KRITIKAI KIADÁS ÉS XML

KIRÁLY PÉTER
Arcanum Kiadó

Kulcsár Péternek
és Zsófiának

„Hát mikor lesz megint alkalmam azt mondhatni:
Nagybecsű ünnep: gyülekezet, magnificenciák,
excellenciás hölgyek és urak!

Bevezető

Az elektronikus kiadás vagy kiadás-előkészítés gyakorlatában számtalanszor beleütközünk abba a problémába, hogy az általunk választott keretrendszer (általában szövegszerkesztő) korlátaihoz kell igazítanunk szövegünket. Vannak gyakorlati akadályok, – ilyenek a karakterproblémák, a különféle jegyzetek és a konverzió(k) kérdése –, és elméletiek: létezhet-e eszközfüggetlen, „tökéletes" szöveg, aminek alapján az összes – legalábbis a mindennapi gyakorlat során felmerülő – kívánságot ki lehet elégíteni. E kiadvány olvasóinak nem kell bizonygatni, hogy tökéletes szöveg nincs,[1] legfeljebb tökéletesebb.

Az ezzel összefüggő (számítógépes) megoldási javaslatok két, radikálisan eltérő irányba tettek lépéseket. Az első azt vallotta, hogy olyan eszközt kell alkotni, ami képes a legextrémebb tipográfiai megoldásokat megvalósítani. Ennek egyik első – és napjainkig élő – terméke a TeX, melyet Thomas Knuth matematikus, a számítástechnika egyik legnagyobb alakja[2] készített. Ez az irányzat elsősorban a szövegben fellelhető formai elemek leírására helyezi a hangsúlyt. A másik módszer a szöveg tartalmi szerkezetét igyekszik megragadni – függetlenül attól, hogy az adott tartalmi

[1] A másodlagos szóbeliséget kutató filozófiai iskola még ennél is tovább megy, amikor határozott meggyőződése, hogy a választott szövegrögzítési (sőt, kifejezési) mód nem csak technológiailag torzítja az alapanyagot, hanem magára a rögzítendő gondolatra is visszahat, azt erősen befolyásolja.

[2] Magyarul is olvasható befejezetelen alapműve *A számítógép-programozás művészete* I-III., Műszaki, Budapest 1994.
(Esterházy Péter: „Fejezet", *Élet és Irodalom*, 2004. okt. 15.)

elemnek milyen formát adunk, vagy hogy egyáltalán tipografizálni lehet-e az adott rendszeren belül. Itt konkrét termék helyett egy logikai-formális szabályrendszer született, az SGML (Structured Generalized Markup Language – kb. szerkesztett általános kódnyelv), majd ennek az egyszerűsítése révén az 1990-es évek közepe táján az XML (eXtendible Markup Language – bővíthető kódnyelv). Az SGML/XML egy 'metanyelv'. Nem azt szabályozza, hogy a *szerző* mezőt hogyan rögzítsük, milyen kiemeléseket használjunk, hanem azt, hogy hogyan hozhatunk létre egy formális nyelvet, s mik azok az alapszabályok, melyeket minden e szerint rögzített szövegnek be kell tartania. Az XML szabályrendszerét betartva viszont könnyen lehet írni olyan DTD-t, vagyis dokumentum-típus-meghatározást, amiben felsoroljuk azon szabályok összességét, amelyeket be kell tartaniuk az általunk előállított szövegeknek. A DTD-vel a szemantikai elemek tetszőleges bonyolultságú hierarchiáját határozunk meg, pl. a *könyv címlap*ból és *oldalak*ból áll, a címlapon adjuk meg a mű *szerző*jét és *cím*ét, az oldalak *bekezdés*ekből, a bekezdések *sor*okból állnak. Vagy, hogy egy kevésbé a kinézetet tükröző példát adjak: a szövegben lehetnek *évszámok*, *földrajzi nevek*, személynevek, *igék*, *főnevek*, *képaláírások*. Az XML szabvány[3], egyszerűsége folytán, épp az utóbbi években lett egyre népszerűbb, és számos nagy szoftvergyártó kezdte valamilyen formában használatba venni. Az XML egyik nagy erőssége, hogy nem kötődik konkrét céghez és teljes mértékben nyílt, átlátható – szemben azokkal a fájlformátumokkal, amelyeket csak azokkal az eszközökkel lehet kezelni (írni, olvasni, feldolgozni) amelyekkel létrehozták. Fontos, hogy az XML 'szöveges' (nem bináris) tárolási módot választott, ami szintén a nyíltságot és az állomány hosszú távú megőrzését erősíti – továbbá a fájlt értelmező program nélkül is bármely közönséges szövegszerkesztővel vagy -nézővel el lehet olvasni. Szempontunkból azonban fontosabb, hogy különféle tudományos társaságok is meglátták az XML-ben rejlő lehetőségeket, és egyre-másra alakítottak ki XML alapú kódnyelveket. A történész-irodalmár szempontjából különösen figyelemre méltó az EAD (Encoded Archival Description – kódolt levéltári leírás) és a

[3] Bár az XML-t szabványnak szokták mondani, valójában nem az ISO által elimert tényleges szabványról van szó, hanem az internetnek önszabályozó módon irányt adó, 1994-ben alapított, több mint 120 szervezetet tömörítő World Wide Web Konzorcium (W3C) dokumentumáról, amit azonban a világ valamennyi programozója igyekszik betartani. A később ismertetett DOM, SAX, XSL, CSS, HTML ugyancsak a W3C szabványai (valamennyi megtalálható a http://www.w3c.org címen).

még *béta* állapotú EAC (Encoded Archival Context – kódolt levéltári összefüggésrendszer), melyet levéltárak és kézirattárak használnak (pl. a jelentős európai kézirattárak közös katalógusát létrehozni tervező Malvine projekt), valamint a TEI (Text Encoding Initiative – szövegkódolási kezdeményezés) kódnyelvi, melyet elsősorban irodalmi és történeti források rögzítésére használnak.[4]

A TEI

A TEI-t létrehozó konzorciumban számos tudományos társaság és tanszék vállal szerepet. Évente konferenciákat tartanak, az egyes részterületeket (így például a kéziratok kritikai kiadását vagy a karakterkódolást) munkabizottságok vizsgálják. A munka komolyságát jelzi, hogy a fejlesztés immár az 5. számú verziónál tart (a fejlesztést 1988 júniusában kezdték, s kétévi munkába került az 1.0-ás változat elkészítése).

A „termék" DTD, vagyis a TEI egy dokumentumtípus-deklaráció, amely a nyelv jelölőelemeit, valamint egymáshoz való viszonyukat határozza meg. Ám mivel a TEI XML alapú, az alapkészletet felül lehet írni, illetve további elemekkel lehet kiegészíteni – amit a TEI-t használó több mint száz projektumnak a többsége kisebb-nagyobb mértékben ténylegesen is kiaknáz.

A TEI moduláris felépítésű, ami azzal az előnnyel jár, hogy a kódolás és a további feldolgozás során nem kell minden szabályra ügyelni, csak azokra, melyek az adott dokumentumtípusra (pl. vers, próza, dráma, szótár), vagy a szöveg kiegészítő elemeire (kritikai apparátus, ugrópontok) vonatkoznak.

> Fontos tudni, hogy az XML állomány önmagában nem alkalmas arra, hogy „emberi fogyasztásra" kerüljön, hiszen nem tartalmaz semmiféle direkt formázást. A dokumentum formázását programmal lehet megoldani. Ehhez rendelkezésre állnak az általánosan használt programnyelvek (C és változatai, Java, Perl, Php stb.) speciális moduljai, melyek az XML feldolgozására használatos függvényeket és metódusokat tárolják. Szerencsére a feldolgozás módszerének algoritmusait is szabványok rögzítik: a DOM (Document Object

[4] EAD: http://lcweb.loc.gov/ead/, EAC: http:/www.library.yale.edu/eac/, TEI: http://www.tei-c.org. Érdekesség, hogy létezik egy 'öszvér' DTD is, a kodikológiai leírásban alkalmazott *Master*, mely a 'nagy szerkezetet' az EAD, a finomabb részleteket a TEI segítségével rögzíti, aminek az eredményességét jelzi, hogy ennek továbfejlesztett változata a TEI tervezett, újabb kiadása részét képezi.

Model – dokumetum-objektum model) amely a dokumentumot mint egy fa-szerkezetű objektumot ábrázol és jár be, valamint a SAX (Simple API for XML – egyszerű programozói felület az XML-hez), amely a dokumentumon sorosan végighaladva eseményvezérelt módon kezeli a benne szereplő elemeket. A komoly programozási tudást igénylő módszereken kívül van egy további mód arra, hogy az XML-t megjelenítsük: az XSL (eXtensible Stylesheet Language – kiterjeszthető stíluslap nyelv), amivel e tanulmány vége felé fogunk újra találkozni.

A dokumentum alapszerkezete

Az XML a következő elemekből épül fel:

- feldolgozási utasítások (pl. a dokumentum karakterkódolási módja)
- dokumentumtípus deklaráció (a használni kívánt szabályrendszer)
- nyitó-, záró- és üres elem (nyitó és záró: <*author*>Cicero</*author*> [jelentése: a szerző neve Cicero]; üres: <*lb n="123"*/> [oldaltörés, kezdődik a 123. oldal])
- elemtulajdonság avagy attribútum (<author lang="lat"> [a szerző nevét latin alakban adtuk meg])
- egyedek (pl. ex*&kesztülhúzottszárú_p_betű*;iment [a latin *per* rövidítésének jele, amit a dokumentum elején definiálni kell])

Az XML elemekről részletekbe menő tájékoztatást találunk számos magyar nyelvű könyvben,[5] úgyhogy térjünk át a TEI elemekre. Íme egy egyszerű példa:

```
<?xml version="1.0" encoding="ISO-8859-2" ?>
<?xml-stylesheet type="text/xsl" href="textology.xsl"?>
<!DOCTYPE TEI.2 PUBLIC „-//TEI P4//DTD Main Document Type//EN"
„tei2.dtd" [
  <!ENTITY % TEI.XML      'INCLUDE' >
  <!ENTITY % TEI.prose    'INCLUDE' >
```

[5] Bevezetésképpen érdemes Michael J. Young: *XML lépésről lépésre* című könyvével kezdeni (Budapest, Szak 2002), majd – ha még maradt kedv – átvenni Neil Bradley könyvét (*Az XML kézikönyv,* Budapest, Szak 2000). Az XML programozása iránt érdeklődők Brett McLaughin Java és XML-éhez fordulhatnak (Kossuth, Budapest 2001).

```
<!ENTITY % TEI.textcrit 'INCLUDE' >
]>
<TEI.2>
<teiHeader>
<fileDesc>
<titleStmt>
<title>Historia Rerum Hungaricarum ...</title>
</titleStmt>
<publicationStmt>...</publicationStmt>
<sourceDesc>OSzKK Fol. Lat. 159. IV. Tom. ff. 2–92.</sourceDesc>
</fileDesc>
</teiHeader>

<text>
<body>
<div0 type="fejezet">
<lg>
<l>Postquam sacra maiestas tanquam pater pientissimus...</l>
</lg>
</div0>
</body>
</text>
</TEI.2>
```
[6]

A dokumentum annak a meghatározásával kezdődik, hogy ez egy XML dokumentum, a használt karakterkódolás az ISO 8859-2 (amiben benne vannak a közép-európai karakterek, néhány kivételtől eltekintve megegyezik a Windows 1250-es kódlappal, de egyéb operációs rendszerek is megértik, mivel ISO szabvány), illetve a megjelenítésnél egy saját készítésű stíluslapot fogunk alkalmazni. Ez után jön a dokumentum típusának meghatározása, amiből kitűnik, hogy ez egy TEI dokumentum és használja az XML, a próza és szövegkritikai apparátus jelölőkészletét.

A TEI.2 elem fogja közre magát a dokumentumot, aminek van egy fejrésze (*teiHeader*), ami a szövegről szóló alapinformációkat tárolja (szerepe ugyanaz, mint a könyvborítóké és a különféle címlapoké vagy a könyvtári katalóguscéduláké) és egy törzse (*text*), ami a dokumentum tulajdonképpeni szövegét tartalmazza. A szöveget számtalan módon tagolhatjuk, attól függően, hogy a szöveg mit kíván meg, vagy hogy mit

[6] Ez és a további jelöletlen példák a *Historia rerum Hungaricarum, 1662–1683* című névtelen história kézirataiból származnak (lelőhelyük OSzK Kézirattár. Fol. Lat. 159. IV. tom. ff. 2–92., Fol. Lat. 460. ff. 2–92., Fol. Lat. 4335).

szeretnénk később a szövegből kinyerni. A Perl programnyelv mottója („There are more than one way") itt is érvényes.[7]

Átírás

A TEI a textológia hagyományának megfelelően elkülöníti egymástól egy adott szöveg átírását és a szövegkritikai apparátust. Ez abban nyilvánul meg, hogy a két kódkészlet más-más modulban (DTD-ben) kapott helyet. Az elsőben (*TEI.transcr*) a források egyedi jellegzetességei (rövidítések, törlések, javítások stb.), a másodikban (*TEI.textcrit*) elsősorban a kútforrások összehasonlítása.

Rövidítés – feloldás: *abbr* és *expan*

A TEI lehetőséget hagy a betűhű átírásra csakúgy, mint a rövidítések feloldására. A TEI-projektumok mind a kettőre szolgálnak példával. Ha a rövidítést eredeti formában akarjuk megtartani, akkor az *abbr*, ha az értelmezését, akkor az *expan* elemet kell használni. A következő két példa egyenértékű:

eventa<expan abbr="&rum;">rum</expan>
eventa<abbr expan="rum">&rum;</abbr>

A *&rum;* egyedet a dokumentum definíciós részében kell meghatároznunk. Ez lehet a rövidítés beszkennelt képe,[8] vagy egy Unicode[9] karakter (ha van ilyen), vagy egy olyan helyettesítő karaktersorozat (pl. HTML kód), ami tipográfialag imitálja a rövidítés külalakját. A két elemben, csakúgy mint az átírás többi elemében további tulajdonságokat határozhatunk meg:

type: egy általunk meghatározott tipológia szerinti típus. A tipológia lehet (könyvtári kifejezéssel élve) poszt- vagy prekoordinált: vagyis akár előre is meg lehet határozni egy kötött választéklistát, de dönthetünk úgy is, hogy a kódolás során adunk a típusoknak értéket.

---

[7] A Perl másik mottójának állítása („A programozó legfőbb erénye a lustaság") itt sajnos nem igazolódik.

[8] Lásd. http://www.ucalgary.ca/~scriptor/cotton/transcription/abbreviations.htm

[9] A karakterek hagyományos tárolási módja (1 byte) 256 féle jel alkalmazását teszi lehetővé, ami sokszor kevésnek bizonyul, ráadásul nincs olyan karakterkészlet, melyen valamennyi kelet- és nyugat-európai jel helyet kaphatna. A Unicode 2 byte-on ábrázolja a karaktereket, amibe elvileg 256^2 vagyis több mint 65 ezer jel megjelenítésére ad lehetőséget, ám ezek egy része még kihasználatlan. Bővebben: http://www.unicode.org/

resp: az adott olvasatért tudományos felelősséget vállaló személy. Ha ugyanazt a forrást többen is kiadták, és egy-egy helyet különféleképpen értelmeztek (ld. a nevezetes „P. dictus" vs. „predictus" vitát), itt adhatjuk meg az egyik vagy másik változat mellett döntő személy(ek) neveit.

cert: a megállapítás bizonyosságának mértéke az adott elektronikus kiadás szerkesztője szerint (aki nem feltétlenül azonos a *resp*-ben megadott személlyel – például felhasználjuk Kemény József némely kétes hitelű forrásközlését). Ezt a TEI a példákban százalékosan adja meg, ami meglehetősen abszurd, hiszen mit jelent az, hogy 70%-ban vagyunk biztosak egy rövidítés értelmében, vagy egy kivakart szó olvasatában? – ám a jelek szerint ez még megváltozhat, ugyanis a TEI egyik tervezett kiegészítésében, a középkori kódexek alapos kézirattári leírását szolgáló MASTER DTD-ben hasonló szerepben felbukkan a *certanity* attribútum, aminek a lehetséges értékei: 'high', 'medium', 'low', amitől ugyan nem leszünk okosabbak, de egy felesleges tudományoskodó mázat legalább elhagy a leírásból.

Téves alak – helyes alak: *sic és corr*

A hibás, pontatlan, eltévesztett szöveg és ennek helyes értelmezése. A *sic* a hibás eredeti, a *corr* a (kiadó által) javított szöveget tartalmazza. A két példa egyenértékű:

> Cruciferi, quorum dux <sic corr="Georgius">Gregorius</sic> Siculus erat...
> Cruciferi, quorum dux <corr sic="Georgius">Georgius/corr> Siculus erat...[10]

Beszúrás: *add, addSpan*

A szerző, az írnok, a későbbi olvasó, jegyzetelő stb. által beszúrt szövegek. Az előbbiekhez képest két további attribútuma van úgymint a *place*: a beszúrás helye (margón, alul, felül stb. – tetszőleges érték megadható); és a *hand*: az a kéz, amelytől a szöveg származik. (Itt jegyzem meg, hogy a TEI-ben számos esetben nem kell, sőt nem tanácsos minden esetben kiírni a közreműködő személy, a kéz, vagy a forrás teljes nevét, hanem a szöveg egy

[10] Király Péter: „Értelme sincs a sok vén pergamennek...", in Bessenyei – Fügedi – Ö. Kövács – Ringer – Schimert (szerk.): *Studia Miskolciensia 3*, Miskolc 1999, 100. o.

meghatározott helyén ezekről (az egyes elemeket egyedi azonosítóval ellátva) listát kell készíteni, és a megfelelő *hand*, *resp*, *wit* stb. attribútumoknál csak az egyedi azonosítókat kell megadni). Az *addSpan* lényegében megegyezik az előbbi elemmel, de ez hosszabb szöveg betoldására vonatkozik és 'üres elem' vagyis nincs lezáró eleme, hanem a *to* jellemzőben megadott „végpont azonosító" határozza meg a beszúrt szöveg végét (végpont azonosítón az *anchor* („horgony") elemet és ennek egyedi *id* tulajdonságát értem).

> Eodem anno civitas <add hand="hand2" place="a két sor között">Leopoldopolis</add> uno lapide ab oppido Galgocz ...
>
> Quo factum itum est versus arcem Lekaviensem, in qua <addSpan to="p.21r.1" hand="hand2"/>comes Emericu Tököly filius praefati comitis, ante obsidionem <anchor id="p.21r.1"/>arcis Arvensis in studiis morabatur...

Törlés: *del, delSpan*
A forrásban szereplő törölt vagy törlendőként jelölt betű, szó vagy bekezdés. Az általános tulajdonságok mellett megjelölhetjük a törlés *status*át: téves, túl sok vagy éppen túl kevés törlés. A *delSpan*-nál ugyanaz a megjegyzés érvényes, mint az *addSpan* esetében.

> ... ex alt~~er~~a crepidine se in aquam praecipitant ...
> ... per ablegatos tractatus <delSpan to="p.69.1"/>(aliquos ablegati essent)<anchor id="p.69.1"/> admissi ...

Kiemelés: *hi*
Ide tartozik minden olyan grafikai-tipográfiai jelölés, ami az adott szövegrészt kiemeli a környezetéből: a ritkított, vastag aláhúzott, kipontozott, nagybetűs, de speciális jelentéssel (pl. fejezet címe) fel nem ruházható írásmódok. Az eddigiektől eltérően ez a TEI általános elemkészletének a része.

Javítás
A TEI-ben nincs a javításnak saját jelölése, az eddigi elemek kombinációjával kell tehát megoldani (a vonatkozó munkacsoport azonban tervbe vette ennek alaposabb kidolgozását).

a corr–sic párossal:

```
... eorum <corr sic="nomine" resp="H">specie</corr> nobilitatem ...
```

ahol 'H' a vonatkozó példány írnoka, tehát H valamilyen módon kijavította az eredetileg leírt szöveget;

a *del–add* párossal:

```
... eorum <del hand="H">nomine</del><add place="a két sor közé"
hand="H">specie</add> nobilitatem ...
```

'H' kitörölte a helytelen szót és a javítást a szó fölé, a két sor közti üres helyre írta be.

Ugyanezt precízebben fogalmazhatjuk a *kritikai apparátussal* (részletesen ld. később).

```
... eorum
<app>
<rdg varSeq="1" hand="H"><del>nomine</del></rdg>
<rdg varSeq="2" hand="H"><add place="a két sor
közé">specie</add></rdg>
</app>
nobilitatem ...
```

Az *app* elem a kritikai apparátust befoglaló eleme, az rdg egy szöveghely olvasatát, a varSeq tulajdonság pedig az olvasat sorrendjét jelenti (tehát kihúzta, *majd* beszúrta.).

> Megjegyzendő, hogy az XML fájlban a sortörésnek, a szóközöknek, a tabulátornak és más láthatatlan elemeknek semmiféle formázási szerepük nincs, a fenti példa teljes szövege ezért megjelenítéskor egy sorba fog kerülni. Ha ezeket a formázásokat használni szeretnénk a megfelelő elemhez (<lb/>) és egyedhez ( ) kell fordulnunk.

A szöveg egymás utáni többszörös átírásait az elemek egymásba ágyazásával is ki lehet fejezni (kitalált példa):

```
<app>
<rdg varSeq="1" hand="H"><del>nomine</del></rdg>
<rdg varSeq="2" hand="H"><del><add>specie</add></del></rdg>
<rdg varSeq="3" hand="H"><add>nomine</add></rdg>
</app>
```

Helyreállítás: *restore*
Abban az esetben használjuk, ha egy törlést vagy más jelölést érvénytelenítünk:

```
<restore hand="H"><del>nomine</del></restore>
```

Hiány: *gap*
Olvashatatlan, kivehetetlen, kivakart vagy az átírásból valamilyen (az előszóban részletezendő) okból kimaradt szöveg jelzése. Tulajdonságai (az általános tulajdonságok mellett): desc: a hiányzó szöveg leírása; *reason*: a hiány oka (olvashatatlan, sérült, lényegtelen szövegrészlet stb.), extent: a hiány mértéke; *agent*: a fizikai sérülés okozója – már amennyiben meg tudjuk állapítani.

```
perlectis evangelicis re<gap desc="olvashatatlan" extent="2–3 karakter"/>isit
... ibidem manserunt.</p>
<p><gap desc="átsatírozott bekezdés" extent="kb. 20 sor, az oldal 2/3-a"/></p>
...
```

A hiány pótlása más forrásból: *supplied*
Az átírónak/kiadónak a sérült vagy olvashatatlan, de egykor ott lévő szöveg helyére írt, más forrásból vett pótlása. Tulajdonságai az előzővel megegyeznek, de hozzájuk jön a *source*: a beillesztett szöveg forrása. Egy oxfordi antifonálé egyik oldalán túl sokat vágtak le a margóból, de párhuzamos szövegek alapján nagy biztonsággal helyre tudták állítani a forrást (Oxford, Bodleian Library Ms Jesus College 10):[11]

```
ad matu<supplied reason="margin trimmed">tinas</supplied> et ad
Ve<supplied reason="margin trimmed">speras</supplied> In
eva<supplied reason="margin trimmed">ngelio</supplied>
```

[11] Forrás: CURSUS. An Online Resource of Medieval Liturgical Texts. http://www.cursus.uea.ac.uk/

A „kéz" változása: *handShift (handList, hand)*
A kéziratnak gyakorta több írnoka, javítója, folytatója van, sőt bizonyos esetben jelezni lehet a tinta színének vagy az írásképnek a változását. Ha több, jól elkülöníthető, esetleg nevesíthető kezet azonosíthatunk, akkor ezeket a fájl fejrészében, a *handList* összefoglaló elemen belül egy-egy (üres) *hand* elem segítségével lehet felsorolni. A *hand* és a *handShift* legtöbb tulajdonsága megegyezik; így *style*: az írás stílusa (humanista, folyóírás stb.); *ink*: a tinta és az írószer jellemzői; character: az íráskép ('iskolás', 'kiírt', 'erőteljesen jobbra döntött'). A *hand*ben megadhatjuk még az írnok nevét, vagy egyéb jellemzőit (*srcibe*) és azt, hogy az adott kéz a kéziratban az első-e vagy túlnyomórészt hozzá lehet-e rendelni (*first*), míg a *handShift*ben a váltás előtti (*old*) és utáni (*new*) kéz azonosítóját.
Egyszerű esetben pedig így is el lehet járni:

```
<p><handShift ink="barna"/>Civitatem Rodus occupatus est per cesarem Thurcarum</p>
<p><handShift ink="fekete"/>...</p>
```

Bizonytalan olvasatok: *damage* és *unclear*
A *gap* és a *supplied* elemek olyan esetekre vonatkoznak, amikor a szöveg teljességgel olvashatatlan. Vannak azonban olyan határesetek, amikor halványan, bizonytalanul, de mégis ki lehet venni valamit az eredeti szövegből. A *damage* és az *unclear* elemek ezeknek a rögzítésére valók, és elsősorban a rongálódott/rongált terület nagyságában különböznek. Mindkettő közös jellemzője a *hand* (ha szándékos törlés történt) és az *agent* (a rongálódás okozója, ha megállapítható – pl. tűz), az előbbiben pedig még a sérülés fokát (*degree*) és kiterjedését (*extent*), utóbbiban pedig a bizonytalanság okát (*reason*) is meg lehet adni.

```
<damage extent="az egész levél" agent="a levelet kötéstábla belsejében találták, szélein nagy kiterjedésű foltok találhatók">
<l>... öltöznek be az erdők</l>
<l>zöldbe...</l>
<l>he, hea, hó!</l>
...
</damage>
```
[12]

[12] „... öltöznek be az erdö... (1510–1542)", in Hargittay Emil (szerk.): *Bevezetés a régi magyar irodalom filológiájába*, Universitas, Budapest 1996, 262–265. o.

Üresen hagyott hely: *space*
Kódexekben gyakori, hogy az iniciálénak helyet hagyott a másoló, de az illuminátorhoz már nem jutott el a kézirat (vagy közben gazdát cserélt, és az új tulajdonos jelvényeit festette bele – bár ez egyáltalán nem tartozik ide), így az eredmény egy üresen hagyott hely lett. Másolásoknál is előfordul, hogy egy-egy olvashatatlan részt üresen hagytak, de azt senki sem pótolta. A *space* pont az ilyen hiányok jelzésére szolgál. Tulajdonságai: *dim*: széltében, vagy hosszában hagyták üresen az adott helyet, *extent*: a kiterjedés betűben vagy valamilyen pontos mérték szerint (mm, cm).

> Metales inter possessiones Nagy Padan et Pethen pro parte Petri filii Ioannis, Andreae filii <space extent="egy szó" /> Ioannis de <space extent="egy szó" /> Padan, Mathaei filii Ioannis de Nagy Padan et nobilium de Pethen.[13]

Szövegkritikai apparátus
A kritikai apparátus a TEI-ben egy külön alkészletet képez. Elsősorban akkor használjuk, amikor több kútfő szövegvariánsait akarjuk összevetni vagy párhuzamosan közölni, de olyan esetekben is (a javításnál láttunk rá példát) amikor a szöveg átírására szolgáló elemkészlet elégtelennek bizonyul egy bonyolult szöveghely elvárt pontosságú rögzítésére. Az apparátus legfontosabb eleme az *app*. Ennek a befoglaló elemnek a fő feladata összegyűjteni az egy adott szöveghelyhez tartozó különféle olvasatokat.

```
<app>
<lem wit="K1 sz">Ifjú olasz király azonban juta</lem>
<rdg wit="delta">Ifjú görög király azonban juta</rdg>
</app>
```
[14]

Apparátus: – *app*
Látható, hogy az *app* elem fogja egybe a különféle olvasatokat (*lem, rdg*). Az apparátust elláthatunk tulajdonságokkal: a szokásos *type* itt is egy saját magunk által meghatározott típust jelöl (hangsúlyozhatjuk például az adott

[13] Köblös József: *A Dunántúli Református Egyházkerület Levéltárának magyar vonatkozású középkori oklevelei*, h. é. n., 100. o.
[14] Enyedi György: *Historia elegantissima* (szerk. Káldos János), Balassi, Budapest 1994, 41. o.

apparátus fontosságát /v.ö. *lectio difficilior*/ vagy másodlagos mivoltát /helyesírási eltérések/ amennyiben szövegtörténeti bizonyításunkban döntő vagy csekélyebb szerepet játszik); a *from* a lemma, vagyis az alapszöveg kezdetének, a *to* a végének helyét jelzi (ezeket az adott helyen egy-egy *id* attribútummal kell jelezni), míg a *loc* a szövegvariáns helyét jelzi abban az esetben, ha az apparátust nem a szövegben, hanem a dokumentum egy másik részén vagy külső fájlban helyezzük el. Az *app* attribútumait az úgynevezett „gyermekelemek" (amiket magába foglal) öröklik, tehát csak akkor célszerű megismételni, ha adott esetben eltérő értéket veszünk fel.

Olvasatok: *lem* és *rdg*
Az olvasat egy adott szöveghelynek egy adott forrásban előforduló szövegvariánsa. A *lem* a szöveghely eredeti, a szerzői szándéknak leginkább megfelelőnek tűnő, annak elfogadott alapszövege (lemmája), míg az *rdg* ugyanennek egy másik olvasata. A szövegrögzítés során nem kell mindig és föltétlenül döntenünk abban a kérdésben, hogy melyik az alapszöveg. Az *app* elemnek azonban legalább egy *lem*-et vagy *rdg*-t mindenképpen tartalmaznia kell, és nyilvánvaló hogy nem lehet több lemmánk egyszerre. A két elemnek megegyeznek a tulajdonságai; *wit*: a forrás siglája (több forrás esetén szóközzel elválasztva). A forrásokat a dokumentum egy pontján mindenképpen fel kell sorolnunk a *witness* elem segítségével (ld. később). Korábbi példánkban a „K1" és az „sz" Enyedi György *Historia elegantissimá*ja kolozsvári nyomtatványának és prágai kéziratának a stemmában alkalmazott jelei. Ha az alapszöveg forrását a bevezetőben egyértelműen tisztáztuk és jeleztük, akkor a lemmában már szükségtelen megismételni. A *cause*: egyénileg alkalmazható tipológia a variáns keletkezési okának meghatározására (pl. 'hasonló végződésű', 'paleográfiai tévesztés', 'haplográfia' azaz ismétlődő betű/szótag egyszeri írása, 'betű-/szóismétlés', 'téves javítás'), *varSeq*: a variáns sorrendiségét jelző szám (ha erre valamilyen okból – pl. a szöveg többszöri átjavítása – szükség van). A *hand*, a *resp* és a *type* tulajdonságokat az átírásnál látottakhoz hasonlóan használjuk.

Lényeges és lényegtelen eltérések:

```
<l id="3">
<app>
<lem>De ez egyre, tudom, nem találtatok</lem>
<rdg wit="delta" type="lényeges">Ez dologról, tudom, nem
```

```
hallottatok</rdg>
</app>
</l>
interea rerum
<app>
<rdg wit="Ko" type="helyesírási">Tökölianarum</rdg>
<rdg wit="H" type="helyesírási">Tokolianarum</rdg>
</app>
```

[15]

A kialakulás okának meghatározása:

```
<app>
<rdg wit="La" varSeq="1">Experiment</rdg>
<rdg wit="Ra2" cause="fel nem ismert rövidítés"
varSeq="2">Eryment</rdg>
</app>
```

[16]

Alapszöveg és az egyik forrásban tévesről helyesre javított szöveg más kéztől:

```
<app>
<lem wit="Ko">intuitu</rdg>
<rdg wit="H" varSeq="1" hand="m1" cause="téves
olvasás">interitu</rdg>
<rdg wit="H" cause="nachgetragen" varSeq="2"
hand="m2">intuitu</rdg>
</app>
```

Nehezen olvasható szöveg, különféle megoldások (Borsa Gedeontól, Mezey Lászlótól, Horváth Ivántól és Szabó Gézától):

```
<div>
<app>
<rdg resp="BG">&ocaudata;lt&ocaudata;&z;nek be az
erd&ocaudata; &z; [&ocaudata;]l.[?]</rdg>
```

[15] Enyedi: i. m., 39. o.

[16] Oxford, Bodleian Library, Rawlinson Poetic 149. A TEI példája „A bath-i asszonyság meséjé"-ből (Chaucer).

```
<rdg resp="ML">...öltöznek be az erdők<lb/>zöldbe...</rdg>
<rdg resp="HI">[...] öltöznek be az erdő zöld [...-i ...-]ben</rdg>
<rdg resp="SzG">. . . . . öltöznék be,<lb/>az erdő zöl<hi
rend="dőlt">d</hi>. . . . . <lb/> . . . . . .ben,</rdg>
</app>
</div>
```
[17]

Itt az *&ocaudata*; az alul farkincás o betű, a *&z*; pedig nyomtatott gót z betű helyettesítésére szolgál Az *lb* sortörést jelent, a *resp* attribútumban megadottak pedig az irodalomtörténészek azonosítói.

Alvariánsok (szövegcsoportok): *rdgGrp*
A *rdgGrp* olyan olvasatokat csoportosít, amelyek meglátásunk szerint genetikus kapcsolatban vagy egyéb rokonsági viszonyban állnak egymással. Attribútumai megegyeznek a *lem* és *rdg* attribútumaival és azok mint szülőelemtől, öröklik őket.

```
<l>
<app>
<lem>Példát szokott venni</lem>
<rdgGrp type="szórendi">
<rdg wit="delta">Szokott venni példát</rdg>
<rdg wit="p">Szokot példátt venni</rdg>
</rdgGrp>
</app>
nem maga <app><lem>házán</lem><rdg wit="delta
p">kárán</rdg></app>,</l>
```
[18]

Láthatjuk, hogy az utolsó olvasatban szóközzel elválasztva egyszerre két forrást is feltüntettünk, továbbá hogy egy sorban akár több eltérést is regisztrálhatunk. *A bath-i asszonyság meséjé*nek (Chaucer) kéziratiban ugyanarra a szóra több alváltozat is előfordul:

[17] „... öltöznek be az erdö...", ld. 12. jegyzet
[18] Enyedi: i. m., 39. o.

```
<app type="lényegi">
<rdgGrp type="alváltozatok">
<lem wit='El Hg'>Experience</lem>
<rdg wit='Ha4'>Experiens</rdg>
</rdgGrp>
<rdgGrp type="alváltozatok">
<lem wit='Cp Ld1'>Experiment</lem>
<rdg wit='La'>Ex&p-underbar;iment</rdg>
</rdgGrp>
<rdgGrp type="alváltozatok">
<lem wit='[nem bizonyított]'>Eriment</lem>
<rdg wit='Ra2'>Eryment</rdg>
</rdgGrp>
</app>
```
[19]

Megjegyzések a forráshelyhez: *witDetail*
Esetenként szükség lehet arra, hogy egy adott olvasatot bővebben kifejtsünk, ahhoz külön megjegyzést illesszünk. A *witDetail* az általános *note* (megjegyzés) elem különleges formája. Két speciális attribútuma van: a *target* utaló az adott olvasat azonosítójára, a *wit* pedig a vonatkozó forrás sziglája. Használatához természetesen a vonatkozó olvasatnak egyedi azonosítót (*id*) kell kapnia, hogy a hivatkozási integritás fönnállhasson.

```
<app type="fontos írásmódbeli eltérés">
<rdg id="f.76.2" wit="H">Kunfstein</lem>
<rdg wit="Ko">Kupstein</rdg>
</app>
<!-- máshol a szövegben -->
<witDetail target="f.76.2" wit="H">a kézirat többi részétől eltérően gót betűs írással</witDetail>
```

A <!-- és --> jelek között szereplő szövegek az XML-ben megjegyzést jelentenek, nem tartoznak bele a szövegbe.

Forráshely bővebb formában: *wit*
Kritikai kiadások átírásánál szükség lehet rá, hogy az utalásrendszer fenntartása céljából a forrást abban az alakban is rögzítsük, ahogy azt a

[19] Ld. TEI dokumentáció.

kiadó közölte (szemben a *lem* és az *rdg* elemek *wit* attribútumában szereplő formalizált alakjával). A *wit* az *rdg*, *rdgGrp* és *lem* elemek után áll.

```
<l id="7.3">
<app>
<rdg wit="Lő">Ördög, ellenség, suk bu, szégyenség nem árthat...</rdg>
<rdg wit="Bá">Ördög, ellenség, suk bu, szegénység nem árthat...</rdg>
</app>
<app type="másodlagos" loc="7.3">
<rdg wit="EB">szégyenség</rdg> <wit>Eckhardt, 1942</wit>
</app>
</l>[20]
```

Vagyis közlünk két eredeti forrást és egy korábbi szövegkiadás megoldását, oly módon, hogy a bibliográfiai utalást az irodalomban bevett rövidítés formájában is megadjuk ('EB'='Eckhardt 1942'). Erre akkor van szükség, ha a megjelenítésnél ezt a rövidített alakot szeretnénk kiírni a teljes címleírás helyett ('Eckhardt Sándor: Balassa Bálint, Budapest 1942') amit a dokumentum egy másik helyén adunk meg.

```
<app><lem wit="K1 sz">Ifjú olasz király azonban juta</lem>
<wit>K<hi rend="alsóindex">1</hi> ...</app>
```

Forráslista: *witList* és *witness*

A forráslista, miként több más hasonló felsorolás (pl. a már idézett *handList*) a dokumentum egy elkülönült részében található. Ebben írjuk le azokat a tudnivalókat, melyek az egyes forrásokra vonatkoznak. A leírás terjedelme nem meghatározott, de mindenképpen célszerű megadni a forrás könyvtári jelzetét, valamint a *sigil* jellemzőben a *szigláját*.

```
<witList>
<witness sigil="K">OSzK, Kézirattár, Fol. Lat. 460. ff. 2–92.</witness>
<witness sigil="B">OSzK, Kézirattár, Fol. Lat. 4335.</witness>
<witness sigil="Ko">OSzK, Kézirattár, Fol. Lat. 159. IV. Tom. ff. 2–92.
(alapszöveg)</witness>
</witList>
```

[20] Stoll Béla: „Szövegkritikai problémák a magyar irodalomban", in Hargittay: i. m., 114–170, 162. o.

Az irodalomban bevett név feltüntetésére használhatjuk az általában kiemelésre használt *soCalled*, bibliográfiai jellegű megjegyzésekre pedig a *bibl* elemet

```
<witList>
<witness sigil="CP"><soCalled>Codex Prayanus</soCalled> <bibl>OSzK, Kézirattár, MNy. 1. Membr., saec. XII. ex. (cca ann. 1192–1193.), ff. XXVIII + 144, 23.5×15 cm.</bibl></witness>
</witList>
```

Több, egymással rokonságban álló forrásból forráscsoportot (pszeudo-forrást) lehet képezni az *included* jellemző segítségével, amiben szóközzel elválasztva adjuk meg az érintett forrásokat:

```
<witList>
<witness sigil="D">Debrecen, 1577 (RMNy 380)</witness>
<witness sigil="p">Petrovay Miklós énekeskönyve</witness>
<witness sigil="zeta" included="p">A &zeta; variáns</witness>
<witness sigil="epszilon" included="D zeta">Az &epsilon; variáns</witness>
</witList>
```

Töredékes források: *witStart, witEnd, lacunaStart, lacunaEnd*
Ha a forrás töredék, töredékek csoportja, vagy olyan teljes szöveg, amiben kihagyások (lacuna) találhatók, akkor szükség van arra, hogy (a *lem* vagy *rdg* elemeken belül) üres elemekkel jelöljük a töredék illetve a kihagyás kezdetét és végét. Erre négy elem kínálkozik: *witStart*: a forrástöredék kezdete vagy folytatása; *witEnd*: a forrástöredék vége vagy felfüggesztése; *lacunaStart*: a *majdnem teljes* forrásszövegben található kihagyás kezdete; *lacunaEnd*: a *majdnem teljes* forrásszövegben található kihagyás vége. Nincs pontosan definiálva, hogy egy forrásszöveget mikor tekintünk töredékesnek és mikor kihagyásoktól szabdaltnak, így az alábbi két példa jelentése ugyanaz.

```
<app>
<lem wit="El Hg">Auctoritee</lem>
<rdg wit="La Ra2">auctorite</rdg>
<rdg wit="X"><lacunaEnd/>auctorite</rdg>
</app>
<app>
<lem wit="El Hg">Auctoritee</lem>
<rdg wit="La Ra2">auctorite</rdg>
```

```
<rdg wit="X"><witStart/>auctorite</rdg>
</app>
```
[21]

Az apparátus hozzákapcsolása a szöveghez

Három módszer kínálkozik arra, hogy összekössük a szöveget a kritikai apparátussal:

- *szöveghelyre utalás* (location-referenced)
- két végpont rögzítése (double-end-point)
- *párhuzamos felosztás* (parallel-segmentation)

Az első kettőt egyaránt lehet használni az alapszövegen belül és kívül elhelyezett (vagy más szóval belső és külső) apparátus esetén. A párhuzamos felosztás nem minden esetben állapítja meg az alapszöveget, és a variánsokat egymás mellett szerepelteti. Ebben az esetben nem lehet az apparátust a szövegen kívülre helyezni, hanem a szövegen belül kell szerepeltetni. Ha egyáltalán alkalmazunk szövegkritikai apparátust (*app* elemet), akkor a választott módszert a szöveg fejrészében az *editorialDecl* elemen belül a *variantEncoding* elemben kell megadnunk: a *method* tulajdonságban a módszer nevét (a lehetséges értékek: *location-referenced*, *double-end-point* és *parallel-segmentation*), a *location*-ben pedig azt, hogy az apparátus a folyószövegen belül vagy kívül helyezkedik el (*internal*, *external*). Vegyük most sorra a három módszert.

A szöveghelyre való utalás módszere

Ez a legtöbb nyomtatott forráskiadvány hagyományos eljárása. Az utalót, a referenciát valamilyen szabály, legtöbbször az oldalszám és sorszám alapján állapítják meg. Külső apparátus mellett történő választás esetén a *variantEncoding* elem tartalmát így kell megadni:

```
<variantEncoding method="location-referenced" location="external"/>
```

A főszöveget a következőképpen kódoljuk:

```
<lg n="146" type="versszak">
<l n="581">A szerelmet bennem nem titkolhatom,</l>
<l n="582">Nem is illik azellen rugódoznom,</l>
<l n="583">Noha tőled, uram atyám, vádlatom,</l>
<l n="584">Érts meg, mely méltatlan azokat hallom.</l>
</lg>
```
[22]

[21] Ld. TEI dokumentáció.

Itt az *lg* és az *l* elemek *n* attribútuma tartalmazza azt a referenciát, amire a jegyzetekben hivatkozni fogunk. A megjelenítés során ezt a referenciát – ha szükséges – a sor mellé illeszthetjük a vizuális tájékoztatás végett. A jegyzetben (amit valahol a dokumentum egy másik pontján helyeztünk el, például a 'Jegyzetek' vagy 'Magyarázatok' szakaszban) a *loc* attribútum segedelmével utalunk vissza a verssor azonosítójára:

```
<app loc="583">
<rdg wit="l">attiam uram</rdg>
</app>
<app loc="584">
<rdg wit="sz">Őriz megh meli</rdg>
</app>
```

Belső apparátus esetén a variantEncoding elem location-je a következő:

```
<variantEncoding method='location-referenced' location='internal'/>
```

A jegyzetet az adott kifejezés helyén, de úgy is meg lehet adni, hogy a sorhoz (bekezdéshez stb.) illesztjük a jegyzetet. Az első módszerrel a sort többfelé törjük:

```
<lg n="18" type="versszak">
  ...
<l>Sem kazdagságában világi sok jokhoz</l>
<l>bátor ne
<app>
<lem>ragazkodgyék</lem>
<rdg wit="NHP"><sic>rakagazkodgyék</sic></lem>
.</l>
  ...
</div>
```
[23]

A szöveghely folytatását a sorhoz illesztett jegyzet alkalmazásával kódoltam. Itt tehát közvetlenül nincs megjelölve az a szó, amelyre a jegyzet vonatkozik:

[22] Enyedi: i. m., 77. o.

[23] Szepsi Csombor Márton: „Magyar versek", in *Összes művei*, (szerk. Kovács Sándor Iván – Kulcsár Péter) /Régi magyar prózai emlékek 1./, Akadémiai, Budapest 1968, 401. o.

```
<lg n="18" type="versszak">
 ...
<l>Tekinczetek engem szemlelyetek testem
<app>
<rdg wit="NHP"><sic>szemlyetek</sic></lem>
</app>
</l>
<l>nisze miképpen járék.</l>
 ...
</div>
```

A két végpont rögzítése

Ennél a módszernél a főszöveg lemmájának elejét és végét pontosan rögzítjük (szemben a szöveghelyre utalással, ahol általában egy szövegblokkra – sorra, bekezdésre – és nem a pontos szöveghelyre utalunk). Ez lehetővé teszi a források (vagy legalábbis lényeges elemeinek) rekonstrukcióját. A jegyzetben (*app*) a lemma elejét a *from* tulajdonsággal, a végét a *to*-val jelezzük. Ezeket a szövegben a már ismertetett *n* és *id* tulajdonságokkal illetve az *anchor* elemmel határozzuk meg.
Külső apparátus esetén a fejrész vonatkozó része így fest:

```
<variantEncoding method='double-end-point' location='external'/>
```

A főszöveg:

```
<l n="2">Virginis innuptae <anchor id="29a"/>sollicitusq[ue]<anchor id="29b"/> thorum?</l>
```
[24]

a jegyzet:

```
<app from="29a" to="29b">
<rdg wit="EV">sollicitasq<abbr>&que;</abbr></rdg>
</app>
```

Ha a megjegyzést illető kifejezés a sor elején áll, nem kell külön *anchor* elemet alkalmaznunk, megteszi a sor saját *id*-je, és igaz ez abban az esetben is, ha a *to* elem a sor végéhez mutat.

[24] Szepsi Csombor Márton: „Europica varietas", uo., 146. o.

A belső apparátus esetén a fejrész:

```
<variantEncoding method='double-end-point' location='internal'/>
```

A főszöveg:

```
<p>... comes Adamus <anchor id="f.4.n.10">Forgách
<app from="f.4.n.10">
<rdg wit="H">Forgácz</rdg>
<rdg wit="Ko">Forgach</rdg>
</app>
partium Cis-Danubianarum generalis...</p>
```

Ha világosabban szeretnénk látni az eltéréseket, akkor az apparátusban a lem segítségével megismételhetjük az alapszöveget.

Átfedő lemmák

Megesik, hogy az egyes szövegvariánsok átfedő szövegrészleteket tartalmaznak: az első szó közös minden szövegben, a második az A és B forrásban, a harmadik pedig a B és C forrásban egyforma. Ekkor lehet ahhoz a macerás, ám de pontos megoldáshoz fordulni, hogy a szöveget a (variánsok szempontjából) lehető legkisebb egységenként megjelöljük.

```
<l n="196">Meggyúlt
<anchor id="196.1"/> szösznek,
<anchor id="196.2"/> higgyed,
<anchor id="196.3"/> füstét
<anchor id="196.4"/> nézhetni.
<anchor id="196.5"/>

<app from="196.1" to="196.5">
<lem wit="sz">szösznek, higgyed, füstét nézhetni.</lem>
<rdg wit="sz">tűznek higiet nehiz föstet nizhetni</rdg>
<rdg wit="l">tőznek langiat nehez oltani</rdg>
<rdg wit="D">tüznek nylvan föstit nézheti</rdg>
</app>
<app from="196.2" to="196.4">
<lem wit="Hg">higgyed, füstét</lem>
<rdg wit="p">nyilvan füstit</rdg>
</app>
</l>
```

[25]

[25] Enyedi: i.m., 51. o.

Láthatjuk, hogy a második apparátusban közölt lemma az elsőben közöltnek részlete. Megjegyzendő, hogy a következőkben tárgyalt harmadik módszerrel, a 'párhuzamos felosztással' nem lehet átfedő lemmákat meghatározni, ott magukat a szövegrészeket kell apróbb részekre széttördelni.

Párhuzamos felosztás

A harmadik módszer abban különbözik a két végpontostól, hogy ebben a szöveg minden egyes pontján a többi olvasat variánsaként adjuk meg az olvasatot. Ennek a legfontosabb következménye, hogy a kódolás alapján az összes szövegvariánst teljes mértékben és viszonylag könnyen reprodukálni lehet, mivel a variánsok pontosan megfeleltethetők egymásnak (pontosan megadjuk, hogy az egyik szereplő szövegrész helyett mi áll a másik forrásban). Ennek megfelelően nincsen az apparátuson kívül eső variánssal rendelkező főszöveg, hiszen minden variánst, így a lemmát is az apparátuson belül adunk meg, amit így csakis a szövegen belül kell elhelyeznünk és semmi esetre sem kívül.

fejrészben:

```
<variantEncoding method="parallel-segmentation" location="internal" />
```

a szövegben:

```
<p>
3.
<app><lem>Szokot</lem><rdg wit="KvárII">Lakott</rdg></app>
vala a szegeny Deáki Püspök igy is kérkedni trefából vagy serio<note
type="fordítás">komolyan</note>: En <app><lem>fütyölni</lem><rdg
wit="Gyfv">fügyölni</rdg></app> nem tudok; En kurva hajat<note
type="magyarázat">Kurva haj: paróka, v.ö. <hi>Nagyenyedi Demokritus</hi>
356a.</note> nem viseltem soha; En idegen Aszszonynak
<app><lem>a</lem><rdg wit="OL" cause="'a' hiányzik"></rdg></app>
melljét nem
<app>
<lem>illettem</lem>
<rdg wit="KvárI Mvh">illettem soha</rdg>
</app>
etc. ...
</p>
```

[26]

[26] Hermányi Dienes József: „[Püspökök élete]. Deaki Josef Püskökről", in *Szépprózai munkái*, szerk. S. Sárdi Margit, Budapest, Akadémiai–Balassi 1992 /Régi magyar prózai emlékek 9./ 79. o.

Látni, hogy a szöveg tördelése lényegtelen, az apparátust el lehet helyezni folyószövegként, és teljesen részekre darabolva is – épp ezért jobb választásnak tűnik ez utóbbi, hiszen ez egyszerűbben átlátható. Megfigyelhetjük, hogy a szöveg hiányára nincs önálló elem: itt egyszerűen egy üres elemet illesztettünk be, de választhattuk volna a következő megoldást is:

```
kurva hajat nem viseltem soha; En idegen Aszszonynak
<app>
<lem>a melljét</lem>
<rdg wit="OL" cause="'a' hiányzik">melljét</rdg>
</app>
nem
```

Valószínűnek tűnik, hogy a TEI következő változatában erre már egy pontosabb megoldást fognak kidolgozni. A szövegben két jegyzet is található: az egyik egy latin kifejezés fordítása, a másik egy magyar kifejezés jelentésének megvilágítása. Ezt két külön típusba soroltuk és majd a megjelenítésnél fogjuk szabályozni, hogy az adott jegyzet hol jelenjen meg.

Megjelenítés és egyéb használat

Ahhoz, hogy az XML fájlunkat akár nyomtatásban, akár a weben meg tudjuk jeleníteni, a szöveget valamilyen módon konvertálni kell.

Nyomtatott szöveg esetében olyan formátumot kell találnunk, amelyiket valamely szöveg- vagy kiadványszerkesztő meg tud nyitni. A bevezetőben esett szó arról, hogy egyre több XML fájlt feldolgozni képes termék van, ám ez nem jelenti azt, hogy ezek bármely DTD-vel elboldogulnak. Nem is csoda, hiszen az XML-ben sehol sem rögzítjük a megjelenítés szabályszerűségeit. Ezek a saját (ti. a programokhoz kötődő) XML formátumok tehát az XML eszközeivel általában a formázás elemeit rögzítik, például azt, hogy miként lehet leírni a 3 ponttal megemelt és bekeretezett felső indexbe tett 10 pontos betűnagyságú számot, vagy hogyan lehet mind vízszintesen, mind függőlegesen középre igazítani egy függőlegesen két összevont rovatba írt szöveget. A gyakorlatban a következő lehetőségeink adódnak:

– *Microsoft Word.* A leggyakrabban használt szövegszerkesztő legújabb verziójának DTD-jét (valamint a dokumentációját) – még mielőtt használata

igazán elterjedt volna – épp a napokban [2004. március végén] vonták ki a nyilvános és ingyenesen hozzáférhető programok köréből. Azt egyelőre nem látni világosan, hogy ez mennyire fogja érinteni a magánfelhasználókat, ám az biztos, hogy más alkalmazások kevésbé fogják export és import formátumként használni, így elterjedtsége is korlátosabb lesz. A TEI honlapról azonban letölthető egy olyan eszköz, amivel a megfelelően formázott Word dokumentumokat TEI formára lehet alakítani.[27]

- *Quark Xpress*. A Quark a legtöbb formázási lehetőséget biztosító, valódi professzionális kiadványszerkesztő. Előnye esetünkben épp a hátránya is: magán (hobbi szintű) felhasználását az ára és a komplexitása lehetetlenné teszi, ráadásul itt az XML be- és kimenetet biztosító modul ún. 'third party software' vagyis ez egy független gyártó terméke, az alapprogramban nincs benne. Ennek ellenére egy (tipográfussal és programozóval felvértezett) kiadónak valószínűleg ez jelenti a legmegfelelőbb megoldást.
- *Frame Maker*. A Quark mellett a másik nagy kiadványszerkesztő eszköz. Saját XML-kezelő moduljai lehetővé teszik a DTD és a Frame Maker-szerkezet, valamint az XSL és a Frame Maker stílusok konverzóját szabályozó megfeleltetések/leképezések megírását, aminek elvégzése után ugyanolyan kényelmesen (WYS/WYG) használható mint a Word. (Jelem írás elkészülte után jelentek meg Váncsa István remek cikksorozatának az XML-lel foglalkozó, és több itt is említett programot alaposabban ismeretető darabjai a számítástechnikában http://www.szamitastechnika.hu/vancsa.php.)
- *PDF*. A PDF beépített módon továbbra sem támogatja az XML fájlokat, viszont ahhoz, hogy PDF dokumentumokat hozzunk létre, a gyártó cég remek API-t (programozóknak szóló függvénykönyvtárat) biztosított

[27] http://www.frank-it-beratung.de/doc2xml/english.html

rendelkezésre, aminek alapján számos nyílt forráskódú alkalmazást készítettek. Az egyik ilyen alkalmazás a FOP, ami egy speciális DTD-nek megfelelő XML fájlt PDF-re tud konvertálni. A feladat tehát az, hogy a TEI fájlunkat FOP-állománnyá alakítsuk. Egy egyszerű FOP-részlet:

```
<fo:block font-size="24pt" text-align="centered">Bevezetés</fo:block>
```

- *OpenOffice.* Az OpenOffice egy nyílt forráskódú, ingyenes, teljes értékű irodai programcsomag, a MS Office remek alternatívája. Ami szempontunkból a legfontosabb, hogy saját beépített XML import/export funkcióval rendelkezik, sőt másféle XML fájlokat is meg tud nyitni. Jelenleg egy szintén viszonylag széles körben elterjedt XML változat, a DocBook formátumát tudja fogadni, de ezt nem holmi dokumentálatlan belső „varázslatok" segítségével, hanem nyílt XSL fájl segítségével. Ennek alapján elkészítették a TEI–OpenOffice konvertert, ami a TEI honlapjáról letölthető és ingyenesen használható.[28]
- *Classical Text Editor.* Az Osztrák Tudományos Akadémián fejlesztettek ki egy sajátos szövegszerkesztőt: ezt ugyanis kifejezetten a kritikai kiadások készítésére szánták. A szerkesztőnek van SGML kimeneti/bemeneti, valamint RTF és PDF kimeneti formátuma. Sajnos az SGML sem konform a TEI-vel, így ebben közvetlenül nem szerkeszthetünk TEI állományt és lehetőségei sem érik el az utóbbi széles spektrumát, de célirányos funkciói, alacsony ára és nagyfokú kényelme miatt (konvertálási kényszer nélkül hozhatunk létre kritikai kiadásokat) van rá esély, hogy a TEI-nél elterjedtebb rendszer legyen.

Átnézve a kínálatot, nyilvánvaló hogy a TEI dokumentum nyomtatásához kisebb-nagyobb mértékben nélkülözhetetlen egyéb számítástechnikai ismeretekkel felszerelkeznünk. Nem kell túlzottan megijedni; ezek az ismeretek (szöveg- vagy kiadványszerkesztési gyakorlattal) néhány hét alatt

[28] http://www.tei-c.org/Software/teioo/

elsajátíthatók. Úgy vélem azonban, hogy ideális esetben a szövegkiadónak egy olyan (könyv)kiadóval kell együttműködnie, ahol van ehhez értő, vagy ezt elsajátító munkatárs.

Nem más a helyzet a szövegek interneten való megjelenítése esetében, viszont itt nagyobb tere lehet a fantáziának, hiszen nem kell a papírmérethez vagy terjedelmi korlátokhoz igazodni. Az internet tipikus fájlformátuma a HTML, ami az SGML „lebutításával" jött létre. A használható eszközkészlet elsősorban a szövegformázásra van fókuszálva, de lehetőség van bizonyos szemantikai struktúra alkalmazására.
Az XML megjelenítésére több módszer kínálkozik. Kezdjük a legegyszerűbbel és haladjunk a nehezebb felé!

Átfedő stíluslapok – CSS

A HTML szabvány kialakítása után viszonylag korán felmerült az igény arra, hogy a viszonylag egyszerűbb formázási lehetőségekkel felruházott HTML fájlok kinézetébe bonyolultabb elemeket illeszthessenek és a sok közvetlen formázás helyett (pl. minden egyes címsornál egyenként megadni a karaktertípust és a méretet) az egész weblapra vagy az egész *site*-ra egyetlen helyen lehessen beállítani ezeket az értékeket. Ennek a megoldása a Cascading Style Sheet (CSS) szabvány. A CSS egyszerűen felülírja a megadott elem kinézetét, ráadásul nagy előnye, hogy elhelyezhető a fájlon kívülre, így egyszerre több állomány is használhatja ugyanazt a definíciót, vagyis minden dokumentumnak egyszerre határozhatjuk meg a kinézetét. A CSS másik újítása az osztályfogalom bevezetése, amivel egyes elemeket csoportba lehet foglalni (például megadni hogy adott bekezdés felsorolás, jegyzet, mottó, szerzőségi közlés, stb.) ami már a szemantikai jelölés felé mutat. A CSS-ben az XML-től eltérően ezek az osztályok nem kötöttek, nincs szabályozó séma, ahogy hierarchia sincs: tőlünk függ, hogy következetesek vagyunk-e vagy sem.

A CSS-t azonban nem csak HTML, hanem XML fájlok megjelenítésekor is használhatjuk. Van azonban néhány korlátja, ami miatt csak a legegyszerűbb esetekben tanácsos kizárólag CSS-sel oldani meg a problémát: a CSS nem szabályozza (pontosabban fogalmazva: nem jeleníti meg) az elemek attribútumait, kizárólag a tartalmát, tehát ezek az információk óhatatlanul elvesznek. Nem gondoskodik az entitások megfelelő behelyettesítéséről, nem számol a struktúra sajátosságaival. Jó kiegészítő eszköz azonban más módszerekkel kombinálva.

Kiterjeszthető stíluslapok – XSL

A már többször emlegetett XSL nyelv szorosan kötődik az XML-hez, elsőrendű feladata, hogy a konvertálást végrehajtsa. A bemenete az XML fájl, de a kimenete tulajdonképpen bármi lehet, leggyakrabban egy másik XML (pl. az OpenOffice vagy a FOP formátuma) vagy HTML. Az XSL az eddig tárgyalt XML, HTML és CSS szabványokkal ellentétben nem leírónyelv, hanem már egy programnyelv (még ha az „igazi" nyelvekhez képes meglehetősen korlátosak az adottságai): vannak benne függvények (újakat is lehet definiálni), ciklusok, változók, feltételes elágazások, rendezés. Segítségével a konverziós feladatokon kívül számos feladatot elvégezhetünk: új szempontok alapján átszervezhetjük a szöveget, elemezhetjük, vagy bizonyos szempontok alapján kivonatolhatjuk a tartalmat (X kéz helyesírási tévesztései), ellenőrizhetünk számos szöveggel kapcsolatos feltevést (igaz-e, hogy az A és B szöveg közötti hasonlóság megközelíti az α és β relációét).

Az XSL néhány tipikus felhasználási módja:

- számoljuk meg, hány javítás (corr elem) van a marosvásárhelyi szövegvariánsban,
- rendezzük sorba a rövidítéseket,
- találjuk meg azokat a sic elemeket, amelyeknek nem adtunk meg corr tulajdonságot,
- írassuk ki az összes olyan bekezdést, melyben így vagy úgy szerepel valamely tisztséggel jelölt egyházi személy (ehhez persze előzetesen tudnunk kell, hogy milyen egyházi tisztségek vannak – ezt a tudást pedig tároljuik egy külső XML fájlban),
- ne a bekezdéseket, hanem a művek címét írassuk ki,
- tördeljük fejezetekre a könyvet.

(A római műegyetemnek a Michelangelo Mózesével büszkélkedő san Pietro in Vincoli templom és a Colosseum közt félúton található jegyzetboltjában forró nyári napokon is kapható Sal Mangano kiváló könyve, az *XSLT Cookbook* (O'Reilly, 2003), melyben a Tisztelt Olvasó számos ehhez hasonló feladat megoldására találhat recepteket.)

Az XML és az XSL fájlokból az xsl értelmező (parser) készít kimenetet. A modernebb internet böngészőkbe ez a program be van építve, ha tehát az XML dokumentum hivatkozik a stíluslapra, akkor a képernyőn már a rendezett kimenetet fogjuk látni. Ugyanakkor léteznek önálló

programok is, használatuknak az az előnye, hogy csak egyszer kell lefuttatni; az olvasó így nem az XML fájlt, hanem a konvertált állományt fogja látni (az első változatban a konverzió minden megtekintéskor lefut – és erőforrásokat köt le).

Az XSL használatára álljon itt egy egyszerű formázási példa:

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

Vagyis, ha a feldolgozó program a title elemmel találkozik, akkor annak tartalmát kurziválja (az i – italic – a HTML-ben a dőlt formázásra szolgál).

Dinamikus HTML

A dinamikus HTML olyan weblap, ami képes bizonyos interakciókra az olvasóval. Például a kurzort egy adott pont fölé helyezve felbukkan egy kis ablakocska, melyben szövegkritikai jegyzetünk található. Ebben tartalmilag nincs semmi új, viszont a stíluslapokon kívül az interaktivitást biztosító programot is meg kell írni valamilyen kliens oldali szkriptnyelven (JavaScript, JScript). Az interaktivitás annyiban befolyásolja stíluslapjainkat, hogy előre meg kell terveznünk, hogy mit teszünk első pillanatban láthatóvá és mit 'kattintásra' – ezzel „tisztítani" tudjuk az alapszöveg tipográfiai megjelenését, hiszen minden pillanatban annyi látszik a képernyőn, amennyi szükséges, a képernyő nincs tele az első látásra zavaró sokszintű jegyzetapparátussal. A szkriptnyelveknek továbbá van egy olyan előnye, hogy a bevezetésben említett DOM metódusok segítségével a felhasználó 'röptében' szűrheti, rendezheti az alapdokumentumot. Hátrányuk viszont, hogy ezek a nyelvek nem váltak olyan szabványokká, mint az XML vagy a stíluslapok, ugyanis a böngészőgyártók saját szája ízük szerint alakították és verzióról-verzióra módosították is őket, vagyis viszonylag munkás olyan kellően összetett programot írni (ha nem éppen lehetetlen), ami minden operációs rendszer minden böngészőváltozatán kielégítő eredménnyel és gyorsasággal fog futni – célszerű tehát a szkriptnyelvet inkább apróbb kiegészítő szolgáltatások megírására használni.

Adatbáziskapcsolatok

Az XML tulajdonképpen olyan adatbázis leírás, ami – a jelenleg uralkodó relációs adatbázismodeltől eltérően – lehetőséget ad kötetlenül vagy szabályosan ismétlődő mezők (mezőcsoportok) és almezők kezelésére, de nincs mögötte adatbáziskezelő 'motor'. Ennek ellenére az XML és az adatbáziskezelés kapcsolatára még mindig inkább a kísérleti fázis jellemző. Az általános, köznapi használatban olyan típusú XML állományoknál van dinamikus fejlődés, amit a relációs módszerrel is ábrázolni (és így annak megfeleltetni) lehet. Pedig mindazokat a feladatokat, amit jelenleg az XSL segítségével oldhatunk meg, sokkal robosztusabban tudná kezelni egy erre a célra épített adatbáziskezelő. Nézzünk meg azonban néhány ma is létező irányzatot:

- *Nagy kereskedelmi adatbáziskezelők* (Oracle, MS SQL): immár rendelkeznek valamiféle 'belső' XML kezeléssel. Ezek azonban komoly programozói felkészülést és még komolyabb anyagi hátteret igényelnek.
- *Nyílt forráskódú adatbáziskezelők.* Két csoportot különíthetünk el: az olyan relációs adatbáziskezelőket, amelyekhez van valamiféle XML modul (pl. PostgreSQL) és azokat, amelyeket kifejezetten az XML kezelésére hoztak létre (eXist, Zebra). Tulajdonképpen mindkét csoport a kísérletezés fázisában van, használatukhoz pedig ugyanúgy programozói ismeret szükségeltetik.
- *Tartalomkezelő rendszerek* (pl. NXT3, Epic, TEXTML). Ezek olyan alkalmazások, amelyek a publikálást és a hatékony visszakeresést helyezik a középpontba. Általában a szabványoktól eltérő, de hatékony megoldásokat alkalmaznak, kezelésüknek nem feltétele a mélyreható programozói ismeret, viszont a magánszemélyek számára nehezen elérhető kategóriába tartoznak. Több magyar kiadó is használ ilyen rendszert (Magyar Nagylexikon, Akadémiai, Balassi, Arcanum).

Összegzés

Az XML és ezen belül a TEI megítélésem szerint már ma is egy olyan eszköz lehet a textológus kezében, aminek a segítségével számos, a szöveg ábrázolását érintő problémát (például a „négyféle jegyzet" problémáját[29]) sikeresen meg lehet oldani. Azonban ma még nélkülözhetetlen hozzá a mélyebb számítástechnikai ismeret, esetenként a programozói készség, hiszen az elméleti szépség mögött még nincsenek se elkészítve, se elterjedve azok a kényelmi szolgáltatások, amik az állomány előállítását és elemzését olyan napi rutinná tudnák tenni, mint amilyenné a szövegszerkesztés és a táblázatkezelés vált az utóbbi tíz évben. Azok a kényelmes használatot biztosító eszközök, melyek rendelkezésre állhatnak (OpenOffice, Classical Text Editor, Folio stb.) alapkiépítésükben nem biztosítanak TEI kompatibilis kimenetet. Fegyvereink használatához egyelőre fegyverhordozóra is szükségünk van!

[29] Hogyan jelenítsük meg és különítsük el egy szövegszerkesztőben a szerző eredeti jegyzeteit, a közreadó kritikai apparátusát, szómagyarázatait és tárgyi jegyzeteit?