MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Tvorba dokumentu v XML

BAKALÁŘSKÁ PRÁCE

Jan Pavlovič

Brno, jaro 2001

Prohlášení

Prohlašuji, že tato bakalářská práce je mým puvodním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: PhDr. Josef Prokeš, Ph.D.

Podekování

Dekuju vsem.

Shrnutí

Psaní dokumentů ve formátu XML DocBook sebou přináší hned několik výhod. Dokumenty lze transformovat do nejrozšířenějších formátů pro prohlížení dokumentů. A tématické vyhledávaní a katalogizace není problémem.

Klíčová slova

HTML, XHTML, XML, SGML, DTD, LaTeX, transformace

Předmluva

Tato práce byla připravena ve formátu XML s využitím DTD DocBook. Závěrečná sazba byla provedena typografickým systémem LaTeX ve stylu fithesis z písma Computer Modern ve variantě CS-font.

Žijeme v době bouřlivého rozvoje informačních technologií a novinky, nad nimiž jsme se v nedávné době rozplývali, jsou již dnes zastaralé. Občas se nám může zdát, při pohledu na jen pár měsíců staré technologie, jak jen jsme mohli žít s něčím tak primitivním a jak jsme se mohli obejít bez vymožeností, jež máme dnes k dispozici. Každý z nás si pamatuje ty chvíle, kdy dojde k rozhodnutí zmodernizovat výpočetní stroj, který používáme k naší každodenní činnosti. Pryč jsou naše představy o koupi pouze nového procesoru či jiných komponent. Nakonec jsme stejně nuceni provést kompletní inovaci většiny zařízení, neboť technologie natolik pokročila, že již není možné staré a nové technologie efektivně kombinovat.

Podobně se tak děje i na poli softwarovém. Kolikrát narážíme na fakt, že náš ještě před časem nový program, ať už textový editor, webový prohlížeč, tabulkový procesor atd., špatně či vůbec neinterpretuje daný dokument, který byl vytvořen novější verzí. A co teprve za pár let. Co když budeme chtít, aby námi dnes napsaný dokument byl k přečtení i za pár let! A to bez shánění historických programů a operačních systémů. Opravdový problém by potom nastal s vizualizací důležitých a cenných textů, jako jsou kupříkladu návody a dokumentace.

Na takové a podobné problémy se pokusíme odpovědět v následujícím textu. Snad přinese i nový náhled na možnosti tvorby dokumentů a komunikace.

Obsah

1 **Úvod** 6
  1.1 *SGML* 6
  1.2 *Historie* 8
  1.3 *Čas pro XML* 8
2 **XML syntaxe** 10
  2.1 *Tvorba DTD* 10
  2.2 *Základy XML* 10
  2.3 *Stylové jazyky* 11
3 **Formát, který vydrží** 13
  3.1 *Na rozcestí* 13
  3.2 *DocBook* 14
  3.3 *Validace* 14
4 **Realizace vizuálních podob** 16
  4.1 *Transformace do PDF a PS* 16
  4.2 *Transformace do LaTeXu* 16
  4.3 *Transformace do RTF* 22
  4.4 *Transformace do HTML* 23
5 **Zhodnocení dostupného softwaru** 24
  5.1 *XML Browsers* 24
  5.2 *XML Editors* 25
  5.3 *XML Parsers* 25
  5.4 *XSLT Engines* 25
  5.5 *XML Formatters* 26
  5.6 *Publishing Systems* 26
6 **Závěr** 27
A **A** 28
B **B** 29

Kapitola 1

Úvod

V dnešní informační době se hlavně klade důraz na efektivitu zpracování. Existuje veliké množství lepších či horších formátů dat. Bylo by nemyslitelné při současné tendenci sjednocování a centralizace, a s tím související i univerzalizace dostupnosti informací, opomíjet existující standardy a vytvářet si vlastní. Chceme-li poskytnout komukoli informaci, musí být především snadno dostupná a nezkreslená. Vzájemná nekompatibilita jednotlivých textových editorů a webových prohlížečů se stává nepřípustnou. Správnou cestou je patrně vytváření dokumentů a aplikací striktně splňujících dané standardy.

S bouřlivým nástupem Internetu přichází i problém, jak se s daným denně rostoucím kvantem dat proklestit a najít informace relevantní. Tím pádem i minimalizovat informační šum, který se k nám dostává. Nejvíce rozšířený jazyk HTML se ukázal jako nedostačující prostředek k zaznamenávaní informací. A to ne jen z důvodu nekompatibility jednotlivých prohlížečů, jež byla způsobena často nepřesným způsobem implementace jednotlivých standardů a obohacováním jazyka individuálními schopnostmi dle každé firmy. Co je však daleko podstatnější je skutečnost, že nám jazyk HTML neumožňuje výstižně popsat logický obsah dokumentu. Jazyk HTML nám umožňuje v celku jednoduše popsat vizuální stav dokumentu, tím však bohužel jeho možnosti končí. S touto omezeností nastává problém, jak v internetovém moři informací vyhledávat. Dosud jedinou možnou metodou bylo fulltextové vyhledávání, které mnohdy nepřináší kýžený výsledek a je nutné zadat co nejvíce klíčových slov k hledanému pojmu.

Potřebujeme jazyk umožňující co největší popsatelnost logického vyjádření dokumentu, jeho flexibilní vizualizaci a přenositelnost neporušující integritu dat.

1.1 SGML

Jako jednou z raných odpovědí na předešlé požadavky byl značkovací jazyk GML (Generalized Markup Language), jehož autory jsou Charles Goldfarb, Edward Mosher a Raymond Lorie. Jedná se o velice obecný značkovací jazyk, který byl vytvořen právě pro překlenutí nekompatibilit jednotlivých systémů a programů.

Jelikož se tento princip jazyka GML velice osvědčil, začala na jeho základě v 80. letech standardizační organizace ANSI vyvíjet jazyk, umožňující definici vlastních značkovacích jazyků, vhodných pro dané použití. S podobnou ideou návrhu přišlo i sdružení GCA (Graphics Communications Association). Již dříve vytvořilo široce použitelný formátovací jazyk GenCode. V rámci projektu ODA (Open Document Architecture), jehož cílem bylo poskytnout standardní architekturu pro vytváření, předávání, uchovávání a zpracování různorodých dokumentů v elektronické podobě, došlo ke spojení obou těchto aktivit. Což mělo za následek vznik jazyka SGML (Standard Generalized Markup Language), definovaný v roce 1986 ISO normou 8879.

SGML je otevřeným standardem nezávislým na platformách, výrobcích nebo aplikacích. Soubory SGML jsou ukládány jako text ASCII, což zajišťuje jejich použitelnost prakticky na libovolné počítačové platformě. Význam a přínos SGML se objevuje ve chvíli, kdy je dokument označen příslušnými značkovacími příkazy. Definování struktury a vnitřních vztahů v do té doby nestrukturované informaci otevírá zcela nové možnosti zpracování, publikování a opakovaného používání této informace. Např. SGML databáze může obsahovat tisíce označkovaných dokumentů a využitím vyznačené struktury z ní lze na přání publikovat obsahově stejné dokumenty mnoha různých forem.

SGML není značkovací jazyk sám o sobě, ale je to metajazyk sloužící k definování různých značkovacích jazyků. Tato definice se nazývá Document Type Definition (DTD). DTD obsahuje definici elementů a jejich atributů, které lze v dokumentu použít. Tato definice je zapsána díky SGML ve standardizované podobě a je snadno zpracovatelná. Celý postup je možné si představit tak, že SMGL je programovací jazyk a DTD je program v něm napsaný. DTD definuje, jakou může mít dokument syntaxi - jména elementů, jejich atributů, použití, vztahy mezi nimi. Co už však nedefinuje, je význam jednotlivých elementů, který jim musí být přiřazen procesorem, který dokumenty daného formátu zpracovává. Podle Extensible Markup Language (XML) 1.0, W3C Recommendation 10-02-1998[1] je procesorem nazýván takový softwarový modul (v dalším textu pro něj budeme používat termín XML procesor, či jen procesor), který je používán pro čtení XML dokumentů a poskytuje přístup k jejich obsahu a struktuře (Pozn.: často se používá též termín parser). Předpokládá se, že XML procesor dělá práci pro jiný modul, nazývaný aplikace. Specifikace popisuje požadované vlastnosti XML procesoru, jak má číst data a tato předávat aplikaci, která je dále zpracovává. Příkladem procesoru budiž např. internetový browser, který HTML tag

```
<A HREF>
```

rozpozná jako link na nějaký zdroj a podle toho jej zobrazí a chová se k němu. Dodejme, že HTML je definováno právě pomocí DTD SGML. Autor dokumentu tedy musí vědět, jaký význam má daný element - jak, kde a s jakými atributy jej použít - a prohlížeč zase musí vědět, jak jej zobrazit a pracovat s ním. To všechno se může dovědět právě v DTD. Aby prohlížeč věděl, jaký DTD má daný dokument použít, začíná každý dokument tzv. prologem, který určuje, podle jakého DTD byl dokument vytvořen. Příkladem prologu budiž např.

```
<!DOCTYPE book SYSTEM "book.DTD">
```

Je tedy možné vytvářet SGML-dokumenty (resp. XML-dokumenty - jak bude níže vysvětleno, XML má k SGML velmi blízko), které vyhovují určitému DTD, např. je možné nadefinovat DTD pro dokumenty určitého typu, které budou všeobecně přijímány, a jednotlivé skupiny uživatelů tak budou mít svůj jazyk, resp. jeho definici pro popis dokumentů. Mohou tak vzniknout specifické DTD, resp. jazyky, jako např. MathML pro matematiku, ebXML, cXML pro e-commerce, CML pro chemii či jakýkoli obor lidské činnosti. Jednou z prvních implementací XML byl tzv. Shakespeare Markup, DTD pro popis Shakespearových dramat.

Se vzrůstající technologickou vyspělostí Internetu přibyly i nároky na vizuální i formální podobu webových stránek. Hlavně pro svoji jednoduchost si v poslední době získal oblibu jazyk HTML. Nenáročné reprezentační stránky si dnes může vytvořit

1. http://www.w3.org/TR/1998/REC-xml-19980210

takřka kdokoli s alespoň základními znalostmi psaní textu. Nabízí se použití nemalého množství WYSIWYG editorů, kteřé programátorskou část za nás udělají sami. S tím dosáhl jazyk HTML svého vrcholu, ovšem i hranic svých možností. Jednoduchost, která byla jeho hlavní výhodou, s sebou přináší jen velice omezené množství použitelných značek, jež dnes již zdaleka nestačí.

Naproti tomu SGML by se mohlo jevit, díky své rozsáhlosti a formovatelnosti, jako ten pravý nástroj pro předávání dat. Umožňuje i používání silně specifických značek, které dokáží velice dobře vymezit logický význam textu a tím i zefektivnit následné vyhledávání. Bohužel, právě pro svoji komplexnost je implementace jazyka SGML velice náročná. V praxi stejně dochází k použití jen malá část jeho možností. Z toho důvodu byla vybrána právě ta relevantní podmnožina, která umožňuje zdárnou tvorbu webových aplikací. A je plně využitelná rostoucím technologickým potenciálem webu nadcházejících let. Uplatnění nachází i v mnoha jiných odvětvích informační komunikace.

1.2 Historie

XML navazuje na principy a konvence jazyků SGML a HTML, vytváří jednoduchý a velice účinný mechanismus pro ukládání, šíření a zpracování informací. Hlavním podkladem pro vznik bylo SGML, ale zdědilo i některé vlastnosti HTML, takže obsahuje dodatečné vlastnosti k jeho použití na Internetu.

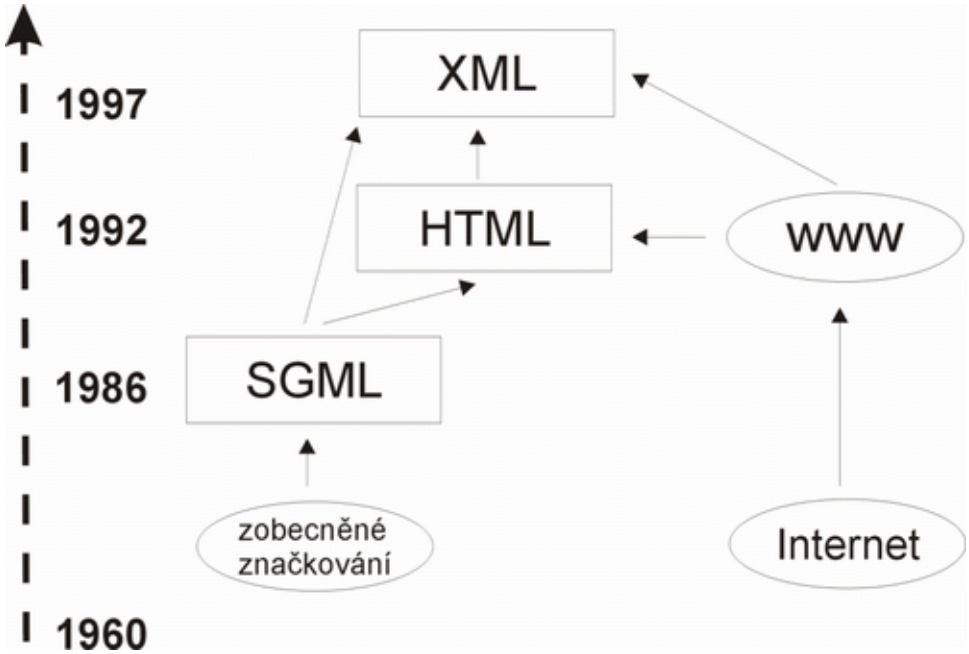
Koncepce obecného značkování vnikla počátkem 60. let, ale nedošlo k jejímu přílišnému rozšíření, dokud nebylo vytvořeno SGML schválené organizací ISO v roce 1986 a v následujících letech se prakticky nezměnilo. Je to určitě známka velké robustnosti SGML již od samého počátku - uběhlo deset let, než nastala potřeba k provedení drobných úprav. Ve skutečnosti byla specifikace SGML až příliš pokročilá, takže některé z jejích vlastností nejsou dodnes zcela využity. Síla SGML byla jeho největší nevýhodou. Aplikace schopné práce s daty SGML se tak ukázaly jako nekompletní, chybující, dlouho vyvíjené a přitom dražší, než proprietální řešení bez SGML.

Tlak na zjednodušení SGML se vhodně kryl s tlakem na vylepšení HTML a konsorcium W3C[2] vidělo jako řešení vytvoření nového jazyka pro zaplnění této mezery. V roce 1996 bylo vytyčeno deset základních cílů, které by měl nový jazyk splňovat. Formálně byla verze 1.0 jazyka XML uvedena 1. února 1998

1.3 Čas pro XML

Nový jazyk vzniklý jako podmnožina SGML, jež si zachovává většinu jeho vlastností, se nazývá XML (eXtensible Markup Language). XML má možnosti tvorby vlastních DTD i jednotlivých značek použitých v dokumentu. Na rozdíl od SGML podléhá jistým omezením ohledně formálního vzhledu daných značek (délka, oddělovače, speciální znaky, použité parametry atd.). Jelikož se jedná o nový standard, nepodléhá problémům s použitím jiných jazyků, jako tomu bylo kupříkladu s HTML. Můžeme si zde nadefinovat vlastní názvy jednotlivých značek, a to dokonce i v libovolném jazyku. Navíc je syntaxe zápisu v XML oproti SGML i HTML poměrně přísná, což umožňuje snazší a jednodušší implementaci na straně aplikací. XML se výborně hodí pro použití v oblasti textových dokumentů a stejně jako SGML již našlo nemalé použití; za zmínku stojí dokumentace v americké obraně a mnoho jiných. Díky schopnosti ukládat logické

2. http://www.w3.org



Obrázek 1.1: Historie značkovacích jazyků

závislosti nachází XML své uplatnění i v mnoha dalších odvětvích datové komunikace. Není v něm problémem uložit údaje z účetnictví, objednávky, faktury a mnoho jiných.

Kapitola 2

XML syntaxe

2.1 Tvorba DTD

DTD (Document Type Definition): tato definice typu dokumentu obsahuje instrukce, které určují pravidla pro konkrétní typ dokumentu. DTD je velice užitečný nástroj, který nám umožní hlídat, zda mají naše dokumenty správnou strukturu. Mezi jedno z nejznámější DTD patří například DTD DocBook, které definuje elemety a atributy vhodné pro značkování dokumentů.

DTD je používáno parserem, který kontroluje, zda tagy v dokumentu odpovídají předem určeným pravidlům pro strukturu dokumentu. To je jedna z výhod, které nám DTD nabízí, tj. kontrola struktury. Další výhodou je pak možnost použití různých transformačních programů, které podle daného DTD převedou dokument z jednoho formátu do druhého.

Na začátku XML souboru musíme určit, jaké DTD bude použito. Obvykle je DTD uloženo v samostatném souboru. Jeho načtení zpřístupníme deklarací:

```
<!DOCTYPE kořenový element SYSTEM ”URL”>
```

URL nám udává, kde je dané DTD uloženo a kořenový element je jméno elementu, ve kterém bude obsažen celý dokument. Klíčové slovo SYSTEM bůže být nahrazeno slovem PUBLIC, což je veřejný identifikátor. Tato metoda identifikace vzdálené informace je flexibilnější, neboť nám nabízí více informací o obsahu datového souboru. Samotné DTD obsahuje deklarace čtyř typů:

- deklarace elementů
- deklarace atributů
- deklarace entit
- deklarace notací

Podrobnější informace o tvorbě DTD lze nalézt v knize: XML pro každého [Ptáček]

2.2 Základy XML

Každý XML dokument se skládá z elementů, které jsou do sebe navzájem vnořené. Elementy se v textu vyznačují pomocí tagů.

```
<para>nějaký text</para>
```

Názvy tagů se zapisují mezi znaky '<' a '>'. Ukončovací tag má před svým názvem ještě znak '/', aby se snadno odlišil od počátečního. Některé elementy nemusejí mít žádný obsah, zápis tagů lze zjednodušit použitím pouze jediného, ukončeného znakem '/'.

```
<br></br> odpovídá <br/>
```

V XML musí být pro každý počáteční tag i tag ukončovací, rovněž nelze v dokumentu elementy křížit.

```
<b>takhle se to <i>dělat</b> nemůže</i>
```

U každého počátečního tagu můžeme použít ještě atributy,obvykle k upřesnění významu dokumentu.

```
<kniha jazyk="čeština">
```

Každý XML dokument musí být celý obsažen v jednom elementu. Nelze, aby se v dokumentu vyskytoval neotagovaný řetězec. Ukázka malého XML dokumentu:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!DOCTYPE lednice SYSTEM "lednice.dtd">
<lednice>
   <popis>Seznam potravin v˜ledničce</popis>
   <odmražení provedeno="1.1.2001"/>
   <seznam přihrádek>
      <přihrádka>máslo</přihrádka>
      <přihrádka>lahváč</přihrádka>
      <přihrádka>lahváč</přihrádka>
   </seznam přihrádek>
</lednice>
```

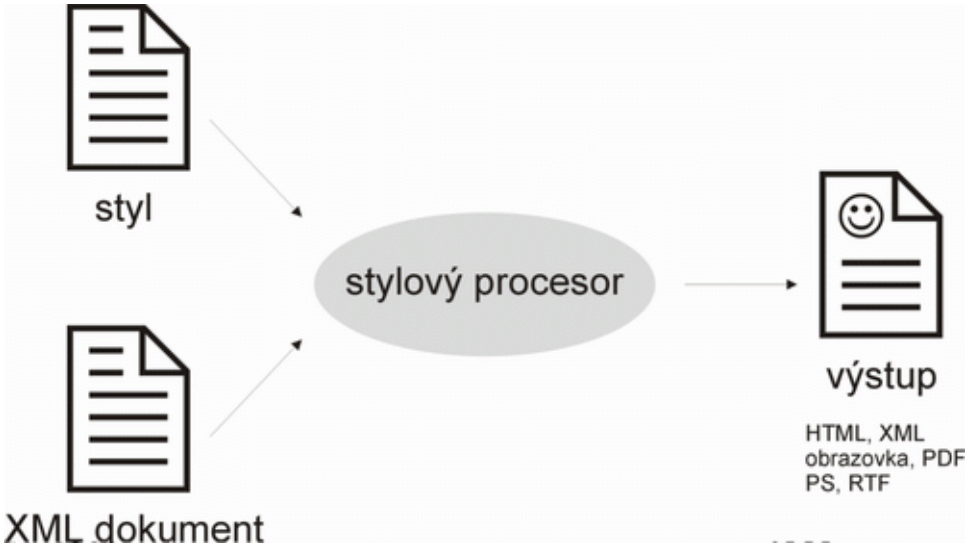
Mnohem více o syntaxi XML se lze dozvědět z knihy: The XML Companion [Bradley]

2.3 Stylové jazyky

Ovlivnit zobrazení jednotlivých značek je možné pomocí stylu (tj. souborů pravidel nebo příkazů, které definují, jak se dokument převede do jiného formátu). Dnes existuje několik stylových jazyků - mezi nejznámější patří asi CSS (Cascading Style Sheets). Pro XML je ale mnohem vhodnější a výkonnější jazyk XSL (eXtensible Stylesheet Language), který dovoluje před samotným formátováním dokument různě upravit a transformovat. Další možností je ještě například DSSSL (Dokument Style Semantics and Specification Language), vyvinutý pro potřeby SGML.

Standard XSL se rozděluje na dvě části:

1. XSLT (XSL Transformation) - obsahuje nástroje pro transformaci XML dokumentu. XSLT lze použít pro transformaci XML dokumentu do XML dokumentu s jinou strukturou, do HTML dokumentu nebo třeba textového formátu. Výstupem může být i dokument, skládající se z formátovacích objektů (viz. níže) - ten pak může být pomocí speciálních programů zformátován (např. do PDF) a zobrazen nebo vytištěn.



Obrázek 2.1: Funkce stylového procesoru.

2. XSLFO (XSL Formated Objects) definuje tzv. formátovací objekty. Ty popisují vzhled stránky (rozměry, způsob zarovnání, použitá písma, barvy ...) a její jednotlivé komponenty, čímž umožňují dokument velmi kvalitně zformátovat pro tisk nebo prezentaci na obrazovce. Formátovací objekty mají syntaxi XML, používají jmenný prostor (většinou značený fo)[1], kořenový element je root. Element fo:root pak obsahuje element fo:layout-master-set (definice layoutu jednotlivých stránek - rozměry, okraje ...) a element fo:page-sequence (obsah jednotlivých stránek). Podrobnějších informace jsou obsaženy na stránkách konsorcia W3C viz. [Clark]

1. http://www.w3.org/1999/XSL/Format

Kapitola 3

Formát, který vydrží

Chceme, aby náš dokument,když už si dáváme práci s jeho vytvořením, byl široce přenositelný, snadno transformovatelný a hlavně jednoduše dostupný. Samozřejmě pro psaní dokumentů bylo již vytvořeno mnoho značně vyspělých nástrojů. Ať bychom se rozhodli pro formát RTF,či LaTeX. Přes mnoho nesporných výhod mají jednu nemalou nevýhodu, a tou je neoddělitelnost vzhledu od logického obsahu. Pokud bychom se rozhodli daný dokument publikovat v jiném formátu či jiném vzhledu, znamenalo by to kompletní vizuální zásah. Bohužel by se jednalo ve značné míře o zásah manuální. Právě tady se nám nabízí jazyk XML, který umožňuje obsahové oddělení od vizuálního. Ovšem XML nekončí svými možnostmi jen u tvorby dokumentů. Není třeba se obávat ztráty pohodlného WYSIWYG rozhraní, jako třeba u Wordu či LyXu, neboť již dnes je na trhu několik slibných editorů a jejich nárůst lze jen očekávat.

Formát XML je značně obecný. Z tohoto faktu vyplývá několik ve své podstatě odlišných způsobů tvorby dokumentů.

3.1 Na rozcestí

V XML si můžeme nejen stanovit způsob výsledného vzhledu, ale i samotný proces zápisu dat v dokumentu. Tím je míněna možnost vytvoření vlastního DTD. To nám přináší značnou výhodu, pokud jsme třeba literárními spisovateli, či tvoříme větší množství publikací a chceme jim dát jednotný vzhled úměrný našim představám. Slastná představa těchto možností bledne s nemalým časovým zatížením při tvorbě vlastního DTD i stylů pro vzhled či transformace. Pokud zrovna nemáme ambice a schopnosti Jiřího Kostka, nabízí se nám použití již vytvořených standardů. Pro tuto volbu svědčí i fakt, že pokud vytvoříme dokument v DTD, kterému budeme rozumět jen my, ztrácí se tím okamžikem veškerá jeho přenositelnost, transformace i pozdější úprava někým jiným (v horším případě i námi samotnými). Například některé nakladatelství může mít požadavek, aby všechny jeho publikace měly jednotný vzhled, vytvoří proto daný styl a požaduje po autorech, aby jejich dokumenty byly s tímto stylem kompatibilní. Či se nám líbí již někým vytvořený styl a hodláme ho aplikovat na svých dokumentech, nemluvíce o faktu, že se prostě nechceme zabývat vytvářením vlastního stylu. Důvodem může být ať již naše lenost či časová vytíženost. Nebo také skutečnost, že na rozdíl od psaní v DTD XML pro tvorbu dokumentů, které je vcelku nenáročně, předpokládá vytváření stylů XSL a XSLT nemalé znalosti. Pokud bychom z jakéhokoli důvodu chtěli použít vlastní množinu pravidel pro XML, tedy vlastní DTD, a zároveň bychom se nechtěli uzavřít světu a jeho možnostem, nezbývá, než vytvořit buď nástroj či styl pro převod na nějaký standart.

3.2 DocBook

Pokud jsme se rozhodli použít XML pro vytvoření dokumentu a jeho následnou publikaci s použitím nějakého standardního DTD, nabízí se nám právě DocBook.Což je dnes asi druhá nejpoužívanější aplikace SGML/XML, hned za jazykem HTML. Vznikl v roce 1991 jako formát založený na SGML, určený především pro výměnu unixové dokumentace. V 90. letech se jeho použití rozšířilo i mezi mnoho velkých firem, které se podílely i na jeho vývoji (např. Novell, HP, Digital, Fujicu a další).

V roce 1999 se péče o DocBook přesunula do sdružení OASIS[1]. Původní verze byla založena na SGML, dnes již je k dostání verze i pro XML[2]. DocBook se od svých počátků vyvinul do systému, který se hodí zejména pro tvorbu počítačové dokumentace. Bez problému ho však lze použít pro zápis libovolných knih a článků. V DocBooku je například vytvořena tato práce, dokumentace k operačnímu systému FreeBSD, ke scriptovacímu jazyku PHP a dokonce také i Linux přechází na DocBook. Používají ho i velká počítačová nakladatelství jako O'Reilly[3].

Výhodou je, že mnoho editorů a nástrojů pro práci s XML v sobě přímo zahrnuje jeho podporu. K DocBooku existuje dokumentace DocBook: The Definitive Guide [Walsh], který je k dispozici i v elektronické podobě[4].

3.3 Validace

Uložení dokumentu v XML nese výhodu možnosti zachytit pomocí elementů strukturu dokumentu. Parser nám umožní kontrolu, zda je dokument správně strukturovaný (well-formed). Lepší parsery kontrolují, zda dokument odpovídá danému DTD, pak se takovýto dokument nazývá validní. Parser využijeme zejména v situaci, kdy chceme ověřit, zda náš dokument neobsahuje nějaké syntaktické a strukturní chyby. Pokud chceme, aby následná transformace XML dokumenty proběhla korektně, je nezbytné, aby takovýto dokument byl validní. V dnešní době existuje mnoho různých druhů lepších či horších parserů. Na samotném vrcholu stojí pravděpodobně parser SP Příloha A.

Pro validaci XML je třeba ještě nastavit některé systémové proměnné,protože jinak bude parser považovat vaše soubory za SGML a bude hlásit chyby. Nastavit je tedy třeba proměnné:

```
SP_CHARSET_FIXED=YES
SP_ENCODING=XML
SGML_CATALOG_FILES=c:\Program Files\sp1_3_4\pubtext\xml.soc
```

validace se provádí příkazem:

```
nsgmls -wxml -s [jmeno_xml_souboru]
```

Je důležité nezapomenout na přepínač *-wxml*, který říká, že se má validovat jako XML. Upozorníme ještě na problém, který může nastat, pokud budeme chtít náhodou validovat dokument v kódování cp1250. Nastavením `SP_ENCODING` na XML říkáme, že SP se pokusí kódování souborů zjistit podle pravidel, která pro to používá XML. SP

1. http://www.oasis-open.org
2. http://www.docbook.org
3. http://www.oreilly.com
4. http://www.docbook.org

pozná a spolupracuje s UTF-8 a iso-8859-2, avšak některé znaky z cp1250 dělají problémy. Můžeme tedy změnit kódování našeho souboru, nebo nastavit `SP_ENCODING` na "windows". Stejně postupujeme, pokud máme problémy s jiným kódováním.

Pokud máme XML dokument, ke kterému neexistuje DTD, můžeme jej pomocí nsgmls také zkontrolovat. Stačí použít parametr *`-wno-valid`* a u dokumentu pouze zkontrolujeme, zda je správně strukturovaný.

Kapitola 4

Realizace vizuálních podob

4.1 Transformace do PDF a PS

Chceme-li náš výtvor prezentovat v tištěné podobě, je nutné ho z XML kódu transformací převést do formátu vhodného pro tisk. Nejlépe k tomu poslouží formát PDF (Portable Document Format) nebo PS (PostScipt). Pro takovouto transformaci existují dva druhy postupu. První je použití XSLT transformací do FO objektů a následný převod do PDF. Na jakoby opačné straně pojetí transformace je převod do zdrojového kódu LaTeXu na překlad do PDF či PS. Oba dva způsoby mají bezesporu své velké výhody a nevýhody.

LaTeX:

+ prakticky neomezená manipulovatelnost s výsledným vzhledem;
+ možnost použití METAFONTU (různá loga atd.);
+ aplikace již vytvořených cls stylů;
+ automatické generování obsahu, literatury a rejstříku;
- alespoň minimální znalost LaTeXu;
- ne příliš dokonalé styly pro převod do LaTeXu;
- poněkud odlišná filozofie od pojetí XML.

FO:

+ možnost použití XSL stylů;
+ jednoduchost transformace;
- alespoň minimální znalost XSL;
- v současné době malá technická podpora pro překlad;
- malá rozmanitost XSL stylů.

Osobně jsem si zvolil metodu transformace přes LaTeX, a to z následujících důvodů. Největší problém je dnes asi právě s absencí rozmanitých XSL stylů pro DocBook, jediné opravdu kvalitní jsou od Normana Walshe Příloha A. Tímto jsme, pokud se neodhodláme dané styly změnit ke svým představám, odsouzení smířit s výsledným vzhledem. Pokud se alespoň trochu vyznáme v používaní LaTeXu, odpadá nutnost pronikat do způsobů vytváření vlastních XSL stylů.

4.2 Transformace do LaTeXu

Transformace XML dokumentu do zdrojového kódu LaTeXu je sice poněkud komplikovanější, ale pokud už se nám podaří transformace do zdrojového kódu LaTeXu, dá se říci, že jsme "za vodou". Následný překlad pomocí cslatexu či pdflatexu je poněkud z jiné oblasti než XML, a proto mu nebude dále věnována pozornost.

K první části transformace použijeme XSLT nástroj J. Clarka XT Příloha A a XML styly pro LaTeX Příloha A. Nejprve je třeba manuálního zásahu do XSL stylů vzhledem k potřebě lokalizace pro český jazyk. K těmto úpravám je nutná jen zběžná znalost

standardu XSL, což je jedna z hlavních předností tohoto způsobu transformace, ovšem na druhou srtanu je třeba se vyznat v psaní dokumentu v prostředí LaTeXu.

v souboru `docbook.xsl`: 17:46 (řádek:sloupec)

```
<xsl:output method="text" encoding="ISO-8859-1" indent="yes"/>
```

na:

```
<xsl:output method="text" encoding="ISO-8859-2" indent="yes"/>
```

`labelid.mod.xsl`: 59 zakomentujeme přidání prázdného řádku (způsoboval problémy při zalomování poznámek)

```
<!-- <xsl:text>&#10;</xsl:text> -->
```

`toc_lot.mod.xsl`: 18 zakomentování (nechceme-li, aby byl vypsán seznam obrázků)

```
<!-- <xsl:text>\listoffigures&#10;</xsl:text> -->
```

postupně změníme řádky osahující:

```
<xsl:text>\documentclass[french,english,a4paper,10pt,final]
{scrreprt}&#10;</xsl:text>
<xsl:text>\else&#10;</xsl:text>
<xsl:text>\documentclass[pdftex,french,english,a4paper,10pt,final]
{scrreprt}&#10;</xsl:text>
```

na:

```
<xsl:text>\documentclass[a4paper,10pt,final]{scrreprt}&#10;</xsl:text>
<xsl:text>\else&#10;</xsl:text>
<xsl:text>\documentclass[a4paper,10pt,final]{scrreprt}&#10;</xsl:text>
```

a obdobně:

```
\ifx\pdfoutput\undefined
\documentclass[french,english,a4paper,10pt,twoside,twocolumn]{article}
\else
\documentclass[pdftex,french,english,a4paper,10pt,twoside,twocolumn]{ar
\fi
```

na:

```
\ifx\pdfoutput\undefined
\documentclass[a4paper,10pt,final]{scrreprt}
\else
\documentclass[pdftex,a4paper,10pt,final]{scrreprt}
\fi
```

dále přidáme řádek obsahující podporu pro češtinu, nejlépe pod uvedené řádky:

```
<xsl:text>% Most Common Packages (? need guru hint)&#10;</xsl:text>
<xsl:text>%-----------------------------------------&#10;</xsl:text>

<xsl:text>\usepackage{czech}         &#10;</xsl:text>
```

`preface.mod.xsl`: 34 zakomentujeme a přidáme řádek (korektní zobrazení názvu předmluvy a začlenění do obsahu)

```
<!-- <xsl:text>&#10;{\sc </xsl:text><xsl:apply-templates select="."
 mode="title.ref"/><xsl:text>}&#10;</xsl:text> -->
<xsl:text>&#10;\chapter*{\prefacename}&#10;</xsl:text>
```

na řádku 45 zakomentujeme vložení odstavce a nahradíme novým řádkem (hezčí výsledné zobrazení)

```
<!-- <xsl:text>&#10;\paragraph*{}&#10;</xsl:text>This is a fixme !!-->
<xsl:text>&#10;</xsl:text>
```

`vars.mod.xsl`: 82:45

```
<xsl:variable name="latex.figure.position">[tbp]</xsl:variable>
```

změníme na: (LaTeX se následně nesnaží sám vhodně umístit danou oblast, ale ponechá ji na stejném místě jako v zdrojovém textu)

```
<xsl:variable name="latex.figure.position">[h]</xsl:variable>
```

dále změníme některá nastavení: 158:38

```
<xsl:variable name="latex.use.babel">1</xsl:variable>
```

na: (vypne multijazykovou podporu)

```
<xsl:variable name="latex.use.babel">0</xsl:variable>
```

159:42

```
<xsl:variable name="latex.use.isolatin1">1</xsl:variable>
```

na: (vypne přidání balíku pro jazykovou sadu iso-8859-1)

```
<xsl:variable name="latex.use.isolatin1">0</xsl:variable>
```

205:41

```
<xsl:variable name="latex.math.support">1</xsl:variable>
```

na: [vypne podporu pro matematické značení (pokud zrovna v textu nepoužíváme žádné složité matematické výrazy, je zcela zbytečná a pouze zdržuje proces překladu)]

```
<xsl:variable name="latex.math.support">0</xsl:variable>
```

`book.mod.xsl`: 18:11

```
<xsl:text> \footnote{</xsl:text><xsl:value-of select="."/><xsl:text>}</
```

odstraníme mezeru (opět pro zkrásnění zobrazení indexu poznámek)

```
<xsl:text>\footnote{</xsl:text><xsl:value-of select="."/><xsl:text>}</x
```

`labelid.mod.xsl`: 64

```
<xsl:text>}{}&#10;</xsl:text>
```

nahradíme (lepší zobrazení indexu poznámky)

```
<xsl:text>}{}</xsl:text>
```

`inline.mod.xsl`: 26:14, 35:14, 44:14, 53:14, 62:14 odstraníme mezery v následujících řádcích pro korektní zvýrazňování:

```
<xsl:text> {\tt </xsl:text>
<xsl:text> {\bf </xsl:text>
<xsl:text> {\em </xsl:text>
<xsl:text> {\bf {\tt </xsl:text>
<xsl:text> {\em {\tt </xsl:text>
```

na

```
<xsl:text>{\tt </xsl:text>
<xsl:text>{\bf </xsl:text>
<xsl:text>{\em </xsl:text>
<xsl:text>{\bf {\tt </xsl:text>
<xsl:text>{\em {\tt </xsl:text>
```

následují úpravy XSL pro použití cls stylu fithesis, používaného k sazbě diplomových a nově i bakalářských prací.

`vars.mod.xsl`: přidáme kód pro zapnutí volitelné podpory stylu fithesis a indikaci používání koncovek rodu ženského v prohlášení

```
            <!-- fithesis -->
<xsl:variable name="latex.fithesis.support">1</xsl:variable>
<xsl:variable name="latex.fithesis.woman">false</xsl:variable>
```

`book.mod.xsl`: přidáme podporu pro včlenění titulní strany a čestného prohlášení

```
<!-- fithesis -->
<xsl:if test="$latex.fithesis.support=1">
<xsl:text>%-----------------------------------------&#10;</xsl:text>
<xsl:text>% Useing fithesis &#10;</xsl:text>
<xsl:text>%-----------------------------------------&#10;</xsl:text>
   <xsl:text>\FrontMatter&#10;</xsl:text>
```

```
   <xsl:text>\ThesisTitlePage&#10;</xsl:text>
   <xsl:text>\begin{ThesisDeclaration}&#10;</xsl:text>
   <xsl:text>\woman</xsl:text>
   <xsl:value-of select="$latex.fithesis.woman"/>
   <xsl:text>&#10;</xsl:text>
   <xsl:text>\DeclarationText&#10;</xsl:text>
   <xsl:text>\end{ThesisDeclaration}&#10;</xsl:text>
</xsl:if>
```

external cover page obalíme tagem, který se uplatní pouze při vypnuté podpoře pro styl fithesis

```
<xsl:if test="$latex.fithesis.support=0">

<!-- Include external Cover page if specified -->
<xsl:text>&#10;\InputIfFileExists{</xsl:text><xsl:value-of
select="$latex.titlepage.file"/>
<xsl:text>}{\typeout{WARNING: Using cover page</xsl:text>
<xsl:value-of select="$latex.titlepage.file"/>
<xsl:text>}}</xsl:text>
<xsl:text>{\maketitle}&#10;</xsl:text>
<xsl:call-template name="label.id"/>
</xsl:if>
```

`toc_lot.mod.xsl`: přidáme do templatu toc následující řádky, pro obnovení arabského číslování

```
<!-- fitimes -->
<xsl:if test="$latex.fithesis.support=1">
<xsl:text>\MainMatter&#10;</xsl:text>
</xsl:if>
```

`preamble.mod.xsl`: do templatu: generate.latex.book.preamble přidáme kód pro provedení začlenění potřebých příkazů stylu fithesis na řádek 125

```
<!-- fithesis -->
   <xsl:call-template name="latex.fithesis"/>
```

v příkazech pro vložení stylu výsledného LaTeXového dokumentu změníme atributy na

```
\documentclass[oneside,11pt,final]{fithesis}
```

a

```
\documentclass[pdftex,oneside,11pt,final]{fithesis}
```

změníme podtrhávání plovoucích objektů figures na řádku: 338

```
<xsl:text> \floatstyle{ruled}&#10;</xsl:text>
```

na

```
<xsl:text> \floatstyle{plain}&#10;</xsl:text>
```

ke konci souboru přidáme kód pro vložení fithesis hlavičky:

```
<!--##################################################
 |    latex.fithesis
 |- ##################################################
 |
 + ################################################## -->

<doc:template name="latex.fithesis" xmlns="">
<refpurpose> Declared fithesis cls </refpurpose>
<refdescription>
</refdescription>
<refreturn>
</refreturn>
</doc:template>

<xsl:template name="latex.fithesis">
<xsl:if test="$latex.fithesis.support=1">
<xsl:text>%-----------------------------------------&#10;</xsl:text>
<xsl:text>% Load fithesis param &#10;</xsl:text>
<xsl:text>%-----------------------------------------&#10;</xsl:text>
   <xsl:text>\thesistitle{</xsl:text>
   <xsl:apply-templates select="bookinfo/title" mode="titlepage.mode"/>
   <xsl:text>}&#10;</xsl:text>
   <xsl:text>\thesisstudent{</xsl:text>
   <xsl:apply-templates select="bookinfo/author/firstname" mode="titlep
   <xsl:text> </xsl:text>
   <xsl:apply-templates select="bookinfo/author/surname" mode="titlepag
   <xsl:text>}&#10;</xsl:text>
   <xsl:text>\thesisyear{</xsl:text>
   <xsl:apply-templates select="bookinfo/pubdate" mode="titlepage.mode"
   <xsl:text>}&#10;</xsl:text>
</xsl:if>
</xsl:template>
```

`docbook.xsl`: následující umožní používání několika přidaných tagů (<bs/>, <small>, </small>)

```
<!-- fithesis -->
<xsl:include href="raven.xsl"/>
```

DTD těchto tagů je třeba přidat do záhlaví XML souboru např.:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
"file:\\\c:\\sgml\\docbook\\docbookx.dtd" [
<!ELEMENT bs EMPTY>
<!ELEMENT small ANY>
]>
```

Tímto jsme dokončili nezbytnou úpravu XSL stylů a nyní je možno je používat pro transformaci do cls stylu fithesis, používaného pro sazbu diplomových prací.

Před započetím vlastní transformace je nutné ještě XML dokument podrobit řádné validaci, viz.: Oddíl 3.3. K transformaci XML dokumentu do LaTeXu je možné použít všestranného transformačního nástroje XT Příloha A. Originální distribuce[1] bohužel nepodporuje kódování našeho rodného jazyka, naštěstí je tu k dispozici verze, kterou upravil Jiří Kosek[2]. K celé proceduře je kromě XT zapotřebí ještě JRE (Java Runtime Environment) Příloha A, zmíněné XSL styly a samozřejmě LaTeX Příloha A. Jako parametry udáme cestu k programu XT, jméno zdrojového souboru XML, cestu k XSL stylům a jméno výstupního souboru.

```
java -cp c:\sgml\xt\xt.jar com.jclark.xsl.sax.Driver [soubor_xml]
c:\sgml\docbook\xsl\latex\docbook.xsl [soubor_tex]
```

Výsledek přeložíme buď příkazem pdfcslatex nebo cslatex. Tímto je celý proces u konce.

4.3 Transformace do RTF

I tato možnost stojí za zmínku, neboť i ona skýtá jisté výhody. Pro tuto transformaci vytvořil Norman Walse DSSSL styly, pomocí kterých je XML dokument v DocBooku převeden do RTF. Bohužel jsou styly napsané v SGML formátu, což značně znesnadňuje jejich případnou úpravu, neboť formát SGML, jak je zmíněno v úvodu, trpí hlavně svou komplexností. Obecně z těchto a podobných problémů standard SGML opouští a přechází se přece jenom k jednoduššímu standardu XML. Nicméně James Clark[3] vytvořil vynikající nástroj na převod do formátu RTF, využívající právě tyto styly.

RTF:

+ snadná úprava v nějakém WYSWYG editoru (Word);
- malá rozmanitost DSSSL stylů.
- ztráta typografické preciznosti.

Právě z posledního důvodu není transformace do RTF příliš vhodná. Nicméně zde existuje postup, jak tento převod uskutečnit. Jeho použití jsem vyzkoušel jak v prostředí Linuxu Red Hat, tak i pod Windows ME. Následující návod je v prostředí Windows, avšak pod operačním systémem Linux je postup takřka stejný.

1. Nejprve si stáhneme DTD pro DocBook. Příloha A
2. Dále jsou nutné DSSSL styly pro DocBook. Příloha A
3. A aplikace Jade 1.2.1 (v distribuci Linuxu je k dispozici OpenJade).Příloha A
4. Rozbalíme kupříkladu do adresářů: DTD `c:\sgml\DocBook` potom ještě DSSSL styly do `c:\sgml\DSSSL\docbook` a Jade `c:\sgml\Jade`
5. Zvalidujeme pomocí SP viz kapitola o validaci. Oddíl 3.3
6. set `SGML_CATALOG_FILES=c:\sgml\Jade\catalog;c:\sgml\DocBook\isoent.cat`

1. http://www.jclark.com/xml/xt.html
2. http://www.kosek.cz
3. http://www.jclark.com

7. Důležité je odstranit z proměnné `SGML_CATALOG_FILES` položku, která by působila chybové hlášky při překladu: `c:\Program Files\sp1_3_4\pubtext\xml.soc`

Spustíme příkazem:

```
jade -t rtf -d c:\sgml\dsssl\docbook\print\docbook.dsl
c:\sgml\Jade\xml.dcl [soubor_xml]
```

Výsledkem je pak soubor v RTF.

4.4 Transformace do HTML

Převod do html je v celku oproti jiným transformacím nejméně problémový. I pro DocBook existují XSL styly. K této transformaci lze použít buď XSLT procesor SAXON Příloha A, který nicméně akceptuje pouze kódování UTF-8. Další alternativou je použít upravený XSLT procesor XT Příloha A, který pro axceptování kódování windows-1250 a iso-8859-2 upravil Jiří Kosek. Celou transformaci provedeme následujícím příkazem:

```
java -cp c:\sgml\xt\xt.jar com.jclark.xsl.sax.Driver [soubor_xml]
c:\sgml\DocBook\xsl\html\docbook.xsl [soubor_html]
```

Kapitola 5

Zhodnocení dostupného softwaru

5.1 XML Browsers

Nezbytnou součástí pro používání XML formátu jsou borwsery, neboli prohlížeče, jež umožní zobrazení XML dokumentu. Jelikož celá filosofie používání XML standardu směřuje k jeho rozšíření jako nástupce HTML a vlastně jako obecného formátu dokumentu ve všech odvětvích, očekává se jeho implementace v nejrozšířenějších aplikacích manipulujících s dokumenty. Tedy můžeme doufat, že se tak brzy stane u i našich oblíbených editorů a neméně tak i u webových prohlížečů. Proto asi nemá velikou cenu zabývat se všemi možnými experimentálními XML prohlížeči, které jsou tak či tak určeny ve většině pouze pro prohlížení formátů XML, takže jenom s nimi bychom se v dnešní době neobešli. Koplexnějším řešením jsou dnes nejvíce rozšířené prohlížeče, které již mají v sobě podporu pro formát XML zahrnutu. Většina trhu webových prohlížečů se dělí mezi tři produkty:

Microsoft Internet Explorer 6
verze: 6.0
výrobce: Microsoft
licence: zdarma
platformy: Windows, Mac

Většina internetové populace používá k surfování po webu právě aplikaci firmy Microsoft: IE, z pohledu implementace XML je tento produkt na úrovni lepšího průměru. Umožňuje i podporu zobrazování pomocí CSS stylů. Podpora pro XSL(T) je založena na starším konceptu XSL. Podporuje také jmenné prostory a VML (Vector Markup Language).

Netscape
verze: 6.01
výrobce: Netscape Communications Corporation
licence: zdarma
platformy: Linux/Unix, Win32, MacPPC

Asi největší konkurent firmy MS. Vývoj jádra samotného Netscapeu se zastavil ve verzi 4.79, následující verze jsou založeny na jádru Mozilly. Tato verze není příliš vydařená, obsahuje mnoho chyb. Netscape podporuje XML, CSS, DOM, jmenné prostory a Xlink.

Mozilla
verze: 0.8.1
výrobce: Mozilla
licence: zdarma
platformy: Linux, Win32

Celá Mozilla je postavena na kódu Open Source, veřejně dostupného z mozilla.org. Obsahuje podporu pro HTML 4.0, XML, CSS, XSLT, DOM jistou částí i pro jmenné prostory. Tato verze má asi nejlépe implementované standardy XML a XSL.

Opera
verze: 5.11
výrobce: Opera Software AS.
licence: zdarma
platformy: Windows, EPOC, BeOS, Linux

Jedná se o velice slibný projekt, který překvapil jak svou rychlostí, tak i svými zobrazovacími schopnostmi. Podporuje CSS, CSS2, WAP-WML a XML. Do budoucnosti této aplikace jsou vkládány velké naděje a již dnes je velkým konkurentem pro největší výrobce webových prohlížečů.

Mezi další webové prohlížeče podporující XML patří: eXeMeL, XML Viewer, JUMBO, InDelv XML browser, DocZilla, Link, HyBrick. Více informací na xmlsoftware[1].

5.2 XML Editors

Editorů je dnes k dispozici opravdu velké množství. Mezi nejznámější patří komerční Epic, nabízející WYSIWYG režim nebo hierarchicky pohled na strukturu dokumentu. Ve WYSIWYG režimu lze navíc zapnout zobrazovaní tagů pro zvýšení přehledu o struktuře dokumentu. Epic má v sobě zabudovaný i formátovací modul, z vytvořených dokumentů lze vygenerovat PostScript. Obsahuje kromě samotného DocBooku i několik dalších DTD založených na DocBooku. Z editoru lze vyvolat i další programy (formátovací procesory ...) a nabízí například i možnost importu/exportu do Wordu. Nevýhodou je ale jeho vysoká cena.

Naopak úplně zdarma je použití kultovního editoru Emacs ve spojeni s balíkem PSGML. Mezi další z produktů nabízených zdarma, který stojí za povšimnutí, je XML Cooktop. Informace o dalších editorech lze nalézt na stránkách xmlsoftware[2]

5.3 XML Parsers

Parser slouží pro validaci (kontrolu struktury) XML dokumentu oproti definovanému DTD. Poté umožní dalším aplikacím (většinou pres standardní rozhraní, např. SAX, DOM) přístup ke struktuře a obsahu dokumentu. Příkladem je třeba XP, SP, Xerces a další. Opět více informací na xmlsoftware[3]

5.4 XSLT Engines

Jedná se nástroje pro transformaci XML dokumentu podle daného XSL. Asi nejznámějšími kandidáty v této kategorii jsou XT, XALAN, SAXON. Poslední ze jmenovaných ovšem nepodporuje žádné z českých kódování. Asi nejlepším se jeví XT, ve verzi, kterou upravil Jiří Kosek. Parserů je v dnešní době opět veliké množství, a to jak ve scriptovacích jazycích (Perl, Python, PHP), tak i pod Javou a C++. Jejich seznam lze nalézt na xmlsoftware[4]

1. http://www.xmlsoftware.com/browsers
2. http://www.xmlsoftware.com/editors
3. http://www.xmlsoftware.com/parsers
4. http://www.xmlsoftware.com/xslt

5.5 XML Formatters

Pod tímto názvem se skrývají nástroje pro zpracování formátovacích objektů. Umožňují z jednoho dokumentu vygenerovat PDF nebo PostScript pro tisk. Nejznámější je asi FOP (součást projektu Apache Cocoon). Jeho největším konkurentem je dnes komerční aplikace XEP (drive FO2PDF) od firmy RenderX. Bohužel FOP stále nepodporuje žádné české kódování, dokonce i při použití UTF-8 nedojde k vysázení některých českých znaků. Další alternativou je použití TeXového balíčku PassiveTeX,který zastane stejnou funkci jako FOP nebo XEP, ovšem i ten má své nemalé omezení. Přehled některých dalších aplikací na xmlsoftware.[5]

5.6 Publishing Systems

Představují komplexní systémy určené k publikování na webu při použití XML, které obsahují všechny potřebné programy z výše popsaných oblastí. Těchto systému je opravdu velice mnoho, avšak většina z nich je určena pro komerční použití. Z těch nekomerčních je velice slibným příkladem projekt Apache Cocoon, ale existuje i množství dalších; viz. xmlsoftware.[6]

5. http://www.xmlsoftware.com/xslfo
6. http://www.xmlsoftware.com/publishing

Kapitola 6

Závěr

V dnešní době je dostupné nemalé množství komplexních řešení, jež nám zprostředkují veškeré potřebné úpravy XML dokumentů. Celá tato technologie je teprve na začátku, a proto nelze očekávat naprosto automatizované řešení daných úkolů. Pořád je třeba řady manuálních úprav a přenastavení. Nejvíce práce je ovšem ve vytváření či v modifikaci XSL stylů pro výsledné zobrazení či transformaci. Ačkoli se může zdát, že otázka univerzálního kódování byla již dávno vyřešena, pořád se lze setkat s nekompatibilitou mnoha aplikací ke znakovým sadám, používaných v našich krajích. XML jako formát má nepopiratelně svou výhodu ve standardu, podle kterého byl vytvořen. To umožňuje použití dokumentů i v budoucnu, kdy bude vytvořen novější software splňující dané standardy. Ovšem ani zde neplatí nic zcela úplně. Formát XML se vyvíjí a v nejbližších letech se i vyvíjet bude. Nicméně s největší pravděpodobností nedojde k jeho výrazným logickým změnám, takže se o budoucnost našich dokumentů nemusíme bát. Pokud je použit formát XML pro tvorbu dokumentů, nelze od něj oddělit i formát transformace výsledného zobrazení XSL. Ten v současné době prožívá asi nejvíce změn. Tím pádem ani podpora jednotlivých webových prohlížečů nedosahuje stejné úrovně, ale výrazně se liší úrovní podpory pro jednotlivé standardy. Nejspíše nás neminou podobné problémy, jako byly u implementace HTML a hlavně CSS stylů.

Samotné psaní dokumentů ve zdrojovém kódu není příliš pohodlné, proto je lepší použití nějakého WYSIWYG editoru, bohužel dnes není dostupný žádný takovýto produkt, který by byl zdarma. Nezbývá než věřit, že se to v brzké době změní. Neboť formátu XML patří zítřek v široké škále odvětví jeho použitelnosti. Způsobů celého vytváření XML dokumentu je několik. V této práci byl použit způsob přes následnou transformaci do LaTeXu. A po jistých úpravách XSL stylu pro tuto transformací, je tím umožněno použít cls styl fithesis, doposud široce používaný pro diplomové práce. Tímto lze sjednotit výsledný vzhled většiny studentských prací. Nač se rozmýšlet, zdali psát dokument ve formátu RFT či v LaTeXu, když je tu možnost psát jej v něčem daleko univerzálnějším a poté ji do požadovaného formátu převést. XML nám přináší možnost široké přenositelnosti do mnoha různých systémů. Poté, co už jednou vtiskneme naše myšlenky do digitální podoby, nemusíme ji přepisovat pro každou variantu použití, nýbrž jednoduchou transformací z ní uděláme kupříkladu publikaci v knižním formátu, webovou stránku, s možností několika verzí pro jednotlivé prohlížeče a platformy, WML stránku, dokument v LaTeXu či ve Wordu, text slajdů na prezentaci a mnoho jiných. Omezení transformací se tím přenáší na vyšší úroveň, než dosud byla forma uložení a s ní pevně spjatý i výsledný vzhled. Doufám, že si v brzké době většina uživatelů oblíbí psaní všech dokumentů v XML a tím se pro ně stane psaní daleko pohodlnější a budou ušetřeni všech dosavadních problémů s nejednotnými formáty a netrivialní konverzí.

Příloha A

A

Odkazy na použitý software:

DocBook http://www.docbook.org/xml/4.1.2
DTD pro DocBook napsané v XML
Dokumentace k DocBooku http://docbook.org/tdg
HTML dokumentace k DTD DocBook
DSSSL styly 1.62 pro DocBook http://www.nwalsh.com/docbook/dsssl
styly pro DocBook napsané v SGML, využívané aplikací Jade pro převod XML do RTF
XSL styly 1.29 pro DocBook: http://www.nwalsh.com/docbook/xsl
styly pro DocBook používané při transformaci do HTML či FO
XSL styly pro převod do LaTeXu http://www.infres.enst.fr/~casellas/docbook.html
styly umožňující transformaci z XML DocBooku do LaTeXu
Jade 1.2.1 http://www.jclark.com/jade
aplikace pro transformaci DocBooku pomocí DSSSL stylů do formátu RTF
JRE 1.3.0_02 http://java.sun.com
Java Runtime Environment
Upravený XT parser od Jiřího Koska http://www.kosek.cz/xml/xt
aplikace umožnující transformaci XSL transformace
PassiveTeX 1.4 http://users.ox.ac.uk/~rahtz/passivetex
makra v TeXu pro převod XSL FO do PDF
xmltex ftp://ftp.tex.ac.uk/tex-archive/macros/xmltex/base.zip
XML parser implementovaný jako makro v TeXu
Fop 0.17.0 http://xml.apache.org/fop
systém pro vytváření a zpracování formátovacích objektů
Xep 2.21 http://www.renderx.com
komerční systém pro vytváření a zpracování formatovacích objektů
Saxon 6.2.2 http://users.iclway.co.uk/mhkay/saxo n
nástroj pro transformaci XML dokumentu podle daného XSL
Sp 1.3.4 http://www.jclark.com/sp
vynikající parser podporující XML
PDFlib 4.0 http://www.pdflib.com
aplikace pro transformaci FO objektů do PDF
TeX Live 5 http://www.texlive.org
rozsáhlý TeXový systém obsahující i nadstavdu LaTeX

Příloha B

B

Součástí práce je i přiložené CD, kde jsou k dispozici jednotlivé ukazky:

- zdrojový kód této práce ve formátu XML
- zdrojový kód této práce ve LaTeX
- tato práce ve formátu PDF, vygenerovaná pomocí aplikace XT, XSL stylů a systému LaTeX
- tato práce ve formátu HTML vygenerovaná aplikací XT a XSL stylů pro transformaci do HTML
- tato práce ve formáru RTF vygenerovaná aplikací Jade pomocí DSSSL stylů
- upravéné XSL styly pro transformaci do LaTeXu
- několik dalších užitečných aplikací

Reference

[Bradley] *The XML Companion*, Neil Bradley, Addison Wesley Longman Limited, Harlow 1998, 0-201-34285-5. 2.2

[Clark] *XSL Transformation (XSLT) Version 1.0)*, James Clark, W3C 1999. 2

[Ptáček] *XML pro každého*, Jiří Ptáček, Grada Publishing, 2000, 80-7169-860-1. 2.1

[Walsh] *DocBook: The Definitive Guide*, Norman Walsh, O'Reilly, Cambridge 1999, 1-56592-580-7. 3.2

Rejstřík

atributy, 11

browser, 24

CSS, 11

DocBook, 14
dokument, 13
DSSSL, 11
DTD, 7

fithesis, 19
FOP, 26

GML, 6

HTML, 6

ISO, 6

JRE, 22

kódování, 14

parser, 25
PassiveTeX, 26
PDF, 16
proměnné, 14
PS, 16
PUBLIC, 10
přepínač, 14

RTF, 13

SGML, 6
SYSTEM, 10

tag, 11
transformace, 16

validace, 14
VML, 24
vzhled, 13

XEP, 26
XML, 8
XSL, 11
XSLFO, 12
XSLT, 11
XT, 22

číslování, 20