TMT 51. évf. 2004. 10. sz.

Bíró Szabolcs
Neumann János Digitális Könyvtár és Multimédia Központ Kht.

A szövegfeldolgozás modern eszközei – az SGML és XML nyelvek*

A Világmindenségnek van egy pontja, ami nem is pont. Nevezhetjük tájnak, vidéknek is. A kiterjedése nagy. Nagy, de mihez képest? Tehát lehet, hogy kicsi. / Lények lakják. Mindegy, milyenek. Értelmesek vagy olyanok, amelyek magukat annak hiszik? A térben helyüket változtatják, méreteikhez képest hihetetlen távolságokat tudnak legyőzni. Tehetik; nem kell tartaniuk az Időtől. Azt is legyőzték már. Legalábbis meg vannak erről győződve. / A közelükbe férkőzöm és birtokomba veszem információikat."

(Felfedező)
Nemere István: Világok világa (Szépirodalmi Könyvkiadó, 1990)

Az elmúlt néhány évtized – melyet előszeretettel nevezünk az informatika korszakának – egyik legfontosabb kérdése, hogy azt a hatalmas mennyiségű információt, melyet nap mint nap „gyártunk" s tárolunk, vajon hogyan, milyen kezelhető, értelmezhető formában leszünk képesek átadni a jövő nemzedékei számára? A fenti, Nemere Istvántól származó idézet kulcsszavait (idő és információ) kiragadva érezhetjük, hogy kezdeti nagy kérdésünk pontosítást követel: milyen „időtálló" formátumban tároljuk és adjuk át információinkat a jövő felhasználóinak?

Hardver- és szoftverkörnyezet – felhasználók, formátumok

A válasz hosszú ideje érlelődött/érlelődik – meggyőződésem, hogy sok helyütt még most sem találták meg. Ennek oka elsősorban az, hogy folyamatosan küzdenünk kell az állandó változásokkal, a napról napra megújuló hardver- és szoftverkörnyezetekkel. Gondoljunk csak bele, az elmúlt évtizedekben a számítógépes szövegek tárolási és megjelenési módját bizony leginkább a mindenkori műszaki (elsősorban hardver-) lehetőségek határozták meg. Az elmúlt 5–10 évben viszont teljesen átrendeződött a hardverpiac: előtérbe kerültek a hordozható számítógépek (laptop, notebook), s bár még mindig drágák, egyre többen használják őket. Tömegközlekedési eszközökön utazva nap mint nap találkozhatunk kézi számítógépekkel (PDA), és a jövő kommunikációs eszközeinek tekinthető ún. „okos" telefonokkal, melyek magukba integrálják a PDA-k és a mobiltelefonok minden előnyös tulajdonságát. Információink tehát hordozhatóvá váltak, magunkkal vihetjük őket bárhova, dolgozhatunk velük – ezáltal praktikusabban kihasználva az időt.

A másik lényeges tényező – melyet nekünk, könyvtárosoknak soha nem szabad figyelmen kívül hagynunk – a szoftverpiac. Ez a másik olyan terület, amely mára hatalmassá nőtt, s melynek eredményeként az előbb említett eszközökön többféle operációs rendszert futtathatunk (Windows, Linux, Mac OS), kereskedelmi és ingyenes programokat használhatunk, aszerint, hogy ki mit engedhet meg magának, mihez ért, vagy netán mit szeret. Következésképpen rengeteg dologra kell odafigyelnünk, ugyanis készülő vagy már meglévő elektronikus/digitális könyvtáraink nagy becsben tartott használói rendkívül sokfélék lehetnek. Őket elsősorban az érdekli, hogy:

- Megtalálják-e a számukra szükséges információt e-könyvtárunk állományában?
- Olvashatják-e PDA-jukon a Word RTF vagy az Adobe PDF formátumában Japánról letöltött útikönyvet?
- Linux operációs rendszeren, Mozilla 1.7 alatt megfelelően látják-e pl. a HTML-ben letölthető Pukánszky–Német-féle Neveléstörténet c. pedagógia tankönyvet?
- Ékezethelyesen kapják-e meg a szövegeket?

Nem csoda hát, ha mindig ott motoszkál a fejünkben: Mi az univerzális csodaformátum? Miben

* A cikk anyaga részben elhangzott előadás formájában a Helyismereti Könyvtárosok Szervezete 2004. július 14–16. között HELYISMERET – DIGITALIZÁLÁS – EURÓPAI UNIÓ címmel, Szolnokon megrendezett XI. országos tanácskozásán.

dolgozzuk fel a szövegeket úgy, hogy azok az egyes hardver- és szoftverkörnyezeteken egyaránt időtállóak és hordozhatóak maradjanak? Fontos kérdés ez, hiszen a hordozhatóság, az adatkommunikáció kulcsa mindig az alkalmazott fájlformátumokban keresendő, az adatformátumokban jelentkező probléma pedig rendkívül hangsúlyos. Gondoljunk csak arra, hogy olyan környezetben, ahol bárki elérhet bármilyen adatot – web –, jó, ha azokat egy mindenki által használható formátumban ábrázolják. Minimális követelmény tehát, hogy az internetes formátumok megkapják a megfelelő nyilvánosságot – ideális esetben pedig nyílt forráskódúak is lehessenek. Kijelenthetjük, hogy a web két ilyen általánosan használt adatformátuma a HTML (Hypertext Markup Language) és a PDF (Portable Document Format). Az előbbi specifikációja nyílt, bárki szabadon elolvashatja, letöltheti – az ajánlást kidolgozó W3C-n (World Wide Web Konzorcium) kívül nincs olyan szervezet, amely további fejlődését kizárólagosan befolyásolhatná. Ezzel szemben a PDF az Adobe cég tulajdona, s bár a formátum specifikációját a cég közreadta, fejlesztése meglehetősen nagy szaktudást igényel.

Mi, digitális dokumentumokkal dolgozó könyvtárosok alapvetően szövegfeldolgozással, adat- és információtárolással foglalkozunk. A tárolt információkat pedig legtöbbször a már említett interneten tesszük elérhetővé felhasználóink számára. Nincs is ezzel semmi baj, hiszen mind a HTML, mind a PDF adatábrázolási formátumok; azt írják le, hogy az adatoknak miként kell festeniük a képernyőn és nyomtatásban. Tudjuk, hogy a PDF kifejlesztésével az Adobe célja az volt, hogy létrejöjjön egy olyan formanyelv, amellyel úgy lehet dokumentumokat formázni, hogy azok mindenhol ugyanolyan méretarányokkal, betűformákkal és tördeléssel jelenjenek meg. Tehát alapvetően megjelenítési, „nyomtatáskész" formátum. A HTML pedig kifejezetten webes megjelenítésre készült.

Mi az oka mégis, hogy időtálló, platformfüggetlen tárolási formátumokként mégsem ezeket szokták emlegetni? A válasz egyszerű: a szóban forgó formátumok nem választják szét a tartalmat a formától – noha nem említettük, ugyanez természetesen igaz a Word RTF állományaira is. Mindez azzal jár, hogy a megjelenítési formátumban tárolt szövegek esetében a metajelek, vezérlőjelek stb. semmit sem mondanak a szöveg tartalmáról, szemantikájáról. A szöveg értelmezésének ugyanis három szintje, megközelítési módja lehetséges: formai (layout), logikai (szintaktikai) és tartalmi (szemantikai). Ez pedig bőven elegendő indok arra, hogy lehetőleg ne használjuk őket tárolási formátumokként, ugyanis az ilyen fájlok hosszú távon a technika/technológia fejlődésével veszíteni fognak értékükből, nem lesznek kompatibilisek a mindenkori értelmező és feldolgozó programokkal, felhasználási lehetőségeik gyakorlatilag „nullára redukálódhatnak".

Az SGML/XML technológia

Az előzőekben említett problémákra már hosszú évek óta létezik megoldás. Egyre gyakrabban hallhatjuk a három- vagy négybetűs rövidítést: SGML vagy XML. Úgy gondolhatnánk tehát, hogy általánosan ismert, kézenfekvő megoldás van a kezünkben. Ugyanakkor, ha a mindennapi megvalósításra kerül a sor, akkor azzal találjuk szembe magunkat, hogy a kivitelezés korántsem ilyen egyszerű. Miért?

Noha a lassan 20 éve ISO szabványként elfogadott SGML (Standard Generalized Markup Language) – szabványos jelölő nyelv dokumentumok belső szerkezetének leírására, beleértve az egyes elemeket jelölő címkék (tagek) definiálásának módját is – segítségével elvben bármilyen dokumentum leírható, függetlenül az azt tároló és megjelenítő számítógépes környezettől, nagyfokú bonyolultsága miatt azonban nem terjedt el sem Európában, sem világszerte a várt mértékben. A technológia viszont a maga nemében nagyszerű és egyedülálló, kár lett volna kiaknázatlanul hagyni. Éppen ezért a webes technológiák fejlesztésével foglalkozó World Wide Web Konzorcium 1998-ban egy új ajánlást – de facto szabványt – jelentetett meg, az XML-t. *Ez az a formátum, amely meg fogja menteni a digitális világot* – vallják sokan. Valóban igazuk lehet, az ajánlás ötletes és használható, ugyanis:

- Levetette mindazokat a bonyolult és csak nagyon ritkán használt opciókat, amelyek az SGML-ben formálisan megvoltak ugyan, de tényleges implementálásuk csak nagy költségekkel és bonyolult módon lett volna megvalósítható.
- Kompatibilis az SGML-lel, hiszen annak részhalmaza.
- Alkalmazások széles körét támogatja.
- Egyszerűen írhatók hozzá feldolgozó programok – automatikusan feldolgozható.
- Egyszerűen használható az interneten.
- Gyorsan tervezhető, könnyen létrehozható.
- A benne tárolt információ autonóm, nincs bezárva különféle kódkészletek, operációs rendszerek és fájlformátumok börtönébe.

- Az XML-ben tárolt információ többször is újrahasznosítható.
- A tárolt információ kereshető, mégpedig minél többféle módon, intuitívan, kereszthivatkozásokkal, tartalmi és szerkezeti szempontok alapján egyaránt.[1]

Mindez nagyon jól hangzik! Végre egy formátum, amiben feldolgozhatjuk, tárolhatjuk digitalizált könyveinket, szövegeinket. Valóban így van, egyértelműen és kategorikusan kijelenthetjük: Európa és a világ – beleértve a könyvtári világot is – az XML irányába halad.

Az előnyök mellett persze arról sem szabad megfeledkeznünk, hogy ezzel az eszközzel korántsem olyan egyszerű bánni, mint egy szöveg-, vagy akár HTML-szerkesztővel. Jóval több előismeretet igényel a technológia alkalmazása, mint azt eleinte gondolnánk. Az első szokatlan elem mindjárt az, hogy egyszerre több fájllal kell dolgoznunk.

1. Először is szükségünk van egy dokumentumtípust definiáló fájlra, röviden DTD-re (Document Type Definition), vagy készítenünk/szereznünk kell egy ún. Schema[2] állományt, amely a születendő SGML/XML, az utóbbi esetében pedig kizárólag XML fájlunk belső szerkezetét írja le. SGML állományokhoz kötelező DTD-t készíteni/alkalmazni, ugyanakkor XML fájlok esetében a DTD vagy XML Schema használata opcionális. Mindkettő pusztán a validitás (érvényesség) meglétét biztosítja. A DTD-k, illetve Schemák meglehetősen összetett adatszerkezetek, amelyeket formálisan mindig a feldolgozni kívánt szöveg/szövegek alkotóelemeivel és azok szerkezetével definiálunk. Ha egy versre gondolunk, akkor annak definíciója például a következő lehet: szerző, cím, versszak és sor. Vagyis az előbbi elemnevek DTD-ben vagy Schemában való meghatározásával egyúttal megmondjuk azt is, hogy milyen elnevezéseket alkalmazhatunk a versünket tartalmazó SGML/XML állományunk szövegének jelölésekor. Csak így érhetjük el, hogy érvényes dokumentumaink szülessenek. Gondoljunk csak bele, hogy HTML-szerkesztés során is kizárólag akkor kaphatunk valid – az ajánlásoknak teljes mértékben megfelelő – oldalakat, ha a HTML nyelv készletét meghatározó SGML DTD-nek megfeleltetve írjuk forráskódunkat.
   Egy olyan DTD-t, vagy akár Schemát, amely(ek) segítségével mondjuk szépirodalmi műveket vagy szakkönyveket dolgozhatunk fel, természetesen nem kell nekünk „lekódolni" – kezdőként valószínűleg nem is sikerülne. Ilyeneket célszerűbb kész formában, nemzetközileg elismert intézmények/projektek oldalairól letölteni (pl. TEI – *www.tei-c.org*, DocBook – *www.docbook.org*).
   A TEI (Text Encoding Initiative = szövegkódolási kezdeményezés) fontos nemzetközi projekt, amelyet 1987-ben három számítógépes nyelvészeti és irodalmi kutatásokkal foglalkozó angolszász tudományos társaság indított el. A projekt feladata irányvonalak kifejlesztése, terjesztése volt a géppel olvasható szövegek – elsősorban próza, vers, dráma, lejegyzett beszéd stb. – kódolására, közvetíthetőségére és cserélhetőségére, valamint javaslatok tétele új szövegek kódolására. A TEI jelenleg konzorciális keretek között működik, eddig négy kiadása jelent meg – P1–P4 –, a TEI P5 év végére várható. A kiadvány legutóbbi verziója már tartalmazza az XML támogatást is, tehát a DTD-nek az SGML mellett egyaránt létezik XML és XML Schema változata.
   A DocBook műszaki dokumentumok, különösen a számítástechnikai terület dokumentációjának létrehozásához szükséges struktúrákat és elemkészletet definiálja. A DTD sokévi munka eredményeként jutott el arra a pontra, ahol már szinte minden olyan elem leírására alkalmassá vált, amely a számítástechnikai dokumentációval kapcsolatban egyáltalán felmerülhet. A DocBook kiválóan alkalmas kézikönyvek, fejlesztési dokumentumok, felhasználói kézikönyvek, tankönyvek stb. jelölésére is.

2. Miután meghatároztuk vagy kiválasztottuk születendő SGML/XML dokumentumunk belső szerkezetét – az ott használatos elemneveket, attribútumokat, entitásokat, vagy ha úgy tetszik egyedeket –, végre elkészíthetjük, lekódolhatjuk első SGML/XML állományunkat. Nem szabad azonban elfeledkeznünk a megjelenítésről sem! Tisztában kell lennünk azzal, hogy SGML dokumentumokat nem tudunk vizuálisan megjeleníteni semmiféle alkalmazásban. Természetesen szerkeszthetők akár a Windows Jegyzettömb programjával is, ám a felhasználók számára „emészthető" formában egyetlen program sem képes megjeleníteni őket. XML fájlok esetében már kicsit más a helyzet, ugyanis ezek fastruktúráját képesek a böngészőprogramok ábrázolni, persze ez korántsem használóbarát. Gondoljunk csak bele: melyik „elvetemült" olvasó szeretné tagelt elemnevek között olvasgatni Petőfi verseit? Valószínűleg egyik sem! Éppen emiatt kell ismét hangsúlyt fektetnünk a megjelenítésre, és visszaeveznünk az adattárolás

„tengeréről" az adatábrázolására. Mondhatnánk azt is, itt köszön vissza a jól bevált HTML szemlélet. Mit jelent ez?

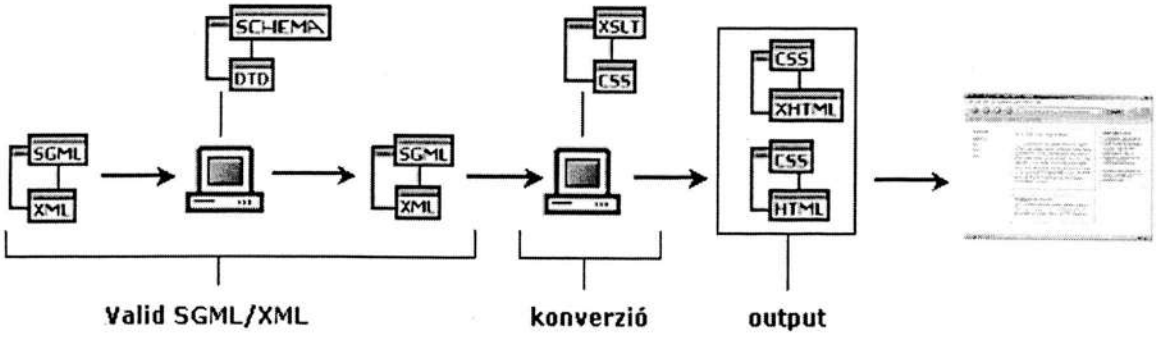
3. Azt, hogy szükségünk van stíluslapokra is. Úgynevezett bővíthető stíluslap(leíró) transzformációs nyelv (eXtensible Stylesheet Language Transformations = XSLT[3]) segítségével „intelligens" stíluslapot készíthetünk, amely felismeri a szöveget tároló központi SGML/XML állományunk belső szerkezeti elemeit, és egy konverziót követően elkészíti azt az oldalt, amely már olvasóink számára is tetszetős és használható. Ilyen stíluslapból több is készülhet, attól függően, hogy milyen formátumot – XML, HTML, XHTML, PDF, Text – szeretnénk végeredményként/kimenetként látni. Az XSLT valójában noha stíluslap, sokkal inkább tekinthető egyfajta sajátos programozási módszernek, nyelvnek, amely beépített vezérlési struktúrákat, nyelvi elemeket, paramétereket, változókat tartalmaz, csak éppen az XML szabályainak megfelelően, egy XML dokumentumként leírva. A félreértések elkerülése végett jegyzem meg: habár az XSLT egy XML-hez kifejlesztett technológia, használható az SGML esetében is. Ennek oka, hogy az XML is az SGML egyszerűsített változata.
   Az XML-ben tárolt szövegeknek létezik egy másik megjelenítési módja is – ez SGML esetében nem használható, kizárólag XSLT-vel összekapcsolva. Mivel tudjuk, hogy az XML állományok tartalommal ellátott belső fastruktúráját a böngészőprogramok képesek megjeleníteni – hiszen beépített XML értelmezőt tartalmaznak –, ezért azokhoz a HTML-ből jól ismert lépcsős stíluslapokat (Cascading Style Sheets – CSS[4]) hozzárendelve, már formázottan láttathatjuk őket. Az ilyen megjelenítésnek azonban több hátránya is van:
   - Egyrészt a böngészőprogram az összes XML elemet olyan sorrendben jeleníti meg a képernyőn, amilyen egymásutániságban azok a forrásdokumentumban fellelhetők. Ez az adottság pedig csak akkor kedvező számunkra, ha ugyanúgy szeretnénk mindent megmutatni, ahogy tároltunk. A gyakorlatban azonban az esetek többségében ez nem így van, hiszen számos esetben előfordulhat, hogy egy adott szöveg bizonyos részét ugyan jelöltük, ám nem akarjuk, hogy az megjelenjen.
   - A másik probléma, hogy az XML és a CSS technológia együttműködése még korántsem mondható vitathatatlanul gyümölcsözőnek, ugyanis a webböngésző alkalmazások nem mindig azt produkálják, amit mi szeretnénk, vagy amire utasítást adtunk – bár ez szerencsére már csak bonyolultabb formázási utasítások esetében tapasztalható.

Következtetésként levonhatjuk, hogy minden olyan esetben, amikor a megjelenítés előtt az SGML/XML állományok valamely elemeit módosítani kell, vagy kódbeli előfordulásukhoz képest át kell helyezni, az XSLT-t kell segítségül hívnunk. Ez a nyelv képes ugyanis értelmezni az SGML/XML szöveg belső logikáját – akár a DTD alapján, vagy anélkül –, s így nyílik lehetőség bizonyos elemek kiválasztására. Például a teljes bibliográfiai leírásból kiemelhetjük csak a címet és a szerzőt, egy TEI fejlécből Dublin Core metaadatcsere-formátumot hozhatunk létre. Lehetőségünk van továbbá SGML/XML állományaink tartalmát tetszés szerinti adatelem alapján sorba rendezni, megjeleníteni, és nem utolsósorban adott elem megjelenítését a tartalomhoz kötni.

4. Végezetül ahhoz, hogy hozzájussunk hőn áhított és szolgáltatható output állományunkhoz, be kell szereznünk egy olyan XSLT feldolgozóprogramot – Xerces, XSLTproc, James Clark's XP, Saxon, Xalan stb. – is, melynek segítségével elvégezhetjük a szükséges fájlkonverziót. Az ilyen alkalmazások általában a bemenetet képező állománynevek megadásával, illetve a kimenet meghatározásával működnek. A Java futtatókörnyezetet igénylő XML-ből való konverzióra alkalmas Xalan esetében ez például a következőképpen nézhet ki:

```
java -cp .\bin\xalan.jar;.\bin\xml-apis.jar;.\bin\
xercesImpl.jar org.apache.xalan.xslt.Process –IN vers.xml
-XSL vers.xsl -XML -OUT vers.html
```

Az előbbi parancs segítségével tehát a bemeneti vers.xml állományukból, a vers.xsl fájlban magadott formázási és konverziós utasítások segítségével kimenetként megkaphatjuk versünket HTML vagy XHTML[5] formában.
Csak érdekességképpen jegyezzük meg, hogy XML fájlokhoz – a CSS-hez hasonlóan – közvetlenül is csatolhatunk transzformációra képes XSL-t, ilyenkor azonban nem jön létre kimeneti állomány, pusztán a böngésző „memóriájában" képződik a látható végeredmény. A megoldás előnye, hogy pl. a kimeneti HTML dokumentumok tárolása megszűnik, így nem igényel többlet tárolókapacitást a weben való szolgáltatás biztosítása sem.



1. ábra **Egy lehetséges SGML/XML feldolgozási folyamat, XSLT+CSS segítségével**

Az *1. ábra* egy XSLT-vel végzett feldolgozási folyamatot mutat be.

Minden XSLT feldolgozószoftver egy XML elemzőt is tartalmaz, amely a már említett két bemeneti dokumentumot – SGML/XML + XSL – egy belső ábrázolási formátumba alakítja át. Az elemző DOM[6] (Document Object Modell) és SAX[7] (Simple API for XML) technológiával egyaránt működhet, az említett ábrázolási formátum pedig lehet fastruktúra, lista vagy egyéb adatszerkezet.
A DOM működésének lényege, hogy felépíti az XML adatfolyamnak megfelelő, objektumokból álló fát, amelynek az elemeit ezután közvetlenül el lehet érni – az objektumok metódusai és adattagjai használhatóak. A DOM fogalma ismerős lehet azoknak, akik a böngészőkben már írtak olyan JavaScript vagy Java programot, amik elérik a Window, Document, Link stb. objektumokat, hiszen az is egy DOM hierarchia.
A SAX nem épít fel DOM szerkezetet, ugyanis itt az elemzés során mindig csak az aktuális elemet látjuk. A SAX emiatt nem teszi lehetővé az XML elemek közvetlen elérését, azokat csak sorosan olvassa fel, aminek következtében események generálódnak. Az elemző alkalmazás ezekre az eseményekre határozza meg az eseménykezelő metódusokat.
A DOM és a SAX területén egyaránt nagy lehetőségeket kínál a Java nyelv, ugyanis ki-ki megírhatja saját XML értelmezőjét, ami persze nem egyszerű feladat, ezért leggyakrabban egy már meglévő, megírt XML értelmezőt használunk saját céljainkra – leggyakrabban a fentebb felsoroltakat használjuk. Egyébiránt az IBM, Oracle, Sun és az Apache rendelkeznek Javában megvalósított implementációkkal. További alkalmazásokat az „XML tools by category = XML eszközök kategóriák szerint" oldalon találhatunk.[8]

A technológia alkalmazásának előnyei

A fenti feldolgozási folyamat áttekintése után bárkiben felmerülhet a kérdés: Miért is jó mindez nekünk? Bonyolult, nagy körültekintést és szakértelmet igényel, és még sorolhatnánk. Nos, képzeljük el, hogy évtizedek múlva már nem használunk Microsoft termékeket, a DOC, az RTF csak emlékeink között élnek, ráadásul a PDF sem használatos már a mindennapi gyakorlatban. Ha szerencsénk van, akkor az idők folyamán egyvalami nem változott: továbbra is szöveges információkkal dolgozunk.

Ha így van, akkor csak elő kell vennünk azt a három állományt – SGML/XML, DTD, XSLT –, amelyek a legegyszerűbb szövegszerkesztővel is szerkeszthetőek. A stíluslapra nincs is igazán szükségünk, hiszen nem tudhatjuk, milyen megjelenítési formátumok lesznek a jövőben. Tudjuk viszont a következőt: készítenünk kell egy konverziós alkalmazást, amely a DTD „segítségével" kiolvassa az adatokat SGML/XML fájljainkból, majd a kimeneti oldalon az adott kor technológiájának, szellemének megfelelő formátumra hozza. Ha így teszünk, bátran kijelenthetjük: sikeresen átörökítettük az információkat a jövő nemzedék számára. A „kekecek" persze mondhatják, hogy konverziót más formátumokkal is végezhetünk, s ebben részint igazuk is van. Ám egy hiányosan definiált, a nyilvánosság számára nem kellően dokumentált, zárt, emberi szem számára szinte értelmezhetetlen fájlból sokkal nehezebb kinyerni az adatokat, információkat. Ne felejtsünk ilyesmikre mindig gondolni már a könyv elektronikus verziójának megszületésekor.

Magyarul, ha írásos anyagot digitalizálunk, dolgozunk fel, akkor előbb mindig el kell döntenünk, hogy azt milyen formátumban tesszük.

Tanácsok a kezdeti lépésekhez

Digitalizálásról itt és most nem szólok, hiszen az külön megérne egy „misét", viszont induló tanácsként érdemes megfogadni, hogy a „nyers" digitalizált, korrektúrázott szöveget először lehetőleg formázási utasítások nélkül, kötetlen formában tároljuk. Ezzel sok időt és energiát megspórolhatunk magunknak. Nagy – több tízezer rekordot tartalmazó – gyűjtemények kialakítása esetén a nemzetközi egységesség biztosítása végett, lehetőleg angol nyelvű, előre kidolgozott dokumentumtípus-definíciókat vagy sémákat alkalmazzunk. A DTD-k, illetve XML Schemák elemneveit ne feleltessük meg a magyar nyelvnek.

SGML/XML állományok készítésekor nagyon ügyeljünk az ékezetek és a „különleges" karakterek helyes kódolására, s már a nyers szövegnél törekedjünk a helyes kódkiosztás meghatározására. Tudjuk, ez nem könnyű, de a különböző kód- és entitástáblázatok mellett már egyre fényesebben tündököl a Unicode[9] csillaga, a megoldás irányát mutatva nekünk, szövegfeldolgozással foglalkozó szakembereknek.

El kell döntenünk azt is, hogy a szoftverpiacon pénzért kapható kódoló rendszerekkel dolgozunke, vagy a már jól bevált, fillérekért, netán ingyen beszerezhető szerkesztőinket használjuk. Noha az előbbiek használata kényelmes (WordPerfect, FrameMaker+SGML, Epic Editor stb.), azonban áruk az egekig szökhet. Az utóbbiak esetében (TextPad, Emacs, jEdit, EditPlus stb.) lassúbb a feldolgozás sebessége, de az anyagi erőforrásokkal jócskán spórolhatunk – nem beszélve arról, hogy forráskódjaink nagy „tudóivá" válhatunk általuk.

Nem utolsósorban pedig szem előtt kell tartanunk, hogy a feldolgozott dokumentumok számbavétele, szolgáltatása és archiválása szempontjából kiemelkedően fontosak a formai és tartalmi jellemzőket leíró metaadatok. A szakértők véleménye szerint ezen a téren semmilyen engedménynek, egyéni megoldásnak helye nincs, kizárólag a nemzetközileg ismert és használt metaadatszabványokat lehet használni. Javasolják – Európában és világszerte is ez a tendencia erősödik –, hogy a közös leíró szabvány a Dublin Core legyen. Ezt támogatja többek közt az OAI nyílt forráskódú metaadat-begyűjtő protokollja, illetve az a ZING névre hallgató protokollcsalád is, amely a Z39.50-et hivatott felfrissíteni.

Az említett tanácsok, lépések betartásával már nyugodtak lehetünk, hiszen megtettük a legfontosabb lépéseket egy nemzetközileg is egységes, szabványos, igényes formában szolgáltatható állomány kialakítása felé.

Összegezés

Ha valaki egy bizonyos technológiáról ír, az azt feltételezi, hogy az illető hisz és – ha nem is vakon, de – megbízik a kérdéses technológiában. Nap mint nap úgy dolgozni, hogy közben az ember hasznavehetetlennek tart egy módszert, vagy nem jósol neki nagy jövőt, enyhén szólva bizarr hozzáállás lenne. Itt és most megnyugtatjuk az olvasókat, e tanulmány szerzője teljes mértékben megbízik a tárgyalt formátumokban, szövegfeldolgozási eljárásokban. (Mindazok ellenére, hogy a TEI vagy a DocBook még nem terjedt el kellőképpen, és az SGML-ből kialakult XML is csak a köztudatba való beépülés fázisában van.)

Rövid „bepillantásunk" alatt észrevehettük, hogy rengeteg területen kell otthonosan mozognunk, ha komolyan akarjuk végezni munkánkat. Terveznünk, kódolnunk, programoznunk kell, és emellett számos szoftver használatát vagyunk kénytelenek elsajátítani. Feladatunk nem egyszerű, de kihívást jelent, ugyanis digitális univerzumunk szélén járunk csupán, ahol még mindig a piaci erők dominálnak. Reméljük, hogy eljön majd az idő, amikor a verseny és a céges érdekek nem széthúzást jelentenek, hanem a közös cél felé irányuló együttes törekvést. Bízzunk benne, hogy előbb-utóbb nem lesznek egymással inkompatibilis szabványok, egyformán működnek majd a böngészőprogramok, és mindenki – minden elektronikus könyvtáros – el tud igazodni a meglévő formátumok között. De addig is létezik megoldás? Teljesen biztosak nem lehetünk benne, de ahogy volt tanárom és kollégám, *Kora András* mondaná: „A World Wide Web Konzorcium ajánlásai talán egy lépéssel közelebb visznek minket egy harmonikusabb digitális világ felé."

Irodalom

BATES, Chris: XML: elmélet és gyakorlat. Bp., Panem Kiadó, 2004. 542 p. ISBN 963 545 393 0

DOMOKOS László: Az SGML és az információs forradalom: *http://www.mek.iif.hu/porta/szint/tarsad/konyvtar/ekonyvt/sgmlinfo/sgmlinfo.htm*

DRÓTOS László: Egy szegény elektronikus könyvtáros panaszai: a digitális szövegformátumok problémái. (Elhangzott az MKE Elektronikus Könyvtár Szekciójának az OSZK-ban 2001. március 1-jén tartott rendezvényén, melynek témája a szövegdigitalizálás volt.) *http://www.oszk.hu/kiadvany/kf/2001/1/drotos.html*

GOLDEN Dániel–TÓTH Tünde–TURI László: Virtuális örökkévalóság: objektumok a digitális könyvtárban. = TMT, 45. köt. 8–9. sz. 1998. p. 299–314.

KORA András: Szemben a változással: időtálló fájlformátumok, szövegfeldolgozás, digitalizálás. = Negyvenéves a szombathelyi könyvtárosképzés. Szombathely, BDF, 2002. p. 116–121. ISBN 963-9290-76-9

World Wide Web Consortium: *http://www.w3.org/*

Jegyzetek

[1] BÍRÓ Szabolcs–KORA András: Kulcs az SGML-hez. Bp., Neumann Kht., 2004. p. 137–139. (akkreditált oktatási tananyag)

[2] Az XML Schema egy XML alkalmazási nyelv, amelynek a célja megegyezik a DTD céljával, tehát típusdefiníciók leírására szolgál. Kifejezetten az XML-hez készült, a DTD-nél egyszerűbb, ráadásul kínálja azt az előnyt, hogy XML szintaktikájú, tehát használatához (a DTD-vel ellentétben) nincs szükség új szintaxis megtanulására.

[3] Az XSLT szintén W3C ajánlás, első verzióját 1999. november 16-án publikálták. Jelenleg az XSLT 2.0 áll kidolgozás alatt, legújabb munkapéldánya 2003. november 12. óta elérhető: *http://www.w3.org/TR/xslt20/*.

[4] A CSS olyan stíluslap-megvalósítás, amely lehetővé teszi, hogy a HTML, XHTML, XML állományok tartalmához egyedi stílust rendelhessünk hozzá. A Cascading Style Sheets W3C ajánlás, melynek jelenleg két érvényben lévő kiadása létezik. Az első kiadás – CSS 1.0 – 1996. december 17-én, annak felülvizsgált verziója pedig 1999. január 11-én jelent meg. A CSS 2.0 1998. május 12-én lett ajánlás, jelenleg ennek felülvizsgált, 2.1-es verziója munkapéldány. A CSS használata egyre népszerűbb a webfejlesztők körében, a technológiát tv- és mobileszközök használatához is folyamatosan fejlesztik.

[5] Az XHTML a kilencvenes évek népszerű, SGML alapú nyelvének, a HTML-nek XML-es alapokra helyezésével jött létre. A nyelv a jelenlegi és jövőbeni dokumentumtípusok és modulok családja, amelyek reprodukálják, részét képezik, és kiterjesztik a HTML 4.01 ajánlást. Az XHTML család dokumentumtípusai XML alapúak, és tulajdonképpen arra (is) lettek tervezve, hogy együttműködjenek az XML alapú felhasználói alkalmazásokkal. A nyelvnek ez idáig két kiadása jelent meg a W3C égisze alatt ajánlás formájában: az XHTML 1.0 (2000. január 26.) és az XHTML 1.1 (2002. augusztus 1.).

[6] A DOM (dokumentumleíró objektummodell) a HTML és XML dokumentumok szabványos ábrázolási módja. Valójában a dokumentumok logikai szerkezetét adja meg, és azt a módot, ahogy a dokumentumhoz hozzá lehet férni és manipulálni lehet azt. 1997 óta W3C ajánlás.

[7] A SAX (egyszerű XML programozási interfész). Nem hivatalos W3C ajánlás, ám de facto szabványnak is tekinthető, hiszen számos XML termék támogatja.

[8] XML tools by category: *http://www.garshol.priv.no/download/xmltools/cat_ix.html*, a letöltés időpontja 2004-08-31.

[9] A Unicode karakterkódolási szabvány. Óriási előnye a többi létező kódtáblával szemben, hogy a világ csaknem összes írással rendelkező nyelvének karaktereit tartalmazza. A Unicode karaktereknek többféle megjelenítési formátumuk van, ilyenek az UTF-8, az UTF-16 és az UTF-32. A legtöbb Windows illesztőfelület például az UTF-16 formátumot használja.

Beérkezett: 2004. VII. 26-án.

A SwetsWise online tartalomszolgálat bővítése újabb kiadókkal

A Swets Information Services bejelentette, hogy hét új kiadóval írt alá együttműködést a *SwetsWise Online Content* tartalomszolgáltatás bővítésére. A SwetsWise egy webalapú, moduláris elemekből épülő szolgáltatás az előfizetések szervezésére és kezelésére, a hozzáférés biztosítására, valamint az online információ területén. A cég 309 kiadótól 8325 teljes szövegű elektronikus folyóiratot forgalmaz, ezek 90%-ban a műszaki, természet- és orvostudományok vezető kiadói. Az újonnan szerveződött kiadók között szerepel a magyar *Akadémiai Kiadó* is.

/Swets-sajtóközlemény, 2004. július 19.
http://www.swets.com/

(J. É.)