muodon vastaavat elementit. Osa elementeistä oli samoja, koska kummankin standardin pohjana oli käytetty DCR:n sanastoa, ja lopuillekin tarvittaville elementeille löytyi vastaavuudet. Tämän jälkeen suunniteltiin rakenteiden muuntaminen, joka perustui yksinkertaisuudessaan kahteen vaiheeseen:

1. Ryhmitellään synonyymit päähakusanansa yhteyteen. Synonyymit löytyivät viittausten perusteella, päähakusanaan synonyymiviittauksella viittaavien artikkelien hakusanat ovat päähakusanan synonyymejä.
2. Jaetaan synonyymiryhmät merkitysten perusteella ryhmiin. Apuna ryhmittelyssä käytettiin alkuperäisten artikkelien merkitysryhmäjakoa.

Projektin tuloksena muunnettiin kuusi eri alojen kaksikielistä sanakirjaa sekä neljä englannin-, ranskan-, italian- ja espanjankielistä tekniikan sanakirjaa termistöksi, joka sisältää yli 1,6 miljoonaa käsitettä.

6.3 Yhdistäminen ja yhteiskäyttö

Erilaisten sanakirjojen yhdistäminen yhdeksi sanakirjatietokannaksi vaatii, että kunkin rakenne on määritelty ja kuvattu yhtenäisellä tavalla. Helpointa on käyttää samaa standardia rakennekuvausta. Mikäli voidaan luottaa siihen että koodaaja on tulkinnut standardia oikein, voidaan myös olla varmoja siitä, että samalla tavalla merkityt rakenteet tarkoittavat samaa asiaa.

Artikkelissa [HaI06] esitellään abstrakti malli, jonka avulla voidaan linkittää toisiinsa yksi- ja monikielisiä sanakirjoja sekä termistöjä. Mallissa sanakirjatietokanta kuvataan

solmuina ja niiden välisinä linkkeinä. Solmuja ovat *hakusana*, *merkitys* ja *käsite*. Sanakirjojen kohdalla merkityssolmu sisältää samat tiedot kuin merkitysryhmä, termistöissä merkityssolmu yhdistää hakusanan käsitteeseen. Luonnollisia linkkejä solmujen välillä ovat sanakirjoista saatavat hakusanan ja sen merkitykset yhdistävät linkit. Termistöissä olevilla linkeillä yhdistyvät hakusanat merkityksiin, merkitykset käsitteisiin ja käsitteet toisiin käsitteisiin. Kirjoittajien mukaan tällä tavalla luokitellusta aineistosta voidaan automaattisesti päätellä seuraavat kolme linkkiä:

- Linkit hakusanasta hakusanaan kuvaavat jotain niiden välistä vastaavuutta (synonyymit, johdokset yms.) saman kielen sisällä.
- Linkit merkityksestä merkitykseen kuvaavat merkitysten välistä vastaavuutta tai käännöstä jollain toisella kielellä.
- Linkit merkityksestä tai käsitteestä toiseen käsitteeseen kuvaavat näiden välistä vastaavuutta ja voivat linkittää yhteen merkityksiä ja käsitteitä samasta tai eri kielistä.

Kuhunkin koneellisesti pääteltyyn linkkiin lisätään merkintä sen luotettavuudesta ja siitä, millaisen prosessin tuloksena linkki on syntynyt. Tällä tavoin linkitetystä aineistosta voidaan löytää tietoa, joka ei sinänsä sisälly yhteenkään mukana olevaan sanakirjaan. Yksikielisten sanakirjojen ja termistöjen yhdistäminen liittää hakusanojen tiedot osaksi käsitehierarkiaa. Monikielisten sanakirjojen suhteen käännöslinkitys voi tuottaa japani–englanti- ja englanti–arabia-sanakirjoista mahdollisuuden löytää käännöksiä suoraan japanista arabiaan.

Ei voida olettaa, että yksi rakennestandardi tai -suositus yleistyisi koko maailmassa, eikä toisaalta mikään rakennekuvaus ja joukko kuvauksia ole riittävä kaikille maailman kieliaineistoille [Sim04b]. Aineistojen laajempaa yhteensopivuutta ja yhteiskäyttöä varten tarvitaan semanttinen malli, ontologia, joka mahdollistaa myös erilaisten rakennekuvausten käytön. Ontologia määrittelee formaalisti tietyn aihealueen käsitteet ja niiden väliset suhteet. Erilaiset rakennekuvaukset voidaan yhdistää ontologiaan määrittelemällä elementtien ja ontologian käsitteiden väliset vastaavuudet jollain formaalilla tavalla. Ontologian käsitteitä voidaan siis käyttää kuvaamaan tai tulkitsemaan eri rakenneosien merkityksiä. Tämä mahdollistaa myös ns. semanttiset haut erilaisista ja eri tavoin koodatuista aineistoista. Toisin sanoen haku voidaan kohdistaa käsitteisiin ja nii-

den hierarkiaan eikä pelkästään merkkijonoihin ja rakennekuvauksessa nimettyihin rakenteisiin.

Tekeillä oleva DCR-käsitteistö (Data Category Registry, katso luku 6.1) on tavallaan myös ontologia. Se määrittelee keskeiset kieliaineiston kuvaamisessa tarvittavat käsitteet ja niiden väliset suhteet. ISO TC37 -komitea pyrkii osana *Language Annotation Framework* -hanketta (LAF) tekemään DCR:stä yhteisen käsitteistön kaikenlaisten kielitieteellisten aineistojen kuvaamiseen sekä määrittelemään tavat, joilla sitä voidaan hyödyntää aineistojen käytössä [IdR07]. Ontologiaksi DCR on rakenteeltaan hyvin yksinkertainen. Se sisältää vain suhteet *yläkäsite* (broader concept) ja *arvo* (has value).

Toinen kielitieteelliselle aineistolle tarkoitettu ontologia on EMELD-hankkeessa[10] kehitetty *General Ontology for Linguistic Description* (GOLD) [FaL03]. Se on rakenteeltaan monimutkaisempi ja sisältää runsaan valikoiman käsitteiden välisiä suhteita. Se on yhdistettävissä muihin ontologioihin ja sitä kautta myös muihinkin kuin kielitieteellisiin sovelluksiin. GOLD on saatavissa W3C:n suosittelemassa OWL/RDF-muodossa, uusin versio 0.45 tosin vasta XML-muotoisena[11]. Kullakin käsitteellä on yksikäsitteinen URI, jonka avulla siihen voidaan viitata. Erityisen metaskeeman (SIL Metaschema) avulla voidaan kuvata minkä tahansa rakennekuvauksen elementit ontologian käsitteiksi. Kuvauksen ei tarvitse rajoittua elementteihin, vaan myös niiden sisältö voidaan kuvata ontologian avulla.

Artikkelissa [Sim04a] on kuvattu projekti, jossa useita erilaisia leksikkoja muunnettiin GOLD-ontologian mukaiseen RDF-muotoon. Muunnosta varten jokaiselle leksikolle kirjoitettiin oma metaskeema, joka kuvasi rakennekuvauksen elementtejä vastaavat ontologian käsitteet. Tämän jälkeen aineisto tallennettiin RDF-tietokantaan hakujen testaamista varten. Suoralla haulla (direct search) voitiin hakea paitsi rakenneosien nimitysten perusteella myös rakenteiden sisältöä, jos se oli kuvattu ontologian avulla. Voitiin siis esimerkiksi hakea eri aineistoista verbit, vaikka ne olisi alunperin kirjoitettu toisistaan eroavalla tavalla (esim. verbi, v, V), kunhan ne oli metaskeemassa kuvattu samaksi käsitteeksi. Epäsuoralla haulla (indirect search) saatiin vielä mielenkiintoisempia tuloksia, koska tällöin hakuja voitiin tehdä ontologiassa määritellyn käsitehierarkian avulla, eli laajemmalla käsitteellä haettaessa löydetään myös sen alakäsitteiden ilmenty-

10 http://emeld.org
11 http://www.linguistics-ontology.org/gold.html

mät. Edellä esitetty verbien haku löytäisi tässä tapauksessa myös transitiiviverbit, refleksiiviverbit jne.

Kaikkiin standardeihin ja ontologioihin liittyy kuitenkin samat ongelmat [Sim04b]:

1. Miten saada malli hyväksytyksi riittävän laajaan yleiseen käyttöön? Molemmat yllä esitellyt ontologiat ovat vielä keskeneräisiä, joten käyttäjiä on vähän.
2. Miten standardia tai ontologiaa pidetään yllä? Lisäysten ja päivitysten pitää olla mahdollisia, mutta lisäysprosessin on oltava luotettava, eikä se saa rikkoa aikaisemman version perustalle rakennettuja sovelluksia ja aineistoja.
3. Miten saada kielitieteellisten ohjelmistojen tuottajat käyttämään mallia sovelluksissaan?

ISO:n hyväksymät asiantuntijat voivat ehdottaa lisäyksiä tai muutoksia DCR:ään. Hyväksymismenettely on määritelty ISO-standardissa, mutta on epäselvää, kuinka kauan jonkin uuden käsitteen tai osahierarkian hyväksyminen vaatii. GOLD on toistaiseksi vain tietyn kehittäjäryhmän päivitettävissä. Muutosten ja lisäysten hyväksymisen prosessit ovat vasta kehitteillä.

6.4 Kotuksen sanakirjat

Pitkissä sanakirjaprojekteissa on oma ajan mittaan hioutunut tarkka käsitteistönsä, minkä takia itse toimitustyössä ei ole järkevää siirtyä standardimuotoon ja sen enimmäkseen yleisen käsitteistön käyttämiseen. Luvussa 5 tehdyn SMS:n muuntamisen tulosten perusteella voidaan tosin sanoa, että jo pelkkä muunnosprosessi on monella tapaa hyödyllinen. Rakenteen analysointi ja kuvaaminen standardimuotoon paljastaa oman rakennekuvauksen ongelmakohdat ja epätarkkuudet. Näiden perusteella alkuperäistä rakennekuvausta voidaan kehittää edelleen. Toinen selkeä hyöty on, että muuntamisen yhteydessä voidaan aineistoon lisätä tietoa ja tarpeen mukaan myös tarkentaa koodausta. SMS:n tapauksessa esimerkiksi sanaluokkien yhteyteen voidaan lisätä niiden standardinmukaiset arvot ja epämääräisesti määritelty tarkenne-rakenne voidaan luokitella täsmällisemmin sen arvojen perusteella (monikollisuus ja sanaluokan tarkenteet erikseen). Rakenteen kehittäminen puolestaan parantaa myös sanakirja-aineiston sisällön laatua.

Epäilemättä standardiin muuntamisesta olisi hyötyä muillekin Kotuksen rakenteisille sanakirjoille. Tällä hetkellä Kotuksessa on ainakin osin rakenteistetussa muodossa jo kuusi yksikielistä ja kaksi kaksikielistä sanakirjaa. Aineistoa on niin vanhasta kirjasuomesta, nykysuomesta, suomenruotsista, suomen ja suomenruotsin murteista kuin suomen sukukielistäkin. Yhtenäinen muoto mahdollistaisi aineistojen keskinäisen vertailun ja ehkä osaltaan auttaisi toimitusten välistä leksikografista yhteistyötä. Myös uusien sanakirjatuotteiden (pohjatyön) tekeminen olemassa olevista sanakirjoista koostamalla voisi helpottua, kun rakenteet olisivat yksikäsitteisesti määriteltyjä.

Suunnitelmissa on myös Kotuksen sanakirjojen yhdistäminen yhteiseen sanakirjatietokantaan tai sanakirjaportaaliin, jonka avulla voisi tehdä hakuja ja yhdistää tietoja useasta sanakirjasta. Tässä tapauksessa yhteisen muodon hyödyt ovat ilmeisiä ja standardin käyttö perusteltua. Hakuliittymäkin on helpompi rakentaa, kun hakua ei tarvitse räätälöidä jokaiselle sanakirja-aineistolle erikseen. Yhteishakumahdollisuuden tarjoamista suurelle yleisölle rajoittavat omalta osaltaan kustannussopimukset, mutta vapaasti julkaistavissa olevaa materiaalia on tekeillä erityisesti sukukielten, murteiden ja etymologian osalta.

Myös ontologioiden käytöstä on ollut puhetta, tosin vain alustavasti, koska niiden käytöstä ei juuri ole kokemuksia. Leksikografien ja kielentutkijoiden kannalta kielitieteellisen ontologian käyttö mahdollistaisi rikkaat hakumahdollisuudet erityisesti sanakirjojen kieliopilliseen tietoon. Houkuttelevaa olisi myös tarjota suurelle yleisölle ontologialla terästettyjä sisällöllisiä hakuja. Tällöin esimerkiksi hakusanalla koira voisi saada tulokseksi myös koiraa kuvaavat vanhan kirjasuomen ja eri suomen murteiden muodot. Ontologioiden käyttöönotto vaatii kuitenkin hyvän esiselvityksen saatavista hyödyistä ja koodauksen tarkentamisen vaatimasta työmäärästä. Lisäksi olisi selvitettävä, millaisia työvälineitä ja ohjelmistoja ontologioiden hyödyntämiseen on olemassa ja miten ne voitaisiin yhdistää Kotuksen sanakirjatyöhön.

7 Yhteenveto

Painettujen sanakirjojen digitointi ja muuntaminen sähköiseen muotoon antaa mahdollisuuksia hyödyntää aineistoa uusilla tavoille. Lisävaatimuksia sähköiselle muodolle asettavat sanakirjojen eri käyttäjäryhmät, joilla on omat tarpeensa aineiston käytön suh-

teen. Näiden lisäksi aineiston on sovelluttava myös koneelliseen prosessointiin ja erilaisten kieliteknologisten sovellusten pohjaksi.

Rakenteistus perustuu sana-artikkelin rakenneosien tunnistamiseen, ulkoasun tai tekstisisällön perusteella, ja eksplisiittiseen merkitsemiseen. Rakenteiden merkitsemistavaksi on vakiintunut W3C:n XML-standardi. Tämän lisäksi tarvitaan rakennekuvaus, joka kertoo koodauksessa käytettävien rakenteiden (elementtien) nimet sekä sen, miten ne voivat sijoittua suhteessa toisiinsa (peräkkäisyys, sisäkkäisyys, toisto).

Rakenneosien merkitsemisessä täytyy tehdä valintoja ja kompromisseja sen mukaan, mikä on koodatun aineiston käyttötarkoitus. Arkistoinnin ja säilyttämisen kannalta on tärkeää säilyttää tekstuaalinen näkymä eli tekstisisältö koskemattomana. Koneellisen hyödyntämisen kannalta taas on järkevämpää koodata leksikaalinen näkymä eli koodata varsinainen tietosisältö yhdenmukaisesti alkuperäisestä muodosta välittämättä. Jälkimmäisen näkymän mukainen tietokantaistettu aineisto saadaan aikaan vain purkamalla sanakirjan implisiittiset viittaukset yksikäsitteiseksi koodaukseksi. Sanakirjan alkuperäisestä rakenteesta riippuen tämä voi olla hyvinkin hankalaa. Jos aineisto on koodattu niin, että kunkin rakenteen merkitys on pääteltävissä sen sijainnista hierarkiassa, tietokantaistus on jossain määrin mahdollista tehdä automaattisesti. Tässäkin suhteessa kannattaa tehdä kompromisseja sen mukaan, mikä on haluttu lopputulos. Hakukäyttöä varten riittää usein erilaisten lyhennysmerkintöjen avaaminen ja aineiston täydentäminen muutamalla sopivalla luokittelulla.

Yksi keskeisistä sanakirjarakenteiden kehittäjistä on ollut Text Encoding Initiative -hanke. Sen puitteissa kehitettiin 1990-luvun alussa laajasti levinnyt suositus humanistisen alan tekstiaineistojen koodaamisesta. Suositus sisälsi yhtenä osanaan rakennekuvauksen painettuna julkaistujen sanakirjojen merkkaamiseen. TEI-koodaukselle on ominaista luokittelun vapaus. Monet elementeistä ovat luokiteltavissa type-attribuutilla, jonka arvo on täysin koodaajan valittavissa. Tämä aiheuttaa hyvin todennäköisesti ongelmia aineistojen sisällöllisen yhteensopivuudessa ja mahdollisesti myös siinä tapauksessa, kun koodaajia on useita.

Sanakirjarakenteiden kehittelyä on jatkanut ISO:n komitea TC37, jonka yleisenä tehtävänä on kieliresursseja koskevien standardien kehittäminen. Sen vuonna 2007 julkaisema ISO 1951:2007 -standardi (XmLex) pyrkii selkeään yksiselitteiseen hierarkkiseen rakenteeseen ja silti mahdollistamaan kaikenlaisten sanakirjojen koodaamisen.

Standardia on vaikea arvioida, koska sitä hyödyntäviä aineistoja ei ole vielä ainakaan julkisesti saatavilla. Tärkeimpänä periaatteena on ollut rakenneosien merkityksen sitominen hierarkiaan niiden järjestyksen sijasta. Tiettyjen attribuuttien (type ja value) arvot on myös standardoitu, mutta niiden käyttö ei estä oman rinnakkaisen luokittelun säilyttämistä. Ikävä kyllä standardi ei ole täysin yksiselitteinen ja sisältää jopa joitakin ristiriitaisuuksia. Olisi ollut hyvä, jos standardissa olisi ollut mukana DTD- tai skeematoteutus, jonka avulla rakenne olisi tullut eksplisiittisesti kuvattua.

Kotuksessa on 1990-luvulta alkaen pyritty rakenteistamaan sanakirjoja. Tarkoituksena parantaa sähköisten versioiden käytettävyyttä sekä sanakirja-aineistojen laatua. Tässä tutkielmassa kuvattu Suomen murteiden sanakirjan rakenteistus oli Kotuksen kolmas rakenteistusprojekti. Oman rakennekuvauksen määrittely on osoittautunut tärkeäksi vaiheeksi sanakirjan rakenteistuksessa, koska sen aikana luodaan yhteinen käsitteistö. Tämän käsitteistön avulla sisällön asiantuntijat eli sanakirjantoimittajat pystyvät hahmottamaan rakennetta, kun implisiittisistä rakenneosista tehdään rakenteistuksessa eksplisiittisiä. Yhteisen käsitteistön käytöllä helpotetaan myös toimitustyötä. Myös runsasta attribuuttien käyttöä pyritään välttämään toimitustyön helpottamiseksi. Aineiston kevyt tietokantaistus eli lyhennysmerkintöjen purkaminen ja yhdyssanojen täydentäminen on vielä kesken.

SMS:n rakennekuvausta verrattiin kahteen edellä mainittuun suositukseen, TEI:hin ja uuteen XmLex-standardiin. Vertailun tuloksena kirjoitettiin XSL-muunnos (liitteessä 2:) SMS:n rakenteesta XmLexin rakenteeseen, jolla testattiin muunnoksen onnistumista käytännössä. Lopputuloksen validoinnissa ongelmia aiheutti standardin DTD-toteutus, joka ei ollut täysin standardin mukainen. Kovin suuria ongelmia muunnoksessa ei tullut vastaan. Esimerkkien kohdalla standardista tuntui puuttuvan yksi hierarkian taso, jota SMS:ssä olisi tarvittu. Muut havaitut ongelmat paljastivat SMS:n rakenteen liian löysästi tai puutteellisesti määritellyt kohdat. Osittain ongelmat johtuivat toimittajien tekemistä vääristä tulkinnoista ja tästä seuranneesta rakenteiden väärinkäytöstä. Tiedon säilymistä muunnoksessa testattiin muuntamalla aineisto takaisin SMS-muotoon. Todettiin, että muunnoksessa häviävät tiedot liittyvät lähinnä ulkoasuun ja nekin voidaan säilyttää muokkaamalla hieman alkuperäistä muunnosskriptiä.

ISO pyrkii varmistamaan standardiensa yhteensopivuuden käyttämällä niiden pohjana yhteistä käsitteistöä. Käsitetietokanta DCR on vapaasti verkossa käytettävissä ja siihen

voi ehdottaa lisäyksiä. Samojen käsitteiden ja yhteensopivien hierarkkisten rakenteiden käyttö mahdollistaa erityyppisten sanakirjojen, leksikoiden ja termistöjen välisen muunnettavuuden.

Semanttisen webin myötä tulleet ontologiat mahdollistavat älykkäiden hakujen tekemisen eri tavoin koodatuista aineistoista. Kielitieteellisten aineistojen ontologiat, ja varsinkin sanakirjakäsitteitä sisältävät, ovat vielä keskeneräisiä, mm. ISO:n DCR-tietokanta ja EMELD-projektista lähtöisin oleva GOLD-ontologia. Kumpikin tarvitsisi lisää käyttäjiä, jotka täydentäisivät käsitteistöä. Toisaalta käyttäjien on vaikea ottaa käyttöön puutteellista sanastoa, johon lisäysten tekeminen on hankalaa ja kestää kauan.

Standardin määrittelemät käsitteet ovat useassa tapauksessa, varsinkin jos sanakirjaa on jo tehty pitkään, liian yleisiä käytettäväksi käytännön toimitustyössä. Omille rakennekuvauksille on edelleen tarvetta, koska niiden avulla voidaan tukea ja ohjata toimittajia artikkelien kirjoittamisessa. Standardin käytöstä voidaan hyötyä etenkin muuntamisprosessin yhteydessä, jolloin tarjoutuu mahdollisuus arvioida oman rakennekuvauksen laatua ja kehittää sitä paremmaksi. Kotuksen suunnitelmissa oleva yhteinen tietokanta vaatii yhteisen muodon, ja siihen tarkoitukseen ISO:n uusi standardi voisi olla varteenotettava vaihtoehto. Ontologioiden käyttöönottoa ei ole vielä harkittu.

EU:n juuri alkanut Clarin-hanke[12] (Common Language Resources and Technology Infrastructure) pyrkii luomaan tutkimusinfrastruktuuria humanistisille aloille. Työryhmä 5.2 keskittyy erityisesti sanastoaineistoihin ja niiden käsittelyyn käytettäviin ohjelmistoihin. Tässä yhteydessä tullaan ottamaan kantaa myös olemassa oleviin standardeihin ja antamaan suosituksia niiden käytöstä. Hanke on tosin vasta aivan alussa, joten konkreettisiä tuloksia joutuu vielä odottamaan.

12 http://www.clarin.eu/

Lähteet

Aap01 Aapala, K. et al. *Leksikografian termistö.* Kotimaisten kielten tutkimuskeskuksen julkaisematon raportti, 2001.

AmT88 Amsler, R., Tompa, F., An SGML-based standard for english monolingual dictionaries. Teoksessa *Information in Text*, Fourth Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary, Waterloo, Kanada, 1988, sivut 61-79.

BCF97 Bergenholtz, H., Cantell, I ., Fjeld, R., *Nordisk leksikografisk ordbok*. Universitetsforlaget, Oslo, 1997.

BeB00 Bell, J., Bird, S., *A preliminary study of the structure of lexicon entries*. Workshop on Web-Based Language Documentation and Description, Philadelphia, 2000. [http://www.ldc.upenn.edu/exploration/expl2000/papers/bell/bell.html, 16.3.2008]

BPS06 Bray T., Paoli J., Sperberg-McQueen C. M., Maler E., Yergeau F., Cowan J., toimittajat, *Extensible Markup Language (XML) 1.1 (Second Edition) W3C Recommendation 16 August 2006*. [http://www.w3.org/TR/2006/REC-xml11-20060816/, 4.1.2008]

EEI00 Erjavec, T., Evans, R., Ide, N., Kilgarriff, A., *The CONCEDE model for lexical databases*. Proc. of Second International Conference on Language Resources and Evaluation (LREC 2000), Pariisi, Ranska, 2000, sivut 355-362.

FaL03 Farrar, S., Langendoen, D. T., A linguistic ontology for the Semantic Web. *GLOT International,* 7, (3)2003, sivut 97–100.

FDM06 Francopoulo, G., Declerck, T., Monachini, M., Romary, L., *The relevance of standards for research infrastructures*. Proc. of The Fifth International Conference on Language Resources and Evaluation (LREC 2006), Genova, Italia, 2006.

Fra06 Francopoulo, G. et al., *Lexical markup framework (LMF)*. Proc. of The Fifth International Conference on Language Resources and Evaluation (LREC 2006), Genova, Italia, 2006.

Gen94 GENELEX Consortium, *Project EUREKA GENELEX, Report on the Semantic Layer*, Version 2.1., 1994. [http://www.tagmatica.fr/doc/En/Semantic/Semantic.htm, 16.3.2008]

GOL07 The GOLD Community, 2007. *General Ontology for Linguistic Description (GOLD) Version 0.45*. [http://www.linguistics-ontology.org/ns/gold/latest/gold.xml, 24.3.2008]

HaI06 Hayashi, Y., Ishida, T., *A dictionary model for unifying machine readable dictionaries and computational concept lexicons*. Proc. of The Fifth International Conference on Language Resources and Evaluation (LREC 2006), Genova, Italia, 2006.

IdR04 Ide, N., Romary, L., *A registry of standard data categories for linguistic annotation*. Proc. of the Fourth Language Resources and Evaluation Conference (LREC 2004), Lissabon, 2004, sivut 135-39.

IdR07 Ide, N., Romary, L., Towards International Standards for Language Resources. Teoksessa *Evaluation of Text and Speech Systems,* Dybkjaer, L., Hemsen, H., Minker, W. (toim.), Springer, 2007, sivut 263-84.

IdV95 Ide, N., Veronis, J., Encoding print dictionaries. *Computers and the Humanities,* 29, 1995, sivut 167-195.

IKR00 Ide, N., Kilgarriff, A., Romary, L., *A formal model of dictionary structure and content*. Proc. of Euralex 2000, Stuttgart, 2000, sivut 113-126.

IVW92 Ide, N., Veronis, J., Warwick-Amstrong, S., Calzolari, N., *Principles for encoding machine readable dictionaries*. Proc. of the Fifth EURALEX International Congress, Tampereen yliopisto, Tampere, 1992.

ISO07 International Standardisation Organisation, *ISO 1951:2007 Presentation/representation of entries in dictionaries*, 2007.

Jan05 Janssen, M., *Open source lexical information network*. Third International Workshop on Generative Approaches to the Lexicon, Geneve, Sveitsi, 2005.

LeD06 Le Meur, A., Derouin, M., *Lemma-oriented dictionaries, concept-oriented terminology and translation memories*. Proc. of The Fifth International Conference on Language Resources and Evaluation (LREC 2006), Genova, Italia, 2006.

LeM06 Le Meur, A., *XmLex and XmLex for bilingual dictionaries.* Kernerman Dictionary News, 14, 2005. [http://www.xmlex.net/lexicography/xmlexintro.pdf, 4.1.2008]

Len00 Lenci, A. et al., SIMPLE: A general framework for the development of multilingual lexicons. *International Journal of Lexicography*, 13, 4(2000), sivut 249-263.

Lit06 Litkowski, K., Computational lexicons and dictionaries. Teoksessa *Encyclopedia of Language and Linguistics*, *2. ed*. Brown K., et al. toimittajat, Elsevier Publishers, Oxford, 2006, sivut 753-761.

Mil90 Miller, G. A. toim., WordNet: An on-line lexical database. *International Journal of Lexicography,* 3, 4(1990), sivut 235-312.

Sim04a Simons, G. et al, *A model for interoperability: XML documents as an RDF database*. Proc. of the EMELD Workshop on Linguistic Databases and Best Practice, Detroit, 2004.

Sim04b Simons, G. et al, *The semantics of markup: Mapping legacy markup schemas to a common semantics.* Proc. of the 4th Workshop on NLP and XML (NLPXML-2004), Barcelona, 2004, sivut 25-32.

Str04 Strandberg, J., *Ei sanat salahan joua: fennistiikan murteenkeruun historiaa 1868–1925*. Pro gradu -tutkielma, Helsingin yliopiston suomen kielen laitos, Helsinki, 2004. [http://urn.fi/URN:NBN:fi-fe20041606]

Suo88 *Suomen murteiden sanakirjan toimitusohjeet* (2. lisätty ja täydennetty painos). Kotimaisten kielten tutkimuskeskuksen julkaisuja 54, Valtion painatuskeskus, Helsinki, 1988.

TEI07 *TEI P5: Guidelines for electronic text encoding and interchange.* [http://www.tei-c.org/release/doc/tei-p5-doc/html/, 4.1.2008]

Tuo89 Tuomi, T., *Suomen murteiden sanakirja, johdanto*. Kotimaisten kielten tutkimuskeskuksen julkaisuja 36, Valtion painatuskeskus, Helsinki, 1989.

Vri06 de Vriend, F. et al. *A unified structure for dutch dialect dictionary data*. Proc. of The Fifth International Conference on Language Resources and Evaluation (LREC 2006), Genova, Italia, 2006.

WPB02 Wittenburg, P., Peters, W., Broeder, D., *Metadata Proposals for Corpora and Lexica*. Proc. of the Third Language Resources and Evaluation Conference (LREC 2002), Las Palmas, Espanja, 2002.

XmL07 XmLex_v00.dtd [http://www.xmlex.net/lexicography/XmLexWorkbench.rar, 8.3.2008]

Liite 1: SMS DTD

Suomen murteiden sanakirjan koodauksessa käytetty DTD on suunniteltu Kotimaisten kielten tutkimuskeskuksessa.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- SMS:n rakennekuvaus                                    -->
<!-- (c) 2000-2006 Kotimaisten kielten tutkimuskeskus       -->

<!ELEMENT sms          (VakioArt | PoikkArt | EtunArt | YhdysJono)+>

<!-- VAKIOARTIKKELI                    -->
<!ELEMENT VakioArt    ( Hakus+, Sanal+, Tark*, SelJakso?, MuotoJakso?, Lev?, Huom*, ViitJakso?,
                        Jasennys?, (MerkRyhma*|EL*) )>
<!ATTLIST VakioArt
          mr-numerointi  (normaali|I|A)     'normaali'
>

<!-- POIKKEUSARTIKKELI                 -->
<!ELEMENT PoikkArt    ( Hakus+, (SelJakso | MuotoJakso | Lev | Huom | EsimJakso | Vrt | Ks |
                        AliHakus | KayttoAla | Korost | TakanViite )*, MerkRyhma* )>

<!-- ETUNUOLIARTIKKELI                 -->
<!ELEMENT EtunArt     ( Hakus+, Sanal*, Tark*, EtunViite, SelJakso? ) >

<!-- YHDYSSANARYHMA/JONO               -->
<!ELEMENT YhdysJono  (( EtunArt | VakioArt )*) >

<!-- HAKUSANA                          -->
<!ELEMENT Hakus       ( #PCDATA ) >
<!ATTLIST Hakus
       Id          ID          #REQUIRED
       Hom         CDATA  #IMPLIED
       Yhdys       (ei|on)     'ei'
       Norm        CDATA   #IMPLIED
>

<!-- TARKENTEET                        -->
<!ELEMENT Sanal       ( #PCDATA )>
<!ELEMENT Tark        ( #PCDATA )>
<!ELEMENT KayttoAla   ( #PCDATA )>

<!-- HUOMAUTUS                         -->
<!ELEMENT Huom        ( #PCDATA | Alue | ViiteHakus | Korost | Ai | Yi )*>

<!-- TYYPPIHUOMAUTUS                   -->
<!ELEMENT TyyppiHuom     ( #PCDATA | Alue | Pit )*>

<!-- JASENNYS                          -->
<!-- Generoidaan automaattisesti       -->
<!ELEMENT Jasennys          ( JMerkRyhma+ )>
<!ELEMENT JMerkRyhma     ( MRviite?, Sanal*, Tark*, SelJakso?, JMerkRyhma*)>
<!ELEMENT MRviite           EMPTY >
<!ATTLIST MRviite
       xref         CDATA     #IMPLIED>
```

```
<!-- SELITE                                  -->
<!ELEMENT SelJakso     ( #PCDATA | TakanViite | KayttoAla | Tark | AliHakus | Korost | Yi )*>

<!-- ALIHAKUSANA                             -->
<!ELEMENT AliHakus     ( #PCDATA )*>

<!-- MUOTOJAKSO                              -->
<!ELEMENT MuotoJakso   ( (Asu|AsuTaiv|Komp|Taiv|Tyvi|Vart)*, MuotoHuom* )>
<!ELEMENT Asu          ( #PCDATA | MuotoLev )*>
<!ELEMENT AsuTaiv      ( #PCDATA | MuotoLev )*>
<!ELEMENT Komp         ( #PCDATA | MuotoLev )*>
<!ELEMENT Taiv         ( #PCDATA | MuotoLev )*>
<!ELEMENT Tyvi         ( #PCDATA | MuotoLev )*>
<!ELEMENT Vart         ( #PCDATA | MuotoLev )*>
<!ELEMENT MuotoLev     ( #PCDATA | Muoto | KayttoAla | Lev )*>
<!ELEMENT Muoto        ( #PCDATA | Ai | Yi )*>
<!ELEMENT MuotoHuom    ( #PCDATA | Muoto | Alue | Pit | ViiteHakus | Korost )*>
<!ATTLIST MuotoHuom
        tyyppi (aineistonsijoitus|ryhmaraja|muukommentti) #IMPLIED>

<!-- LEVIKKI                                 -->
<!ELEMENT Lev          ( #PCDATA | KayttoAla | Alue | Pit | Korost )*>
<!ELEMENT Alue         ( #PCDATA )>
<!ATTLIST Alue
        jarjnro        CDATA    #IMPLIED
>
<!ELEMENT Pit          ( #PCDATA )>
<!ATTLIST Pit
        jarjnro        CDATA    #REQUIRED
>

<!-- VIITTAUKSET                             -->
<!ELEMENT ViitJakso    ( Osia?, Rinn?, Ks?, Vrt?, Yhd? )>
<!ELEMENT ViiteHakus   ( #PCDATA )>
<!ATTLIST ViiteHakus
        xref           CDATA    #IMPLIED
        HomViite       CDATA    #IMPLIED
        SignViite      CDATA    #IMPLIED
>
<!ELEMENT Rinn         ( #PCDATA | ViiteHakus | TakanViite | Pit )*>
<!ELEMENT Vrt          ( #PCDATA | ViiteHakus | TakanViite | Pit )*>
<!ELEMENT Ks           ( #PCDATA | ViiteHakus )*>
<!ELEMENT Yhd          ( #PCDATA | ViiteHakus )*>
<!-- Etunuoliviittaus -->
<!ELEMENT EtunViite    ( ViiteHakus+ )>
<!-- Osaluettelo -->
<!ELEMENT Osia         ( #PCDATA | ViiteHakus )*>
<!-- Takanuoliviittaus -->
<!ELEMENT TakanViite   ( ViiteHakus+ )>

<!-- MERKITYSRYHMA                           -->
<!ELEMENT MerkRyhma    ( Sanal*, Tark*, SelJakso?, MuotoJakso?, Lev?, Huom*, ViitJakso?,
                             EL*, MerkRyhma* )>
<!ATTLIST MerkRyhma
        Id             ID                   #IMPLIED
>
<!ELEMENT Sign         (#PCDATA) >
```

```
<!-- ESIMERKIT                                  -->
<!ELEMENT EL              ( EsimJakso+ )>
<!ATTLIST EL
            tyyppi   (normaali|poikkeava)       'normaali'
>
<!ELEMENT EsimJakso  ( Esim*, Pit )>
<!ELEMENT Esim          ( #PCDATA | Mur | Sit | SuljeSel | Vrt | KayttoAla | TakanViite )*>
<!ELEMENT Mur           ( #PCDATA | SuljeSel | Huutomerkki | Harvennus | Ai | Yi )*>
<!ELEMENT Sit           ( #PCDATA | Mur | SuljeSel )*>
<!ELEMENT SuljeSel      ( #PCDATA | TakanViite | KayttoAla | Sit | Vrt )*>

<!-- Kommenttihuutomerkki: [!] -->
<!ELEMENT Huutomerkki    EMPTY >

<!-- Ulkoasurakenteet: ylaindeksi, alaindeksi, kursiivi ja harvennus. -->
<!ELEMENT Yi            ( #PCDATA )>
<!ELEMENT Ai            ( #PCDATA )>
<!ELEMENT Korost        ( #PCDATA )>
<!ELEMENT Harvennus  ( #PCDATA )>
```

Liite 2: sms2xmlex.xsl

Luvussa 5 kuvatussa muunnoksessa käytetty XSL-muunnosskripti, jonka avulla voidaan muuntaa SMS:n DTD:n mukainen aineisto XmLex-muotoon.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
                    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                    xmlns:xlink="http://www.w3.org/TR/xlink"
>

<xsl:output method="xml" omit-xml-declaration="no" indent="yes"
            doctype-system="XmLex_V00.dtd"
/>

<xsl:template match="/">
  <xsl:apply-templates/>
</xsl:template>

<!--
Sanakirja:
sms => Dictionary
-->
<xsl:template match="sms">
  <Dictionary>
     <xsl:apply-templates/>
  </Dictionary>
</xsl:template>

<!--
Artikkelit:
VakioArt, EtunArt ja PoikkArt (@mr-numerointi)
=> DictionaryEntry(@style)
-->
<xsl:template match="VakioArt|EtunArt|PoikkArt">
  <DictionaryEntry>
    <!-- id-tunniste poimitaan hakusanasta-->
    <xsl:attribute name="identifier">
      <xsl:value-of select="Hakus/@Id"/>
    </xsl:attribute>
    <!-- homonyyminumero poimitaan hakusanasta-->
    <xsl:if test="Hakus/@Hom">
      <xsl:attribute name="homographNumber">
        <xsl:value-of select="Hakus/@Hom"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test="@mr-numerointi">
      <xsl:attribute name="style">
        <xsl:value-of select="@mr-numerointi"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:apply-templates/>
  </DictionaryEntry>
</xsl:template>
```

```
<!--
Yhdyssanajonot:
YhdysJono => NestEntry
-->
<xsl:template match="YhdysJono">
  <NestEntry>
    <!-- luodaan identifier hakusanan id:n perusteella -->
    <xsl:attribute name="identifier">
      <xsl:value-of select="concat('yhd-', *[1]/Hakus/@Id)"/>
    </xsl:attribute>
    <xsl:apply-templates/>
  </NestEntry>
</xsl:template>


<!--
Artikkelin "sisällysluettelo" laajoissa artikkeleissa:
Jasennys
=> poistetaan
-->
<xsl:template match="Jasennys"/>


<!--
Hakusana:
Hakus(@Yhdys, @Norm)
=> HeadwordCtn
     /Headword(@freeType)
     /SearchForm
-->
<xsl:template match="Hakus">
  <HeadwordCtn>
    <Headword>
      <xsl:if test="@Yhdys='on'"> <!-- yhdyssana -->
        <xsl:attribute name="freeType">compound</xsl:attribute>
      </xsl:if>
      <xsl:apply-templates/>
    </Headword>
    <!-- normalisoitu muoto -->
    <SearchForm><xsl:value-of select="@Norm"/></SearchForm>
  </HeadwordCtn>
</xsl:template>


<!--
Sanaluokka:
Sanal => PartOfSpeech(@freeType)
(poimitaan mukaan myös tätä seuraava tarkenne)
-->
<xsl:template match="Sanal">
  <PartOfSpeechCtn>
    <PartOfSpeech>
      <xsl:attribute name="freeValue"><xsl:apply-templates/></xsl:attribute>
      <!--<xsl:attribute name="value">STANDARDIARVO<xsl:attribute>-->
    </PartOfSpeech>
    <!-- Poimitaan mukaan mahdolliset seuraavat tarkenteet -->
    <xsl:if test="following-sibling::Tark[position()=1]">
          <xsl:apply-templates select="../Tark" mode="sanaluokkaan"/>
    </xsl:if>
```

```
  </PartOfSpeechCtn>
</xsl:template>


<!--
Tarkenne:
Tark => sisällöstä riippuen: Subcategorisation tai GrammaticalNumber (mon.)
            yhdistetään PartOfSpeechCtn:n, jos edellä sanaluokka
            yksinäinen tarkenne etunuoliartikkelin alussa ongelmallinen
-->
<xsl:template match="Tark" mode="sanaluokkaan">
  <xsl:choose>
    <xsl:when test="contains(.,'mon.')">
      <GrammaticalNumber>
        <xsl:attribute name="freeValue"><xsl:apply-templates/></xsl:attribute>
        <xsl:attribute name="value">plural</xsl:attribute>
      </GrammaticalNumber>
    </xsl:when>
    <xsl:otherwise>
      <Subcategorisation><xsl:apply-templates/></Subcategorisation>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!-- edellä sanaluokka, poimittu jo, ei tarvitse toistaa -->
<xsl:template match="Tark[preceding-sibling::Sanal[position()=1]]"/>

<!-- edellä ei sanaluokkaa -->
<xsl:template match="Tark[not(preceding-sibling::Sanal[position()=1])]">
  <xsl:choose>
    <!-- Subcategorisation, etunuoliartikkelin alussa ilman sanaluokkaa -->
    <xsl:when test="parent::EtunArt and not(contains(.,'mon.'))">
      <PartOfSpeechCtn>
            <!-- RAKENNEVIRHE, tässä pitäisi olla myös sanaluokka -->
        <Subcategorisation><xsl:apply-templates/></Subcategorisation>
      </PartOfSpeechCtn>
    </xsl:when>
    <!-- GrammaticalNumber (monikollisuus) -->
    <xsl:when test="contains(.,'mon.')">
      <GrammaticalNumber>
        <xsl:attribute name="freeValue"><xsl:apply-templates/></xsl:attribute>
        <xsl:attribute name="value">plural</xsl:attribute>
      </GrammaticalNumber>
    </xsl:when>
    <!-- Subcategorisation, muualla -->
    <xsl:otherwise>
      <Subcategorisation><xsl:apply-templates/></Subcategorisation>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--
Käyttöalamerkintä:
KayttoAla
=> sisällöstä riippuen: RangeOfApplication (tai Register, SubjectField, SenseQualifier)
-->
<xsl:template match="KayttoAla">
  <RangeOfApplication><xsl:apply-templates/></RangeOfApplication>
</xsl:template>
```

```
<!--
Huomautus:
Huom => Note
-->
<xsl:template match="Huom">
  <Note><xsl:apply-templates/></Note>
</xsl:template>

<!--
Huomautuksia hakusanan tyypeistä:
TyyppiHuom => Note
-->
<xsl:template match="TyyppiHuom">
  <Note><xsl:apply-templates/></Note>
</xsl:template>

<!--
Merkityksenselite:
SelJakso => Definition
-->
<xsl:template match="SelJakso">
  <Definition><xsl:apply-templates/></Definition>
</xsl:template>

<!--
Alihakusana:
AliHakus => HiddenEntry
-->
<xsl:template match="AliHakus">
  <HiddenEntry><xsl:apply-templates/></HiddenEntry>
</xsl:template>

<!--
Muotoluetteloiden kokoelma:
MuotoJakso => poistetaan (sisällön kannalta turha ryhmittely)
-->
<xsl:template match="MuotoJakso">
  <xsl:apply-templates/>
</xsl:template>

<!--
Asuluettelo levikkeineen:
Asu => VariantBlock
-->
<xsl:template match="Asu">
  <VariantBlock><xsl:apply-templates/></VariantBlock>
</xsl:template>

<!--
Asujen ja taivutusten luettelo levikkeineen:
AsuTaiv => VariantBlock
(taivutusmuotoja on hankala erottaa muista muodoista)
-->
<xsl:template match="AsuTaiv">
  <VariantBlock><xsl:apply-templates/></VariantBlock>
</xsl:template>

<!--
```

```
Komparaatiomuotojen luettelo levikkeineen:
Komp => InflectionBlock
-->
<xsl:template match="Komp">
  <InflectionBlock><xsl:apply-templates/></InflectionBlock>
</xsl:template>

<!--
Taivutusmuotojen luettelo levikkeineen:
Taiv => InflectionBlock
-->
<xsl:template match="Taiv">
  <InflectionBlock><xsl:apply-templates/></InflectionBlock>
</xsl:template>

<!--
Tyvimuotojen luettelo levikkeineen:
Tyvi => DerivationBlock
-->
<xsl:template match="Tyvi">
  <DerivationBlock><xsl:apply-templates/></DerivationBlock>
</xsl:template>

<!--
Vartalomuotojen luettelo levikkeineen:
Vart => DerivationBlock
-->
<xsl:template match="Vart">
  <DerivationBlock><xsl:apply-templates/></DerivationBlock>
</xsl:template>

<!--
Rakenteiden Asu, AsuTaiv, Komp, Taiv, Tyvi ja Vart alussa oleva otsikkoteksti:
=> FreeComment(@type=heading)
(välilyönnit ja -merkit poistetaan)
-->
<xsl:template match="Asu/text()|AsuTaiv/text()|Komp/text()|Taiv/text()|Tyvi/text()|Vart/text()">
  <xsl:if test=".!=" and normalize-space(.)!=',' and .!='.'">
    <FreeComment type="heading">
      <xsl:value-of select="."/>
    </FreeComment>
  </xsl:if>
</xsl:template>


<!--
Muoto ja sen levikkitieto:
MuotoLev
=> sijainnista hierarkiassa riippuen:
InflectionCtn, DerivationCtn tai VariantCtn
-->
<xsl:template match="MuotoLev">
  <xsl:choose>
    <xsl:when test="parent::Komp or parent::Taiv">
      <InflectionCtn><xsl:apply-templates/></InflectionCtn>
    </xsl:when>
    <xsl:when test="parent::Tyvi or parent::Vart">
      <DerivationCtn><xsl:apply-templates/></DerivationCtn>
    </xsl:when>
```

```
    <xsl:when test="parent::Asu or parent::AsuTaiv">
      <VariantCtn><xsl:apply-templates/></VariantCtn>
    </xsl:when>
    <xsl:otherwise>
      <!--muita vaihtoehtoja ei pitäisi olla kuin ym.-->
      <VIRHE/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--
Muoto:
=> sijainnista hierarkiassa riippuen:
Inflection, Inflection(@type=comparative), Derivation, Variant tai Fragment
-->
<xsl:template match="Muoto">
  <xsl:choose>
    <xsl:when test="ancestor::Komp">
      <Inflection>
        <xsl:attribute name="type">comparative</xsl:attribute>
        <xsl:apply-templates/>
      </Inflection>
    </xsl:when>
    <xsl:when test="ancestor::Taiv">
      <Inflection><xsl:apply-templates/></Inflection>
    </xsl:when>
    <xsl:when test="ancestor::Tyvi or ancestor::Vart">
      <Derivation><xsl:apply-templates/></Derivation>
    </xsl:when>
    <xsl:when test="ancestor::Asu or ancestor::AsuTaiv">
        <Variant><xsl:apply-templates/></Variant>
    </xsl:when>
    <xsl:otherwise>
      <!-- parent::MuotoHuom -->
      <Fragment><xsl:apply-templates/></Fragment>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--
Muotoihin liittyviä vapaamuotoisia huomautuksia:
MuotoHuom => LinguisticNote
-->
<xsl:template match="MuotoHuom">
  <LinguisticNote>
    <xsl:attribute name="freeType">
      <xsl:value-of select="@tyyppi"/>
    </xsl:attribute>
    <xsl:apply-templates/>
  </LinguisticNote>
</xsl:template>


<!--
Levikkitieto:
Lev => FreeTopic (standardista puuttuva leksikaalinen elementti)
-->
<xsl:template match="Lev">
  <FreeTopic type="levikki">
```

```
    <xsl:apply-templates/>
  </FreeTopic>
</xsl:template>

<!--
Aluelyhenne:
Alue => GeographicalUsage(@freeType=alue,@type=used)
-->
<xsl:template match="Alue">
  <GeographicalUsage>
    <xsl:attribute name="type">used</xsl:attribute>
    <xsl:attribute name="freeType">alue</xsl:attribute>
    <xsl:apply-templates/>
  </GeographicalUsage>
</xsl:template>

<!--
Pitäjälyhenne
Pit => GeographicalUsage(@freeType=pitäjä)
-->
<xsl:template match="Pit">
  <GeographicalUsage>
    <xsl:attribute name="type">used</xsl:attribute>
    <xsl:attribute name="freeType">pitäjä</xsl:attribute>
    <xsl:apply-templates/>
  </GeographicalUsage>
</xsl:template>

<!--
Artikkelitason viittaukset koottuna yhteen ryhmään:
ViitJakso => poistetaan turha ryhmittely
-->
<xsl:template match="ViitJakso">
  <xsl:apply-templates/>
</xsl:template>

<!--
Viitehakusana (linkki):
ViiteHakus(@xref, @HomViite, @SignViite)
=> Ptr(@xlink:href)
  HomographNumber
  SenseNumber
-->
<xsl:template match="ViiteHakus">
  <Ptr>
    <xsl:attribute name="xlink:href">
      <xsl:value-of select="@xref"/>
    </xsl:attribute>
    <xsl:if test="@HomViite">
      <HomographNumber><xsl:value-of select="@HomViite"/></HomographNumber>
    </xsl:if>
    <xsl:apply-templates/>
    <xsl:if test="@SignViite">
      <SenseNumber><xsl:value-of select="@SignViite"/></SenseNumber>
    </xsl:if>
  </Ptr>
</xsl:template>

<!--
```

```
Rinnakkaisviittaus (synonyyminen):
Rinn => SeeAlso (ei voida käyttää See-elementtiä, koska se on sallittu vain merkitysryhmissä)
-->
<xsl:template match="Rinn">
  <SeeAlso style="Rinn"><xsl:apply-templates/></SeeAlso>
</xsl:template>

<!--
Vertaa-viittaus:
Vrt => SeeAlso
-->
<xsl:template match="Vrt">
  <SeeAlso style="Vrt"><xsl:apply-templates/></SeeAlso>
</xsl:template>

<!--
Katso-viittaus:
Ks => SeeAlso
-->
<xsl:template match="Ks">
  <SeeAlso style="Ks"><xsl:apply-templates/></SeeAlso>
</xsl:template>

<!--
Yhdyssanaviittaus:
Yhd => SeeAlso
-->
<xsl:template match="Yhd">
  <SeeAlso style="Yhd"><xsl:apply-templates/></SeeAlso>
</xsl:template>

<!--
Etunuoliviittaus (hakusana selitetty toisessa artikkelissa):
EtunViite => SeeAlso
-->
<xsl:template match="EtunViite">
  <SeeAlso style="EtunViite"><xsl:apply-templates/></SeeAlso>
</xsl:template>

<!--
Viittaus hakusanan osiin:
Osia => SeeAlso
-->
<xsl:template match="Osia">
  <SeeAlso style="Osia"><xsl:apply-templates/></SeeAlso>
</xsl:template>

<!--
Takanuoliviittaus (rinnakkaisviittausta vastaava viittaus takaisinpäin):
TakanViite => SeeAlso (tässäkin pitäisi oikeastaa käyttää See-elementtiä)
-->
<xsl:template match="TakanViite">
  <SeeAlso style="TakanViite"><xsl:apply-templates/></SeeAlso>
</xsl:template>

<!--
Merkitysryhmä:
MerkRyhma => SenseGroup (Group=Grp)
-->
```

```
<xsl:template match="MerkRyhma">
  <SenseGrp><xsl:apply-templates/></SenseGrp>
</xsl:template>

<!--
Esimerkkilohko
EL(@tyyppi=normaali|poikkeava)
=> ?ExampleBlock(@style)
-->
<xsl:template match="EL">
  <!-- HUOM: ExampleBlock ei määritelty standardissa!
    <ExampleBlock>
      <xsl:attribute name="style">
        <xsl:value-of select="@tyyppi"/>
      </xsl:attribute>
      <xsl:apply-templates/>
    </ExampleBlock>
  -->
  <xsl:apply-templates/>
</xsl:template>

<!--
Esimerkkijakso (yhdistää esimerkin ja pitäjän):
EsimJakso => ExampleCtn
-->
<xsl:template match="EsimJakso">
  <ExampleCtn>
    <xsl:if test="not(Esim)">
      <Example/><!-- pelkkä pitäjä ei vars. esimerkkiä -->
    </xsl:if>
    <xsl:apply-templates/>
  </ExampleCtn>
</xsl:template>

<!--
Esimerkki:
Esim => Example
-->
<xsl:template match="Esim">
  <Example><xsl:apply-templates/></Example>
</xsl:template>

<!--
Murre-esimerkki:
Mur => Example(@style=dialect)
-->
<xsl:template match="Mur">
  <Example style="dialect"><xsl:apply-templates/></Example>
</xsl:template>

<!--
Sitaatti-esimerkki:
Sit => Citation tai Example(@style=citation)
-->
<xsl:template match="Sit">
  <Example style="citation"><xsl:apply-templates/></Example>
</xsl:template>

<!--
```

```
Esimerkin keskellä sulkeissa oleva selite:
SuljeSel => Gloss
-->
<xsl:template match="SuljeSel">
  <Gloss><xsl:apply-templates/></Gloss>
</xsl:template>

<!-- Korostukset -->
<xsl:template match="Yi">
  <sup><xsl:apply-templates/></sup>
</xsl:template>

<xsl:template match="Ai">
  <sub><xsl:apply-templates/></sub>
</xsl:template>

<xsl:template match="Korost">
  <i><xsl:apply-templates/></i>
</xsl:template>

<xsl:template match="Harvennus">
  <span style="sparse"><xsl:apply-templates/></span>
</xsl:template>

<!--
Toim. huom.:
Huutomerkki => <Gloss>[!]</Gloss>
-->
<xsl:template match="Huutomerkki">
  <Gloss>[!]</Gloss>
</xsl:template>

</xsl:stylesheet>
```