VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

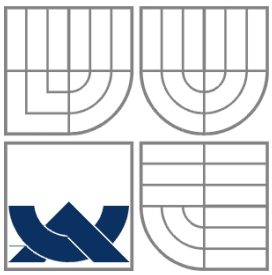
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FORMÁT XML PRE ZNAČKOVANIE SLOVNÍKOV

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE MARTIN ROJČEK
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FORMÁT XML PRO ZNAČKOVÁNÍ SLOVNÍKŮ

XML DICTIONARY TAGGING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN ROJČEK

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2007

Zadání bakalářské práce/4373/2006/xrojce00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Rojček Martin**
Obor: Informační technologie
Téma: **Formát XML pro značkování slovníků**
Kategorie: Databáze

Pokyny:

1. Seznamte se s nástroji pro práci s XML, XMLSchema atd.
2. Na základě existujících řešení navrhněte a realizujte systém pro správu slovníkových dat ve formátu XML.
3. Diskutujte výhody a nevýhody daného řešení na základě porovnání obsahu několika testovacích překladových a výkladových slovníků.
4. Vytvořte dokumentaci k realizovanému systému.

Literatura:

- podle dohody

Při obhajobě semestrální části projektu je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrž Pavel, doc. RNDr., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Martin Rojček**
Id studenta: 84167
Bytem: Liptovská Sielnica 98, 032 23 Liptovská Sielnica
Narozen: 11. 02. 1985, Ružomberok
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Formát XML pro značkování slovníků
Vedoucí/školitel VŠKP: Smrž Pavel, doc. RNDr., Ph.D.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Bakalárska práca opisuje ukladanie slovníka do vhodnej XML podoby. Zaoberá sa hlavne nástrojmi na popis štruktúry (DTD, XML Schéma, Relax NG a pod.) a transformáciu (XSLT) XML dokumentov. Popisuje najznámejšie formáty uloženia slovníkov do XML (OLIF, ISLE/MILE apod.) a praktické využitie jedného zo spôsobov – uloženie pomocou super-jednoduchého ac.dtd. Implementácia poukazuje na relatívne jednoduchú prácu so slovníkmi uloženými v takejto podobe. Posledná časť je venovaná získavaniu štatistík, na čo bol použitý skriptovací jazyk python.

Kľúčové slová

slovník, značkovanie, XML, transformácia, XSLT, DTD, Relax NG, OLIF, ISLE/MILE, python.

Abstract

This Bachelor's thesis describes data stacking of vocabulary into proper XML structure. It deals with structure destription tools(DTD,XML Schema, Relax NG and others) and transformation (XSLT) of XML documents. It describes best known vocabulary storing formats into XML (OLIF, ISLE/MILE and others) and practical advantage to take one of method - store with helping of simplest ac.dtd. Implementation shows on relatively easy work with vocabularies stored in this form. Last part considers about getting statistic data with using Python scripting language.

Keywords

dictionary, tagging, XML, transformation, XSLT, DTD, Relax NG, OLIF, ISLE/MILE, python.

Citácia

Martin Rojček: Formát XML pre značkovanie slovníkov, bakalárska práca, Brno, FIT VUT v Brne, 2007

Formát XML pre značkovanie slovníkov

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením doc. RNDr. Pavla Smrža, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Martin Rojček
9.5.2007

Poďakovanie

Ďakujem vedúcemu bakalárskej práce doc. RNDr. Pavlovi Smržovi, Ph.D. za jeho trpezlivosť a odbornú pomoc pri vypracovávaní bakalárskej práce.

Obsah

Úvod

Táto práca sa zaoberá slovníkmi a poukazuje na vhodnosť uloženia a manipulácie s nimi vo formáte XML, pričom využíva príbuzné formáty a technológie, akými sú XSL, XML Schéma apod. Vhodnosť XML spočíva v tom, že XML forma obsahuje okrem samotných dát aj popis ich štruktúry. Je nezávislá na použitom systéme a platforme, a má ďalšiu radu výhod.

Cieľom tejto práce je zoznámenie sa s XML a pochopenie súvislostí využitia tohto jazyka pri ukladaní slovníkových dát. Tiež poukazuje na relatívne jednoduchú prácu so slovníkmi uloženými v takejto podobe.

Práca je rozdelená do niekoľkých celkov, pričom v prvých teoretických častiach je opísaná práca s XML a podpornými nástrojmi. V ďalších častiach je už opísaná práca so slovníkom praktickým spôsobom – zápis slovníka v XML, jeho transformácia a vytvorenie štatistík.

Prvá časť je zameraná na teoretický úvod preniknutia do problematiky, týkajúcej sa jazyka XML a príbuzných technológií. Týmito technológiami sú hlavne nástroje na popis štruktúry dokumentu, akými sú: DTD, XML Schéma, Relax NG, Schematron a nástroje na transformáciu dokumentov XML - XSLT.

Druhá časť sa venuje slovníkom všeobecne, popisuje rôzne spôsoby uloženia slovníkov vo formáte XML (superjednoduché, OLIF atď.).

Tretia časť popisuje implementáciu jednotlivých operácií s daným slovníkom. Použitými nástrojmi boli XSLT a skripty v jazyku python. V tejto časti sa tiež nachádza popis získavania štatistík.

1 XML a súvisiace technológie

S jazykom XML sa už určite stretla väčšina ľudí. Totiž má široké možnosti využitia. Uvediem ich aspoň niekoľko: využitie v internetových aplikáciách, účtovníctve, dátových skladoch, IT systémoch apod. Ja sa jazykom XML sa v tejto kapitole zaoberám z dôvodu jeho vhodnosti pre uloženie slovníkov. XML poskytuje výhodný spôsob opisu štruktúrovaných dát slovníka v otvorenom textovom formáte. Uvediem, čo to vlastne XML je a ako vyzerá jeho syntax. Podľa toho je zreteľnejšie, prečo je XML vhodným formátom v súvislosti s ukladaním slovníkových dát.

1.1 XML

Definícia 1: XML (Extensible Markup Language - rozšíriteľný značkový jazyk) je podmnožina SGML definujúca jazyk pre výmenu štruktúrovaných dát [1].

Definícia 2: XML je množina pravidiel, ktoré rozdeľujú dokument do rôznych častí. Je to metajazyk definujúci syntax definície iných, doménovo špecifickejších, sémantických a štruktúrovaných značkovacích jazykov [2].

XML dokument obsahuje dáta. Neobsahuje prezentačnú vrstvu a tiež nemá za úlohu definovať štruktúru dokumentu. Tieto úlohy preberajú iné jazyky. Architektúra XML je rozčlenená na niekoľko vrstiev:

1. Dáta – nositeľom dát je samotný XML dokument.
2. Štruktúra – štruktúru dokumentu určuje DTD alebo XML Schéma (príp. Relax NG).
3. Prezentácia – prezentačná vrstva je zabezpečovaná štandardom XSL.

O XML sa často hovorí v súvislosti s tvorbou webových stránok a teda aj v súvislosti s jazykom HTML. Práve s HTML býva jazyk XML často porovnávaný a skutočne s ním má aj veľa spoločného. XML, na rozdiel od HTML, môže definovať vlastné značky (**tagy**). Na základe toho je možné označiť význam jednotlivých častí textu. Pričom HTML umožňuje len definovať vzhľad textu pri jeho zobrazení.

Napriek tomuto prirovnaniu XML nemá za úlohu nahradiť HTML. XML je určené do oblastí, kde sa HTML, respektíve iné jazyky, nedokázali uplatniť. Jedná sa hlavne o komunikáciu medzi strojmi, programami, kde je vyžadovaná jasná a jednoduchá syntax a jednoznačne daná štruktúra, ktorá umožňuje jednoduché a rýchle spracovanie programom.

Prácu s XML uľahčuje to, že celý formát je založený na obyčajnom texte. Použitie textového formátu môže pripadať niekomu ako zbytočné plytvanie miestom. Dnes sa však omnoho väčší dôraz kladie na zrozumiteľnosť a jednoduchú prácu s dátami. To, že ušetríme pár kilobajtov pamäte už nie je také relevantné. V XML preto môžeme vytvárať dokumenty, ktoré obsahujú texty v mnohých jazykoch [3].

Jazyk XML by mal byť použiteľný aj v budúcnosti, keď bude musieť vyhovieť ďalším požiadavkám. Preto je XML navrhnutý tak, aby bol ľahko rozšíriteľný. Stal sa odporúčaním konzorcia W3C. XML je otvorený formát, čo znamená, že jeho špecifikácia je každému zdarma k dispozícii na serveri konzorcia W3C.

1.1.1 Syntax XML

XML dokument sa skladá z elementov. Element môže obsahovať ďalšie elementy alebo dáta. Element môže mať dané atribúty, ktoré určujú vlastnosti elementu.

Syntax: **Element**

```
<meno_elementu atribút=”hodnota atribútu”>

dáta, ktoré sú obsahom elementu

</meno_elementu>
```

Základné syntaktické pravidlá:

- Každý element musí mať uzatvárací tag
  `<element>obsah elementu</element>`,
  alebo je zapísaný skrátene (v tom prípade nemôže obsahovať žiadne ďalšie elementy alebo text):
  `<element/>`.
- Názvy elementov rozlišujú veľké a malé písmená (case sensitive):
  `<Zly_priklad>obsah</zly_priklad>`
  `<Dobry_priklad>obsah</Dobry_priklad>`.
- Elementy sa nesmú prekrývať (musia byť korektne zahniezdené):
  `<zly_priklad><b>obsah</zly_priklad></b>`
  `<dobry_priklad><b>obsah</b></dobry_priklad>`.
- XML dokument musí mať koreňový element. Všetky ostatné elementy v ňom obsiahnuté.
- Hodnota atribútu musí byť v apostrofoch.
- Apostrofy alebo úvodzovky ohraničujú hodnotu atribútu:
  `<element atribút=”hodnota atribútu”>`
  `<element atribút=’hodnota atribútu’>`.
- Komentár:
  `<!-- komentár -->`.

Definícia: XML dokument je **well-formed (správne vytvorený),** ak spĺňa základné syntaktické pravidlá uvedené vyššie [5].

Definícia: XML dokument je **valid (platný)**, ak je správne vytvorený a spĺňa definíciu dokumentu danú v externom súbore (DTD alebo XML schéma) [5].

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE kniha SYSTEM "osoba.dtd">
<osoba>
    <meno>Milan</meno>
    <priezvisko>Pula</priezvisko>
    <dob>25-April-1969</dob>
</osoba>
```

Ak program, ktorý spracúva XML súbor, v ňom nájde chybu, podľa špecifikácie XML zastaví (na rozdiel od HTML, kedy boli možné rôzne scenáre spracovania chybného HTML súboru v rôznych programoch). Tým sa veľmi zjednodušuje práca programátora pri návrhu programu.

Escape characters (špeciálne znaky) [6] sú znaky, ktoré sú spracovávané špeciálnym spôsobom a preto ich musíme nahradiť reťazcami znakov:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	Quotation mark

1.1.2 XML parsery

V mojej aplikácii bolo potrebné pracovať s XML dokumentom. Najčastejšia operácia s XML dokumentom je jeho načítanie, k tomuto účelu slúžia knižnice nazývané parsery. Ich úlohou je čítanie a analýza XML dokumentu a následný prevod do infosetu (definície abstraktného dátového modelu XML dokumentu). Súčasťou infosetu už nie sú samotné znaky, ale elementy, atribúty a textové obsahy elementov. V zásade existujú dva hlavné prístupy pri parsovaní XML dokumentu – prístup založený na stromovej štruktúre a prístup založený na udalostiach. Každý z týchto prístupov reprezentujú parsery DOM a SAX. Sú najznámejšie a aj najpoužívanejšie, preto som sa ich snažil bližšie priblížiť.

1.1.2.1 XML DOM

XML Document object model (XML objektový model dokumentu) je platformovo a jazykovo nezávislé rozhranie, ktoré umožňuje programátorom vytvárať aplikácie upravujúce obsah, štruktúru a štýl dokumentov. Je založený na štandarde organizácie W3C [6].

XML DOM predstavuje stromovú štruktúru XML dokumentu. Koreňovým vrcholom je **documentElement** objekt. Na ďalšej úrovni stromu sú **childNode** objekty. Programátor pri úprave dokumentu reprezentovaného prostredníctvom DOM pracuje so stromom, používa metódy na pridávanie, mazanie, úpravu objektov.

XML parser je program (resp. knižnica), ktorý XML dokument načíta do pamäte počítača ako štruktúru objektov podľa XML DOM – **infoset**. Tiež umožňuje ďalšie úpravy a s jeho pomocou dokument môžeme uložiť do súboru. Ak XML súbor obsahuje referenciu na DTD alebo XML schému, XML parser vykonáva aj funkciu validátora – overuje, či dokument spĺňa danú definíciu.

Riešenie podľa štandardu DOM môže byť pri veľkých XML dokumentoch náročné na pamäť – celý XML dokument sa nahráva do pamäti a v závislosti od konkrétnej implementácie môže zaberať v operačnej pamäti dva až desať krát väčší priestor ako na disku.

1.1.2.2 SAX

Simple API for XML je jednoduché rozhranie pre XML. Pri svojom vzniku bolo určené len pre jazyk Java, potom však vznikli implementácie aj do iných jazykov. Jedná sa o najznámejšie rozhranie riadené udalosťou (event-based). V praxi to znamená, že sú definované funkcie, ktoré sú vyvolané v okamžiku, kedy parser narazí na začiatok elementu, na obsah elementu, na koniec elementu, na komentár, na inštrukcie pre spracovanie apod. Funkcii sú potom predané všetky potrebné parametre ako napr. názov elementu. Výhoda udalosťami riadeného prístupu je v rýchlosti a malej spotrebe pamäti. XML súbor sa prechádza postupne a nie je nutné nahrať celý dokument do pamäti. Naopak, nevýhodou je, že je nutné spracovať dokument behom jedného sekvenčného priechodu.

1.2 Popis štruktúry dokumentu

XML dokumenty môžu mať ľubovoľnú štruktúru a možno v nich používať ľubovoľné značky. Príliš veľa voľnosti však škodí, preto sú potrebné jazyky pre popis štruktúry dokumentu XML. Tieto jazyky umožňujú definovať, aké elementy a atribúty môžeme v XML dokumente používať, ako ich vzájomne kombinovať, čo môžu obsahovať. Štruktúra doku-

mentu XML vlastne definuje nový značkovací jazyk, ktorý má syntax XML, ale používa nami vytvorené značky.

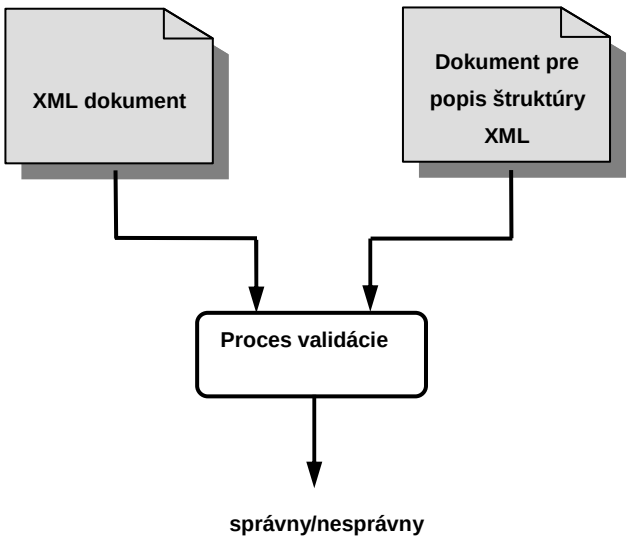
Výhoda formalizovanej definície spočíva v tom, že je jednoznačná a znemožňuje rôzne interpretácie, na rozdiel od definície popísanej v prirodzenom jazyku. Všetci, ktorí chcú dokumenty XML odpovedajúce danej schéme spracovávať, preto vedia, ako môžu dokumenty vypadať.

Vzhľadom k tomu, že schéma jednoznačne definuje, ako môže dokument XML vypadať, môžeme ho samozrejme použiť i na validáciu. Jedná sa teda o najčastejšie použitie schém. **Validácia** je proces, pri ktorom sa overuje, či konkrétny dokument XML vyhovuje všetkým obmedzeniam definovaným v schéme. Výhodou validácie je uľahčenie vývoja aplikácií. Tie nemusia pri čítaní dát z XML prevádzať kontrolu správneho vstupu, pretože väčšina prípadných chýb odhalí validácia.

Niektoré jazyky pre popis štruktúry umožňujú pre obsah jednotlivých elementov a atribútov určiť ich dátový typ, ako číslo, reťazec, dátum a pod. V priebehu validácie potom môže byť jednotlivým častiam XML dokumentu priradený ich dátový typ. Aplikácia pri čítaní dokumentu pracuje rovno s otypovanými dátami a nielen s textovými reťazcami, ako je vo svete XML bežné [7].

V neposlednom rade slúži schéma ako dokumentácia pre značkovací jazyk, ktorý definuje. Naviac pre mnoho jazykov pre popis schém existujú nástroje, ktoré z pôvodnej schémy vygenerujú prehľadnú dokumentáciu, napr. v podobe hypertextovo previazaných webových stránok.

Pomocou štruktúry XML dokumentu popísanej pomocou DTD, XML Schéma alebo Relax NG je možné verifikovať syntaktickú správnosť XML dokumentu.



Obrázok 1.: Proces validácie XML dokumentu

1.2.1 DTD (Document Type Definition)

Hoci je DTD starší jazyk na popis štruktúry dokumentu, stále sa používa a aj ja som využil pri transformácii vstupných súborov slovníka do validných XML dokumentov na základe súboru ac.dtd, ktorý je určený pre superjednoduché slovníky.

DTD je jazyk na popis štruktúry dokumentu pochádzajúci ešte z jazyka SGML. DTD vďaka tomu má jedinú veľkú výhodu – podporuje ho veľké množstvo aplikácií. To je bohužiaľ v súčasnosti jedinou výhodou. Za najväčší nedostatok môžeme označiť nulovú podporu pre menné priestory.

XML menný priestor – namespace - umožňuje jednoznačnú identifikáciu elementov a atribútov XML priradením rovnakého jednoznačného identifikátora URI [8].

Druhým nedostatkom je nemožnosť určenia dátového typu pre obsah elementov a atribútov. SGML, a neskôr XML, bolo primárne navrhnuté pre značkovanie dokumentov textovej povahy, v ktorých je v podstate všetko text. XML sa však začalo masovo používať i na výmenu štruktúrovaných dát medzi informačnými systémami – okrem textu sa používajú číselné hodnoty, menové údaje, dáta a pod. Aby bola validácia skutočne účinná, je potrebné obmedziť štruktúru vzájomného zanorenia elementov, ale i obsah týchto elementov (príp. i atribútov) ich dátovým typom.

Tretí problém spočíva v tom, že DTD má síce jednoduchý a kompaktný zápis. Jeho syntax však nenájdeme na inom mieste v XML.

1.2.2 XML Schéma

Je ďalšou z možností, ktorou som mohol kontrolovať, či sú transformované XML dokumenty validné. Oproti DTD má mnoho výhod, no ešte lepším nástrojom na kontrolu validnosti je ďalej spomínaný Relax NG.

Ku vzniku XML Schéma viedla potreba náhrady DTD. Jeho rysy je možné zhrnúť do niekoľkých bodov:

- Sú vytvorené pomocou XML, majú XML syntax, pre ich spracovanie je možné použiť štandardný XML software.
- Umožňujú presnejšie určovanie typov dát (reťazec, booleovský typ, čísla, dátumy a pod.).
- Je možné presne kvantifikovať počet opakovaní elementu, elementy zoskupovať v záväznom poradí a pod.
- V jednom dokumente možno kombinovať i viac schém.

- Pracujú s objektovo orientovanými konceptmi, napr. s dedičnosťou a zapúzdrenosťou.

Problémom XML Schéma je jej pomerne zložitá a nie vždy celkom pochopiteľná špecifikácia. Obsahuje veľký počet kľúčových slov a ručné písanie schémy bez podporného nástroja je tiež veľmi náročné.

1.2.3 Relax NG (REgular LAnguage for XML Next Generation)

Relax NG je jazyk pre popis štruktúry XML dokumentov. Je navrhnutý združením OASIS, dnes je už aj ISO štandardom (ISO/IEC 19757-2). Má dve formy zápisu: kompaktnú a XML syntax. Kompaktná syntax je prehľadne čitateľná, a uľahčuje tak návrh aj správu schém. XML syntax používa pre svoj zápis súbor XML značiek (tagov). Na rozdiel od XSD súborov, je tento formát jednoduchší a napriek tomu umožňuje zapísať väčšinu z toho, čo umožňuje XSD (XML Schema Definition). XSD je iný názov pre XML Schéma, ktorý som opisoval vyššie.

Samozrejmosťou je podpora dátových typov, pre ktorú používa tento jazyk dátové typy prevzaté z jazyka XML Schéma. Pre elementy a atribúty je možné definovať i regulárne výrazy, ktorým majú vyhovovať. Kvôli týmto vlastnostiam jazyku Relax NG stúpa popularita vývojárov a získal si aj oficiálnu podporu veľkých organizácií, ktoré v ňom objavili veľmi silný nástroj (paradoxne aj združenie W3C, ktoré stojí za vznikom XSD).

Relax NG sa poučil z predchádzajúcich chýb. Vychádza z jazykov RELAX a TREX.

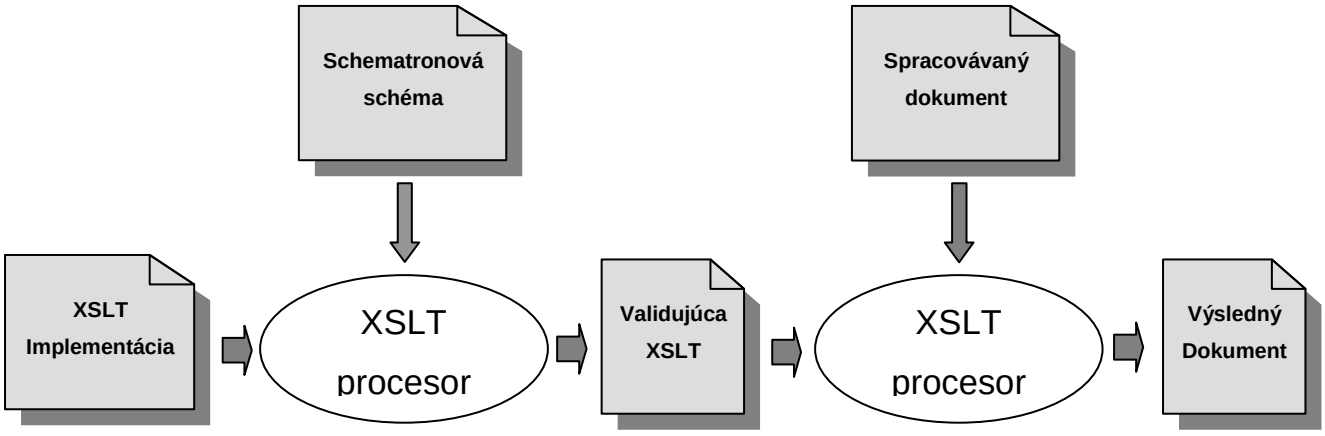
TREX (Tree Regular Expressions for XML) je jazyk na validáciu XML dokumentov [10]. TREX vzor presne určuje vzor pre štruktúru a obsah XML dokumentu. TREX vzor tak identifikuje triedu z XML dokumentov pozostávajúcich z tých dokumentov, ktoré vyhovujú vzoru. TREX vzor je sám XML dokumentom [11].

Syntax Relax NG je veľmi intuitívna. Vyniká taktiež dobrou rozšíriteľnosťou (napríklad je možné použiť rôzne knižnice dátových typov). Elegantne je riešená i modularita definíc. Jazyk Relax NG je naviac založený na silnom matematickom modeli. Vzhľadom k použitiu open source nástrojov pri spracovaní a validácii XML dokumentov nie je problém jeho slabá podpora v komerčných nástrojoch (napríklad oproti XML Schema).

1.2.4 Schematron

Schematron uvádzam hlavne z dôvodu jeho odlišnej filozofie od predchádzajúcich nástrojov na popis štruktúry dokumentu. Hoci ešte nie je veľmi rozšíreným jazykom, je to mocný nástroj, ktorý má mnoho výhod, ktoré v tejto časti spomínam. Je veľmi vhodný na použitie práve pre slovníky, aj vďaka svojím vlastnostiam, ktoré iné jazyky neumožňujú – napríklad dokáže kontrolovať obsahy elementov v závislosti na iných elementoch.

Schematron je špeciálny validačný jazyk umožňujúci definovať sadu podmienok, ktorým musí dokument vyhovieť. Podmienky sa pritom zapisujú pomocou dotazovacieho jazyka XPath, ktorý ponúka široké možnosti výberu a spracovania dát nachádzajúcich sa v dokumente XML a zvládne aj jednoduché výpočty. Použitie jazyka XPath má dve veľké výhody – sú k dispozícii pomerne silné vyjadrovacie prostriedky XPathu a na validáciu dokumentu postačuje XSLT procesor, pretože schematronovú schému možno previesť na XSL transformáciu. Tento prístup umožňuje reprezentovať mnoho druhov štruktúr a obmedzení, ktoré by boli nepohodlne a ťažko reprezentovateľné v jazykoch založených na gramatikách. V jazykoch založených na gramatikách sa definujú možné obsahy jednotlivých elementov. Všetky tieto definície spolu potom vymedzujú množinu všetkých prípustných dokumentov. Všetko sa musí definovať postupne a často-krát zbytočne prácne.



Obrázok 2.: Validácia XSLT procesorom

Definícia jazyka Schematron je veľmi jednoduchá. Obsahuje niekoľko elementov. Ak teda užívateľ ovláda XPath, nie je veľkým problémom naučiť sa i Schematron.

XPath je jazyk pre hľadanie informácií v XML dokumente. Je používaný na riadenie prostredníctvom elementov a atribútov v XML dokumente [12].

Schematron je v podstate syntézou už existujúcich, rozsiahle používaných a štandardných technológií, čo mu zaisťuje vysokú životaschopnosť. Čitateľnosť Schematronu je veľmi dobrá, ak používateľ ovláda XPath. V opačnom prípade sa môže ľahko orientovať podľa slovných vyjadrení, ktoré by mali popisovať každý jednotlivý test alebo súbor testov.

Schematron je založený na úplne odlišnom princípe ako skôr zmienené schémové jazyky, ktoré viac-menej definujú gramatiku pre dokument XML. Pomocou Schematronu je možné zapísať tvrdenie o prítomnosti alebo absencii určitých vzorov v dokumente. Validácia oproti Schematronu potom vracia zoznam tvrdení, ktorý vznikne kontrolou vzorov oproti dokumentu. Nemusia sa definovať všetky elementy a väzby medzi nimi, ale stačí sa z praktického hľadiska sústrediť na konkrétne obmedzenie, ktoré chceme kontrolovať. V Schematrone je možné popísať niektoré závislosti, ktoré boli v iných jazykoch nepopísateľné. Často sa používa spoločne s jazykmi založenými na gramatikách (DTD, XML Schema, TREX, ...). Taktiež môže byť vnorený do XML Schema alebo do Relax NG.

Je založený na dvoch jednoduchých akciách:

- **Nájdenie** kontextových uzlov v dokumente (obvykle elementov) na základe nejakého výrazu jazyka XPath.
- **Overenie,** či každý z týchto uzlov splňuje určitú podmienku, ktorá je opäť vyjadrená pomocou XPath.

Umožňuje vytvárať a kombinovať dva druhy schém:

- Element **report** umožňuje stanoviť, akým variantom jazyka sa zaoberáme, teda informuje o kladných vlastnostiach inštancie.
- Element **assert** umožňuje overiť, že dokument odpovedá nejakej schéme, teda detekuje chyby.

Nevýhodou Schematronu je, že je pomerne prácne popísať celú štruktúru dokumentu a všetko sa musí robiť ručne.

Výsledok validácie dokumentu nie je iba binárna informácia (validný/chybný), ale je to zložitejšia množina hodnôt.

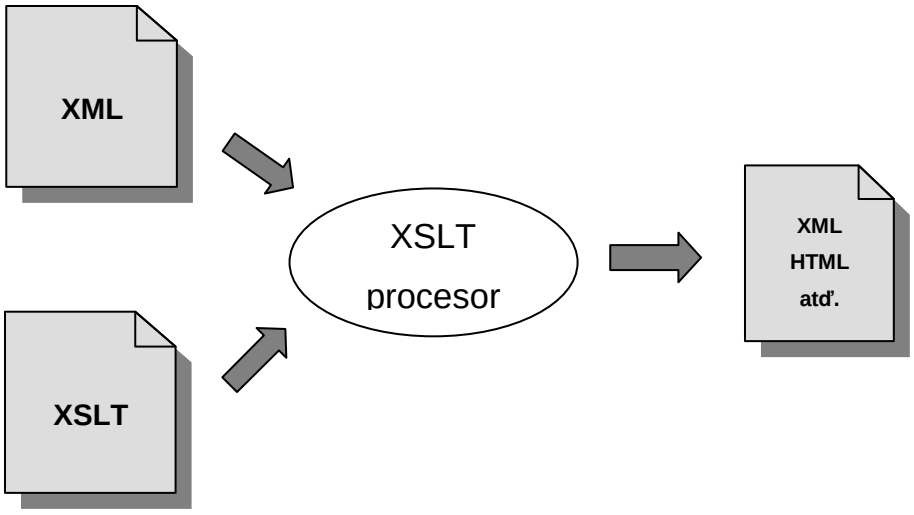
Jednotlivé množiny vzorov sa môžu aktivovať a deaktivovať podľa potreby [14].

1.3 Transformácie XML, XSLT

Pri ďalšom spracovávaní XML dokumentu je často potrebné transformovať tento XML dokument na iný XML dokument, prípadne iný dokument. Ja som potreboval transformovať XML

dokument na iný XML dokument podľa DTD (ac.dtd). Technológia XSLT ponúka možnosť tejto transformácie elegantným spôsobom.

XSLT (eXtensible Stylesheet Language Transformations) je jazyk pre transformáciu XML dokumentov. Jeho najväčšou výhodou je fakt, že využíva syntax jazyka XML. Princíp XSLT je pomerne jednoduchý: vstupom do XSLT procesoru je zdrojový dokument a XSLT štýl, výstupom je potom výsledný dokument, ktorým môže byť iný XML, HTML alebo len textový dokument.



Obrázok 3.: Princíp transformácie XML pomocou XSLT

Definícia: XSL transformácia je operácia na strome XML dokumentu, ktorej výsledkom je strom elementov a atribútov [15].

XSLT je časťou XSL, ktorý tvorí tzv. „zlatú strednú cestu“ medzi jednoduchým CSS a zložitým DSSSL.

CSS (Cascading Style Sheets) je jazyk pre popis spôsobu zobrazenia stránok napísaných v jazykoch HTML, XHTML alebo XML. Bol navrhnutý štandardizačnou organizáciou W3C. Hlavným zmyslom je umožniť návrhárom oddeliť vzhľad dokumentu od jeho štruktúry a obsahu [16].

Aj keď sa stále vytvárajú nové a dokonalejšie verzie CSS, je tento jazyk pomerne málo efektívny. Druhým extrémom je jazyk DSSSL, ktorý bol vytvorený pre použitie v SGML. Bol síce vydaný ako štandard ISO, ale pre svoju nesmiernu zložitosť je dnes takmer bez softvérovej podpory. **DSSSL** (Document Style Semantics and Specification Language) je jazyk na špecifikáciu štýlových šablón pre SGML dokumenty, založený na

podmnožine programovacieho jazyka Scheme. Je špecifikovaný štandardom ISO/IEC 10179:1996 [17].

XSL šablóna je XML súbor, v ktorom sa nachádzajú pravidlá pre XSL transformáciu. Sú tu definované vzory a šablóny, pričom XSL procesor hľadá vzor v strome XML súboru, keď ho nájde, nahradí ho šablónou. Výsledný súbor môže byť úplne iný, na rozsah zmien nie sú kladené žiadne obmedzenia. Výstup sa väčšinou skladá z nových dát, dát skopírovaných z originálneho dokumentu a značiek. Každá šablóna určuje dve veci: na akej časti výstupného dokumentu je použitá a ako bude táto časť transformovaná do výstupného dokumentu.

Pri XSL transformácii prečíta XSL procesor dokument XML a XSL šablónu a podľa inštrukcií nájde v štýle XSL, ako má výsledný dokument vypadať. Každý XML dokument je strom zostavený z uzlov. Jeho obsah je považovaný za množinou uzlov (prvkov, komentárov, atribútov, menných priestorov) usporiadanou do určitej hierarchie. XSLT procesor predpokladá XML strom obsahujúci sedem druhov uzlov:

1. Koreňový uzol (je to niečo iné ako koreňový prvok v XML dokumente)
2. Uzol prvku
3. Textový uzol
4. Uzol atribútu
5. Uzol menného priestoru
6. Uzol obsahujúci spracovávajúci inštrukcie
7. Uzol komentára

XSLT ponúka pri tvorbe šablón ďalšie nepreberné možnosti. V jednotlivých šablónach môžeme používať podmienky a cykly, elementy je možné radiť na základe rôznych kritérií atď. Pomocou XSLT je možné odstraňovať, vytvárať, preskupovať, triediť, či opakovane používať prvky zdrojového dokumentu. XSLT samozrejme ponúka prostriedky pre automatické číslovanie – môžeme číslovať kapitoly, obrázky, položky zoznamu apod. Štýl možno rozdeliť na viac častí a navzájom kombinovať. Pre zložitejšie štýly sa hodí možnosť používania premenných a parametrov.

Dnes je k dispozícii niekoľko softvérových XSLT procesorov. Väčšina je napísaná v Jave, niektoré sú k dispozícii aj v C++. Medzi voľne šíriteľné procesory patrí XT (http://www.jclark.com/xml/xt-old.html), Saxon (http://saxon.sourceforge.net/) alebo Xalan (http://xml.apache.org/xalan-j/). Vlastným XSLT sa môže pochváliť aj Microsoft, Oracle alebo IBM.

2 Slovníky

Keďže táto bakalárska práca je zameraná hlavne na slovníky, je dôležité lepšie popísať túto problematiku. V tejto kapitole sa teda snažím objasniť, čo to vlastne slovník je a aké slovníky poznáme. Vo svojej praktickej časti práce so slovníkom som použil jednoduché uloženie slovníka podľa ac.dtd vzhľadom k tomu, že nebolo potrebné používať zložitejší štandard. Jednotlivé podkapitoly sú venované štandardom uloženia slovníkov do XML, pretože tieto štandardy ponúkajú ďalšie možnosti uloženia slovníka, ktoré by sa dali využiť pre túto realizáciu.

Slovník je najčastejšie abecedne radený zoznam slovnej zásoby, vysvetľujúci slová z rôznych hľadísk. Zostavovaním slovníkov sa zaoberá lingvistická disciplína nazývaná lexikografia [3].

Vo väčšine slovníkov sú slová zachytené iba vo svojom základnom tvare, tzv. lemmate. Slovníky sa vyskytujú tradične najčastejšie v knižnej podobe. V poslednej dobe sa však objavujú aj digitálne slovníky, dostupné na CD alebo internete.

Podľa rozsahu sa slovníky rozdeľujú:

- malé, vreckové (do 10 000 hesiel)
- stredné (50 000 až 60 000 hesiel)
- veľké (nad 60 000 hesiel)

Podľa typu sa rozdeľujú na:

- slovníky výkladové (jednojazyčné) – sú napísané v jednom jazyku, pri každom slove možno nájsť informácie v rovnakom jazyku. Ďalej ich možno rozdeliť:
  - slovníky súčasného jazyka (významové, pravopisné, frazeologické, slovníky synoným, slovníky cudzích slov atď.)
  - slovníky jednotlivých historických období a slovníky etymologické
  - slovníky popisujúce slovnú zásobu pracovných skupín
  - špeciálne slovníky (retrográdne, frekvenčné, valenčné atď.)
- slovníky prekladové (viacjazyčné) – slúžia pre preklad z jedného jazyka do druhého, k slovám jedného jazyka obsahujú jeho preklad v druhom jazyku, často i s výslovnosťou, komentármi, frázami alebo inými sprievodnými informáciami. Niektoré väčšie prekladové slovníky obsahujú i druhú časť, v ktorej sú slová pre spätný preklad z druhého jazyka do prvého. Tieto slovníky môžu byť i špecializované, napr. sa obmedzovať len na odborné termíny z niektorej oblasti.

Radenie slov v slovníku

V slovníkoch tých jazykov, ktoré používajú alfabetický alebo slabičný systém zápisu, sa slová zoraďujú abecedne, alebo v analogickom fonetickom poradí. Slová alebo znaky v jazykoch s logografickým systémom zápisu, ako napríklad čínština, sa zoraďuje podľa jednej z mnohých schém, založených na zložkách znaku, počte ťahov a celkovom tvare znaku.

2.1 TBX

Prvým zmieňovaným štandardom je TBX. V súčasnej dobe nemá veľký význam, no je dôležitý z hľadiska vývoja štandardov. Je prvým ustáleným štandardom popisujúcim uloženie slovníkov v elektronickej podobe.

Prvý výmenný formát pre terminológiu sa volal MATER; definoval, ako sa mali dáta ukladať na magnetickú pásku, špecifikoval, medzi inými vecami, sekvenciu bajtov, dĺžku pásky, veľkosť bloku atď. Tento formát bol prevedený do MicroMater (pre počítačovú výmenu), a neskôr do MARTIF, prvého, na SGML založeného formátu. MARTIF podstúpil niekoľko štandardizačných krokov. Aktuálny stav formátu je XLT (XMLbased Formats for Lexicon and Terminology Exchange), ktorý je rámcový pre niekoľko častí štandardu závislého na rozdielnom používaní prípadov užitia, najznámejšia forma z nich je TBX (Term Base eXchange).

TBX modeluje terminologické vstupy. Takéto vstupy sú budované na základe rozdielov medzi centrálnymi konceptmi (ktorými sú sémantické jednotky) a vzťahmi (ktoré určujú jednotky v rozdielnych jazykoch). Rozdiel medzi konceptmi a vzťahmi je základný člen v TBX architektúre s názvoslovnými vstupmi, je organizovaný konceptmi. Koncepty sú základné sémantické entity, môžu mať globálne atribúty (uložené v pomocnej časti) ako príslušná oblasť, súvisiace koncepty, definície, príklady, vzorky vety atď. Potom sú popísané v jazykovo-príbuzných názvoch sekcií („Nastaveniach jazyka“), ktoré sa skladajú z termínových informačných skupín obklopujúcich jednodielne vzťahy.

Vstupy vzťahov tvoria jadro TBX súboru, ktorým je XML dokument zostavený z nasledovných zložiek:

- **Hlava**, ktorá popisuje zoznam s výnimkou niektorých globálnych a organizačných informácií (obsah, validácia stavu, kontakt, zakódovanie, revízie, atď.).
- **Telo**, ktoré sa skladá zo súboru vstupov, jeden na koncept v databáze. Telo môže mať začiatočné a konečné elementy.

TBX patril ku prvým štandardom týkajúcich sa slovníkov. Postupne ho nahradili ďalšie – keď začala práca na výmennom formáte pre strojový preklad slovníkov, rýchlo sa prišlo na to, že MARTIF/TBX nebude schopný uspokojiť požiadavky pre takéto meniace sa slovníky. Jazykový popis, dostupný v TBX štandarde, nebol prijateľný pre jazykovú výmenu. Je v ňom zahrnutých len veľmi málo vysvetliviek (ako slovný druh, gramatický rod, číslo) a odvoláva sa na veľmi málo jazykov. Obzvlášť, nebola tam žiadna predstava o základných rysoch ako výmena v MT (Machine Translation – strojový preklad), ako skloňovanie foriem, syntaktické typy, obsah štruktúry, významové rysy atď.

2.2 OLIF

OLIF je významnejším štandardom oproti TBX, ktorý, ako prvotný štandard v tejto oblasti, obsahoval veľa nedostatkov. OLIF sa ich snaží odstrániť a hlavne doplniť pravidlá, s ktorými predchádzajúci štandard nepočítal.

OLIF (Open Lexicon Interchange Format) je otvorený štandard pre lexikálne/názvoslovné kódovanie dát. Aj keď pôvodný účel OLIFu bolo poskytovať výmenu lexikálnych dát medzi vlastnými strojovými prekladovými (MT – machine translation) slovníkmi, neskôr bol vyvíjaný ako štandard na všeobecnejšiu výmenu dát v oblasti jazykových technológií. Bol definovaný v EC projekte nazývanom OTELO. Neskoršie verzie boli vyvíjané OLIF konzorciom. Vyvíjaný štandard ponúka rozsiahlu podporu pre jazykovú výmenu a reprezentáciu dát. Zrealizovaný ako XML Schéma (OLIF v. 2.1) a XML DTD (OLIF v. 2) so zastupujúcim nastavením názvoslovných a lexikálnych rysov ponúka voľby pre jazykové aplikácie [21]:

- slovná/názvoslovná výmena dát
- slovník a terminologický manažment
- termínová extrakcia
- riadiaci jazyk
- vyhľadávanie informácií
- slovníkový vývoj
- ontológie

Aktuálna verzia pridala hlavu štruktúry ako TBX, rezervu mnohojazyčné ontológie, lepšie XML členenie a niekoľko nástrojov a podporovaných komponent.

2.2.1 OLIF metamodel

Základná architektúra OLIF bola konceptovo-založená (t.j. základná jednotka je sémantická entita); ale rozdielna od TBX, koncepty v OLIF sú definované pre daný jazyk. Koncepty tvoria uzly OLIF vstupu. Medzi konceptmi sú väzby, ktoré mieria z jedného konceptu k inému; tieto väzby môžu byť jednojazyčné (v prípade slovníka synonymických spojení) alebo mnohojazyčné (v prípade prekladov). Následkom toho môže byť metamodel OLIFu charakterizovaný nasledovným spôsobom:

1. Je konceptovo-založený, ale koncepty sú jednojazyčné a majú jazykové poznámky.
2. Je mnohojazyčný (môžu byť väzby z konceptu pre mnoho cieľov jazykových uzlov), ale orientované (väzby majú zdroj a cieľ, a nemôžu sa jednoducho vrátiť späť).

2.2.2 Heslá v OLIF

OLIF definuje heslo (entry) iným spôsobom ako ac.dtd. Heslo v slovníku zahŕňa slovný výraz a jeho príslušné časti (v prekladovom slovníku je to preklad slova, jeho výslovnosť, gramatické poznámky apod.) OLIF vstup je charakterizovaný štyrmi typmi informácie: kanonická forma (popisuje malé rozdiely medzi slovami), jazyk, slovný druh a sémantická značka.

Vstupné uzly môžu mať jazykové poznámky. Také poznámky sa vzťahujú na jazykové a názvoslovné položky, ktoré môžu byť výmennými a zoskupené podľa hodnôt jazykových popisov: morfologické, syntaktické, sémantické a organizačné informácie.

Heslá môžu byť prepojené väzbami. Poznáme dva základné typy väzieb:

- Väzby, ktoré kombinujú jednojazyčné vstupy (krížové odkazy)
- Väzby kombinujúce vstupy rozdielnych jazykov (preklady).

Väzby sú orientované, t.j. vedú zo zdrojového vstupu (charakterizovaného kľúčovým popisom) k cieľovému vstupu (charakterizovaného ďalším kľúčovým popisom).

2.2.3 OLIF ako výmenný formát

Koncept výmenného formátu odráža skutočnosť, že rôzne systémy používajú rôzne interné reprezentácie pre slovníkový materiál: napríklad niektoré slovníky rozoznávajú medzi jednoduchým slovom a mnohovýznamovým slovom, iné nie.

Ako výsledok každý systém podieľajúci sa na výmene musí poskytovať prevodníky z a do OLIF, čím je vlastný formát konvertovaný do výmenného formátu. Také prevodníky čelia počtu výziev, stanovená podmienka takýchto konverzií by mala byť plnoautomatická a kompletná, t.j. konverzia slovníka vstupného súboru do OLIF a späť by mala mať výsledok 1:1 v tom istom slovníkovom súbore.

2.3 ISLE/MILE

ISLE je štandard podporovaný EC a NSF v programe HLT (Human Language Technology). Je pokračovaním dlhého trvania iniciatívy EAGLES.

Cieľom ISLE je vyvíjať, šíriť a propagovať faktografické HLT normy a smernice pre jazykové zdroje, pomôcky a produkty vnútri medzinárodných rámcov (v tomto kontexte EU-US medzinárodnej výskumnej spolupôsobiacej iniciatívy).

MILE (Multilingual ISLE Lexical Entry) je štandard, ktorý je výsledkom skúmania založený na EAGLES/PAROLE. Vyjadruje reprezentáciu mnohojazyčných informácií v rámci vrstvených slovníkov reprezentujúcich štandard. Na rozdiel od OLIF, MILE neobsahuje len informačné položky, ktoré sú k dispozícii v dnešných MT slovníkoch, ale zamýšľajú vyjadriť kompletný slovný popis, vrátane sémantickej reprezentácie a mnohojazyčnosti.

MILE nie je výmenný štandard, ale reprezentuje štandard a môže byť mapovaný do niekoľkých rozdielnych výmenných formátov. MILE si možno predstaviť ako vysoko modulárnu a snáď vrstvenú štruktúru (s rozdielnymi hodnotami doporučení), za účelom zlepšiť flexibilitu reprezentácie a integrácie existujúcich zdrojov.

Tak ako je vrstvený prístup, MILE heslá môžu definovať prepísanie podmienok za účelom priamej špecifikácie obmedzujúcich nastavení podľa prenosného kontextu (napr. cieľový priamy predmet musí byť v množnom čísle) majúceho vplyv na jednojazyčný popis vstupu.

3 Implementácia

V tejto kapitole opíšem postup práce pri operáciách so vstupnými textovými, neskôr XML dokumentmi. Uvediem niekoľko príkladov pre lepšiu predstavu, a tiež naznačím problémy, na ktoré som pri implementácii narazil. Samotná implementácia bola rozdelená do niekoľkých pracovných fáz. Prvou fázou bolo správne previesť vstupné textové dokumenty slovníka do XML podoby. Druhou bola kontrola prevedených XML dokumentov podľa ac.dtd. Zo vzniknutých XML dokumentov som sa snažil získať určité štatistiky.

3.1 Použitie XML

V prvej kapitole som sa priblížil jazyk XML a tiež poukázať na jeho vlastnosti. Podľa týchto vlastností, hlavne výhod, je možné vyvodiť závery, prečo je tento formát vhodný pre použitie v súvislosti so slovníkmi. Tieto vlastnosti ešte zhrniem. Formát XML pre tvorbu slovníku bol teda zvolený na základe nasledujúcich predností tohto jazyka:

- Formát jazyka je textový a tým prístupný pre každého.
- Každý dokument má pevnú štruktúru, ktorá je definovaná v deklarácii typu dokumentu.
- Výhoda transformácie do iného formátu pomocou XSL.
- Možnosť prepojení s inými dokumenty.
- Jednoduché prispôsobenie slovníku užívateľským predstavám, príkladom môže byť vizualizácia.

XML síce neinterpretuje význam jednotlivých značiek, ale pomocou značiek sa často určuje význam ich obsahu. Majme napríklad časť XML kódu s týmto obsahom: `<slovo>obsah</slovo>`. Tento význam je človeku zrozumiteľný, a to i bez akéhokoľvek formálneho určenia. V slovníku potrebujeme práve určiť významy jednotlivých častí - slov v slovníku a popísať štruktúru tohto slovníka. Každý XML dokument, teda aj slovník, je možné predstaviť si ako usporiadaný strom. Táto všeobecnosť, resp. neurčitosť, je súčasne prednosťou i slabinou jazyka XML. Pomocou XML je možné spracovať rôzne typy štruktúrovaných dát prostredníctvom jednoznačnej syntaxe. Hoci som spomínal, že pri slovníku môžeme popísať význam jednotlivých častí, XML sa priamo nezaoberá použitými dátami a sémantikou. Význam jednotlivých častí slovníka teda určíme názvom jednotlivých tagov, pričom tomuto významu rozumie človek, nie však počítač.

3.2 Prevod vstupných súborov do XML

Je potrebné popísať, akú formu vlastne mali vstupné dáta, preto sa v tejto kapitole venujem práve spomínanému problému, a tiež prvotnej fáze práce so vstupnými dátami – upravenie vstupných súborov do správne vytvoreného XML.

Pracoval som s prekladovým anglicko-českým slovníkom. Vstupné súbory slovníka boli v textovom formáte. Je to 21 súborov so slovami po a až v. Pre predstavu uvediem časť slovníka – uloženie anglického slova *kickoff*:

```
<hwen1>kickoff</hwen1> <pronun1>[kikof] </pronun1>
<sectn1>1 </sectn1><indcs1>sport. </indcs1><hwecs1>výkop</hwecs1>
<indcs1>v kopané </indcs1>
<sectn1>2 </sectn1><indcs1>hovor. </indcs1><hwecs1>začátek,</hwecs1>
<hwecs1>otevření,</hwecs1> <hwecs1>zahájení</hwecs1>
<indcs1>podniku </indcs1>
<partp1></partp1>
<phrwen1>~ circle</phrwen1> <phrecs1>středový kruh</phrecs1>
<indcs1>v kopané
@@@@@
</indcs1>
```

V danom slovníku sa je obsiahnutých mnoho elementov. Popíšem iba tie, ktoré sa nachádzajú v predchádzajúcom príklade. Element <hwen1> obsahuje anglické heslo, <pronun1> jeho výslovnosť. <sectn1> určuje poradie významov pre anglický výraz. Element <indcs1> je v češtine vyjadrená príslušnosť k určitým situáciám, v ktorých sa prekladané slovo používa. <hwecs1> je preklad anglického výrazu (jeho význam sa často vzťahuje k elementu <indcs1>). Elementy <phrwen1> a <phrecs1> sa týkajú fráz - <phrwen1> obsahuje frázu, v ktorej sa daný anglický výraz často vyskytuje, a <phrecs1> je preložená fráza do češtiny.

Je zrejmé, že vstupné slovníkové dáta už obsahujú elementy. V podstate by stačilo premenovať vstupné súbory s príponou .txt na .xml a mali by sme správne vytvorené (well-formed) XML dokumenty. Vstupné súbory však potrebovali viac úprav. Správne vytvorený XML dokument musí obsahovať namiesto znakov <, >, &, ‘, “, reťazce znakov <, >, amp;, ', ", ktorými som pôvodné znaky nahradil.

Základnou požiadavkou pre správne vytvorený dokument je, aby bol celý jeho obsah uzatvorený práve v jednom elemente – nazýva sa koreňový element. Preto bolo potrebné

vložiť tento koreňový element, v našom prípade <superitem> a prológ do každého súboru.

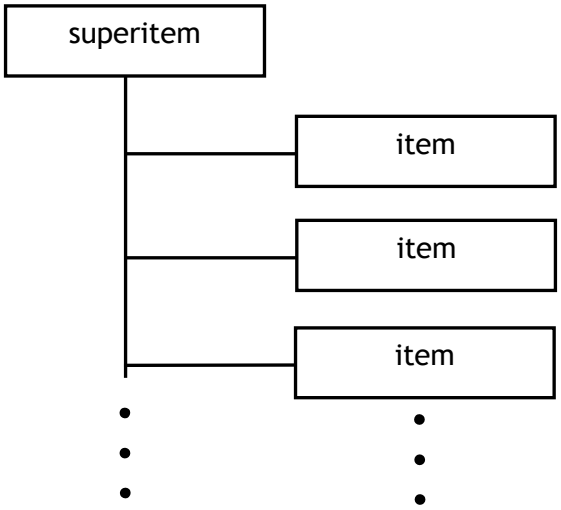
Prológ

Úvodný oddiel dokumentu sa môže a mal by sa začínať deklaráciou XML [18], ktorá špecifikuje použitú verziu XML. Konkrétne deklarácia XML, ktorú som použil, vyzerá nasledovne:

```
<?xml version="1.0" encoding="utf-8"?>
```

V súčasnosti jediná platná verzia je 1.0. V deklarácii môže byť nepovinne uvedená aj jazyková špecifikácia. Použil som deklaráciu, ktorá hovorí, že dokument je vytvorený v 8-bitovom kódovaní UNICODE UTF-8.

Kvôli ľahšej práci pri ďalšom spracovaní XML súborov - transformácii pomocou XSLT bolo vhodné uzavrieť anglický výraz a všetky jeho príslušné časti pomocou elementu <item> do jedného celku. Na to bolo potrebné vyhľadať element <hwen1>, pred ktorým som vkladal element </item>, ktorým som ukončoval predchádzajúci celok a elementom <item> som začínal ďalší. Takýmto spôsobom XML dokumenty získali lepšiu štruktúru, ktorá vyzerá takto:



Obrázok 4.: štruktúra upravených vstupných dokumentov

Podobne ako v predchádzajúcom prípade, potreboval som pre ďalšie jednoduchšie spracovanie obaliť ďalšie význam hlavného výrazu do jedného elementu. Nazval som ho <sec>. Vkladanie tohto elementu nebolo celkom triviálne, preto som použil skript napísaný v jazyku python. V tomto programe som prechádzal všetkými riadkami každého zo súborov slovníka a hľadal element <sectn1>, ktorý určuje poradie významu hlavného

výrazu. Týmto elementom sa začína časť, ktorú chceme obaliť, preto po nájdení <sectn1> a zistení, že je tento element prvým významovým elementom, vkladám element <sec>. Po nájdení ďalších významov konkrétneho výrazu bolo potrebné ukončiť predchádzajúcu významovú časť. Takéto ukončenie je potrebné aj v prípade, že sa končí celkový spracovávaný výraz, t.j. narazíme na element </item>.

Po týchto úpravách boli vstupné dokumenty správne vytvorenými (well-formed) XML dokumentmi a mohol som ich ďalej spracovávať.

3.3 Transformácia XML dokumentov pomocou XSLT

V kapitole 1.3 som spomínal, že XML dokument môžeme pretransformovať pomocou XSLT do rôznych výstupných formátov, akými sú XML, HTML, textový dokument a ďalšie. V našom prípade bolo potrebné previesť správne vytvorené (well-formed) XML dokumenty na validné XML dokumenty podľa ac.dtd.

Na transformáciu som potreboval vytvoriť dokument XSLT. Je to „klasický" XML dokument, pretože používa XML syntax. Aby bolo možné prvky v XSLT dokumente kombinovať, používajú sa menné priestory, o ktorých som hovoril v kapitole 1.2.1. Pri XSLT sa používa menný priestor s prefixom xsl. Celý štýl je uzatvorený v prvku <stylesheet>:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

Adresa http://www.w3.org/1999/XSL/Transform identifikuje menný priestor.

Samotný štýl sa skladá predovšetkým zo šablón, ktoré definujú, ako sa majú jednotlivé časti XML dokumentu pretransformovať. Základný tvar šablóny je:

```
<xsl:template match="výraz">
     obsah_šablóny
</xsl:template>
```

Obsah šablóny presne definuje, ako sa časti transformovaného dokumentu vyhovujúce výrazu budú spracovávať. V obsahu šablóny som väčšinou používal ďalšie konštrukcie XSLT, pričom sa dajú použiť aj prvky z výsledného dokumentu.

XSLT procesor na začiatku svojej práce načíta vstupný XML dokument a vytvorí si jeho stromovú štruktúru. Tento strom prechádza v poradí, v akom sú prvky uvedené v dokumente (prechod do hĺbky). Ak je nájdená šablóna zodpovedajúca uzlu v strome,

začne sa jej obsah zapisovať na výstup. Dôležité je, že ďalší potomkovia uzla, pre ktorý bola vybraná šablóna, už nie sú automaticky spracovávaní. Toto správanie som však často upravoval pomocou XSLT inštrukcie <xsl:apply-templates/>. Táto inštrukcia hovorí, aby sa porovnal každý prvok, ktorý je potomkom daného uzla, so šablónami v XSL súbore. Atribút *select* určuje potomka, ktorý má byť spracovaný:

```
<xsl:template match="superitem">
     <xsl:element name="superentry">
          <xsl:apply-templates select="item"/>
     </xsl:element>
</xsl:template>
```

Prvok xsl:element vkladá prvok do výstupného dokumentu. Meno prvku je predané ako atribút name.

Dôležitým, často používaným prvkom v mojom XSL štýle, je xsl:value-of, ktorý kopíruje hodnotu uzlu zo vstupného dokumentu na výstupný. Atribút *select* prvku xsl:value-of určuje, ktoré uzly majú byť vybrané.

```
<xsl:value-of select="." />
```

Tento prvok vracia množinu uzlov, ktoré obsahuje uzol predaný funkcii v parametri. Jednoducho povedané, vstupný obsah konkrétneho uzla sa prekopíruje na výstup. Obvykle uzol, ktorý práve spracovávame a uzol kontextu sú rovnaké uzly.

Pomocou xsl:attribute som vkladal atribúty do výstupu. Atribúty sa do výstupného dokumentu zahrnú jednoducho tak, že použijeme v šablóne ich reťazcové vyjadrenie.

```
<xsl:template match="hwen1">
     <xsl:element name="orth">
          <xsl:if test="following-sibling::hwsen[1]">
               <xsl:attribute name="type">homonym:<xsl:value-of
               select="../hwsen" /></xsl:attribute>
          </xsl:if>
          <xsl:value-of select="." />
     </xsl:element>
</xsl:template>
```

xsl:if v predchádzajúcom prípade som použil pre rozhodovanie. Tento prvok buď zahrnie, alebo nezahrnie určitú časť kódu do výsledného dokumentu, v závislosti na vstupných dátach. Podmienka sa zadáva v atribúte test a je to XPath výraz, ktorý vracia logickú hodnotu. V našom prípade je to following-sibling, ktorý zahŕňa všetkých potomkov uzla, ktorého sa týka – konkrétne je to uzol s názvom hwsen.

```
<xsl:template match="undef1" mode="insec">
      <xsl:element name="indicator">
            <xsl:choose>
                  <xsl:when test=" .='v. ' ">
                        <xsl:attribute name="type">ref</xsl:attribute>
                  </xsl:when>
                  <xsl:when test=" .='= ' ">
                        <xsl:attribute name="type">ref</xsl:attribute>
                  </xsl:when>
                  <xsl:when test=" .='též ' ">
                        <xsl:attribute name="type">othform</xsl:attribute>
                  </xsl:when>
                  <xsl:when test=" .='v. též ' ">
                        <xsl:attribute name="type">othform</xsl:attribute>
                  </xsl:when>
                  <xsl:when test=" .='/ ' ">
                        <xsl:attribute name="type">alt</xsl:attribute
                  </xsl:when>
                  <xsl:otherwise>
                        <xsl:attribute name="type">others</xsl:attribute>
                  </xsl:otherwise>
            </xsl:choose>
            <xsl:value-of select="." />
      </xsl:element>
</xsl:template>
```

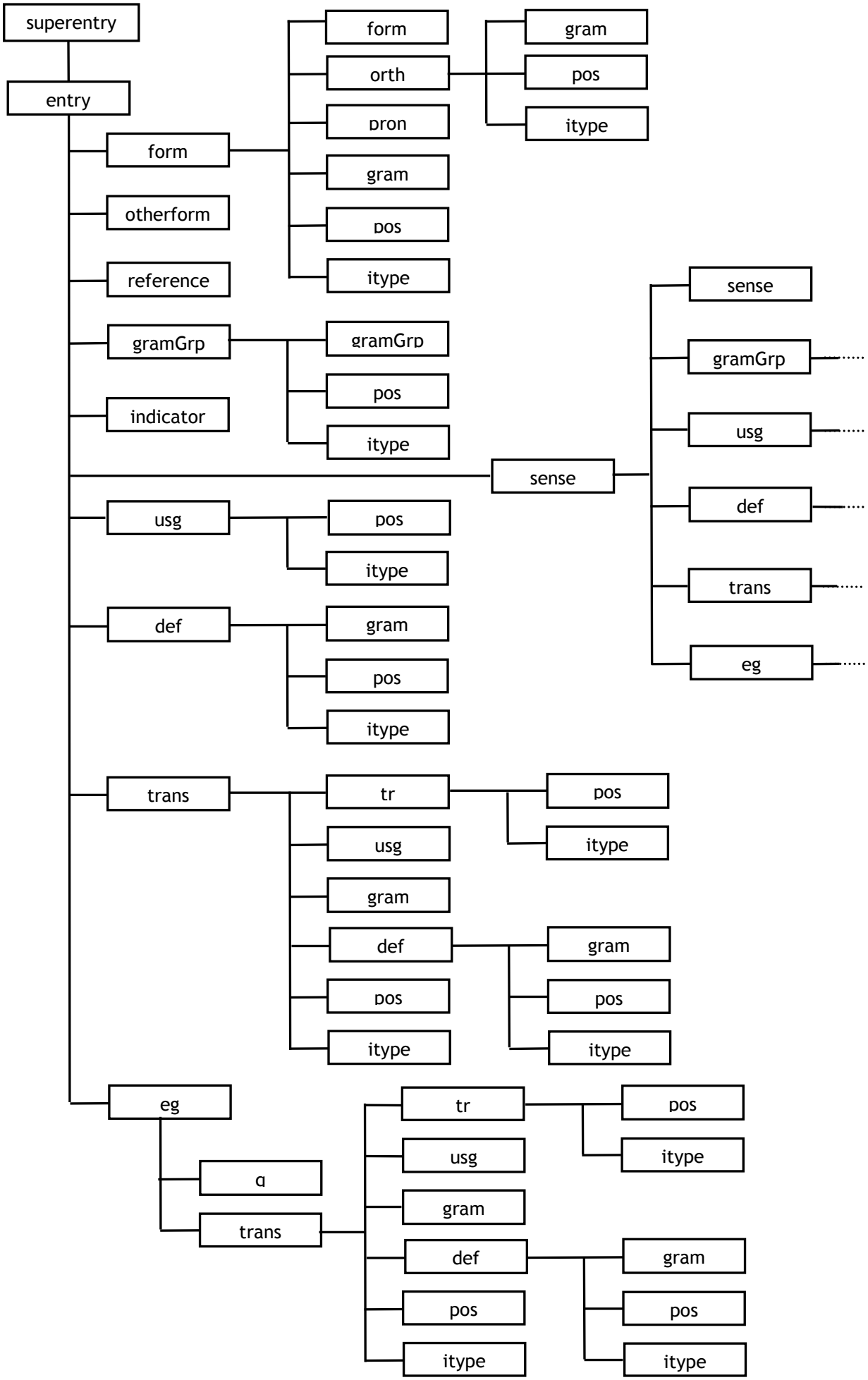
Prvok `xsl:choose` vyberá jednu z viacerých možností podľa rôznych podmienok. Každá podmienka a jej priradená výstupná šablóna je zaistená potomkom `xsl:when`. Atribút `test` prvku `xsl:when` je výraz výberu s booleovskou hodnotou. Ak vyhovuje viac podmienok, je prevedená iba prvá podmienka. Ak nevyhovuje žiadna, použije sa hodnota prvku `xsl:otherwise`, ktorý je potomkom prvku `xsl:choose`.

Pomocou týchto prvkov som teda navrhol XSLT štýl. Samozrejme jazyk XSL je oveľa rozsiahlejší, pre túto konkrétnu transformáciu však stačili spomínané prvky.

Na obrázku 5 je znázornená stromová štruktúra pretransformovaných slovníkových XML dokumentov. Koreňovým elementom je element `superentry`, ktorého bezprostrednými potomkami sú elementy `entry`, obsahujúce všetky príslušné časti anglického výrazu. V jeho potomkovi `form` sa nachádza ortografia anglického výrazu (`orth`), gramatické poznámky (`gram, pos`), výslovnosť anglického výrazu (`pron`), celé znenie skratky (`itype`), ak je obsiahnutá elementom `orth`. Prvok `gramGrp` zahŕňa skupinu gramatických poznámok: `gram, pos, itype`. Element `usg` vyjadruje poznámky k výrazu alebo k jeho niektorému významu – upresňuje dané slovo pre kontext jeho použitia (napr. *kniž.* alebo *zkratka*). Ďalším prvkom je prvok `def` určujúci slovné spojenia týkajúce sa konkrétneho anglického slova. Prvok `trans` znamená preklad – môže to byť priamo preklad anglického výrazu (`tr`), upresňujúcej poznámky k tomuto výrazu (`usg`) a ďalších už spomínaných elementov. Príklad, pomocou ktorého sa v slovníku vysvetľuje bližšie význam prekladaného slova, je v tomto prípad obsiahnutý elementom `eg`. V jeho

potomkovi q je tento príklad zapísaný v angličtine, v potomkovi trans, ktorý môže obsahovať už spomínané elementy, sa nachádza preklad príkladu. Element sense obsahuje ďalší význam prekladaného anglického výrazu. V tomto prvku sú v zahrnuté takmer všetky prvky nachádzajúce sa v elemente entry.

Poslednými elementmi v transformovanom slovníku sú indicator, reference a otherform. Je to špeciálnejšia skupinka prvkov, pretože sa nenachádzajú v ac.dtd, ale bolo potrebné ich vytvoriť, aby sa nestratili dáta, ktoré obsahujú. Element indicator napomáha určiť nadchádzajúci element. Element reference zahŕňa odkaz na príbuzný alebo podobný výraz v slovníku. V elemente otherform sa nachádzajú ďalšie formy anglického výrazu.



Obrázok 5.: Stromová štruktúra transformovaného slovníka

3.4 Získanie štatistík slovníka

Pod pojmom štatistika rozumieme odbor zaoberajúci sa skúmaním a kvantitatívnou charakteristikou rozličných javov [19]. Presnejšie ide o teoretickú a praktickú činnosť, ktorá skúma stav, priebeh a výsledky rozličných javov a činností na základe ich kvantitatívnej analýzy. Z toho môžem odvodiť: slovníková štatistika potom skúma a hodnotí výsledky stavu slovníkov na základe získaných číselných údajov o slovníkoch.

Mojou úlohou teda bolo získať určité číselné údaje o spracovávanom slovníku. Tento problém som rozdelil na dve časti. V prvej som sa snažil získať základné údaje o slovníku, ako je počet slov, počet jednovýznamových a viacvýznamových slov, počet homoným a variácií výrazu apod. Druhá časť slúžila na získanie ďalších, doplnkových údajov napr. celkový počet významov, aký je ich priemerný počet na slovo, ktoré slovo má maximálny počet významov apod.

Na získavanie štatistík som použil skriptovací jazyk python.

Pre prácu s XML dokumentmi som zvolil rozhranie DOM, ktoré pristupuje k XML dokumentu objektovým spôsobom. Z XML dokumentu sa v pamäti počítača vytvorí jeho reprezentácia v podobe objektu, ktorého metódy slúžia pre ďalšiu prácu s dátami. Štruktúra dokumentu je vyjadrená hierarchiou čiastočných objektov, ako sú napr. „element“, „attribute“ apod. Neskôr som zistil, že bolo lepšie použiť SAX, pretože pri rozhraní DOM sa naplno prejavila najväčšia nevýhoda jazyka python – rýchlosť. DOM je totiž veľmi náročný na pamäť, čo pri súboroch, ktoré som spracovával nebol zanedbateľný fakt. SAX na rozdiel od rozhrania DOM neudržuje celý XML dokument v pamäti, ale iba práve čítanú časť. Je teda vhodnejší pre obsluhu veľkých XML dokumentov, na čo som bohužiaľ prišiel až po implementácii rozhraním DOM.

Hlavná myšlienka implementácie získavania jednoduchých štatistík bola prechádzať XML dokumentmi a získavať potrebné údaje. Na to, aby som prechádzal jednotlivými elementmi XML stromu každého spracovávaného dokumentu, som potreboval iterátor, ktorý mi to umožní. Pri implementovaní iterátora som využil rekurzívny prístup. Takto implementovaný iterátor vracal hodnoty uzlov v opačnom poradí.

Prechádzal som teda každým XML dokumentom a počítal príslušné elementy. S rozhraním DOM sa jednoducho vytvorí stromová štruktúra výstupného dokumentu – vytvoria sa uzly a určí sa, potomkom ktorého uzla je vytvorený uzol. Na koniec sa do neho zapíše hodnota, ktorú uzol bude obsahovať vo výstupnom dokumente.

Výsledný XML dokument s konkrétnymi výsledkami spracovávaného slovníka jednoduchých štatistík vyzerá takto:

```
<statistics>
    <entries> 103077 </entries>
    <orths>
        <simple count="98693">
            <onesense> 71658 </onesense>
            <moresense> 27035 </moresense>
        </simple>
        <homonym> 4361 </homonym>
        <variation> 2017 </variation>
    </orths>
    <pronunciation> 100928 </pronunciation>
    <usage> 182411 </usage>
    <abbrevs> 14795 </abbrevs>
    <examples>
        <english> 89265 </english>
        <czech> 462910 </czech>
    </examples>
</statistics>
```

Elementy	Význam elementov
<entries>	celkový počet záznamov v slovníku
<simple count= >	počet kľúčových slov v angličtine
<onesense>	počet jednovýznamových hesiel
<moresense>	počet viacvýznamových hesiel
<homonym>	heslá označené ako homonymá
<variation>	heslá označené ako variácie – iné tvary anglického výrazu
<pronunciation>	počet hesiel doplnených o výslovnosť
<usage>	vysvetľujúce poznámky
<abbrevs>	elementy vysvetľujúce význam skratiek
<english>	anglické frázy a výrazy
<czech>	české frázy a výrazy

V druhej časti získavania pokročilejších štatistík bol postup podobný ako v prvom prípade. Pomocou rovnakého iterátora som prechádzal opäť stromovú štruktúru XML dokumentov. Zistiť počet záznamov a významov nebolo zložité. Zisťoval som však, aký je maximálny počet významov anglického hesla, a tiež zistiť, o ktoré konkrétne heslo sa jedná. Pri prechode stromovou štruktúrou slovníka som hľadal element sense. Pri narazení na tento element, určujúci ďalšie významy anglického hesla, som porovnal

atribút, pomocou ktorého zistíme úroveň daného významu – maximálna úroveň určuje počet významov. Vďaka porovnávaniu úrovní významu s maximálnou hodnotou sa na konci dozvieme maximálny počet významov hesla. Premenná `indexOrth` slúži na indexovanie v poli anglických slov (`elmntOrth[]`), ktoré sme si predtým naplnili. Tento index (zvyšuje sa pri každom výskyte elementu `orth`) slúži v tomto prípade na nájdenie slova s maximálnym počtom významov.

Pri zisťovaní, či sa v anglických heslách nachádzajú slová obsahujúce písmená s diakritikou som použil jednoduchý regulárny výraz:

```
orth = node.getElementsByTagName('orth')[0].firstChild.nodeValue
if re.search(u'[áčďéěíňóřšťůýž]', orth):
  countOrthWithSpecChar = countOrthWithSpecChar + 1
```

Výsledný XML dokument s konkrétnymi výsledkami spracovávaného slovníka ďalších štatistík vyzerá takto:

```
<statistics>
    <entries> 103077 </entries>
    <sense>
        <maxsense> 93 </maxsense>
        <maxsenseword> go </maxsenseword>
        <avgsense> 2.2500 </avgsense>
    </sense>
    <orthwithouttrans> 19687 </orthwithouttrans>
    <orthwithspecchar> 209 </orthwithspecchar>
</statistics>
```

Elementy	Význam elementov
`<entries>`	celkový počet záznamov v slovníku
`<sense>`	celkový počet významov hesiel
`<maxsense>`	maximálny počet významov jedného hesla
`<maxsenseword>`	heslo s maximálnym počtom významov
`<avgsense>`	priemerný počet významov na heslo
`<orthwithouttrans>`	počet hesiel bez prekladu
`<orthwithspecchar>`	počet hesiel obsahujúcich písmená s diakritikou

3.4.1 Vyhodnotenie štatistík

Štatistiky dokázali, že sa jedná o bežný prekladový slovník. Výsledky štatistík, dá sa povedať, vyšli podľa očakávania. Hoci, ako laika, ma niektoré hodnoty prekvapili. Napríklad som očakával iný pomer viacvýznamových k jednovýznamovým heslám. Tento pomer je približne 1:2,5. Jedno z najväčších položiek boli vysvetľujúce poznámky, čo poukazuje na význam, ktorý sa kladie na pochopenie daného hesla, príp. príkladu. Najväčšiu časť slovníka tvoria české výrazy a frázy, čo nie je prekvapujúce vzhľadom k tomu, na čo je slovník určený. Opäť to poukazuje na snahu čo najlepšie vysvetliť význam prekladaného hesla. Takmer všetky heslá sú doplnené o výslovnosť. Počet významov na jedno heslo je viac ako dva významy (priemerne 2,25 významu na heslo). Najväčší počet významov má slovo *go*. Je to až 93 významov. Takmer 20 000 slov nie je preložených. Sú to väčšinou skratky slov, kde sa skôr určuje, čo tá skratka znamená – jej celé znenie v angličtine. Zaujímavé bolo zistenie, že v niektorých anglických výrazoch sa nachádzajú písmená s diakritikou. Nie je ich veľa – 209, ale v tomto prípade môžem povedať, že je to dosť. Podľa slov, ktoré patria do tejto skupiny, usudzujem, že sú to väčšinou prevzaté slová iného pôvodu (pravdepodobne francúzskeho).

4 Záver

Cieľom práce bolo transformovať vstupné slovníkové dáta do XML a získanie informácií o transformovanom slovníku v XML. Operácie so slovníkovými dátami boli celkovo časovo náročné, aj vďaka veľkosti súborov, v ktorých boli uložené.

Pri transformácii bolo potrebné pochopiť štruktúru vstupných dát a ich súvislosť s požadovanou výstupnou podobou. To bolo miestami zložitejšie ako pochopenie jazyka XSLT, ktorým som previedol transformáciu. Nebolo potrebné použiť zložitejší spôsob uloženia slovníka do XML, preto som použil superjednoduchú formu. Rozšírenie práce teda môže spočívať v ukladaní do iného formátu, ktorý ponúka širšie možnosti. Takým formátom je napríklad OLIF.

Musím priznať, že pri implementácii skriptu v jazyku python som zvolil nesprávne rozhranie, ktorým bolo DOM. Ako som už spomínal, DOM je veľmi náročný na pamäť, čo sa značne prejavilo pri spustení skriptu. Chybou bolo, že som sa nechal zlákať jednoduchosťou jeho použitia a neskúsenosťou v tomto smere. Určite bolo vhodnejšie použiť rozhranie SAX – konečný výsledok by sa nezmenil, ale rýchlosť určite.

Vďaka štatistikám som sa dozvedel zaujímavé informácie o slovníkových dátach, s ktorými som pracoval. Je oveľa lepšie svoje domnienky premeniť na reálne údaje. V tomto smere sa dá určite program vhodne rozšíriť, pretože otázok, týkajúcich sa „vlastností“ slovníka, ktoré by sme radi vedeli, je mnoho.

Práca s jazykom XML priam vyžaduje použitie ďalších jazykov a formátov. Pomocou nich som spracovával slovníkové dáta, čo je len malá oblasť ich využitia. Novinkou pre mňa bol taktiež jazyk python, ktorý som si obľúbil. Vlastne so skriptovacími jazykmi som sa prakticky stretol až „tu“. Vďaka tejto práci som teda spoznal pre mňa nové programovacie prístupy a prostriedky, čo hodnotím ako najväčší prínos pre mňa. To však určite lepšie ocením až v budúcnosti - pri stretnutí s nástrojmi, ktoré som práve tu používal.

Literatúra

[1] http://www.w3.org/TR/REC-xml/

[2] http://www.w3.org/XML/1999/XML-in-10-points

[3] http://cs.wikipedia.org/wiki/Slovn%C3%ADk

[4] Kosek, J. *XML pro každého*. Podrobný průvodce, Praha, Grada Publishing, spol. s.r.o., 2000

[5] http://www.w3schools.com/xml/xml_dtd.asp

[6] http://www.w3schools.com/dom/dom_intro.asp

[7] http://www.kosek.cz/xml/schema/uvod.html#vyznam

[8] http://www.w3.org/TR/REC-xml-names/

[9] http://cs.wikipedia.org/wiki/XSD

[10] http://www.thaiopensource.com/trex/

[11] http://www.thaiopensource.com/trex/tutorial.html

[12] http://www.w3schools.com/xpath/default.asp

[13] http://xml.coverpages.org/schematron.html

[14] http://www.ldodds.com/papers/schematron_xsltuk.html

[15] http://en.wikipedia.org/wiki/XSLT

[16] http://cs.wikipedia.org/wiki/Cascading_Style_Sheets

[17] http://en.wikipedia.org/wiki/DSSSL

[18] Extensible Markup Language (XML) 1.0 (Second Edition) - REC-xml-20001006, W3C

[19] Krátky slovník slovenského jazyka. Bratislava, Veda 1997, s. 716

[20] http://cs.wikipedia.org/wiki/Skriptovac%C3%AD_jazyk

[21] http://www.olif.net/

Zoznam príloh

Príloha 1. CD