XML-põhise dokumendihalduse võimalikkusest Eesti Wabariigis

Jan Willemson

20. veebruar 2001. a.

Sissejuhatus

Uksest ja aknast sisse murdev digitaalajastu pakub meile meie igapäevases töös palju uusi võimalusi, kuid esitab samas ka senitundmatuid väljakutseid. Nii näiteks on heli pakkimise meetodid jõudnud nõnda kaugele, et korraliku kvaliteediga MP3-esid võib igaüks endale mõne minuti jooksul mitme CD jagu koduarvutisse tõmmata; samas valmistab see palju peavalu muusikatöösturitele, kes tahaksid kuulajate arvelt oma kukrut täita.

Uusi võimalusi ja probleeme näeme iga päev ka elektroonilises dokumendihalduses. Ühest küljest on tekstiredaktorid jõudnud niikaugele, et ka eriettevalmistuseta Inimene neid suurema vaevata kasutada suudab, kuid teisest küljest sellega arvuti kasulikkus dokumendi elutsükli jaoks ka lõppeb. Peale valmimist trükitakse kiri, arve, tõend vms välja, kirjutatakse käsitsi alla ning lükatakse kausta. Samas kehtib Eestis juba peaaegu aasta otsa digitaallkirja seadus ja kõik saavad aru, et üks 45 gigabaidine kõvaketas võtab arhiivis tunduvalt vähem ruumi, kui temale mahtuvate dokumentide väljatrükid paberkaustadena.

Miks me siis ikka veel kõiki arvutustehnika poolt pakutavaid võimalusi ei kasuta? Ühe põhjusena saab välja tuua riiklikul ja asutuste tasemel vastava asjaajamiskorra puudumise. Hullem veel – keegi ei kujuta endale täpselt ette, kuidas selline kord võiks välja näha, sest keegi pole vastavat korda kunagi näinud. Seega tuleb Eestis lahendada küllalt ebameeldiv ülesanne: luua esimene kogu riiki hõlmav digitaalse asjaajamise süsteem, mis hõlmaks nii dokumentide loomise, liikumise, signeerimise kui ka arhiveerimisega seotud küsimused. Ebameeldivusi põhjustab seejuures Eesti väiksus: kui meie poolt hoole ja armastusega välja töötatud lahendus ei ühildu nt Euroopa Liidu tulevaste dokumendihalduspõhimõtetega, oleme palju aega ja ressurssi ilma asjata raisanud. Samas pole mõtet oodata mitu aastat kuni euromasinavärk meile järele jõuab.

Ülalmainitud probleemide lahendamiseks loodi 2000. aastal Riigikantselei juurde Eesti dokumendihalduse programmi töörühm. Rühma senise tegevuse tulemused on kokku võetud mitmetesse aruannetesse, mille huvitatud lugeja leiab Riigikantselei veebist lehelt [1].

Üks võimalik baastehnoloogia, millest dokumendihalduse kontekstis viimasel ajal üha sagedamini kuulda võib, on XML (*eXtensible Markup Language*). Jämedalt öeldes annab XML igale kasutajale võimaluse defineerida oma vajadustele vastav HTML-i sarnane keel ning luua selles keeles suhtlevad rakendused. Hea tahtmise korral võib Inimene XML-dokumendi lähtekoodist aru saada, millline omadus võimaldab Inimesel vahelduseks end ka arvutustehnika üle valitsejana tunda.

Tuleb kahetsusega tõdeda, et XML-i kui fenomeni ümber hõljub arvukalt müüte ja lootusi, nii loodetakse temast ühtset ja ülemaailmset dokumendiformaadistandardit, mis lahendab probleemid digitaalsignatuuridega, arhiveerimisega, eri platvormide vahelise kommunikatsiooniga jne. Kas ja mil määral seda kõike oodata võib, peakski järgnevast muuhulgas selguma. Samuti analüüsib artikkel mõnesid dokumendihaldusprogrammi ülalviidatud aruannetes esitatud seisukohti, püüab prohvetlikult ennustada, kuhupoole maailm digitaalses dokumendihgalduses järgmise paari aasta jooksul liikuma hakkab, ja mida Eesti selle liikumise taustal ette peaks võtma.

Milleks üldse uus formaat?

Loomulikult pole elektroonilise andmevahetuse probleem uus ja arvutustehnika paarikümneaastase ajaloo jooksul on selleks otstarbeks välja pakutud palju erinevaid formaate, mille põhjal bitte mõistlikuks tekstiks tõlkida. Miks ei võiks mõnda neist digitaalses dokumendihalduses tarvitada? Sellest arusaamiseks vaatleme lühidalt, milliseid tingimusi sobilikule formaadile esitatakse.

- Ta peab olema avaliku ja piisavalt lihtsa kirjeldusega, et garanteerida ühene arusaadavus ka aastakümnete möödudes ning võimaldada mitteeksperdist kohtunikulgi lahendada vaidlusi digitaaldokumendi bitijada interpretatsiooni adekvaatsuse üle. Probleemidest, mis vastasel juhul tekkida võivad, saab lugeda nt artiklist [2].

- Ta peab sisaldama piisavalt võimsaid vahendeid dokumentide mitmekülgseks vormistamiseks (pealkirjad, tabelid, graafika, loendid).

- Oleks väga hea, kui kasutatav formaat võimaldaks dokumendi osi konteksti alusel eristada, nt tuua eraldi välja kirja kirjutaja, saaja, kuupäeva, digitaalallkirja jne.

- Kuna vormistajateks saavad enamasti tavalised kontoriametnikud, kes tahavad asja võimalikult mugavalt kaelast ära saada, peab nende käsutuses olema valitud formaadis suhtlev graafiline (nt WYSIWYG) liides.

Miks selles valguses ükski olemasolevatest dokumendiformaatidest ei kõlba? Järgnev põhjuste loetelu ei pretendeeri küll ammendavusele, kuid toob siiski esile põhiprobleemid, mis peaksid motiveerima uue standardi loomist.

- **7-bitine ASCII** – üks vanim, üldiselt tunnustatuim ja kõige ühesemalt arusaadav formaat. Samas ei võimalda ta eriti mitmekülgset vormistust, sest kuigi ASCII-kunstist on saanud lausa eraldi kunstivool, jääb korralik graafika ikkagi vaid unistuseks.

- ***Bitmap*** – mõnes mõttes kõige universaalsem, kuivõrd *bitmap*-pildina võib esitada mida tahes. Nii näiteks läks Eestiski vahepeal moodi veebilehe vormistamine ühe GIF-pildina, et garanteerida ühesugune kuva kõigis brauseritest. Samas on *bitmap*'ina loodud tekstidokumendi parandamine hiljem liiga keeruline.

- **MS Wordi DOC** – küll väga võimas, mugav ja laialt kasutatav, kuid salajase definitsiooniga ning ebastabiilne, ühe Wordi versiooniga kirjutatud fail ei pruugi teisega loetav olla (ei alt üles ega ülalt alla). Seega tuleks arhiivis peale dokumentide säilitada ka kõiki MS Office'i versioone (koos *service pack*'idega) või faile iga natukese aja tagant ümber konverteerida.

- **MS Wordi RTF** – väidetavalt avaliku kirjeldusega ning versiooniti muutumatu, kuid praktikas pole keegi suutnud tõestada, et MS Word ise avaldatud definitsioonile vastavaid RTFe väljastab ja ka erinevate Wordi versioonide arusaamad samast RTF-failist ei lange alati kokku.

- **LaTeX** – alates 1980.dest stabiilsena püsinud väga võimas keel, mis on suunatud eeskätt küljendustöödele ja teadustekstide kirjutamisele, kuid võimaldab loomulikult ka kõike muud. Samas puudus LaTeXil kuni 1990.te aastate lõpuni mugav graafiline kasutajaliides.

- **PostScript** ja **PDF** – Adobe'i poolt välja töötatud avalikud formaadid, mõlemad orienteeritud lõpptrükikuju esitamisele. PostScript-failide redigeerimiseks mugavat tarkvara praktiliselt ei eksisteeri, PDF-failide tarvis tuleks kasutada Adobe'i Acrobat-perekonna tooteid. Viimased on mõeldud enam professionaalseks küljendustööks ega tasu end lihtsas kontoris ära.

- **HTML** – samuti põhiosas standardne ja stabiilne, kuid ei paku praktiliselt mingeid võimalusi laiendamiseks HTML-is mitte ette nähtud dokumendielementidega.

Küll aga vastab HTML küllalt hästi teistele olulistele nõudmistele: avalikkus, graafiline redigeeritavus ja piisavad vahendid elementaarseks vormistustööks (loendid, tabelid jne). Lisades siia veel mehhanismi, kuidas defineerida juurde rakenduse spetsiifilisi elemente, saaksime enamvähem kõiki vajadusi rahuldava lahenduse.

Esimeses lähenduses võibki XML-i vaadelda lihtsalt kui laiendatavat HTML-i. Järgnevas jaotises uurime XML-i ehitust (ja ehitusvigu) lähemalt.

XML-dokumendi struktuur – süntaks *vs* semantika

Nagu ülalöeldust järgneb, oli XML-i disaini üks põhiprintsiipe luua igale kasutajale võimalus defineerida oma vajadustele vastav dokumendi struktuur. Enamasti kirjeldatakse seda struktuuri ehk tulevase dokumendi süntaksit spetsiaalse DTD-faili (*Document Type Definition*) abil.

Oletame näiteks, et Alice ja Bob ajavad omavahel äri ning tahavad saata üksteisele digitaalseid arveid. Arvel peab kirjas seisma saatja nimi ja töökoht, maksja nimi ja töökoht ning arve suurus dollarites. Siis võiks vastav DTD `cheque.dtd` välja näha järgmine:

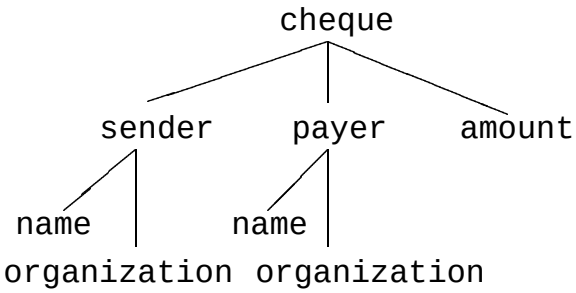
```
<!ELEMENT cheque (sender,payer,amount)>

<!ELEMENT sender (name,organization)>
<!ELEMENT payer (name,organization)>

<!ELEMENT name (#PCDATA)>
<!ELEMENT organization (#PCDATA)>

<!ELEMENT amount (#PCDATA)>
```

Niisiis tohib loodavas dokumendis kasutada kuut elementi, neist kolm sisaldavad omakorda alamelemente, ülejäänud kolm aga suvalist konteksti sobivat (või sobimatut) teksti. Selle DTD põhjal moodustatud arve `cheque.xml` võiks välja näha järgmine:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cheque SYSTEM "cheque.dtd" >
<cheque>
  <sender>
    <name>Alice</name>
    <organization>Alice Inc.</organization>
  </sender>
```

```
  <payer>
    <name>Bob</name>
    <organization>Bob Ltd.</organization>
  </payer>
  <amount>2000</amount>
</cheque>
```

Lugeja peaks pöörama tähelepanu asjaolule, et nii DTD kui vastav dokument järgivad sama struktuuri, mida võib esitada ka puuna:



Kes ütleb, mida selline dokument tähendab? Esimesel pilgul tundub kõik kena, kuivõrd XML-märgendite nimed (nagu `payer` ja `name`) kõnelevad iseenda eest. Kui Alice ka tahaks hiljem minna kohtusse ning väita, et Bob pidas ülaltoodud dokumenti signeerides silmas mitte maksekohustuse võtmist, vaid midagi muud, jääks ta ilmselt kaotajaks. Küllap mõtlesid umbes nii ka XML-formaadi väljatöötajad, kuid kahjuks pole kõik nii ilus.

Nagu juba toodud lihtsast näitest näha võib, on XML-fail keskmisele bürokraadile Bobile käsitsi kirjutamiseks liiga keeruline ning ta nõuab arvatavasti oma töö kergendamiseks graafilist liidest. See aga annab talle peale mugavuse lähtekoodi üldse mitte vaadata ka võimaluse kirjutada alla soovimatule dokumendile.

Oletame näiteks, et naljahambast programmeerija on Alice'i ja Bobi raamatupidamistarkvara pannud suhtlema järgmise DTD (mida suitsukatteks nimetatakse endiselt `cheque.dtd`) põhjal:

```
<!ELEMENT marriageproposal (girltopropose,proposer,
      yearofmarriage)>

<!ELEMENT girltopropose (name,organization)>
<!ELEMENT proposer (name,organization)>

<!ELEMENT name (#PCDATA)>
<!ELEMENT organization (#PCDATA)>
<!ELEMENT yearofmarriage (#PCDATA)>
```

Bobile saabunud dokument näeb siis seestpoolt välja selline:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE marriageproposal SYSTEM "cheque.dtd">
<marriageproposal>
  <girltopropose>
    <name>Alice</name>
    <organization>Alice Inc.</organization>
  </girltopropose>
  <proposer>
    <name>Bob</name>
    <organization>Bob Ltd.</organization>
  </proposer>
  <yearofmarriage>2002</yearofmarriage>
</marriageproposal>
```

Bobi kasutajaliides näitab talle arvet koos kõigi rekvisiitidega ning Bob signeerib selle pahaaimamatult. Kuna väljastpoolt vaadates töötab kõik õigesti (kui Alice tipib summaks $2002, siis näeb Bob seda täpselt sama moodi), ei oska keegi pikka aega pettust kahtlustada. Ühel päeval avaldab programmeerija aga Alice'ile saladuse. Kas nüüd võib Alice kohtus väita, et Bob on oma allkirjaga kinnitanud lubadust ta naiseks võtta?

Toodud näitest võime teha kaks olulist järeldust.

1. Kuigi XML-dokumendi elementide sisu saab kirjeldada märgiste nimede abil, ei ole märgise nimi ja tähendus *á priori* seotud. Selles mõttes sarnanevad XML-märgised muutujanimedele programmeerimises: neid võib nimetada mõistlikult, kuid see pole kohustuslik.

2. Seega tuleb kasutatavate dokumendielementide tähendused fikseerida kusagil mujal, nt seaduses, määruses vms. Lisaks läheb vaja tarkvara, mis

   (a) oskaks defineeritud märgenditega adekvaatselt ümber käia ja

   (b) oleks usaldatav.

Tarkvara usaldamise küsimus on praktilises andmeturbes üks olulisemaid, kuid kahjuks ka väga keeruline ning senini ilma hea lahenduseta. Seepärast läheme järgnevas kergema vastupanu teed ja uurime, millist XML-teadlikku tarkvara hetkel laias ilmas leida võib ning millist vaja läheks.

XML-tarkvarast

Kuidas tarkvara XML-dokumenti näitab?

Paneme tähele, et kubmki senikirjeldatud XML-dokumendi koostisosadest (lähtekood ja DTD) ei ütle poolt sõnagi, kuidas dokumenti kasutajale näidata tuleks – milliseid fonte kasutada, kui palju jätta taandrida, mis värviga esitada viited jne. Selle ülesande täitmiseks saab XML-failile analoogiliselt HTML-iga lisada laadilehti (*stylesheets*).

Laadilehtedest algavad aga probleemid. Nimelt on välja pakutud vähemalt kolm konkureerivat stiilikirjelduskeelt: CSS (*Cascading Style Sheets*), XSL (*eXtensible Stylesheet Language*) ja DSSSL (*Document Style Semantics and Specifications Language*). Neist viimane leiab põhiliselt rakendust üsna spetsiifilistes küljendustöödes, esimesed kaks aga püüavad rahuldada laiatarbevajadusi.

CSS-id töötavad üsna sirgjooneliselt, lubades omistada XML-dokumendis esinevatele (või täpsemalt DTD-s defineeritud) märgenditele visuaalseid atribuute (kuvatava teksti font, värv, joondamine, reavahetuste esitamine jne). XSL seevastu kujutab endast rohkem programmeerimiskeelt, millega määratakse lähtekoodi teisendusi nt HTML-iks, et seejärel dokumenti tavalise brauseriga näidata. Sellisena pakub XSL rohkem võimalusi kui CSS, kuid neis võimalustes peituvad ka ohud. Nii võib XSL-i abil XML-dokumendi teksti muuta ekraanil näitamiseks hoopis teistsuguseks ning Alice ja Bob jäävadki kohtus vaidlema, kumba versiooni Bob õieti signeeris.

Samuti kannatavad CSS ja XSL kõvasti XML-vahendite üldise häda käes: neid pole mitme arendusaasta jooksul suudetud lõplikult standardiseerida. CSS1 [3] lasti välja 1996. ja CSS2 [4] 1998. aastal, 3. versiooni [5] kallal käib praegu arendustöö. XSL v1.0 [6] on W3C arendusmeeskonna poolt tõestetud *Candidate Recommendation* ehk soovituse soovituse staadiumisse. Igahtahes ei saa ükski tarkvaraarendaja hetkel lasta välja standardsele CSS-ile või XSL-ile vastavat redaktorit, kuivõrd standardit lihtsalt ei eksisteeri.

Loomulikult ei oota suured kompaniid W3C järel, kuni too standardite lõppversioonidega maha saab. Olemasolevate draftide annab arendustööd teha küll ja nõnda saavadki nii Internet Explorer kui Netscape XML-ist juba praegu aru. Probleem seisneb selles, et IE ja NS-i tõlgendused ei lange mitte alati kokku – IE on suuna võtnud XSL-i, NS aga CSS-i toetamisele [7].

Kujutame nüüd ette, et Alice tahab Bobile signeerimiseks ette sokutada võltsdokumenti. Alice teab, et Bob vaatab XML-faile Netscape'iga, seega valmistab Alice ette dokumendi, millele lisab nii CSS-i kui XSL-i ja hoolitseb, et XSL-i abil vaadatuna oleks tulemus meelepärane talle, CSS-i abil vaadatuna aga Bobile. Bob signeerib pahaaimamatult kõik (ka XSL-faili!) ja hiljem kohtus võib Alice Internet Exploreriga dokumenti näidates oma õigust nõuda.

XML-teadliku tarkvara hetkeseisust ja tulevikust

Jämedalt öeldes jagunevad kõik hetkel saadaolevad (või vähemalt artikli autorile kätte sattunud) XML-teadlikud tarkvaratükid kaheks.

- Üks osa programme ei tea midagi CSS-idest ega XSL-idest, XML-dokumenti näitadakse puuna (või mõnes ekvivalentses tarkvaratootja poolt leiutatud vaates) vastavalt kasutaja poolt viidatud DTD-le. Selliste programmide hulka kuuluvad näiteks XMLSpy, Morphon, Xeena jt. Nad on väga universaalsed selles mõttes, et aktsepteerivad suvalist DTD-d, kuid samas jäävad oma kasutusmugavuselt WYSIWYG redaktoritele tugevalt alla.

- Leidub ka täis-WYSIWYG editore, mis XML-faile väljastavad, kuid need töötavad vaid ühe või mõne DTD põhjal. Levinud näideteks sobivad kõik HTML-redaktorid (kuivõrd HTML pole muud kui XML-i erijuht ühe konkreetse DTD jaoks). Samuti kasutab XML-i StarOffice'i järglane OpenOffice, mille DTD võib leida allikast [8].

Millist tarkvara oleks vaja? Ideaalne XML-redaktor

- võtaks üheks sisendiks DTD,

- teiseks sisendiks CSS-i või XSL-i ja

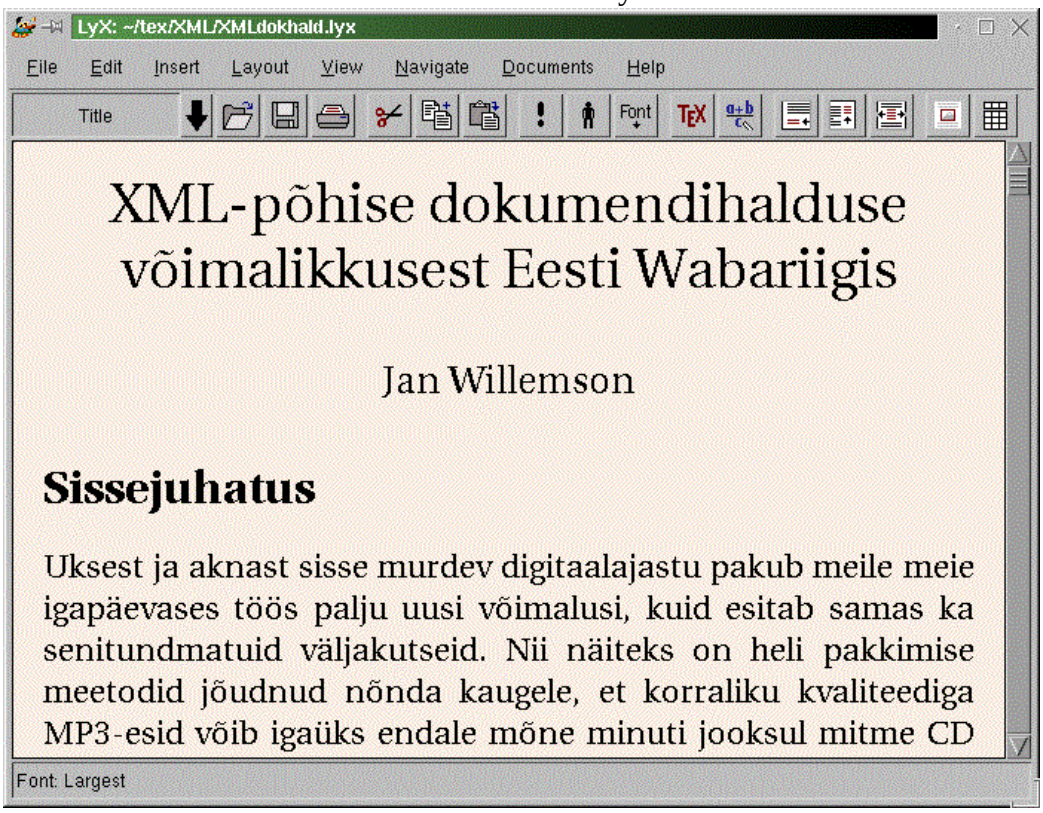
- näitaks neile vastavat Wordi-laadset WYSIWYG-redaktorit.

WYSIWYG-ideoloogiale XML-redaktori loomisel tekib aga mitmeid probleeme. Üheks neist osutub ootamatult mure, mille WYSIWYG oma arust lahendab – kasutajate laiskus. Enamik lihtbürokraate ei viitsi luua pealkirju *heading*-stiiliga või kasutada tabeli tegemiseks vastavat keskkonda. Palju lihtsam on ju tühikutega joonadada ning pealkirjad suurde rasvasesse kirja keerata. XML-dokumendi koostamine aga eeldab, et kasutaja kirjeldab ära struktuuri, eraldades ilmutatult kuupäeva, pealkirja, allakirjutaja jne. Niisiis ei aita korrektsete dokumentide sünnile kaasa see, kui defineerida Wordile kaunid laadilehed, sest teksti saab kujundada ka "jõuga".

Kuidas sundida kasutajat hästistruktireeritud teksti kirjutama? Vastus saab olla ainult üks: tehes halva struktuuri viljelemise raskemaks või koguni võimatuks. Võimalikuks alternatiiviks WYSIWYG-ile pakub autor siinkohal WYSIWYM-ideoloogiat (*What You See Is What You Mean*). Sellise redaktori näiteks sobib LyX, mida võib vabavarana tõmmata veebilehelt [9]. LyXi tööaken on toodud joonisel 1.

Redaktori nupuriba ei paku võimalusi joondamiseks ega fondi suuruse muutmiseks, samuti pole mõtet kümme korda järjest tühikuklahvi toksida, sest üle ühe tühiku ekraanile lihtsalt ei teki. Samuti ei säili akna laiuse muutmisel teksti jaotus

Joonis 1: Tekstiredaktori LyX tööaken



ridadeks. Pealkirja, autori nime, loendeid jms saab kätte ainult siis, kui kasutada vstavat stiili stiilide loetelust. Dokumendi viimiseks trükikujule kasutatakse LaTeXi abi ja tänu sellele on LyX oma minimalistlikule kasutajaliidesele vaatamata väga võimas. Samuti saab iga kasutaja vastavalt oma vajadustele dokumendiklasse juurde defineerida või liidestada LyXi uute failiformaatidega. Hetkel on redaktoriga vaikimisi kaasas moodul, mis võimaldab väljundformaadina kasutada XML-i lähedast sugulast SGML-i (*Standard General Modelling language*), sobiva DTD põhise XML-i filtri lisamine tähendab vaid väga lühiajalist tööd.

Igal juhul soovitab artikli autor kõigil LyXi proovida veendumaks, et WYSIWYG ei pruugigi olla kõige mugavam võimalus. Nagu jooniselt 1 näha, on LyXi kasutatud ka käesoleva kirjatüki valmimisel. Ja – *last but not least* – autori mittematemaatikust abikaasa ei taha pärast selle redaktoriga tutvumist teisi enam nähagi!

Tuleme nüüd tagasi käesoleva alajaotuse alguses sõnastatud ideaalse XML-editori kontseptsiooni juurde ja küsime, kuivõrd tõenäoline on üldse sellise tarkvara ilmumine, mis etteantud suvalise DTD ja CSS-i/XSL-i peale Wordi laadset tekstitoimetit näitab. Kõige tõsisemaks takistuseks kujuneb arvatavasti laadileh-

tede arvestamine dokumendi kuvamisel. Nii CSS-i kui XSL-i struktuur võib olla väga keeruline ja nendega reaalajas arvestamine tähendab sisuliselt igal tekstitoimetussammul kogu dokumendi uuesti ülerehkendamist. See ei tähenda, et vastav tarkvara ei võiks ühel hetkel valmida, kuid kardetavasti ei saa ta olema sugugi lihtne ja järelikult ka mitte odav ega (esialgu eriti) töökindel.

Microsoft on juba 1990.te aastate keskpaigast lubanud, et MSOffice hakkab XML-i toetama. Siiani pole seda juhtunud ei Office 97 ega 2000 puhul. Hiljuti Office XP-ks ümber nimetet 2002 kohta jagatakse jälle ohtralt lubadusi – eks tähenda ju XP-gi *eXPerience / XML Powered*. Küll aga pole autorile kirjutamise hetkel veel teada,

1. kas uus MSO saab hakkama suvalise DTD põhise WYSIWYG tööga või piirdub OpenOffice'i stiilis vaid mõnega DTD-ga ja

2. kas kõigi märgendite vahele kirjutatavate andmete kodeering on avalik.

Pangem tähele, et täiesti avalikule ja standardsele dokumendiformaadile üleminek pole Microsoftile majanduslikult kasulik, sest siis ei peakski keegi enam MSOffice'it ostma paljalt sellepärast, et kõigil teistel see on ja ainus tarkvara, mis MSO formaadis korrektselt kõneleb, on MSO ise.

Kuidas siis lahendada praegu valitsev probleem, et enamus kontoreid kasutab Wordi ja Excelit, kuid võib juhtuda, et XML-põhise riikliku dokumendihaldusprogrammi jaoks nad ei sobi ja võibolla ei saa kunagi sobima?

Ühe lahendusena pakkus professor Tanel Tammet välja nn lihtsustatud XML-i kontseptsiooni [10]. Lihtsustatud XML kujutab endast kirjelduskeelt, milles saaks dokuemnte luua olemasolevate vahenditega (nt Word) ja need pärast lihtsalt (nt Wordi makrokeelse programmi abil) õigesse XML-i teisendada. Nii võiks lihtsustatud XML-is kirjutet tekstilõik näha välja

```
;; See on kommentaar
Linn:: Tallinn
Nimi::
Nimi eesnimi:: Juhan
Nimi perekonnanimi:: Kask
```

mis teisendatakse kujule

```
<linn>Tallinn</linn>
<nimi>
   <eesnimi>Juhan</eesnimi>
   <perekonnanimi>Kask</perekonnanimi>
</nimi>
```

Selle lähenemise juures võib esile tuua mitu puudust. Kõigepealt on lihtsustatud XML-i käsitsi kirjutada ainult karvavõrd kergem kui täismahulist XML-i, ilusa graafilise kautajaliidese mõõtu ei anna ta kindlasti välja. Lisaks võib probleeme tekitada teksti töötlemine Wordi makrode abil. Lubades kontorite Wordides ametlikult makrode kasutamise sisse lülitada, muutuksid kõik Eesti riigiasutused viiruste leviku paradiisiks, kuivõrd hinnanguliselt on umber 80% kõigist hetkel levivatest viirustest MSOffice'i makroviirused. Samuti pole Microsoft suutnud oma makrokeelt stabiilsena hoida, nii muutub .NET-platvormil VisualBasic tunduvalt, seega tähendaks selle platvormi juurutamine kogu riigi dokumendihaldustarkvara ümberkirjutamist.

Tarkvara – järeldused

Mida siis kokkuvõtteks tähendaks XML-põhisele dokumendihaldusele üleminek Eesti riigi jaoks tarkvara mõttes? Kõigepealt, nagu ülalöeldust järeldub, pole hetkel terves maailmas ideaalset tarkvara (veel) olemas. See jätab meile kolm võimalust:

1. kirjutada vastavad programmid ise,
2. oodata, kuni keegi teine need kirjutab või
3. kasutada mõnd ajutist hädalahendust, nt defineerida Wordi jaoks mingid laadilehed ja salvestada dokumendid HTML-is või minna üle lihtsustatud XML-ile.

Esimesel juhul kindlustame me 100% kohalikele vajadustele vastava tulemuse, kuid riskime oma hambad murda arendustööga (graafilise redaktori kirjutamine pole sugugi triviaalne) ning juurutamisega (Juulale justiitsministeeriumist ja Martale majandusministeeriumist on Word ikka armsam).

Teisel juhul tehakse redaktori loomise töö küll meie eest ära, kuid maksma läheb selle ostmine meile (so Eesti asutustele kokku) vähemalt samapalju kui ise kirjutamine. Samuti pole hetkel selge, kas mõni laiatarbetarkvara järgmine versioon (MSOffice, Corel WordPerfect, Sun Star/OpenOffice vms) toetab suvaliste DTD-dega XML-i või mitte.

Kolmanda võimaluse kaalumisel tahaks kõigepealt meelde tuletada rahvatarkust: miski pole püsivam kui ajutine asi. Üleminekulahenduse juurutamine võtab vähemalt samapalju võhma kui lõpplahendusegi puhul, lisaks anname me Wordipõhiselt jätkates Juulale ja Martale endiselt võimaluse sisuliselt struktureerimata dokumente koostada. See tähendab, et kunagi "tõelisele" XML-redaktorile minnes tuleb võõrutusvaevad nagunii üle elada ja nende vaevade edasilükkamine ei anna lõpuks mingit efekti.

Tarkvarajaotuse kokkuvõtteks lubab kirjatüki autor endale subjektiivse järelduse, et kolmest ülalvaadeldud lahendusest on kõige kindlamini sihile viiv (aga ka kõige ketserlikum) esimene. Selle kohaselt peaks Eesti riik kuulutama välja riigihankekonkursi WYSIWYM-redaktorile, mis võimaldaks etteantud DTD-de põhjal XML-dokukmente koostada, lugeda, saata ja välja trükkida. Samuti tuleks palgata (vähemalt alguses) üsna suur koolitajate-käehoidjate armee, kes seletaks Juulale ning Martale, et enam ei saagi saatja aadressi kirjutamiseks tühikuklahviga lehe paremasse serva sõita, vaid selleks tuleb kasutada mingit totakat stiili.

Kas selline lahendus võib läbi minna? Jah, kui Eesti riik suudab laadademokraatiale vahelduseks ka järjekindel olla. Kõlab arvatavasti naiivselt, kuid käesoleva artikli autor usub sellisesse võimalusse.

Kuidas signeerida XML-dokumenti

Klassikalises krüptograafias on digitaalallkirjade andmiseks kujunenud välja teatavad põhimõtted. Nii vaadeldakse signeeritavat dokumenti kui ühte faili või koguni bitijada, mille sisul pole mingit tähtsust. Sellest failist ja allakirjutaja salajasest võtmest lähtudes moodustatakse teine bitijada, mida nimetataksegi signatuuriks. Seejuures sõltub viimane algsest dokumendist nii kõvasti, et dokumendi vähimalgi muutmisel annab allkirja kontrollimise protseduur negatiivse vastuse.

Ühest küljest tundub selline lähenemine ideaalne: pärast signeerimist ei saa dokumenti enam muuta ja nii kaob pabermaailmas tuntud probleem, kus piisava koguse tühja pinna korral võib ka juba allkirjastatud aktile suvalist teksti lisada või laenulepingult markeriga nulle kustutada. Samas vaatavad arhivaarid asjale oma pilguga. Nende jaoks dokumendi elu allkirjastamisega mitte ei lõppe, vaid hoopis algab, ning pärast signeerimist tahaksid nad lisada veel mitmeid elemente (nn manuseid): registreerimisnumbri, kooskõlastue, arhiivi saabumise kuupäeva jne. Seejuures peaks dokument koos allkirja ja manustega moodustama midagi võimalikult tavalise arhiiviühiku sarnast, “ühe tüki” või lihtsamalt öeldes – ühe faili.

Kuidas aga saavutada olukorda, kus mitte kogu fail pole signatuuri all, või veel hullem, signatuur sisaldub allkirjastatud dokumendis endas? Ainuvõimalik lahendus on signeerida dokument vaid osaliselt, näidates seejuures teatud reeglite abil ära, millise osa eest allkirjastaja vastutab ja millised võivad hiljem tema teadmata muutuda.

Niisuguse metoodika realiseerib W3C kandidaatsoovitus [11]. See lubab kasutada mitmeid XML-dokumendi muutmise mehhanisme (nagu XPath ja XSLT) ning jätta nende abil osa dokumenti signatuuri alt välja. Tugevasti lihtsustatult võiks signeeritud XML-dokument välja näha selline:

```
<doc>
<a>Tekst A</a>
<b>Tekst B</b>
<sig1>
   <ärajätureegel1>
   Jätame ära kõik, mis pole Tekst A,
       sh selle signatuuri
   </ärajätureegel1>
   <sig_a>Teksti A signatuur</sig_a>
</sig1>
<sig2>
   <ärajätureegel2>
   Jätame ära kõik, mis pole Tekst B,
       sh selle signatuuri
   </ärajätureegel2>
   <sig_b>Teksti B signatuur</sig_b>
</sig2>
<manused>...</manused>
</doc>
```

Paneme tähele, et tegelikult sisaldab toodud näide endas kaht dokumenti. Jättes välja märgendite `<a>...</a>` ja `<sig1>...</sig1>` vahel asuva osa, saame ikkagi korrektselt singeeritud andmed. Kas see oleks samaväärne mõne nulli kustutamisega laenulepingult? Kui leping on loodud XML-põhisena ning osad nullid signeeritud eraldi, siis küll. Samas ei hakka ilsmelt keegi paari numbrit eraldi allkirjastama, nii et laenulepingu puhul selline rünne ohtlikuks ei osutu, kuid keerulisemate dokumentide korral võib "endale jalga tulistamise" vältimine tunduvalt raskemaks minna.

Tuleme veelkord tagasi käesoleva jaotise algusesse, kus küsisime, kuidas kindlustada allkirjastatavate andmete ja allkirja püsimine "ühes tükis". Kas klassikaline, eraldi loodud signatuur jalutabki lähtedokumendist sõltumatult mööda arvuti kõvaketast ringi? Ei, reeglina kasutatakse nende kooshoidmiseks spetsiaalseid konteinerformaate, mille üheks levinumaks näiteks võib tuua CMS-i (*Cryptographic Message Syntax*, [12]). Operatsioonisüsteemi poolt vaadates on CMS-konteineri näol tegemist ühe failiga, kust vastavat struktuuri tundva programi abil saab kätte kõik vajalikud osised: dokumendi, signatuuri, manused jne.

Ka ülalkirjeldatud signeeritud XML-formaat pole sisuliselt muud kui üks konteiner, milles osiseid eristava srtuktuurina kasutatakse CMS-i asemel XML-i. Õilsa eesmärgi nimel on W3C töörühm küll palju pingutanud, kuid kahjuks tähendavad tulemusse kavandatud võimsamad vahendid ka suuremat eksimiseohtu.

Lisaks võib ülalmainitud kandidaatsoovituses välja tuua veel mitmeid küsi-

tavusi. Ühe probleemina võib signeeritav XML-dokument sisaldada endas viiteid välistele allikatele. Nii näiteks on selge, et koos lähtekoodiga tuleb allkirjastada ka info nende andmete graafilise esituse kohta nt CSS-i või XSL-i kujul, sest vastasel juhul võiks Alice luua Bobi poolt signeeritud dokumendile endale meelepärase visuaalse kuju. Samas moodustavad CSS-id ja XSL-id reeglina omaette faili, mis pole osa baasdokumendist.

Analoogiliselt saab ka praktiliselt kõiki teisi dokumendi moodustamise ja signeerimisega seotud reegleid anda ette viidetena; see käib nii lähteandmete kanoniseerimise algoritmi, sõnumilühendi leidmise algoritmi, signeerimisalgoritmi kui mitmesuguste signeerimiseelsete teisenduste kohta.

Koheselt tekib küsimus: kas kõik viidatud allikad tuleks allkirjastada? Kuna igaühest neist võib sõltuda dokumendi väljanägemine, signeerimisprotsess jne, siis paranoiliselt võttes jah. Samas võivad need allikad sisaldada viiteid edasi, lisaks jäävad nad sageli väljapoole allakirjutaja kontrolli. Millist osa välistest dokumentidest signeerida, jääbki W3C kandidaatsoovituses vähemalt hetkel üsna uduseks.

Kokkuvõtted

Kahtlemata on XML loodud lähtudes soovist maailma aidata ja tema väljatöötajatele ei saa kerglast asjassesuhtumist kindlasti kuidagi ette heita. Samas kasutatakse seda kolmetähelist lühendit maailmas üsna palju ka lihtsalt kuuma lendsõnana – iga tarkvaratootja tunneb enesel lasuvat kohustust lisada oma produkti reklaamlehele fraas “Toetab XML-i”, saamata lõpuni aru, mida XML-i toetamine tegelikult tähendab ja teadmata, kas see 100% üldse võimalik on.

Nii saigi käesolev artikkel kirjutatud teadlikult natuke vaimustust maha jahutavana. Mitmete autori poolt esitatud seisukohtadega võib kindlasti vaielda, kuid ehk leidis hea lugeja enda jaoks mõneski XML-i aspektis uudse vaatenurga.

Saagu siinkohal lõpetuseks ühes kohas üles loetud mõned XML-i põhiprobleemid ja disainivead, mida iga selle keele juurutaja silmas peaks pidama.

- XML on loodud olema võimas, paindlik ja igas suunas lainedatav, kuid samas jätsid tema loojad arvestamata, et keerulisi ning universaalseid struktuure toetava tarkvara kirjutamine võib osutuda väga kalliks.

- XML-i arendus toimub mõnes mõttes liiga demokraatlikult: paljud olulised osad jäävad kauaks beetastaadiumisse ja sama ülesande lahendamiseks võib sageli leida mitmeid konkureerivaid väljatöötlusi. Samuti pole juurutajad kaitstud selle eest, et mingil hetkel tekib W3C egiidi all uus XYZ-i loov töörühm, millega kõik poole mängu pealt arvestama peavad hakkama.

- Kuigi XML-is on loodud võimalus dokumendi elementide sisu kirjeldamiseks elementidele tähenduslike nimede andmise läbi, ei ole märgise nimi sisuga siiski *á priori* seotud. Loodetud selguse toomise asemel võib see asjaolu põhjustada hoopis arusaamatusi.

- XML-i signeerimisel jääb segaseks, millisest vaatest dokumendile tähenduse omistamisel lähtuda tuleb. Samuti on seni lahendamata probleem välisviidetega ja vastamata küsimus, miks ikkagi kasutada konteinerformaadina XML-i, kui standardne CMS pakub hoolikamal vaatlusel praktiliselt samu võimalusi ning hoiab samas tänu selgemale struktuurile paremini ära "endale jalga tulistamist".

Kõike ülalöeldut ei tule võtta lauskriitikana. XML annab võimaluse teha nii mõndagi, ainult me ise ei tohiks soovida võimatut ja peaksime olema valmis selleks, et meie valikud ühel etapil võivad tugevasti kitsendada valikuvabadust järgmisel.

Üks põhiküsimusi, mis riigiasutusi digitaalse dokumendihalduse programmi juures huvitas, kõlab nii: kas XML-ist saab imerohi, mis tagab digitaalse asjaajamise sõltumatuse mõnest konkreetsest kommertstarkvarapaketist? Vastus on filosoofiline: jah ja ei. XML kujutab endast siiski ainult keelt, milles kõnelevad osapooled mõistavad teineteist vaid vastastikuse tahtmise korral. Ka XML-põhist süsteemi saab disainida halvasti, kaua ja kulukalt. Millise tee Eesti riik valib, jääb juba meie riigiisade tarkuse otsustada.

Viited

[1] Eesti Vabariigi dokumendihalduse programmi kodulehekülg, `http://www.riik.ee/dh/`

[2] Dokumendiformaadid ja nende turvaprobleemid, Jan Willemson, *Arvutimaailm*, 9.2000, lk 19-21

[3] Cascading Style Sheets, level 1, W3C Recommendation, 17 December 1996, revised 11 Jan 1999, `http://www.w3.org/TR/REC-CSS1/`

[4] Cascading Style Sheets, level 2 CSS2 Specification, W3C Recommendation 12 May 1998, `http://www.w3.org/TR/REC-CSS2/`

[5] CSS3 module: W3C selectors, W3C Working Draft, 5 October 2000, `http://www.w3.org/TR/css3-selectors/`

[6] Extensible Stylesheet Language (XSL) Version 1.0 W3C Candidate Recommendation 21 November 2000, `http://www.w3.org/TR/xsl/`

[7] W3C Standards Support in IE and the Netscape Gecko Browser Engine, http://home.netscape.com/browsers/future/standards.html

[8] StarOffice XML File Format Working Draft, http://xml.openoffice.org/xml_specification_draft.pdf

[9] Tekstiredaktori LyX koduleheküIg, http://www.lyx.org/

[10] Soovitused riigiasutuste elektrooniliseks asjaajamiseks: XML-põhine dokumendihaldus, Tanel Tammet, 28. juuni 2000, http://www.riik.ee/dh/ylevaade/xmlallkiri.pdf

[11] XML-Signature Syntrax and Processing, W3C Candidate Recommendation, 31.October-2000, http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/

[12] Cryptographic Message Syntax, RFC2630, http://www.ietf.org/rfc/rfc2630.txt