Perinnedatan automaattinen muotoilu XML-tekniikoin

Merja Ek, Heli Hakkarainen,
Pekka Kilpeläinen, Tommi Penttinen

Report A/2002/3

ISBN 951-781-263-9

UNIVERSITY OF KUOPIO

Department of Computer Science and Applied Mathematics

P.O.Box 1627, FIN-70211 Kuopio, FINLAND

Perinnedatan automaattinen muotoilu XML-tekniikoin

Merja Ek, Heli Hakkarainen, Pekka Kilpeläinen, Tommi Penttinen

Kuopion yliopisto, Tietojenkäsittelytieteen ja sovelletun matematiikan laitos
PL 1627 (Microkatu 1 D), 70211 Kuopio
{merja.ek, heli.hakkarainen, pekka.kilpelainen, tommi.penttinen}@cs.uku.fi

Tiivistelmä. XML-tekniikat tukevat dokumenttitiedon moninaiskäyttöä: eri esitysmuodot tulostusta ja digitaalista mediaa varten voidaan tuottaa samasta XML-muodosta. XML-tekniikoiden soveltaminen edellyttää perinnedatan muuntamista XML-muotoon. Tähän on olemassa menetelmiä, mutta vastaavaa muotoilumääritysten muuntamista XML-muotoon ei juurikaan ole tarkasteltu. Esimerkiksi XSL on voimakas XML-dokumenttien muotoilukieli, mutta kymmenien erilaisten elementtityyppien käsittely vaatii silläkin kymmenien vastaavien muotoilusääntöjen kirjoittamisen. Mikäli perinnedataan on sovellettu eksakteja muotoilumäärityksiä, datan XML-muodolle olisi hyvä pystyä tuottamaan niiden perusteella muotoilu mahdollisimman automaattisesti. Tarkastelemme XML-tekniikoiden sovellusesimerkkinä muotoiltuna tulostettavan riviperustaisen aineiston muuntamista XML-muotoon. Esitämme tätä varten kehittämämme arkkitehtuurin, joka muuntaa ja muotoilee annetun aineiston sen rakennetta ja muotoilua kuvaavan kontrollitiedoston ohjaamana automaattisesti. Arkkitehtuuri perustuu käsiteltävien tiedostojen XML-muotoon konvertointiin kehittämällämme ”XML-käärintäkielellä” nimeltä XW. Kerromme myös järjestelmän toteuttamisesta saaduista XML-tekniikoiden soveltamiskokemuksista.

1 Johdanto

XML-tekniikat tukevat dokumenttitiedon moninaiskäyttöä: eri esitysmuodot esimerkiksi tulostusta, verkkolevitystä ja CD-ROM-versioita varten voidaan tuottaa samasta XML-muodosta. Perinnedatan ja sen muotoilumääritysten muuntaminen XML-muotoon on kuitenkin usein käytännössä työlästä. Datan XML-muotoon muuntamista voi helpottaa soveltamalla korkeantasoista, tarkoitusta varten kehitettyä kieltä. ”XML-käärintäkieli” XW [1, 2] on XML-perustainen kuvauskieli määrämuotoisen datan muuntamiseksi XML-muotoon. XW-kääre on XML-muotoinen hierarkkinen malli lähtöaineiston rakenteelle ja samalla sitä noudattavalle tulosaineiston rakenteelle. Lähtöaineiston osien peräkkäisyyttä kuvataan peräkkäisillä elementeillä ja sisäkkäisyyttä sisäkkäisillä elementeillä. Osien vaihtoehtoisuus, valinnaisuus ja toistuvuus sekä osia erottavat erotinmerkkijonot kuvataan tämän mallin osana omilla XW-elementeillään ja -attribuuteillaan.

Datan konvertointi XML-muotoon on yleisesti tunnistettu ja runsaasti tarkasteltu ongelma [3]. Sen sijaan vastaavaa olemassa olevien muotoilumääritysten konvertoimista XML-muotoon soveltuvaksi on tarkasteltu vähemmän. Vaikka esimerkiksi XSL [4] on voimakas XML-dokumenttien muotoilukieli, kymmenien erilaisten ele-

menttien muotoileminen vaatii silläkin kymmenien vastaavien muotoilusääntöjen kirjoittamisen. Mikäli perinnedataan on sovellettu eksakteja muotoilumäärityksiä, myös vastaavien XML-dokumenttien muotoilu pitäisi pystyä tuottamaan niistä mahdollisimman automaattisesti.

Luvussa 2 kuvaamme esimerkkinä tarkastelemamme aineiston, riviperustaisen laskuaineiston sekä sitä kuvaavan kontrollitiedoston, joka kuvaa aineiston rakenteen ja muotoilun. Aineiston XML-konversio ja muotoilu voidaan automatisoida kontrollitiedoston ohjaamana. Luvussa 3 kuvailemme tämän mahdollisuuden toteuttavan arkkitehtuurin. Automatisointi perustuu kontrollitiedoston muuntamiseen XML-muotoon. Kontrollitiedoston XML-käärintä XW-kielellä esitellään luvussa 3.1. Myös varsinainen laskuaineisto voidaan konvertoida vastaavasti. Tarvittava kääreenkuvaus voi olla työläs kirjoittaa käsin, mutta XW-kielen yksinkertaisuuden takia se voidaan generoida automaattisesti kontrollitiedoston XML-muodosta (luku 3.2). Vastaavasti muotoilu laskuaineistolle voidaan tuottaa yleisellä muotoiluskriptillä, joka yhdistää kontrollitiedoston muotoilutiedot aineiston XML-esitykseen (luku 3.3). Tällainen automatisoidun muotoilun periaate on lupaava: sen avulla voi käsitellä eri aineistoja niiden kontrollitiedoston ohjaamana ilman rutiininomaisten muunnos- ja muotoilumääritysten kirjoittamista. Arkkitehtuuri on toteutettavissa olemassa olevilla XML-välineillä. Toteutusten laajamittaiseen käytettävyyteen liittyy kuitenkin tekniikoiden uutuuteen liittyviä ongelmia, joita käsitellään mm. suorituskykyarvioita sisältävässä luvussa 4.

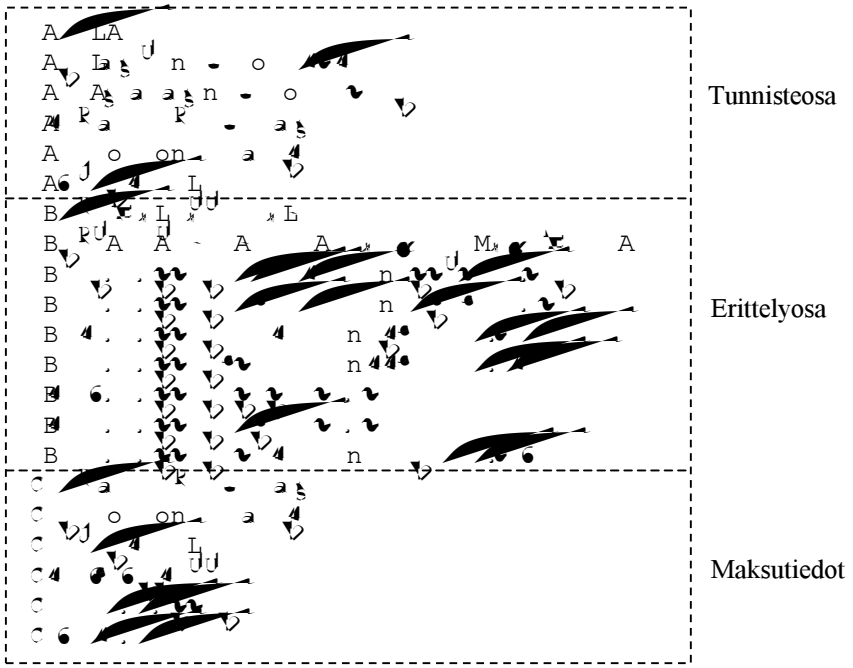
2 Esimerkkiaineistot

Esimerkkiaineistona tarkastellaan rivipohjaista puhelinlaskuaineistoa. Aineisto koostuu yksittäisistä laskuista, jotka jakautuvat edelleen kolmeen osaan: laskun tunnisteosaan (A), erittelyosaan (B) ja maksutietoihin (C). Laskun maksutiedot sisältävät laskutusosoitteen, viitenumeron, eräpäivän ja loppusumman (kuva 1). Kunkin rivin merkitys osoitetaan sen alussa olevalla rivitunnisteella (`A1`, `A2` jne).[1]

Laskuaineistoon liittyy kontrollitiedosto (kuva 2), joka kuvaa aineiston rakenteen ja muotoilun. Kontrollitiedostossa on rivitunnisteittain kerrottu aineiston kunkin rivin merkitys ja muotoilu. Kontrollitiedosto sisältää laskuaineistoa vastaavasti tunnistetiedot-osan, erittelyn ja maksutiedot. Kontrollitiedoston tunniste- ja maksutietorivien rakenne on seuraava: ensimmäisenä on rivin tunniste, toisessa kentässä kyseisen tiedon tai elementin nimi, ja näitä seuraavat tulostukseen käytettävä fontti, pistekoko ja tuloskentän x- ja y-koordinaatit (kentän vasemman alanurkan sijainti).

Laskun erittely- eli B-rivit on kuvattu eri tavalla. Niille ei ole määritelty y-koordinaattia, sillä ne tulostetaan järjestyksessä peräkkäin. Erittelyrivit koostuvat soluista, joille on määritelty x-koordinaatti ja muita muotoiluja. Erilaisia soluja yhdistelemällä saadaan tuotettua erilaisia rivejä. Kontrollitiedostossa erittelyrivit kuvataan luettelemalla rivin tunnisteen ja nimen jälkeen rivin sisältösolujen nimet.

[1] Aineisto on pelkistys todellisesta laskuaineistosta, joka sisältää enemmän yksilöitävää tietoa.



Kuva 1. Esimerkkilaskudata.

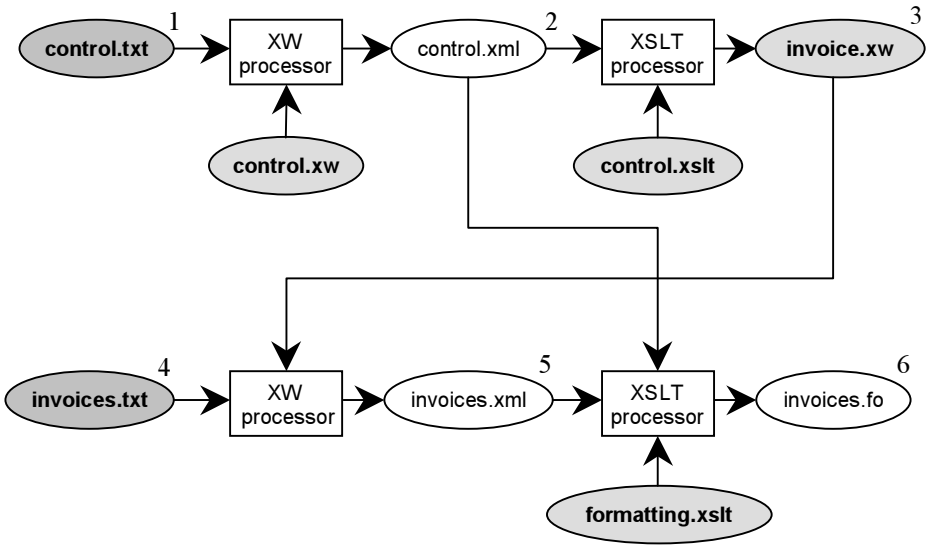
```
A1|Otsikko LASKU|Helvetica|12|76|65
A2|Laskunumero|Helvetica|10|432|65
A3|Asiakasnumero|Helvetica|10|432|80
...
B1|Otsikko PUHELUERITTELY|otsikkosolu1
B2|otsikkorivi|otsikkosolu1|otsikkosolu2...
B3|erittelyrivi puhelu|solu1|solu2|solu3|solu4|solu5
B4|erittelyrivi tekstiviesti|solu1|solu4|solu5
C1|Maksajan nimi|Courier|9|58|665
...
Erittelyn solut:
otsikkosolu1|Helvetica|12|40|left
otsikkosolu2|Helvetica|12|120|right
...
solu1|Helvetica|10|40|left
solu2|Helvetica|10|120|right
solu3|Helvetica|10|200|right
solu4|Helvetica|10|280|right
solu5|Helvetica|10|360|right
```

Kuva 2. Esimerkkiaineiston kontrollitiedosto.

Solujen muotoilu kuvataan kontrollitiedoston lopussa. Niihin liittyy muiden muotoilumääritysten lisäksi tieto siitä, tasataanko tieto solun oikeaan vai vasempaan reunaan (”right”/”left”).

3 Muotoiluprosessin automatisointi

Seuraavaksi kuvaamme, kuinka laskudatan XML-konversio ja muotoilu voidaan automatisoida dataa kuvaavan kontrollitiedoston ohjaamana. Kontrollitiedosto on prosessissa keskeisessä asemassa, koska se kuvaa datan rakenteen, (elementtien niminä käytettävät) osien nimet sekä muotoilutiedot. Kuva 3 esittää laskudatan käsittelyprosessia. Ensin kontrollitiedosto (1) muunnetaan käsittelyn helpottamiseksi XML-muotoon (2). Varsinaisen laskudatan XML-muotoon muuntava kääre (3) voidaan seuraavaksi generoida kontrollitiedoston XML-muodosta XSLT:llä. Tämän kääreen ohjaamana XML-muotoon (5) muunnettu laskuaineisto (4) voidaan lopulta muotoilla yleisen XSLT-skriptin (formatting.xslt) ohjaamana. Sama muotoiluskripti pätee eri aineistoihin, sillä se soveltaa aineistoa kuvaavan kontrollitiedoston (2) sisältämiä määrityksiä. Tuloksena syntyvä laskuaineiston XSL-FO-muotoinen muotoiltu esitys (6) voidaan lopulta tulostaa PDF:ksi, PostScriptiksi jne.



Kuva 3. Muotoiluprosessin automatisointi.

3.1 Kontrollidatan XML-käärintä

Muotoiluprosessin automatisointi perustuu kontrollitiedoston muuntamiseen XML-muotoon. Kuvaamme tämän suoraviivaisen muunnoksen XW-käärintäkielellä. Kuvassa 4 on esitetty XW-kääre laskuaineiston kontrollitiedoston muuntamiseksi XML-muotoon (kuva 5).

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <xw:wrapper ...>
3  <kontrollitiedosto>
4     <tunnistetiedot xw:childterminator="\n">
5        <rivi xw:starter="\^A"...>
6           <tunniste>A<xw:collapse/></tunniste>
7           <nimi xw:childseparator="\s">
8              <xw:collapse xw:maxoccurs="unbounded">
9                 <xw:collapse/>_
10             </xw:collapse>
11          </nimi>
12          <fontti/> <koko/> <x/> <y/>
13       </rivi>
14    </tunnistetiedot>
15    <erittely xw:childterminator="\n">
16       <erittelyrivi xw:starter="\^B"...>
17          <tunniste>B<xw:collapse/></tunniste>
18          <nimi xw:childseparator="\s">
19             <xw:collapse xw:maxoccurs="unbounded">
20                <xw:collapse/>_
21             </xw:collapse>
22          </nimi>
23          <solu xw:maxoccurs="unbounded"/>
24       </erittelyrivi>
25    </erittely>
26    <maksutiedot xw:childterminator="\n">
27       <rivi xw:starter="\^C"...>
28          ...
29       </rivi>
30    </maksutiedot>
31    <erittelyn_solut xw:starter="\^Erittelyn solut:\n"...>
32       <solu xw:maxoccurs="unbounded" xw:childseparator="|">
33          <nimi/> <fontti/> <koko/> <x/> <a/>
34       </solu>
35    </erittelyn_solut>
36 </kontrollitiedosto>
37 </xw:wrapper>
```

Kuva 4. Kontrollitiedoston XML-muotoon muuntava XW-kääre.

Kääreessä kuvataan kontrollitiedoston (rivit 3-36) koostuvan neljästä osasta: tunnistetiedoista (rivit 4-14), erittelystä (rivit 15-25), maksutiedoista (rivit 26-30) ja erittelysolujen muotoilutiedoista (rivit 31-35). Kutakin osaa kuvataan XW-kääreessä elementillä. Nämä XW-nimiavaruuteen kuulumattomat tuloselementit tulostetaan

kirjoitetussa muodossa tulokseen. Jos elementillä on kääreessä lapsielementtejä, tulostuvat samat lapsielementit tulokseen. Kääreen tyhjien elementtien sisällöksi tulee tulokseen lähtöaineiston vastaavan osan (teksti)sisältö. Elementtiä vastaavan osan aloittava tai lopettava merkkijono määritellään attribuuteilla `xw:starter` ja `xw:terminator`. Vaihtoehtoisesti osan aliosille voi määritellä esimerkiksi lopettavan merkkijonon sisältävää osaa vastaavassa elementissä attribuutilla `xw:childterminator` (esim. rivi 4). Osan toistuminen ilmaistaan XML Schema –tyyliin [5] attribuuteilla `xw:minoccurs` ja `xw:maxoccurs`. Elementti `xw:collapse` tuottaa tulosdokumenttiin itseään vastaavan syöteosan sisällön. Sen avulla esimerkiksi laskun osan nimeä kuvaavan kentän sisältö muutetaan tulosdokumenttiin `nimi`-elementin sisällöksi (kuva 4, rivit 8-10 ja kuva 5, rivi 6).

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <kontrollitiedosto>
3     <tunnistetiedot>
4        <rivi>
5           <tunniste>A1</tunniste>
6           <nimi>Otsikko_LASKU_</nimi>
7           <fontti>Helvetica</fontti> <koko>12</koko>
8           <x>76</x> <y>65</y>
9        </rivi>
10       ...
11    </tunnistetiedot>
12    <erittely>
13       <erittelyrivi>
14          <tunniste>B3</tunniste>
15          <nimi>erittelyrivi_puhelu_</nimi>
16          <solu>solu1</solu> <solu>solu2</solu>
17          <solu>solu3</solu> <solu>solu4</solu>
18          <solu>solu5</solu>
19       </erittelyrivi>
20       ...
21    </erittely>
22    ...
23    <erittelyn_solut>
24       <solu>
25          <nimi>otsikkosolu1</nimi>
26          <fontti>Helvetica</fontti> <koko>12</koko>
27          <x>40</x> <a>left</a>
28       </solu>
29       ...
30    </erittelyn_solut>
31 </kontrollitiedosto>
```

Kuva 5. Kontrollitiedosto XML-muodossa.

3.2 Käärekuvauksen automatisointi

Laskuaineisto on XML-käsittelyä varten muunnettava XML-muotoon. Tuotettava XML-muoto on esitetty kuvassa 6.[2] Tämä XML-muunnos voidaan tehdä XW-kääreenkuvauskielellä. Esimerkiksi laskuaineiston (kuva 1) ensimmäinen rivi

```
A1|LASKU
```

muunnetaan kuvan 6 rivin 5 esittämään muotoon. Kuvassa 7 on XW-kääre aineiston muuntamiseksi XML-muotoon. Muunnos tapahtuu tunnistamalla rivi tunnisteestaan ”A1” (kuva 7, rivit 7–13). Vastaavasti tunnistetaan ja muunnetaan XML:ksi myös muut laskuaineiston rivit, vaihtaen vain rivitunnistetta ja luotavan elementin nimeä.

Laskuaineistot saattavat olla rakenteeltaan rikkaita, ja muunnosmäärittelyn (esim. XW-kääreen) laatiminen aineistolle vaatii paljon rutiininomaista kirjoittamista. Muunnos voidaan automatisoida käyttäen XML-muotoista kontrollitiedostoa parametritiedostona, josta aineiston rivitunnisteet ja XML-muodossa käytettävien elementtien nimet haetaan. Näin kaikki luotavat elementit saadaan muodostettua samalla kaavalla.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <data>
3   <lasku>
4      <tunnistetiedot>
5         <Otsikko_LASKU_>LASKU</Otsikko_LASKU_>
6         <Laskunumero >Laskunumero: 14045</Laskunumero >
7         <Asiakasnumero_>Asiakasnumero: 73052</Asiakasnumero_>
8         ...
9      </tunnistetiedot>
10     <erittely>
11        ...
12        <erittelyrivi_puhelu_>
13           <solu1>2.8.2002</solu1> <solu2>118</solu2>
14           <solu3>14 min</solu3> <solu4>20010</solu4>
15           <solu5>5.02</solu5>
16        </erittelyrivi_puhelu_>
17        ...
18     </erittely>
19     <maksutiedot>
20        <Maksajan_nimi_>Pauli Puhelias</Maksajan_nimi_>
21        <Maksajan_katuosoite_>Johdonmutka 24</Maksajan_katuosoite_>
22        ...
23     </maksutiedot>
24  </lasku>
25 </data>
```

Kuva 6. Laskuaineisto XML-muodossa.

Kontrollitiedosto sisältää kaiken tarvittavan tiedon laskuaineiston muuntamiseksi, joten kun kontrollitiedosto on XML-muodossa, siitä voidaan generoida laskuaineis-

[2] Käytämme elementtiniminä kontrollitiedoston XML-käärinnän yhteydessä poimittuja tunnisteita kuten `Otsikko_LASKU_`.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <xw:wrapper xmlns:xw="http://www.cs.uku.fi/XW/2001"
3        xw:inputencoding="ISO-8859-1" xw:outputencoding="ISO-8859-1"
4        xw:sourcetype="text">
5     <data>
6        <lasku xw:maxoccurs="unbounded">
7           <tunnistetiedot>
8              <xw:collapse xw:starter="\^A1" xw:childseparator="|">
9                 <xw:ignore/> <!-- tunnistekentän (tyhjä) loppu -->
10                <Otsikko_LASKU_/>
11             </xw:collapse>
12             ...
13          </tunnistetiedot>
14          <erittely>
15             <xw:CHOICE xw:maxoccurs="unbounded">
16                ...
17                <erittelyrivi_puhelu_ xw:childseparator="|"
18                      xw:starter="\^B3">
19                   <xw:ignore/>
20                   <solu1/> <solu2/> <solu3/> <solu4/> <solu5/>
21                </erittelyrivi_puhelu_>
22                ...
23             </xw:CHOICE>
24          </erittely>
25          <maksutiedot>
26             <xw:collapse xw:childseparator="|" xw:starter="\^C1">
27                <xw:ignore/>
28                <Maksajan nimi_/>
29             </xw:collapse>
30             ...
31          </maksutiedot>
32       </lasku>
33    </data>
34 </xw:wrapper>
```

Kuva 7. Laskuaineistolle automaattisesti muodostettava XW-kääre.

tolle XW-kääre. Tämä vaihe toteutettiin XSLT:llä [6]. XSLT-skripti saa syötteenä XML-muotoisen kontrollitiedoston. Edellä esitetty XW-kääreen osa (kuva 7, rivit 7–13) luodaan XSLT:llä seuraavasti:

```
<xsl:template match="rivi">
   <xw:collapse  xw:starter="\^{tunniste}"
                 xw:childseparator="|">
      <xw:ignore/>
      <xsl:element name="{nimi}"/>
   </xw:collapse>
</xsl:template>
```

Kontrollitiedoston `rivi`-elementin `tunniste`-lapsen sisältö otetaan siis attribuutin `xw:starter` arvoksi, ja sen `nimi`-lapsi tuottaa nimen luotavalle elementille. Kaikki XML-muotoisen kontrollitiedoston tunnistetietojen ja maksutietojen `rivi`-elementit voidaan käsitellä tällä samalla XSLT-kaavaimella.

Myös `erittelyrivi`-elementeistä luotavaan XW-kääreen osaan (kuva 7, rivit 17–21) tarvittavat nimi ja rivin aloittava tunniste tuotetaan XML-muotoisesta kontrollitiedostosta samalla tavalla:

```
<xsl:template match="erittelyrivi">
   <xsl:element name="{nimi}">
      <xsl:attribute name="xw:starter">\^<xsl:value-of
         select="tunniste"/></xsl:attribute>
      <xsl:attribute
         name="xw:childseparator">|</xsl:attribute>
      <xw:ignore/>
      <xsl:for-each select="solu">
         <xsl:element name="{.}"/>
      </xsl:for-each>
   </xsl:element>
</xsl:template>
```

Lisäksi jokaista erittelyrivin solua kohden tuotetaan kääreenkuvaukseen vastaavan niminen elementti (kuva 7, rivi 20).

Näin kaikki kääreen muodostamiseen tarvittavat tiedot saadaan kontrollitiedostosta, ja kääre voidaan muodostaa automaattisesti koko laskuaineistolle. Tuloksena saatu XW-kääre (kuva 7) muuntaa varsinaisen laskudatan XML-muotoon (kuva 6). Kontrollitiedostoa hyväksikäyttäen XW-kääre — joka saattaa todellisen laskuaineiston tapauksessa olla jopa kymmeniä sivuja pitkä — saatiin tuotettua kahdella yksinkertaisella ja lyhyellä muunnoksella.

3.3 Muotoilumäärittelyn automatisointi

Laskudatan XML-muotoon muuntamisen jälkeen sen muotoiluun voidaan käyttää XML-välineitä. Muotoilun automatisointia varten laskuaineistolle kirjoitettiin XSLT-skripti, joka kontrollitiedostoa hyväksikäyttäen muuntaa aineiston muotoilun sisältävään XSL-FO-muotoon.

Muotoiluskripti saa syötteenä XML-muotoisen laskuaineiston. Skripti toteuttaa muotoilun noutamalla elementtien muotoilutiedot (kirjasintyyppi ja -koko sekä sijainti) elementtinimen perusteella XML-muotoisesta kontrollitiedostosta ja muodostamalla niillä parametroidun muotoiluolion. Esimerkiksi tunnistetietojen lapsielementtien XSL-muotoilu on kuvattu kuvan 8 riveillä 3–20.

Yksittäisten elementtien tietoja on pystyttävä asettelemaan lopputulokseen mielivaltaisesti. Tähän asetteluun käytettiin XSL:n `block-container`-muotoiluoliota. Sille voidaan antaa koordinaatteina objektin sijainti muodostettavalla sivulla, kun attribuutin `position` arvo on `"absolute"` (kuva 8, rivit 15–20).

Maksutietojen elementit käsitellään samalla tavalla kuin tunnistetietojen elementit. Erittelyrivien muotoileminen on hieman haastavampaa. Erittelyrivien tiedoilla ei ole y-koordinaattia, sillä rivit tulostetaan vain järjestyksessä allekkain. Jokainen rivi voi koostua eri määrästä eri tavoin sisennettäviä soluja. Tämän takia erittelyrivien muotoilu toteutettiin käsittelemällä jokaista erittelyriviä omana taulukkonaan, jolloin jokaiselle riville voidaan määritellä solukohtaisesti sarakkeiden leveydet (kuva 9).

```
<xsl:template match="tunnistetiedot/*">
   <xsl:variable name="elementtinimi" select="name()"/>
   <xsl:variable name="rivi" select=
      "document($kontrollitiedosto)/kontrollitiedosto/
      tunnistetiedot/rivi[nimi=$elementtinimi]"/>
   <xsl:variable name="fontti" select="$rivi/fontti"/>
   <xsl:variable name="koko" select="$rivi/koko"/>
   <xsl:variable name="x" select="$rivi/x"/>
   <xsl:variable name="y" select="$rivi/y"/>
   <fo:block-container height="1cm" width="20cm" top="{$y}pt"
      left="{$x}pt" position="absolute">
      <fo:block font-family="{$fontti}" font-size="{$koko}pt">
         <xsl:apply-templates/>
      </fo:block>
   </fo:block-container>
</xsl:template>
```

Kuva 8. Muotoiluskriptin tunnistetiedot muotoileva osa.

```
<xsl:template match="erittely/*">
   <fo:table>
      <xsl:variable name="erittelyrivinimi" select="name()"/>
      <xsl:for-each select=
         "document($kontrollitiedosto)/kontrollitiedosto/
         erittely/erittelyrivi[nimi=$erittelyrivinimi]/solu">
         <xsl:call-template name="sarake"/>
      </xsl:for-each>
      <fo:table-body>
         <fo:table-row>
            <xsl:for-each select="*">
               <xsl:variable name="solunimi" select="name()"/>
               <xsl:variable name="rivi" select=
                  "document($kontrollitiedosto)/kontrollitiedosto/
                  erittelyn_solut/solu[nimi=$solunimi]"/>
               <xsl:variable name="fontti" select="$rivi/fontti"/>
               <xsl:variable name="koko" select="$rivi/koko"/>
               <xsl:variable name="a" select="$rivi/a"/>
               <fo:table-cell padding="0.5mm">
                  <fo:block font-family="{$fontti}"
                     font-size="{$koko}pt" text-align="{$a}">
                     <xsl:apply-templates/>
                  </fo:block>
               </fo:table-cell>
            </xsl:for-each>
         </fo:table-row>
      </fo:table-body>
   </fo:table>
</xsl:template>
```

Kuva 9. Erittelyrivien muotoilu.

```
1  <xsl:template name="sarake">
2     <xsl:param name="max_pituus" select="450"/>
3     <xsl:variable name="solunimi" select="text()"/>
4     <xsl:variable name="x" select=
5           "document($kontrollitiedosto)/kontrollitiedosto/
6           erittelyn_solut/solu[nimi=$solunimi]/x"/>
7     <xsl:variable name="nextsolunimi" select=
8           "following-sibling::solu[1]"/>
9     <xsl:variable name="nextx" select=
10          "document($kontrollitiedosto)/kontrollitiedosto/
11          erittelyn_solut/solu[nimi=$nextsolunimi]/x"/>
12    <xsl:choose>
13       <xsl:when test="following-sibling::solu">
14          <fo:table-column column-width="{$nextx - $x}pt"/>
15       </xsl:when>
16       <xsl:otherwise>
17          <fo:table-column column-width=
18                "{$max_pituus - $x}pt"/>
19       </xsl:otherwise>
20    </xsl:choose>
21 </xsl:template>
```

Kuva 10. Erittelyrivien sarakkeiden määrittely.

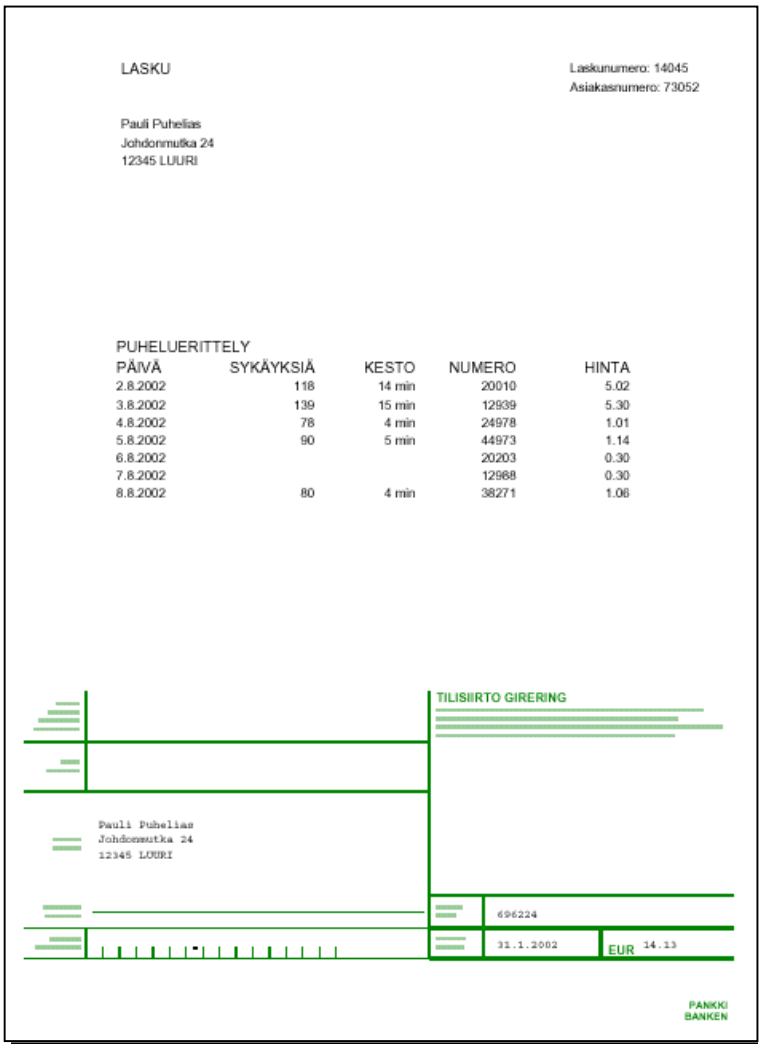
Taulukon muotoilu tapahtuu niin, että ensin tuotetaan sarakkeiden leveydet (kuva 9, rivit 4–7 ja kuva 10). Ne saadaan laskettua vähentämällä solua seuraavan solun x-koordinaatista solun oma x-koordinaatti (kuva 10, rivit 14 ja 17). Sen jälkeen erittelyrivin jokainen `solu` muotoillaan omaksi taulukon solukseen soveltaen kontrollitiedostosta saatavia fontti-, koko- ja tasaustietoja (kuva 9, rivit 9–21).

Muotoiluskripti tuottaa XSL-FO-muotoisen laskun, josta lasku voidaan tulostaa esimerkiksi PDF-muotoon (kuva 11).

4 Arviointia

Mittasimme laskuaineiston muuntamis- ja muotoiluaikoja kokeellisesti. Kontrollitiedoston käsittelyn vaatimaa aikaa ei testattu, koska käytännössä kontrollitiedosto on kiinteän kokoinen, ja se käsitellään vain kerran yhtä aineistotyyppiä kohden.

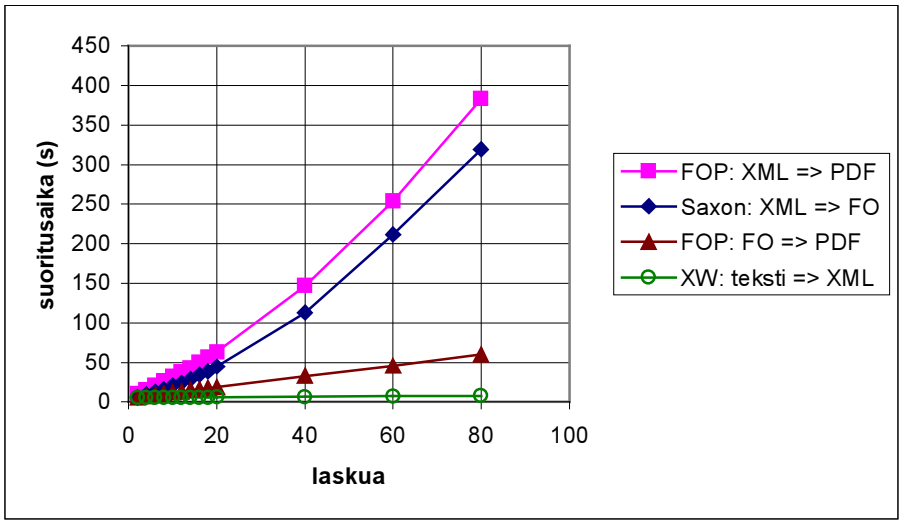
Testaus suoritettiin SunOS 5.8 -käyttöjärjestelmässä 750 MHz prosessoria käyttävällä Sun Fire 280R -palvelimella ja Java-versiolla J2SE v1.4.0. XSLT-prosessorina käytettiin sekä Xalania [7] että Saxonia [8]. PDF-muotoilu suoritettiin Apachen FOP-prosessorilla [9]. Käytettyä prosessoriaikaa mitattiin Unixin `time`-komennolla. Lähtöaineistona oli kaksi tyypillistä laskua (neljä sivua) sisältävä 6,6 kilotavun tekstitiedosto ja siitä kopioimalla muodostetut monikerrat. Mittasimme erikseen laskuaineiston muuntamisen XW:llä XML:ksi ja sen muuntamisen edelleen PDF:ksi. XML–PDF-muunnos suoritettiin kahdella eri tavalla: yhdessä askeleessa FOPilla käyttäen sen sisäänrakennettua XSLT-prosessoria (Xalan) ja kahdessa askeleessa suorittaen XSLT-muunnos Saxonilla ja PDF-muotoilu FOPilla. Tulokset kuvassa 12.



Kuva 11. Muotoiltu lasku PDF-muodon kautta tulostettuna

Tuloksista nähdään, että XW:llä suoritetun XML-konversion vaatima aika on murto-osa muotoilun vaatimasta ajasta. Tästä taas selvästi suurin osa kuluu XSLT-muunnokseen, jonka vaatima aika kasvaa lopullisen PDF-muotoilun ajantarvetta nopeammin. 80 laskun aineistolla suoritusaikojen ero on jo viisinkertainen. Käytetyllä XSLT-prosessorilla ei ollut merkittävää vaikutusta suoritusaikaan.

Nykyisellään kokeiltu käsittelytapa ei vaikuta käytännölliseltä vaihtoehdolta massatulosteiden muotoiluun: minuutissa muotoiltiin ainoastaan 20 laskua. Toisaalta pienivolyymisten verkkolaskuaineistojen käsittelyyn XSLT-prosessoreiden nykyinenkin suorituskyky voi olla riittävä.



Kuva 12. Laskuaineiston vaatimia muunnos- ja muotoiluaikoja

Automaattisen muunnoksen ja muotoilun perimmäinen tavoite on pystyä käsittelemään erilaisia aineistoja sisällöstä riippumatta samoilla muunnosohjelmilla räätälöimättä niitä eri aineistoille sopiviksi. Edellä esitetyt muunnoskuvaukset ovat pitkälti sisällöstä riippumattomia. Aineistoriippumattomien muotoiluskriptien käyttö aiheuttaa sen, että kaiken lopputulokseen halutun tekstin on sisällyttävä elementteihin. XML-aineistoihin tulee siksi esimerkiksi sellaisia kömpelön tuntuisia elementtejä kuin `<laskunumero>Laskunumero: 14045</laskunumero>`. Sisältöriippumattomuus mahdollistaa kuitenkin sen, että kerran kirjoitettuja muunnosohjelmia voidaan käyttää lukemattomien aineistojen käsittelyyn. Riittää, että aineisto on erotinmerkein eroteltua, ja sille on olemassa vastaava kontrollitiedosto. Kolmen suhteellisen lyhyen skriptin ohjaamana koko muunnos- ja muotoiluputki toimii automaattisesti.

XML-tekniikoiden käytössä on vielä toistaiseksi ongelmana niiden uutuus. Törmäsimme kokeilussa erityisesti XSL-prosessorien puutteisiin. Ilmeisesti täydellisiä XSL-toteutuksia ei ole.

Kiitokset

Edellä esitetty tutkimus on tehty XRAKE (XML-rajapintojen kehittäminen) –hankkeessa, jonka rahoittavat TEKES/EAKR, Deio Oy, Enfo Group Oyj, Järvi-Suomen Ohjelmapalvelu Oy, Kuopion yliopistollinen sairaala, Medigroup Oy, SysOpen Oyj ja TietoEnator Oyj.

Lähteet

1. Ek, M., Hakkarainen, H., Kilpeläinen, P., Kuikka, E., Penttinen, T.: Describing XML wrappers for information integration. In: Proc. of XML Finland 2001, Tampere, Nov. 2001, 38–51
2. Ek, M., Hakkarainen, H., Kilpeläinen, P., Penttinen, T.: Declarative XML wrapping of data. Report A/2002/2. Univ. of Kuopio, Dept. of Comp. Sci. and Applied Math., 2002
3. Waldt, D.: Getting data into XML: Data collection and conversion techniques. Abstract. In: Proc. of XML Europe, Barcelona, May 2002
4. Adler, S. et al, editors: Extensible Stylesheet Language (XSL) Version 1.0. W3C Rec., October 2001, http//:www.w3.org/TR/xsl/
5. Thompson, N., Beech, D., Maloney, M., Mendelsohn, N. editors: XML Schema Part 1: Structures. W3C Rec., May 2001
6. Clark, J. editor: XSL Transformations (XSLT) Version 1.0. W3C Rec., Nov. 1999
7. Apache Xalan version 2.2.D11, 2001, http://xml.apache.org/xalan-j/index.html
8. Saxon version 6.4.4, 2001, http://saxon.sourceforge.net/saxon6.4.4/index.html
9. Apache FOP version 0.20.4, 2002, http://xml.apache.org/fop/index.html