VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

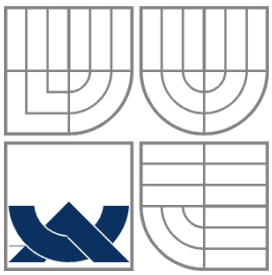
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTERAKTÍVNA VIZUALIZÁCIA XML

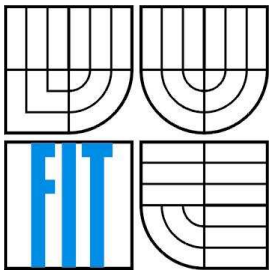
BAKALÁRSKÁ PRÁCA
BACHELOR´S THESIS

AUTOR PRÁCE MICHAL FILUS
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTERAKTÍVNA VIZUALIZÁCIA XML

INTERACTIVE VISUALIZATION OF XML

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL FILUS

VEDOUCÍ PRÁCE
SUPERVISOR

ING.PETR CHMELAŘ

BRNO 2008

Abstrakt

Práca sa zaoberá problematikou vizualizácie XML dokumentov. Cieľom bolo získanie informácii z oblasti XML dokumentov, spoznať ich štruktúru, jazyky pre ich popis a nakoniec vytvorenie funkčnej aplikácie na ich vizualizáciu. V práci sa ďalej nachádza stručný popis programovacieho jazyka Java, je poukázané na jeho výhody a vhodnosť použitia pre programovanie aplikácii pracujúcich s XML. Sú vysvetlené dôvody vizualizácie XML a obsahuje prehľad existujúcich aplikácii s ich stručným popisom. Následne je v práci spracovaný návrh aplikácie pre vizualizáciu XML a popis jej implementácie.

Výsledkom práce je vytvorená aplikácia XMLvisualizer naprogramovaná v jazyku Java.

Kľúčové slová

XML, vizualizácia XML, DTD, XSD

Abstract

This work deals with the problem of visualization of the XML documents. The goal of it was to receive information about XML documents, to become a knowledge of their structure, to know the languages for their description and finally, to create functional application for visualization. This work contains a short description of the programming language Java, it points out to their advantages and the availability of use in programming application which works with XML documents. The reasons of XML visualization are given in this work, which includes brief descriptions of existing applications for visualization, too. Thereafter the scheme of the application and description of its implementation is worked out in it.

The result of this work is an application called XMLvisualizer created in Java programming language.

Keywords

XML, XML visualization, DTD, XSD

Citácia

Michal Filus: Interaktívna vizualizácia XML, bakalárska práca, Brno, FIT VUT v Brně, 2008.

Interaktívna vizualizácia XML

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením ing.Petra Chmelaře. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Michal Filus
8. máj 2008

Poďakovanie

Moje poďakovanie patrí vedúcemu bakalárskej práce ing. Petru Chmelařovi za jeho odbornú pomoc a cenné rady pri riešení tejto práce.

Obsah

1 Úvod

V súčasností si nevieme predstaviť život bez informačných technológii. Zasahujú do všetkých oblastí našich záujmov. Nastala potreba efektívneho a rýchleho prenosu informácii, a v súvislosti s tým aj potreba formátu, ktorý by umožňoval prenos dát medzi rôznymi aplikáciami. Doterajšie možnosti prenosu informácii neboli univerzálne ani otvorené, prenos medzi rôznymi prostrediami predstavoval problém. Obrat nastal s príchodom jazyka XML, ktorý priniesol do tejto oblasti veľký pokrok. Formát bol otvorený, umožňoval výmenu informácii medzi rôznymi platformami operačných systémov a aplikáciami. Vďaka svojím vlastnostiam preto XML našlo uplatnenie v mnohých oblastiach.

XML ako textový formát je pre človeka čitateľný. Pri rozsiahlejších dokumentoch je však orientácia v dokumente problematická. Zlepšenie prináša vizualizácia XML dokumentov. Jazyk XML definuje štruktúru dokumentu, neurčuje však jeho vzhľad. Tento sa musí k dokumentu externe pridávať napríklad pomocou CSS. Tento postup sa však nedá obecne aplikovať na všetky dokumenty, pretože by si to vyžadovalo priamy zásah do ich štruktúry. Existuje niekoľko aplikácii, ktoré obecnú vizualizáciu umožňujú, väčšinou sú však zamerané špecificky na určitý typ dokumentov.

Tento dokument sa zaoberá problematikou vizualizácie obecných XML dokumentov. Prvá polovica práce je venovaná teoretickému úvodu do problému. Skladá sa z troch častí. V prvej je spracovaná teória k XML, ich históriou, použitím a jazykmi pre ich popis. Druhá časť je venovaná jazyku Java, spracovaním XML v jave a posledná rozoberá problém vizualizácie a predstavuje niektoré dostupné aplikácie v súčasnosti. Zvyšná polovica je venovaná samotnému riešeniu problému vizualizácie, návrhom aplikácie a popisom jej implementácie. V závere sa nachádza zhodnotenie výsledkov práce a možnosti ďalšieho vývoja aplikácie.

2 Teória

2.1 XML

2.1.1 Úvod

XML (*eXtensible Markup Language)*, v preklade rozšíriteľný značkovací jazyk, ktorý bol vyvinutý a štandardizovaný konzorciom W3C (World Wide Web Consortium). Pochádza z rodiny jazykov SGML (*Standard Generalized Markup Language*), z ktorej pochádza aj známy značkovací jazyk HTML používaný pri tvorbe WWW. V súčasnej dobe sa jedná o jeden z najdôležitejších formátov výmeny dát štruktúrovaným spôsobom. Jeho rozšírenie sa značne zvýšilo v posledných rokoch vďaka jeho dobrým vlastnostiam, viac na [2].

- Popisuje dáta nezávisle na ich platforme, čím sa stáva veľmi silným prostriedkov pre zaistenie prenositeľnosti dát. Nezávislosť sa dosahuje tým, že obsahom XML dokumentu sú textové informácie, dokument ako taký je textový súbor
- XML je otvorený formát, značky je možné doplňovať podľa potreby
- Značky jazyka majú pevnú gramatiku, ktorú je nutné striktne dodržovať. Naopak porušenie týchto pravidiel sa dá vďaka tejto gramatike ľahko skontrolovať.
- Značkovanie v XML jasne popisuje význam uložených dát na rozdiel od HTML, kde značky prevažne určujú, ako sa majú dáta zobraziť, viac na [11].

2.1.2 História

Predchodcom XML bol jazyk SGML**,** norma ISO 8879, ktorý vznikol pre potreby americkej armády. Bol vyvinutý pre jednotný spôsob správy technickej dokumentácie. Jazyk SGML bol navrhnutý s veľmi komplexnou gramatikou. Z tohto dôvodu je jeho spracovanie veľmi zložité, a začali vznikať nové jazyky redukciou možností pôvodného SGML. Najznámejším derivátom bol jazyk HTML, ktorý však nie je vhodný pre výmenu dát.

Na základe skúseností s HTML sa začal v roku 1996 vyvíjať jazyk XML. Vývoj bol ukončený v roku 1998 a výsledkom bola verzia 1.0 jazyka, ktorá je dnes bežne používaná. Novšia verzia 1.1 obsahuje oproti staršej len minimálne zmeny.

2.1.3 Použitie

Použitie jazyka XML sa značne líši od spôsobu použitia HTML. Podobnosť týchto dvoch jazykov pramení z toho, že obidva sú odvodené od štandardu definície jazyka SGML. Jazyk XML nebol navrhnutý pre zobrazovanie dát, umožňuje odčlenenie obsahu od dizajnu.

Hlavnou výhodou XML dokumentov je, že informácie v nich uložené sú použiteľné viacerými typmi aplikácii, tento formát je veľmi univerzálny. Je možné zdieľať informácie medzi programami bez nutnosti akejkoľvek konverzie.

XML dokumenty je možné rozdeliť do dvoch veľkých skupín.

- **Dátovo orientované dokumenty**

  Hlavným účelom je prenos dát medzi aplikáciami. Typické pre tieto dokumenty je používanie veľkého množstva rôznych značiek, ktoré obsahujú štruktúrovanú informáciu, ktorá má podobu čísel alebo krátkych znakov. Tieto druhy dokumentov sú dnes často využívané napríklad ako konfiguračné súbory pre rôzne aplikácie.
- **Textovo orientované dokumenty**

  Tento typ dokumentov je špecifický tým, že obsahuje menší počet značiek a väčšiu časť zaberá samotná informácia. Značky obaľujú reťazce, ktoré predstavujú vety. Tieto

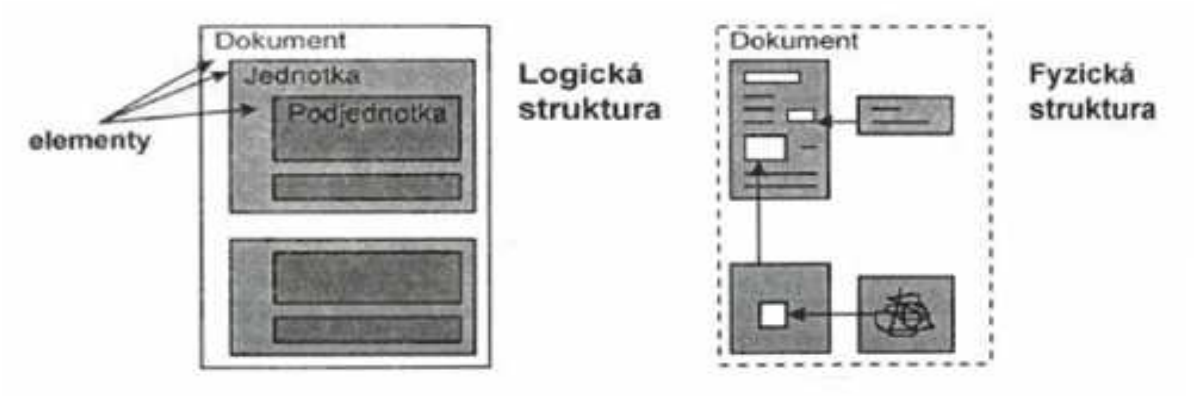
dokumenty nachádzajú uplatnenie v príprave kníh, WWW stránok alebo programovej dokumentácie.

S nasadením XML sa značne počíta v oblastiach webových aplikácii, kde možnosť definovania vlastných značiek, ktoré presne označia významy jednotlivých častí stránky, bude mať pozitívny vplyv na presnosť a rýchlosť vyhľadávania informácii (viď [13]).

2.1.4 Štruktúra jazyka

Dokumenty XML obsahujú informácie v štruktúrovanej forme. Dokument sa skladá z textového obsahu a textových značiek (*tagov*). Môžeme definovať svoje vlastné značky, ktoré určujú štruktúru dát. Syntaktické pravidlá XML dokumentu sú veľmi jednoduché a tiež veľmi presné. Jazyk je ľahký na naučenie a použitie v praxi.

Dokument XML ma logickú a fyzickú štruktúru. Logická štruktúra rozdeľuje dokument do pomenovaných jednotiek a podjednotiek, nazývaných *elementy*. Fyzická štruktúra umožňuje pomenovať a uložiť samostatne časti dokumentu, nazývané *entity*, niekedy aj v rôznych dátových súboroch, aby mohli byť informácie opakovane použité a aby bolo možné vkladať odkazy na dáta, ktoré neodpovedajú štandardu XML, ako sú napríklad obrázky, viac na [8].



Obr 2.1 Logická a fyzická štruktúra dokumentu (prevzaté z [8])

Pravidlá:

- V prvom riadku každého dokumentu je deklarácia XML dokumentu. Je v ňom informácia o použitej XML verzii a informácia o použitom kódovaní v dokumente.
- Elementy tvoria vlastnú stavbu dokumentu. Každý element v XML dokumente musí mať začiatočnú a koncovú značku, elementy bez koncovej značky nie sú povolené. V prípade, že element neobsahuje žiadne dáta, je povolený zápis <element/> ktorý je totožný so zápisom <element></element>. Značky sú citlivé na veľkosť písmen.

- Dokument musí obsahovať koreňový element, tento element je hlavný, všetky ostatné elementy sú do neho vnorené. Vnorené elementy musia byť úplne obsiahnuté v nadradenom elemente, počiatočné a koncové značky sa nemôžu navzájom prekrývať.
- Element môže mať hodnotu a svoje atribúty. Hodnota predstavuje text vložený medzi počiatočnú a koncovú značku elementu. Atribút je vlastnosť elementu, ktorá má hodnotu. Hodnoty musia byť uzavreté v apostrofoch, inak preklad dokumentu spôsobí chybu.
- Názov elementu je ľubovoľný, XML je otvorený jazyk, musí však vyhovovať pravidlám [13].

2.1.5 Jazyky pre popis schémy XML dokumentu

Hlavnou funkciou schémových jazykov je definícia nového jazyka. V praxi to znamená, že v tomto formáte určujeme, ktoré značky je možné v XML dokumente používať a v akých budú vzájomných vzťahoch. Tento schémový súbor popisuje nami vytvorený XML formát. Existuje niekoľko jazykov pre popis schémy dokumentu.

2.1.5.1 DTD

DTD (*Document Type Definition*) je historicky prvým jazykom pre popis schémy XML. Pochádza z textovo orientovaných dokumentov. Je používaný k dvom účelom. Poskytuje syntax k popisu a obmedzeniam logických štruktúr dokumentu, k čomu slúži deklarácia elementov a atribútov a ďalej nachádza použite vo vytvorení logického dokumentu z fyzických entít, kde deklarujeme entity a notácie. Z dnešného pohľadu je však považovaný ako omyl pri vývoji XML dokumentov pre jeho mnohé nevýhody, viac na [11].

- Syntax jazyka je jednoduchá. DTD však nie je napísané v XML jazyku. Z tohto dôvodu nie je možná kontrola správnosti vlastného DTD dokumentu, čo je pri XML samozrejmosť.
- Nemá možnosť popísať dátové typy dokumentu, rozsah typov a ich obmedzenie
- DTD nepodporuje prácu z mennými priestormi , čo je nevýhodné pri práci s veľkými XML dokumentmi
- Jazyk DTD je používaný pre popis dokumentov od samého začiatku, preto je aj dnes využívaný napriek svojím nevýhodám. Vytvorená schéma je vo väčšine prípadov oddelená od XML dokumentu a samostatný súbor s DTD sa k nemu pripája pomocou značky `DOCTYPE`, viď [11, 7].

list.xml

```
<?xml version="1.0"?>
<!DOCTYPE list SYSTEM
"adresarová  cesta/list.dtd">
<list>
<komu>Katka</komu>
<od>Michal</from>
<hlavicka>Stretnutie</hlavicka>
<telo>Uvidíme sa dnes večer pre kinom</telo>
</list>
```

list.dtd

```
<!ELEMENT list (komu, od, hlavicka, telo)>
<!ELEMENT komu (#PCDATA)>
<!ELEMENT od (#PCDATA)>
<!ELEMENT hlavicka (#PCDATA)>
<!ELEMENT telo (#PCDATA)>
```

Obr 2.2 Príklad XML dokumentu a jeho DTD

2.1.5.2 XML Schema

Jazyk *W3C XML Schema* je v súčasnosti najdôležitejším jazykom pre popis schémy dokumentu. Od roku 2001 sa stal všeobecne uznávaným štandardom, ktorý je široko podporovaný. Predstavuje alternatívu k formátu DTD, ktorá je založená na XML. V priebehu krátkej doby sa však ráta s jeho použitím vo väčšine aplikácii a nahradeniu jazyka DTD. Často sa jazyk XML Schema označuje ako XSD (*XML Schema Definition*).

Pomocou XSD je možné definovať:

- Elementy a atribúty, ktoré je možné použiť v dokumente
- Vnorené elementy a počet týchto elementov
- Prázdne elementy a elementy, ktoré môžu obsahovať text
- Dátové typy pre elementy a atribúty
- Predvolené alebo fixné hodnoty pre elementy a atribúty

XSD odstraňuje mnohé nevýhody, ktoré má jeho predchodca, jazyk DTD. Je rozšíriteľný pre splnenie požiadaviek v budúcnosti, obsahuje veľa možností a je silnejší. Sám o sebe je XML dokumentom, nie je teda potrebné sa učiť nový jazyk pre jeho zápis a XSD dokumenty je možné vytvárať a editovať v ľubovoľnom XML editore. Podporuje definície dátových typov a menné priestory. Na obrázku 2.3 je ukážka jednoduchého XML dokumentu a jeho XSD schémy. Ako je možné vidieť, mnohokrát je schéma rozsiahlejšia ako samotný dokument, ktorý popisuje (viď [7]).

list.xml

```
<?xml version="1.0"?>

<list
xmlns="urn:xfilus00:schemata:ukazka.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:xfilus00:schemata:ukazka.1.0 list.xsd">

<komu>Katka</komu>
<od>Michal</od>
<hlavicka>Stretnutie</hlavicka>
<telo>Uvidíme sa dnes večer pred kinom</telo>
</list>
```

list.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:xfilus00:schemata:ukazka.1.0"
xmlns="urn:xfilus00:schemata:ukazka.1.0"
elementFormDefault="qualified">

<xs:element name="list">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="komu" type="xs:string"/>
      <xs:element name="od" type="xs:string"/>
      <xs:element name="hlavicka" type="xs:string"/>
      <xs:element name="telo" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Obr 2.3 Príklad XML dokumentu a jeho XSD schémy

2.1.6 Spracovanie XML dokumentov

Pre spracovanie XML dokumentov sa používajú nástroje zvané *parsery*. Tieto nástroje umožňujú načítavanie elementov, ich vytváranie a editáciu, zápis do dokumentu alebo transformácie XML do iných formátov. Parser nám cez programátorské rozhranie vytvára abstraktný model dokumentu, vďaka čomu pracujeme s jeho jednotlivými časťami.

Existuje viac spôsobov spracovania XML, každý z nich ma definované svoje rozhranie, ktorému vyhovuje určitý typ parseru. Rozhrania je možné rozdeliť do dvoch veľkých skupín.

2.1.6.1 Prúdové spracovanie dokumentu

Typickým predstaviteľom je rozhranie SAX (*Simple API for XML*). Princíp tohto typu spracovania je v sekvenčnom prechádzaní dokumentu, kde je pre každú ucelenú časť vyvolaná udalosť. Tieto udalosti sú potom programátorsky spracované, čo v konečnom dôsledku znamená získavanie informácii o elementoch a atribútoch.

Výhody prúdového spracovania sú v jeho veľkej rýchlosti a pamäťovej nenáročnosti. Práca s rozhraním je jednoduchá, je všeobecne rozšírené a podporované. Medzi jeho nevýhody patrí jeho základná vlastnosť, sekvenčné spracovanie. V praxi to znamená načítavanie dokumentu jedným smerom. Nie je možné sa vracať a dokument sa musí začať čítať od začiatku.

2.1.6.2 Stromová reprezentácia dokumentu

Základným predstaviteľom tohto typu spracovania je DOM (*Document Object Model*). Hlavný rozdiel oproti prúdovému spracovaniu spočíva v načítaní celého XML dokumentu do pamäti počítača. Z toho vyplýva väčšia pamäťová náročnosť a menšia rýchlosť spracovania, pretože v pamäti sa musí

vytvoriť celá stromová štruktúra dokumentu. Preto je možné načítavať len dokumenty, ktoré majú určitú veľkosť, ktorá závisí od veľkosti operačnej pamäti hostiteľského počítača. Rozhranie je obecné a práca s ním je o niečo zložitejšia ako v prípade SAX.

Tento prístup však umožňuje spracovanie údajov voliteľným smerom, je možné sa vracať, preskakovať, obecne sa pohybovať vo vytvorenom strome. Na rozdiel od prúdového spracovania je možné do dokumentu zapisovať, vytvárať nové elementy a atribúty, meniť ich hodnoty, viac na [13].

2.2 Programovací jazyk JAVA

2.2.1 História jazyka

Programovací jazyk *Java* bol vytvorený firmou *Sun Microsystems*. Pôvodne bol tento jazyk určený pre ovládanie spotrebnej elektroniky a iných vstavaných systémov ktoré sú riadené zabudovaným mikroprocesorom. Na začiatku vývoja mal jazyk názov *Oak,* ktorým ho nazvali podľa dubu stojaceho pre domov pána Groslinga, vedúceho vývojového týmu.

Spočiatku sa projektu vývoja jazyka príliš nedarilo, neskôr si však v roku 1993 firma Sun uvedomila narastajúcu dôležitosť rýchlo sa rozširujúceho Internetu a možnosť použitia Javy pre programovanie aplikácií pre WWW(*World Wide Web*). Táto skutočnosť dala vývoju projektu náhle iný smer a v krátkom čase bol vytvorený prvý internetový prehliadač *HotJava* v ktorom boli použité *applety* Javy. Tento prehliadač ukázal, aké sú možnosti jej využitia a od tohto momentu popularita jazyka narastala až sa stal jedným z najpoužívanejších programovacích jazykov súčasnosti. Oficiálne predstavenie jazyka prebehlo na konferencii SunWorld v roku 1995 (viď [9]).

Vývoj jazyka sa nezastavil a stále sú vyvíjane jeho nové verzie JDK (Java Development Kit). JDK predstavuje balík potrebných programov pre prácu v Jave. Aktuálnou verziou JDK je 1.6.

2.2.2 Charakteristika jazyka

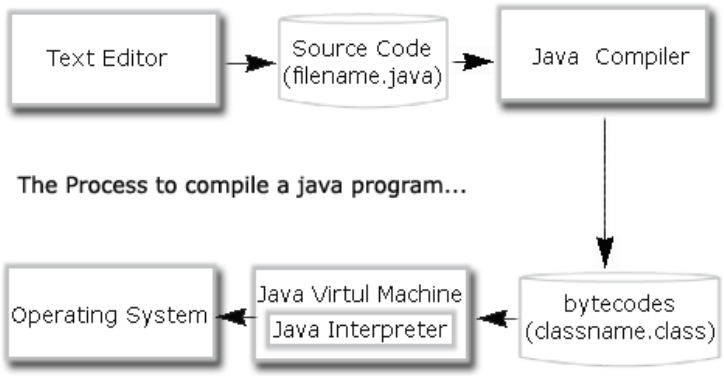
Firmou Sun je Java charakterizovaná ako jednoduchý, objektovo orientovaný, interpretovaný, robustný, prenositeľný jazyk, ktorý je nezávislý na architektúre.

Počet konštrukcií jazyka sa pohybuje na nízkej úrovni. Syntax jazyka zabraňuje alebo obmedzuje vznik ťažko identifikovateľných chýb pri programovaní. Mnohé konštrukcie boli vytvorené podľa vzoru jazykov C/C++, ktoré sú značne rozšírené, preto mnohým programátorom, ktorý poznajú dané jazyky, nerobí problém prejsť na jazyk Java. Nie sú používané hlavičkové súbory. Spoľahlivosť jazyka spočíva v tom, že nepodporuje priamy prístup do pamäti. Kontroly správneho prístupu k položkám sa vykonávajú počas behu programu. Prínosom je takzvaný *garbage collection*, ktorý automaticky uvoľňuje pamäť a zároveň zlučuje voľné a už nepoužívané pamäťové segmenty.

Java je objektovo orientovaná, práca s ňou je prirodzenejšia pretože objekty modelujú modely reálneho sveta lepšie než dátové štruktúry, ktoré sú oddelené od procesného modelu. Objekt reálneho sveta má svoj stav a správanie. Analogicky objekt v objektovo orientovanom programovaní má svoje vlastnosti a svoje metódy. Je schopný odosielať a prijímať správy a môže obsahovať iné objekty.

2.2.2.1 Spôsob spracovania programu

Zvláštnosťou Javy je, že preklad neprebieha do jazyka relatívnych adries, ale do pseudojazyka nazývaného *byte-code*. Tento jazyk je nezávislý na cieľovom počítači. To v praxi znamená, že programátora nemusí zaujímať aký operačný systém používa užívateľ jeho programu. Jedinou podmienkou je, aby pre tento systém existovala Java platforma, ktorou je súhrn programov interpretujúcich jazyk. Preložený program je zavádzaný do pamäti počítača, súčasne prebieha overenie pseudojazyka. Toto všetko je možné vykonať jednotne vďaka nezávislosti pseudokódu na platforme. Po overení je program spustený pomocou interpreteru.



Obr. 2.4: *Spôsob kompilácie programu v jave (prevzaté z [5])*

Problémom interpretovaných jazykov je ich pomalosť v porovnaní s kompilovanými jazykmi. V Jave je tento problém čiastočne vyriešený pomocou tzv. *JIT kompilátoru*, ktorý v dobe zavádzania programu z disku do pamäti počítača preloží program do strojového jazyka konkrétneho počítača. To má za následok výsledné zrýchlenie aplikácie, viac na [9].

2.2.2.2 Štruktúra programu

Pri vytváraní programu vznikajú triedy. Triedy, ktoré podľa ich účelu je možné zaradiť do skupín, vytvárajú balíčky (*package*). Rovnako tieto sú zoskupované do logických celkov. Takto vzniká hierarchický strom balíčkov, pod balíčkov (*subpackage*) a tried (*class*) v nich uložených. Takéto delenie umožňuje veľmi pohodlnú orientáciu v programe, pretože názvy balíčkov intuitívne napovedajú o účeli tried ktoré obsahujú. Základná štruktúra programu je ukázaná na obrázku 2.5.

Program po spustení vypíše na obrazovku text „ahoj“. Na príklade je možné vidieť, že aspoň jedna trieda musí byť označená ako verejná a obsahovať statickú hlavnú metódu (*main*), ktorá zavádza program.

```
public class Nazov_programu{
    public static void main(String[] args){
        System.out.println("Ahoj");
    }
}
```

Obr 2.5 Príklad programu v jazyku Java

2.2.3 Grafické užívateľské prostredia v Jave

Grafické užívateľské prostredie, anglicky *Graphics User Interface* (ďalej *GUI*), predstavuje rozhranie medzi užívateľom a programom, ktorý je spustený na počítači. V Java sa GUI tvorí pomocou prvkov, ktoré sa nazývajú *komponenty*. Tieto prvky sú väčšinou štandardizované. To znamená že funkcie jednotlivých prvkov sú rovnaké ako v iných programovacích jazykoch.

Program s grafickým užívateľským rozhraním sa od konzolovej aplikácie líši tým, že je riadený udalosťami (*event*). Každá komponenta má svoju skupinu pre ňu špecifických udalostí. V Jave je systém komponent a ich udalostí spracovaný veľmi precízne a umožňuje vytváranie užívateľsky príjemných aplikácii.

Java obsahuje dve základné knižnice pre vytváranie GUI.

AWT

AWT (*Abstract Windowing Toolkit*) je staršia knižnica používaná od verzie Javy JDK 1.0. Základnou vlastnosťou tejto knižnice je, že komponenty prispôsobujú svoj vzhľad podľa operačného systému, kde je aplikácia spustená. Funkčnosť prvkov ostáva rovnaká. Táto skutočnosť môže byť chápaná ako výhoda, keďže užívateľ vidí prostredie na aké je zvyknutý, ale taktiež aj ako nevýhoda, pretože niekedy sa nepodarí dosiahnuť rovnaké rozmiestnenie komponentov v zobrazovanom okne pri spustení programu na rôznych platformách, čo spôsobuje problémy.

Používanie knižnice AWT je dnes na ústupe, preferuje sa novšia JFC Swing. Výnimkou je však jej použitie v apletoch, pretože niektoré internetové prehliadače nepodporujú bezvýhradne novšiu knižnicu Swing a použitie AWT zaručuje kompatibilitu, viac na [10].

JFC Swing

Knižnica JFC Swing (*Java Foundation Classes*) sa v Jave nachádza od verzie JDK 1.2. Knižnica vychádza zo staršej AWT, používa rovnaký model udalostí, niektoré z nich bez zmeny preberá.

Obsahuje mnohé nové komponenty a ich udalosti, ktoré rozširujú možnosti tvorby grafických prostredí v Jave. Swing na rozdiel od AWT obsahuje voľbu nazývanú *Look-and-feel*, ktorá umožňuje definovať aký vzhľad bude mať naša aplikácia. Prvou možnosťou je, že vzhľad komponent sa bude

preberať z hostiteľského operačného systému rovnako ako je to pri AWT, alebo bude štandardný a naša aplikácia bude vyzerať na všetkých platformách rovnako.

Swing ďalej podporuje technológie pre získavanie informácii z užívateľského prostredia pre aplikácie, ktoré umožňujú prácu hendikepovaným užívateľom. Je podporovaná tvorba aplikácii v mnohých jazykoch, obsahuje tisíce rozdielnych znakov z jazykov ako sú japončina alebo kórejčina, viac na [6].

2.2.4 Použitie v aplikáciách

Java je všestranný jazyk využívaný v mnohých druhoch aplikácii. Nachádza sa v aplikáciách s viacerými vláknami (*multithread applications*), vo výkonných aplikáciách bežiacich na serveroch (*Java Enterprise Edition*), ďalej sú to aplikácie na prenosných vstavaných zariadeniach (*Java Micro Edition*) alebo aplikácie distribuované na sieti (*applets* alebo *Java Web Start*). Java umožňuje vytváranie prenositeľných aplikácii s grafickým užívateľským prostredím a spracovanie semištruktúrovaných dát (*XML*) (viď [14]).

2.2.5 Vývojové prostredia

V súčasnosti je na trhu mnoho vývojových nástrojov určených pre vývoj aplikácii v Jave. Najznámejšie z nich sú JBuilder od firmy Borland, Visual Age od firmy IBM, Netbeans, Eclipse. Netbeans a Eclipse sú zadarmo a ich zdrojové kódy sú voľne k dispozícii (*open-source*). Na ich vývoji sa môžu podieľať samotní užívatelia, ktorí takto prispievajú k skvalitneniu produktu. Výhodou oboch programov je, že sú vytvorené v Jave. To znamená, že sú používané na rôznych platformách a projekty v nich vytvárané sú taktiež prenositeľné. To prispelo k ich celkovej obľúbenosti.

2.2.6 Spracovanie XML v Jave

Pre prácu s XML dokumentom je potrebné použiť nástroj nazývaný *parser*. Je možné použiť knižnicu priamo z jazyka Java alebo externú knižnicu. Pre Javu ich existuje mnoho, *XOM*, *XP parser*, *JAXB*, *Piccolo*, *kXML*, *Xerces*, *Crimson*. Jedny z najpoužívanejších sú *dom4j* a *jdom*. Pomocou týchto nástrojov je možné pracovať s XML dokumentmi rozhraním SAX alebo rozhraním DOM.

Existujú taktiež voľne šíriteľné programy, ktoré slúžia na vytváranie schém z XML dokumentov a rôzne prevody medzi jednotlivými schémami. Jedným z týchto programov je napríklad *dtdGenerator*, generujúci DTD schému dokumentu alebo aplikácia *dtd2xsd* ktorá vytvára XSD schému pomocou DTD schémy na vstupe.

2.3 Problém vizualizácie XML dokumentov

XML dokumenty majú nesporne veľa výhod spojených s uchovávaním a prenosom informácii. Problémom, ktorý sa týka hlavne dátovo orientovaných dokumentov, je však ich zlá čitateľnosť pre človeka, ak sa jedná o dokumenty väčšieho rozsahu. Keď dokument obsahuje elementy ktoré majú väčšie množstvo atribútov, alebo je zanorenie elementov v dokumente vysoké, je zložité sa v ňom spoľahlivo orientovať.

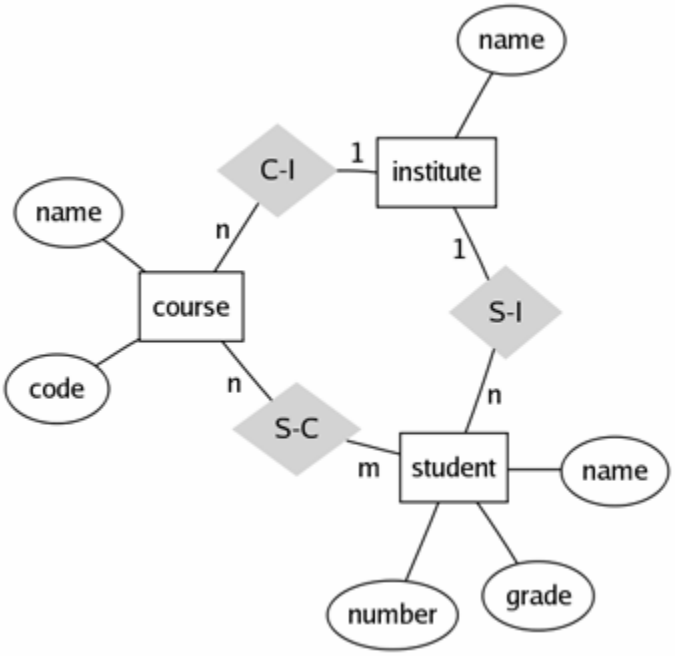
Jedným z možných riešení je vizualizácia dokumentu. Tieto môžu byť vizualizované viacerými spôsobmi, napríklad ako spojenie tabuliek, alebo ako relačný diagram objektov, kde objekty sú tvorené samotnými elementmi dokumentu. Prioritou vizualizácie by vždy malo byť zobrazenie čo najväčšieho počtu informácii čo najprehľadnejšou formou.

2.3.1 Dostupné aplikácie v súčasnosti

Je dostupných niekoľko aplikácii umožňujúcich vizualizáciu XML dokumentov. Niektoré z nich sú voľne šíriteľné, iné je potrebné si zakúpiť. Líšia sa kvalitou vypracovania a taktiež možnosťami ktoré ponúkajú.

2.3.1.1 GraphViz

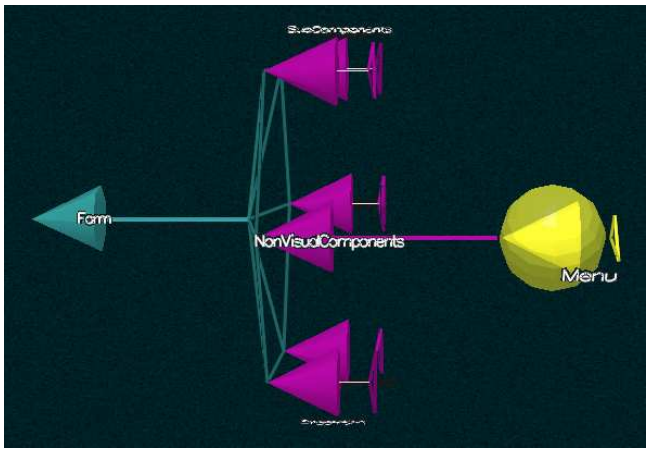
GrapViz je voľne šíriteľný nástroj na zobrazovanie štruktúrovaných dát pomocou grafov a sietí. Skladá sa z niekoľkých častí zobrazujúcich viaceré typy informácii, umožňuje napríklad zobrazenie konečných automatov alebo ER diagramov využívaných v softvérovom inžinierstve, viac na [3].



Obr 2.6 Ukážka vizualizácie pomocou aplikácie Graphviz (prevzaté z [3])

2.3.1.2 Hydra3d

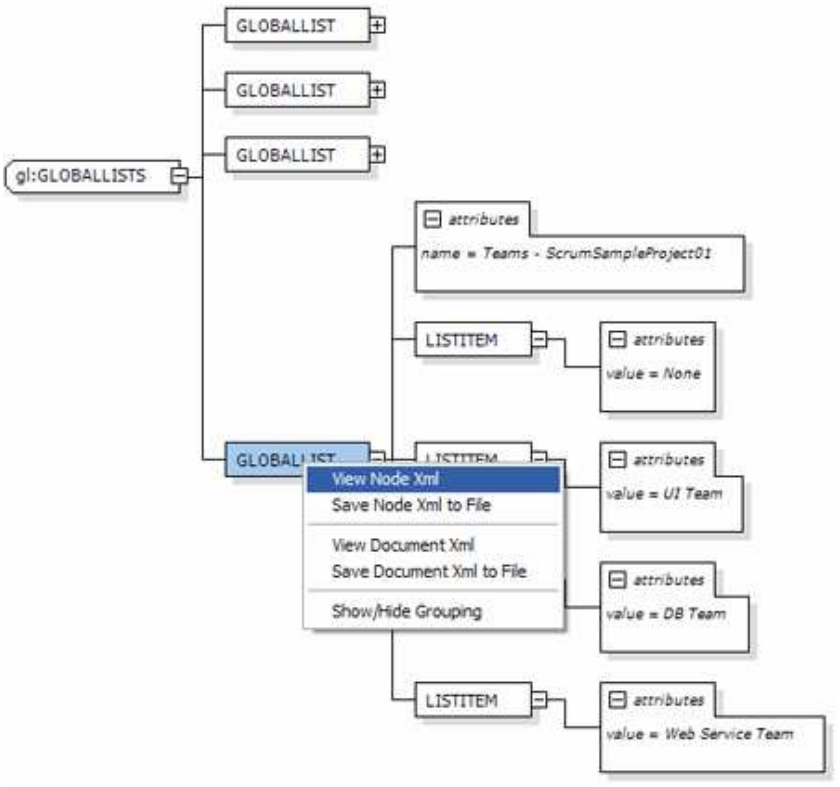
Hydra 3D je trojdimenzionálny nástroj pre vizualizáciu dokumentov pod operačným systémom UNIX a Windows. Umožňuje prezeranie a editovanie dokumentu v 3D priestore. Aplikácia spolu so zdrojovými kódmi je voľne šíriteľná, viac na [4].



Obr 2.7 Ukážka 3D vizualizácie dokumentu pomocou aplikácie Hydra3D (prevzaté z [4])

2.3.1.3 Conchango Xml Visualizer

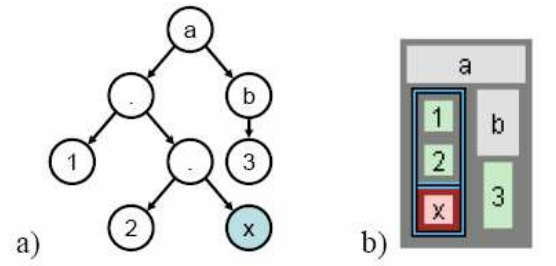
Aplikácia slúži na vizualizáciu XML dokumentov v prostredí Visual Studio 2005. Je naprogramovaná v jazyku C#. Umožňuje dynamicky pracovať s otvoreným dokumentom, ukladanie jeho uzlov do jednotlivých súborov. Aplikácia je voľne prístupná na internetovej stránke autora, viac na [1].



Obr 2.8 Ukážka vizualizácie dokumentu pomocou aplikácie XML Visualizer (prevzaté z [1])

2.3.1.4 XML vizualizácia s využitím prekresľovania stromov

Páni J.Jelínek a P.Slavík spracovali článok s názvom *XML Visualization Using Tree Rewriting* v ktorom sa zaoberajú novou metódou vizualizácie XML dokumentov. Princíp spočíva v transformácii dokumentov podľa určitých pravidiel a v zobrazení vzájomných vzťahov medzi jednotlivými elementmi (viď [12]).



Obr 2.9 Ukážka stromu XML dokumentu (prevzaté z [12])

3 Návrh a tvorba aplikácie

Cieľom bolo vytvoriť jednoduchú aplikáciu na vizualizáciu XML dokumentov a zobrazenie ich schém a definícii. Bola vyvíjaná s požiadavkou na použitie pod rozličnými platformami. Grafické prostredie bolo vytvárané ako pekne spracované a užívateľsky prívetivé.

Program je zameraný hlavne na dátovo orientované dokumenty. Prioritou pri vizualizácii dokumentu bolo zobrazenie čo najväčšieho počtu informácii súčasne. Musí umožňovať zobrazenie dokumentov s rozdielnym kódovaním, aby bolo možné vizualizovať dokumenty v rozličných jazykoch.

Je umožnená interaktívna editácia dokumentu. To značí pridávanie a mazanie elementov na rôznych miestach, editáciu ich atribútov a hodnôt.

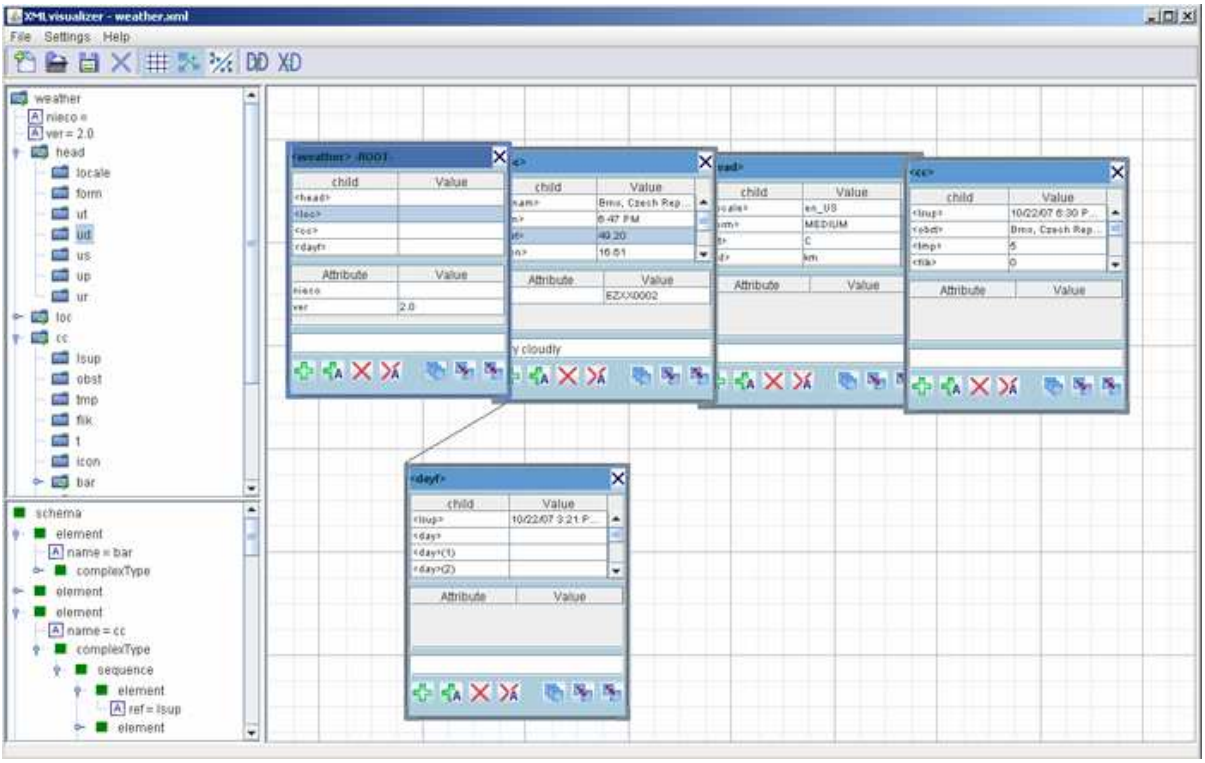
Aplikácia dostala pracovný názov XMLvisualizer.

3.1 Návrh užívateľského rozhrania

Hlavné okno programu obsahuje menu pre hlavnú orientáciu v programe, pod ním sa nachádza lišta s nástrojmi, ktoré slúžia pre urýchlenie práce s aplikáciou.

XML dokument je zobrazovaný dvoma spôsobmi.

- Stromové zobrazenie dokumentu
- Zobrazenie elementov ako objektov na panely elementov a vykreslenie relácii medzi nimi



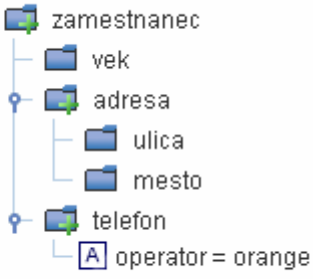
Obr 3.1 Ukážka otvoreného dokumentu v aplikácii XML visualizer

3.1.1 Stromové zobrazenie dokumentu

Stromové zobrazenie slúži hlavne na rýchlu orientáciu v dokumente. Výhodou tohto spôsobu zobrazenia je prehľadnosť, pretože je ľahké zistiť zanorenie elementu, pod aký rodičovský element patrí a rovnako koľko má potomkov a atribútov.

V strome sú pomocou ikoniek jasne rozlíšené rodičovské elementy, to znamená elementy ktoré obsahujú aspoň jedného potomka, a elementy, ktoré ako koncové uzly obsahujú spravidla text. Pri atribútoch elementu je zobrazovaná ich hodnota.

```
<zamestnanec>
  <vek>21</vek>
  <adresa>
    <ulica>L.Svododu</ulica>
    <mesto>Lučenec</mesto>
  </adresa>
  <telefon operator="orange">
    905 344 622
  </telefon>
</zamestnanec>
```



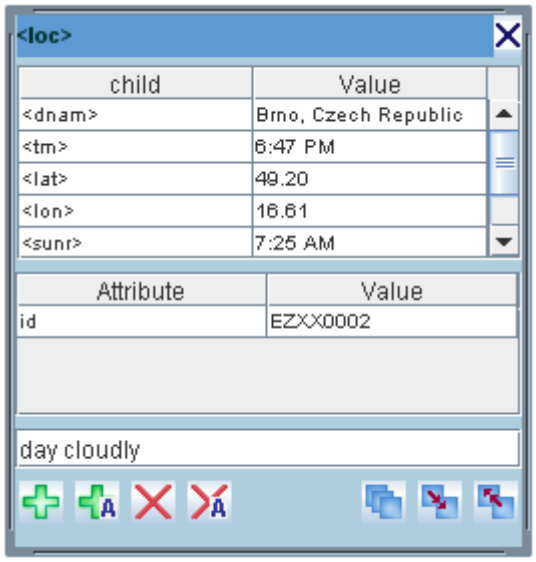
Obr 3.2 Návrh stromového zobrazenie dokumentu

3.1.2 Zobrazenie elementu a jeho atribútov

Pri zobrazení elementu ako objektu je snahou vykresliť čo najväčšie množstvo informácii na čo najmenšej ploche. Pre užívateľa je potrebné ľahko zistiť zaradenie elementu v XML dokumente, poznať jeho zanorené elementy, vlastné atribúty, ich hodnoty a hodnotu samotného elementu. Tieto objekty sú zobrazené na panely elementov vo forme vnútorných okien. Pre zlepšenie orientácie sa po nájdení ukazovateľom myši na titulok okna elementu zobrazí popis elementu (*tooltip*), v ktorom sú zobrazené všetky rodičovské uzly až po koreňový element.

Objekt elementu obsahuje tabuľku, v ktorej sú uvedené názvy jeho potomkov, pri každom z nich je uvedená aj jeho hodnota. Po kliknutí na položku sa tento potomok zobrazí do panelu pod svojho rodiča. Pod tabuľkou potomkov je zobrazená tabuľka atribútov elementu. Vedľa každého atribútu sa nachádza položka s hodnotou. Túto je možné editovať priamo v tabuľke.

S objektom je možné po panely voľne pohybovať, pričom existuje možnosť pohybu elementu aj so zobrazenými synovskými objektmi. Objekt elementu obsahuje tlačidlá na zobrazenie alebo skrytie všetkých svojich potomkov z panelu elementov.



Obr 3.3 Návrh objektu elementu ako vnútorného okna

3.1.3 Zobrazenie vzájomných vzťahov

Vzájomné vzťahy medzi jednotlivými elementmi na panely sú zobrazené pomocou spojníc. Tieto spojnice sú vykresľované dynamicky, čo znamená, že sledujú pohyb elementov.

Pri väčšom počte objektov na paneli však vzniká problém s neprehľadnosťou, je zložité správne rozoznať príslušnosť potomkov ku svojmu rodičovi. Objekt elementu preto obsahuje tlačidlo pre zoradenie. Po jeho použití sa synovské elementy zoradia vedľa svojho rodiča, čím sa panel elementov výrazne sprehľadní. Príklad zobrazenia elementov na panely sa nachádza na obrázku 3.1.

3.1.4 Vytváranie dokumentov a ich editácia

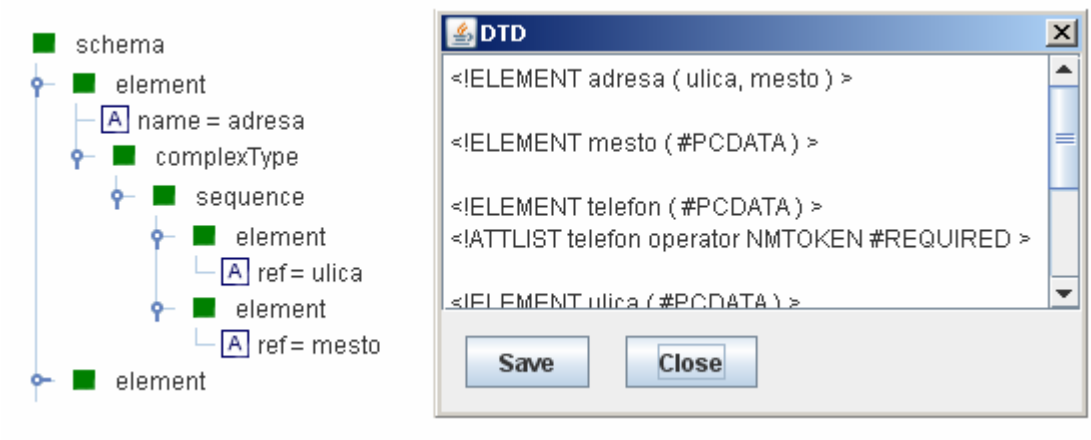
Dokumenty otvorené aplikáciou je možné editovať pomocou vyskakovacieho menu (*popup menu*), ktoré sa zobrazí po kliknutí na uzly elementov v stromovom zobrazení alebo priamo editáciou hodnôt samotného elementu po kliknutí na zobrazené tlačidlá.

Elementy sa dajú pridávať a mazať, pri mazaní elementu sa vymaže celý podstrom, ktorého je element vlastníkom. Rovnako je možné pridávanie a mazanie atribútov elementu.

Zmeny v dokumente je možné uložiť, rovnako sa dá dokument uložiť pod iným názvom.

3.1.5 DTD a XSD

Program umožňuje zobrazenie definície dokumentu DTD a jeho schémy XSD. Výhodou XSD oproti DTD je, že sám o sebe je XML dokumentom. Preto je vhodné zobraziť ho v stromovom zobrazení, z dôvodu práce s XML dokumentom samotným a jeho schémou. Druhou možnosťou je zobrazenie DTD a XSD schém dokumentu textovou formou po kliknutí na ikony v lište nástrojov. Následne sa zobrazí dialógové okno obsahujúce text DTD alebo XSD. Schémy je možné ukladať do súborov.



Obr 3.4 Ukážka stromového zobrazenia XSD schémy a dialógového okna s DTD

3.1.6 Nastavenia programu

Program obsahuje nastavenia, ktoré skvalitňujú prácu so zobrazenými elementmi na panely. Všetky tieto nastavenia sú voliteľné a je možné ich vypnúť. Pri štarte programu sú načítavané, pri ukončení sa ukladajú. Spolu s týmito nastaveniami sú ukladané aj adresárové cesty posledného otvoreného dokumentu a posledného uloženého dokumentu, čo prispieva k pohodliu užívateľa.

Prvým z nastavení programu je zobrazenie mriežky. Mriežka uľahčuje manipuláciu s objektmi na ploche, ľahšie sa určuje ich orientácia. Nastavenie pohybu synovských elementov spolu s ich rodičom, slúži na prehľadnú manipuláciu s elementmi na panely. Posledným nastavením je zobrazenie elementov na panely. Sú dve varianty zobrazenia. Prvou je, že všetky elementy na panely budú vytvárať jediný strom a medzi ľubovoľnými dvoma elementmi bude existovať priame alebo nepriame spojenie. Druhou variantou je zobrazenie viacerých stromov súčasne, keď nemusí byť

medzi elementmi žiadna viazanosť. Táto varianta je dobrá napríklad pre porovnávanie jednotlivých uzlov stromu.

4 Model aplikácie (UML)

Aplikácia sa skladá zo štyroch funkčných celkov.

- **GUI**

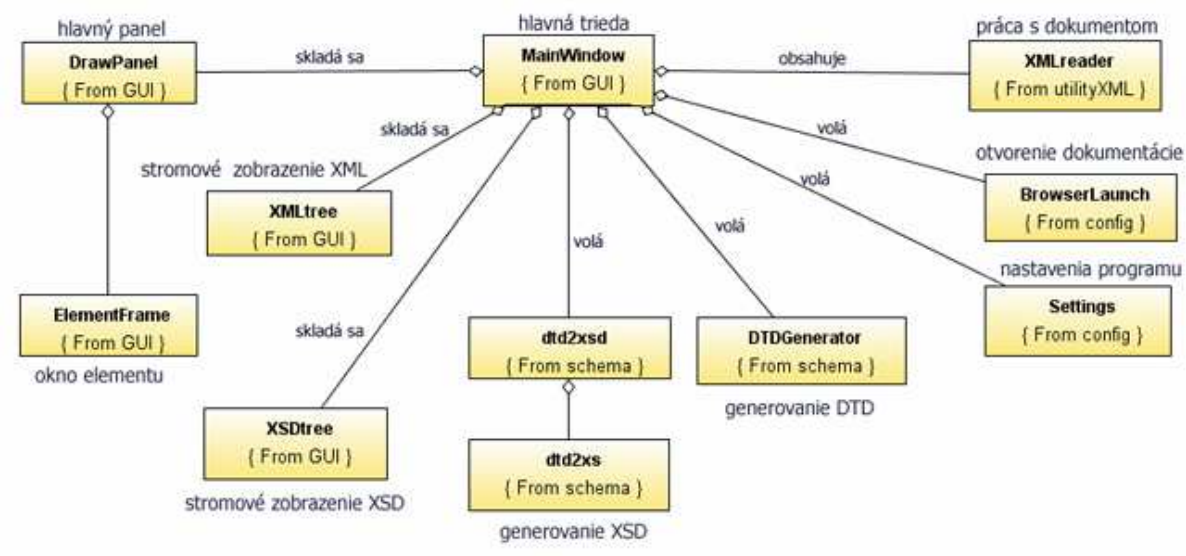
  V tomto balíčku sa nachádzajú triedy, ktoré priamo zobrazujú jednotlivé časti aplikácie. Obsahuje hlavnú triedu, ktorou sú ovládané všetky ostatné triedy v programe.
- **utilityXML**

  Obsahuje triedu pre prácu s XML dokumentom, jeho načítanie a ukladanie.
- **config**

  Triedy pre prácu s konfiguračným súborom a zobrazenie užívateľského manuálu pre aplikáciu.
- **schema**

  Obsahuje triedy pre vytvorenie DTD a XSD schém XML dokumentu.

Pre spracovanie XML dokumentu sú používané externé knižnice JDOM, ktorá slúži na parsovanie XML, a knižnica XML, ktorá je využívaná pri vytváraní XSD schémy dokumentu. Na obrázku 3.5 je uvedený diagram tried vytvorený pomocou jazyka UML, norma ISO/IEC 19501. Podrobnejšie informácie o jednotlivých triedach, ich atribútoch a metódach sú uvedené v programovej dokumentácii v prílohe na CD.



Obr 4.1 Diagram tried aplikácie XMLvisualizer

4.1 Implementácia

4.1.1 Voľba implementačného jazyka

Vzhľadom na požiadavky aplikácie sa ako vhodný programovací jazyk použila Java. Java je objektovo orientovaná, je multiplatformová a umožňuje pohodlné vytváranie grafických užívateľských prostredí. Má kvalitnú dokumentáciu, ktorá významne uľahčuje prácu na aplikácii. Pre Javu existuje taktiež veľké množstvo knižníc pre spracovanie XML dokumentov. Hlavnou výhodou je symbióza spolu s XML jazykom, keďže obidva jazyky sú platformovo nezávislé. Pri implementácii bola použitá Java vo verzii JDK 1.6.

4.1.2 Voľba vývojového prostredia

Z množstva vývojových prostredí pre Javu som sa rozhodol pre Netbeans. Moje rozhodnutie ovplyvnili fakty, že je voľne dostupné a v minulosti som s ním mal možnosť pracovať. Obsahuje nástroje pre efektívne písanie zdrojového kódu a vytváranie grafických užívateľských prostredí. Netbeans je používaný na viacerých platformách a projekty ním vytvorené sú prenositeľné. Pri implementácii boli použité Netbeans vo verzii 6.

4.1.3 Spôsob parsovania XML dokumentu

Aplikácia bola navrhovaná za účelom interaktívnej vizualizácie a editácie dokumentov, čo znamená opakované prechádzanie stromu dokumentu, prácu s elementmi, mazanie a doplňovanie častí dokumentu a rovnako editácie hodnôt elementov a atribútov. Z dvoch možných rozhraní SAX a DOM, ktorými je možné v Jave spracovať dokument, som sa jednoznačne rozhodol pre DOM, ktorý spĺňa všetky vyššie uvedené požiadavky. Jeho jedinou nevýhodou, ktorá sa prejavila pri záverečnom testovaní aplikácie, je jeho pamäťová náročnosť. Preto je potrebné pri väčšom XML dokumente nastaviť Jave pri spúšťaní aplikácie pomocou parametrov väčšiu použiteľnú pamäť.

Spomedzi mnohých možných parserov pre Javu som si zvolil parser JDOM. Dôvodom bola jeho jednoduchosť a kvalitná dokumentácia.

4.1.4 Stromové zobrazenia

Pre stromové zobrazenia dokumentu a XML schémy je použitá komponenta z knižnice Swing *JTree*. Strom dokumentu sa vytvára pomocou rekurzívnej funkcie, ktorá zanorením prechádza celý dokument a nakoniec vráti stromové zobrazenie dokumentu. Jednotlivé uzly stromu majú zaregistrovanú udalosť na kliknutie, čo vedie k následnému zobrazeniu menu nad danou položkou. Pomocou tohto menu je potom možné zobraziť element, ktorý prislúcha aktívnemu uzlu stromu, pridať atribút, potomka alebo zmazať element. Každá zmena v dokumente musí byť zaregistrovaná

pre kontrolu dokumentu pri jeho uzatváraní. Po zmene dokumentu je nutné strom vytvoriť nanovo, aby bol vždy aktuálny.

Ak dokument obsahuje elementy, ktoré majú rovnaký názov, sú pre ne vytvárané indexy. Tieto indexy pri položkách stromu musia zodpovedať indexom elementov na panely. Toto opatrenie umožňuje jednoduchú orientáciu v dlhších zoznamoch rovnakých položiek.

4.1.5 Zobrazenie elementu

Element je na panely zobrazovaný pomocou komponenty *JInternalFrame*. Táto komponenta umožňuje zmenu svojej veľkosti, čo je potrebné pri zobrazení elementu s veľkým počtom synovských uzlov a atribútov.

Synovské elementy sú zobrazené pomocou tabuľky *JTable*. Po kliknutí na položku tabuľky sa synovský element zobrazí vedľa svojho rodiča. Aby mal element prehľad o svojich potomkoch, obsahuje zoznam zobrazených synovských uzlov. Je potrebný pri mazaní elementu, pretože po odstránení rodiča z plochy musia byť odstránené všetky jeho podradené elementy ktorých je priamym aj nepriamym vlastníkom. Mazanie prebieha pomocou rekurzívnej funkcie, ktorá zanorovaním odstraňuje všetky synovské uzly z panelu.

Atribútu elementu sú zobrazené taktiež pomocou tabuľky. Stĺpec tabuľky v ktorom sa nachádza hodnota atribútu je editovateľný, priamo v tabuľke je možné ju meniť. Po zmene sa prekreslí strom dokumentu.

Objekt elementu ďalej obsahuje rozšírenia pre zrýchlenie práce s potomkami elementu. Prvým je funkcia pre zoradenie potomkov, kde sa prechádza zoznam zobrazených potomkov a menia sa ich súradnice podľa pozície vlastníka. Druhým je funkcia pre zobrazenie všetkých potomkov elementu, kde sa zobrazia všetky dosiaľ skryté synovské elementy. Posledným rozšírením je funkcia, ktorá skryje všetky synovské elementy a zároveň ich podstromy, prechádza zoznam zobrazených elementov a postupne pomocou rekurzie uzatvára. Podrobnejší užívateľský návod sa nachádza v prílohe.

4.1.6 Zobrazenie elementov na panely

Ako hlavná plocha pre zobrazenie objektov elementov slúži špeciálny druh panelu *JDesktopPane*. Medzi jeho vlastnosti patrí, že komponenty sú zobrazované na hladinách. Pomocou špeciálnych funkcií je možné jednotlivé komponenty zobraziť do popredia, alebo naopak, nechať ich prekrývať ostatnými.

Elementy sú na panely zobrazované podobným spôsobom, ako objekty schém v jazyku UML. Každý element sa na panely môže nachádzať iba jedenkrát. Medzi rodičom a jeho potomkami sú dynamicky vykresľované spojnice, ich súradnice sa vypočítavajú z aktuálnej pozície dotyčných elementov, podľa toho je rozhodované, z ktorých rohov objektu elementu budú vedené. Po kliknutí na

objekt elementu sa tento nastaví ako aktívny, čo znamená jeho zobrazenie v popredí a farebné odlíšenie.

Rozlišujú sa dva spôsoby zobrazenia objektov elementov na panely. Prvá možnosť dovoľuje vytváranie len jedného stromu. To znamená, že pri zobrazení elementu, ktorý nie je priamym potomkom ani rodičom žiadneho už zobrazeného elementu, je potrebné vykresliť všetky elementy doplňujúce cestu medzi nimi. Zoznam týchto elementov sa vytvára pomocou špeciálnych funkcii, ktoré prehľadávajú strom dokumentu a ukladajú potrebné položky do tohto zoznamu. Elementy v zozname musia byť medzi sebou taktiež prepojené. Výsledkom týchto operácii je, že každý element na panely je v priamom alebo nepriamom vzťahu s iným elementom. Druhou možnosťou je zobrazenie viacerých stromov súčasne, kde nemusia byť medzi nimi žiadne vzájomné vzťahy.

4.1.7 Vytvorenie DTD a XSD z XML dokumentu

Pre vytvorenie DTD a XSD schém z XML dokumentu boli použité voľne dostupné nástroje, ktoré po úpravách boli zakomponované do aplikácie. Ako prvá sa pomocou programu `DTDgenerator` vygeneruje DTD a z tejto sa pomocou ďalšieho programu `dtd2xd` vygeneruje XSD schéma dokumentu. Tak sú získané obidva typy schém pre zobrazenie.

4.1.8 Editácia dokumentu

Dokument je možné editovať z časti stromového zobrazenia dokumentu a potom z objektu elementu na panely. Editačné funkcie sú v oboch prípadoch rovnaké. Prevedie sa zmena v dokumente, nastaví sa stavová premenná ktorá určuje, že sa dokument zmenil. Po tejto zmene je potrebné prekreslenie stromu dokumentu a je potrebné vykonať úpravy na panely elementov, ak sa zmena týka zobrazených elementov.

4.1.9 Problém kódovania dokumentov

Program XMLvisualizer bol vytváraný ako multiplatformový. Jedným z problémov, ktorý táto požiadavka prináša, je kódovanie dokumentov. Rôzne operačné systémy využívajú odlišné znakové sady a rovnako aj XML dokumenty pod nimi vytvárané. Štandardnou znakovou sadou XML dokumentu je *utf-8,* ktorá je používaná ako predvolená, ak nie je určená iná.

Pri načítaní dokumentu sa tento prevedie na kódovanie utf-8, ktoré zaručuje jeho bezproblémové spracovanie knižnicou JDOM a Javou. V prípade uloženia dokumentu je taktiež ukladaný pod kódovaním utf-8.

4.1.10 Ukladanie a načítavanie nastavení programu

Všetky nastavania sú ukladané pri ukončení aplikácie do súboru `config.xml`. Pri štarte aplikácie sú rovnako z tohto konfiguračného súboru načítané. Pre ukladanie dát je použitý jazyk XML, ktorý svojou štruktúrou umožňuje bezproblémové nastavenie jednotlivých položiek.

Nastavenia sú ukladané pomocou funkcií, ktoré zisťujú ich aktuálny stav a prevedú zápis do XML súboru. Pri otváraní dokumentu sa vždy ukladá aktuálny adresár, v ktorom sa dokument nachádza, rovnako je to aj pri ukladaní súboru. Ak pri načítavaní adresára z nejakého dôvodu tento adresár neexistuje, nastaví sa aktuálny adresár ako predvolený podľa zvyklostí operačného systému.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <opendir dir="C:\Documents and Settings\User\Desktop" />
  <savedir dir="D:\XMLfiles\" />
  <gridenable value="true" />
  <movechildenable value="false" />
  <moretreesenable value="false" />
</configuration>
```

Obr 4.2 Ukážka XML dokumentu pre ukladanie konfigurácie

5 Záver

5.1 Zhodnotenie dosiahnutých výsledkov

Práca sa zaoberá problémom vizualizácie XML dokumentov. Prvá polovica je venovaná úvodu do problému, zhrňuje základné poznatky o XML dokumentoch a základné informácie o jazyku Java. Sú v nej zhodnotené taktiež niektoré aplikácie pre vizualizáciu dokumentov dostupné v súčasnosti. Druhá polovica sa zaoberá samotným návrhom aplikácie, obsahuje množstvo obrázkov, ktoré navrhujú jej vzhľad a poskytujú náhľad na jej možnosti. Za návrhom nasleduje implementácia, kde sú popísané a zdôvodnené riešenia problémov spojených s vývojom aplikácie.

Výsledkom práce je v aplikácia XMLvisualizer. Spĺňa všetky zadaním požadované vlastnosti. Aplikácia navyše obsahuje možnosti interaktívnej editácie dokumentu, DTD a XSD schémy dokumentu je možné ukladať. Grafické prostredie je jednoduché a užívateľsky prívetivé. Pre skvalitnenie a zrýchlenie práce boli implementované rozšírené nastavenia zobrazovania dokumentu, všetky nastavenia sú automaticky ukladané a pri štarte aplikácie načítavané.

5.2 Vlastné zhodnotenie

V priebehu práce na aplikácii som sa zlepšil v programovaní v jazyku Java, naučil som sa používať princípy objektového programovania. Snažil som sa klásť dôraz na kvalitné a početné komentáre v zdrojovom kóde a na znovupoužiteľnosť tried a metód. Tieto opatrenia by mali v budúcnosti uľahčiť prácu na zdokonaľovaní aplikácie, ktorej by som sa rád naďalej venoval.

Ako ďalší osobný prínos vidím v pochopení problematiky XML dokumentov. Spoznal som výhody tohto formátu, ale rovnako aj jeho obmedzenia. XML dokumenty budú v budúcnosti určite široko využívané pre svoje vlastnosti a ich pochopenie umožní programátorovi použitie XML dokumentov v širokom spektre aplikácii.

5.3 Možnosti ďalšieho vývoja

Napriek tomu, že je súčasné riešenie prakticky kompletné a vyhovuje všetkým požadovaným vlastnostiam, bolo by možné previesť úpravy pre zdokonalenie aplikácie.

Veľkosť stromu elementov na hlavnom panely je limitovaná veľkosťou okna. Pri rozsiahlejších dokumentoch je potrebné pre operácie so stromom používať posuvník, čo nemusí byť vždy praktické. Zlepšenie by nastalo implementáciou nástroja, ktorý by zobrazoval strom na ploche v menšom meradle a pomocou neho by bolo možné ovládať zobrazenie len niektorej časti hlavného stromu elementov.

Pri editácii dokumentu by bolo možné spracovať funkciu pre krok späť, ktorá by umožnila vrátiť zmeny v dokumente. Problém, s ktorým by sme sa mohli pri implementácii stretnúť, by bolo napríklad mazanie väčšej časti dokumentu, ktorý by bolo potrebné pre návrat niekde uložiť. To však môže byť nemožné pri rozsiahlejších dokumentoch, kde by ľahko mohlo dôjsť k zaplneniu pamäti. Riešenie existuje v forme ukladania zmien do externého XML dokumentu. Analogicky ku funkcii krok späť je potom možné implementovať jej opačnú variantu, krok dopredu.

Vhodná by bola kombinácia riešenia zobrazenia pomocou panelov elementov s riešením vizualizácie XML dokumentov napríklad pomocou prekresľovania tabuliek. Užívateľ by si mohol vybrať spomedzi troch typov zobrazení, pretože každé z nich je vhodné pre zobrazenie iného typu štruktúry XML dokumentov.

Ďalšie významné zlepšenie by mohla priniesť metóda analýzy dokumentu, kde by bol pri štarte aplikácie dokument zanalyzovaný a podľa jeho štruktúry by sa zvolil vhodný typ vizualizácie. Táto by bola potom prispôsobená, aby poskytovala užívateľovi čo najväčší prehľad o dokumente.

Literatúra

[1] Conchango Xml Visualizer for Visual Studio 2005 (RTM). online, November 2005. URL <*http://blogs.conchango.com/howardvanrooijen/archive/2005/11/24/2424.aspx*>

[2] Extensible Markup Language (XML). online, Január 2008. URL <*http://www.w3.org/XML/*>

[3] Graphviz - Graph Visualization Software. online, Marec 2008. URL <*http://www.graphviz.org/*>

[4] Hydra3d. online, Apríl 2008. URL <*http://hydra3d.sourceforge.net/*>

[5] Introduction to Java. online, Apríl 2008. URL <*http://www.roseindia.net/java/java-introduction/*>

[6] The Java tutorials. online, Marec 2008. URL <*http://java.sun.com/docs/books/tutorial/*>

[7] XML Schema Tutorial. online, Apríl 2008. URL <*http://www.w3schools.com/schema/default.asp*>

[8] Bradley, N.: *XML: kompletní průvodce*. Praha: Grada publishing, prvé vydanie, 2000, ISBN 80-7169-949-7.

[9] Herout, P.: *Učebnice jazyka Java*. České Budějovice: Kopp, druhé vydanie, 2000, ISBN 80-7232-115-3.

[10] Herout, P.: *Java grafické uživatelské prostředí a čeština*. České Budějovice: Kopp, prvé vydanie, 2001, ISBN 80-7232-237-0.

[11] Herout, P.: *Java a XML*. České Budějovice: Kopp, prvé vydanie, 2007, ISBN 978-80-7232-307-4.

[12] Jelinek, J.; Slavik, P.:XML visualization using tree rewriting. In *Proceedings of the 20th spring conference on Computer graphics*, New York: ACM, 2004, ISBN 1-58113-967-5, s. 65–72, online. URL <*http://portal.acm.org/citation.cfm?id=1037219*>

[13] Kosek, J.: *XML pro každého: podrobný průvodce*. Praha: Grada publishing, prvé vydanie, 2000, ISBN 80-7169-860-1, online. URL <*http://www.kosek.cz/xml/xmlprokazdeho.pdf*>

[14] Kočí, R.: Prednášky k predmetu Seminář Java. Január 2007.

Zoznam príloh

6 Prílohy

6.1 Príloha 1 – Užívateľský manuál

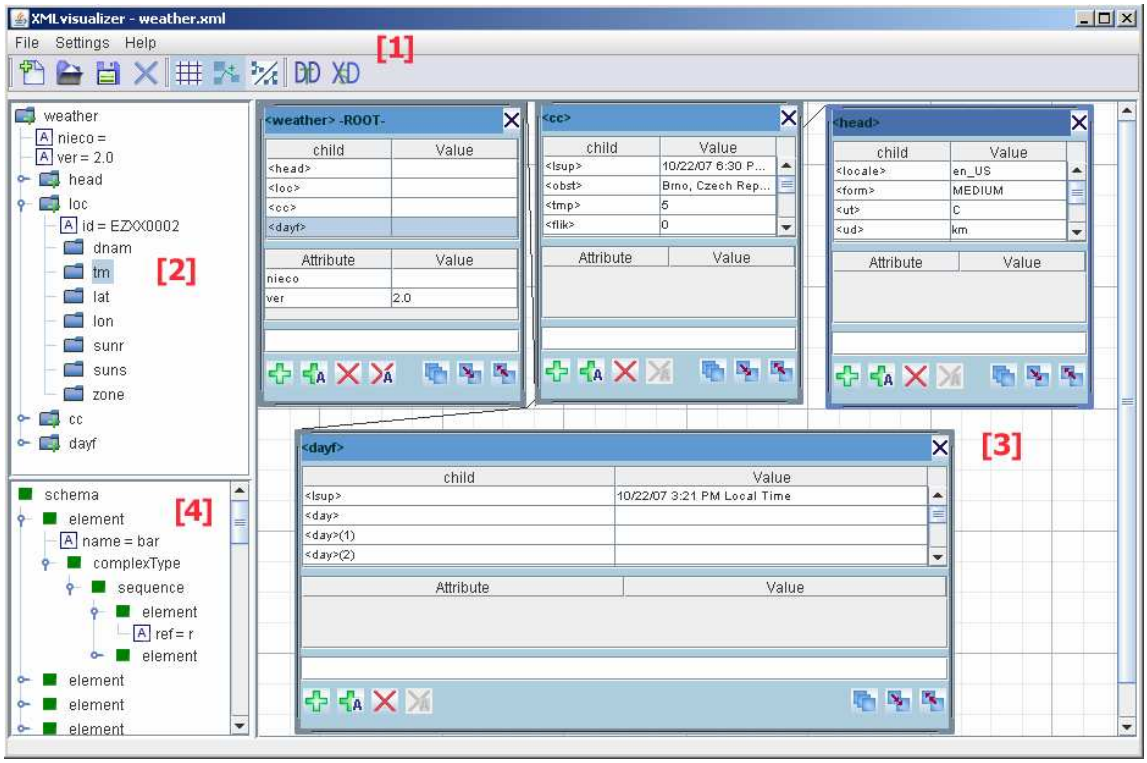
6.1.1 Úvod

XMLvisualizer je nástroj slúžiaci na vizualizáciu XML dokumentov, prednostne je zameraný na zobrazenie dátovo orientovaných dokumentov. Je naprogramovaný v jazyku Java vo verzii 1.6 v spojení s vývojovým prostredím Netbeans verzia 6.

Hlavné okno aplikácie a jej spustenie

Aplikácia zobrazuje XML dokument dvoma spôsobmi. Prvým je stromové zobrazenie dokumentu a druhým je zobrazenie elementov XML dokumentu ako objektov na hlavnom panely. Spustenie je možné príkazmi `java –jar XMLvisualizer.jar` z adresára preloženého programu.

1. Hlavné menu aplikácie a nástrojová lišta s nastaveniami
2. Stromová reprezentácia XML dokumentu
3. Hlavný panel aplikácie s vnútornými oknami zobrazujúce elementy
4. Stromová reprezentácia XSD schémy dokumentu



Obr 6.1 Hlavné okno aplikácie

6.1.2 Vizualizácia

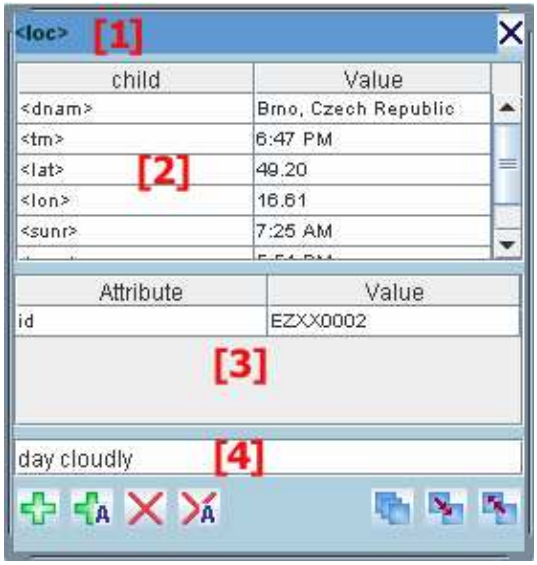
Otvorenie, uzatvorenie a ukladanie dokumentu

Operácie s dokumentom je možné robiť z hlavného menu programu alebo pomocou ikoniek v nástrojovej lište. Každá operácia má svoju klávesovú skratku.

Obr 6.2 Operácie s dokumentom

Práca s oknom elementu

Element je na panely zobrazený ako vnútorné okno. V okne sa nachádza tabuľka potomkov elementu. Obsahuje názov synovského elementu a jeho hodnotu. Po kliknutí na riadok tabuľky sa zobrazí na panely vybratý element. Element môže byť na panely zobrazený len jeden krát. Nie je dovolené zobraziť dvakrát rovnaký element. Pod touto tabuľkou sa nachádza tabuľka atribútov elementu. Skladá sa z dvoch stĺpcov, pričom jeden obsahuje názov atribútu a druhý jeho hodnotu. Hodnota je editovateľná priamo v tabuľke. Okno elementu ďalej obsahuje textové pole, v ktorom je zobrazených prvých päťdesiat znakov jeho hodnoty ak nejakú má. Hodnotu je možné editovať, pri dátových orientovaných dokumentoch však táto zmena spôsobí stratu potomkov elementu a ich podstromov. Na spodku okna elementu sa nachádzajú tlačidlá pre editáciu alebo zobrazenie potomkov elementu alebo atribútov.



1. Titulka okna s názvom elementu. Pomocou tejto lišty je možné pohybovať s elementom po panely. Po nadídení nad titulku kurzorom myši sa zobrazí popis s pôvodom elementu
2. Tabuľka potomkov elementu
3. Tabuľka atribútov elementu
4. Hodnota elementu

Obr 6.3 Vnútorné okno elementu

 Pridanie nového potomka elementu

 Pridanie nového atribútu elementu

 Odstránenie elementu označeného v tabuľke elementov

 Odstránenie atribútu označeného v tabuľke atribútov

 Zoradenie všetkých potomkov elementu zobrazených na hlavnom panely

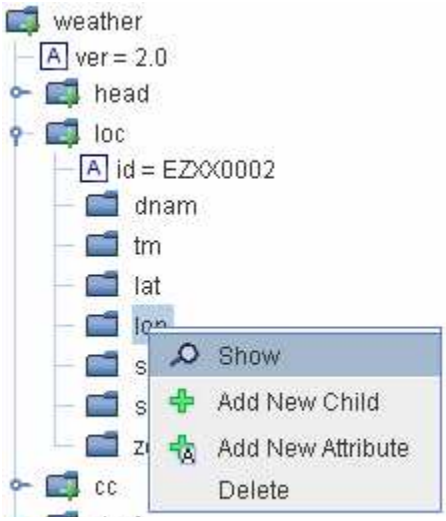
 Zobrazenie všetkých potomkov elementu

 Odstránenie potomkov a ich podstromov z hlavného panelu

Obr 6.4 Popis pre tlačidlá elementu

Práca so stromom dokumentu

Stromové zobrazenie slúži na rýchlu orientáciu v dokumente. Strom obsahuje tri druhy položiek. Prvou je element s potomkami alebo atribútmi, druhou je atribút a jeho hodnota a poslednou je element bez potomkov a atribútov, ktorý zvyčajne obsahuje text. Po nájdení kurzorom myši nad položku stromu sa zobrazí automaticky jej popis. Kliknutím pravým tlačidlom myši na položku zobrazíme menu. Pomocou tohto menu potom ďalej môžeme zobraziť vybratý element dokumentu na panely, pridať nový element alebo priradiť atribút, alebo tento element odstrániť.



Obr 6.6 Ukážka stromového zobrazenia

 Element ktorý ma potomkov alebo atribúty

 Prázdny element alebo element s textom

 Ikona pre atribút elementu

Obr 6.5 Popis pre ikony v stromovom zobrazení

Nastavenia vizualizácie

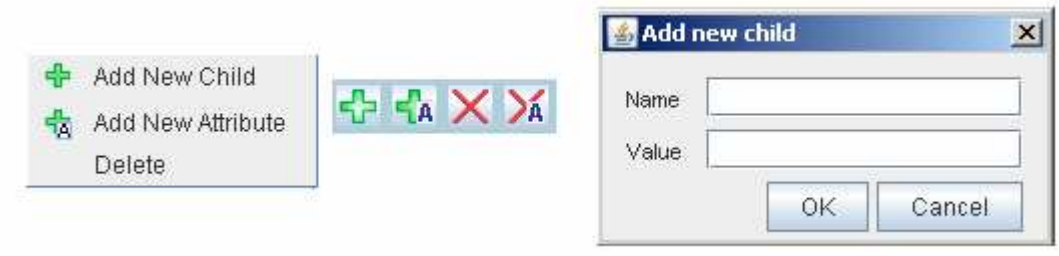
Aplikácia umožňuje nastavenie niektorých vlastností, ktoré ovplyvňujú vizualizáziu a umožňujú pohodlnejšiu manipuláciu s dokumentom.

1. Zobrazí mriežku na hlavnom panely
2. Nastavenie pohybu potomkov elementu s elementom. Keď je toto nastavenie povolené, synovské elementy menia polohu podľa rodičovského elementu, ak je zakázané, pohyb elementov je voľný.
3. Nastavenie povoľuje alebo zakazuje zobrazenie viacerých stromov elementov na hlavnom panely.

Obr 6.7 Nastavenia programu

6.1.3 Editácia dokumentu

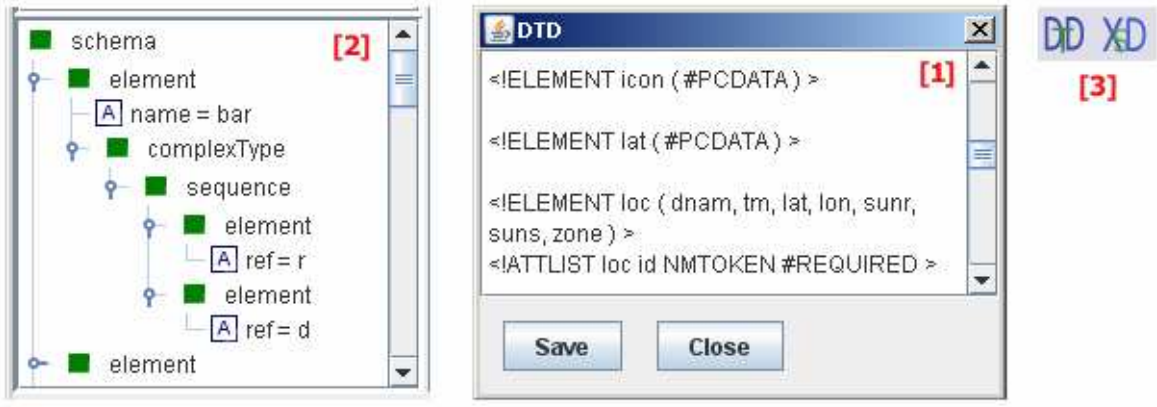
Dokument je možné editovať pomocou menu zobrazeného nad položkami v stromovom zobrazení, alebo priamo z okna konkrétneho zobrazeného elementu na hlavnom panely. Po kliknutí na tlačidlo pridania elementu alebo atribútu sa v oboch prípadoch zobrazí dialóg obsahujúci dve textové polia. Pole s názvom musí byť povinne vyplnené, hodnota nemusí byť zadaná. Ak niektorá z hodnôt nezodpovedá norme XML, ktorá určuje aké znaky môžu obsahovať názvy elementov, je zobrazené chybové hlásenie a operácia pridania skončí chybou.



Obr 6.8 Ukážka možností editácie dokumentu a dialógového okna

6.1.4 Schémy dokumentu

Aplikácia umožňuje zobrazenie DTD a XSD schémy XML dokumentu. Tieto sú zobrazené po kliknutí na ikony v lište nástrojov v hlavnom okne [3]. Následne sa otvorí dialóg so schémou [1]. Schému dokumentu je možné uložiť. Keďže XSD schéma je vytvorená vo formáte XML, je rovnako zobrazovaná ako strom pravidiel [2] pod stromovým zobrazením dokumentu.



Obr 6.9 Ukážka zobrazenia DTD a XSD schém dokumentu

6.1.5 Odporúčania

Problém s nedostatkom pamäti (Java heap space)

Pri potrebe zobrazenia väčších XML dokumentov je nutné pred spustením aplikácie zväčšiť pamäť pridelenú Jave. To sa vykoná pomocou parametra `–Xms300m` kde hodnota `300m` znamená v tomto prípade 300MB RAM. Táto hodnota by mala postačovať na väčšinu rozsiahlejších dokumentov. V prípade, že pamäť nebude dostatočná, program zobrazí dialóg s informáciou o chybe spôsobenej s nedostatkom pamäti a bude ukončený.

6.2 Príloha 2 – CD

Priložené CD obsahuje elektronickú podobu tohto textu `bcpraca.pdf`. A adresári `/src` sa nachádzajú zdrojové texty aplikácie a potrebné externé knižnice. Preložený a spustiteľný projekt sa nachádza v adresári `/dist`, v podadresári `/javadoc` je programová dokumentácia. Pre prácu s väčšími dokumentmi je možné aplikáciu pod prostredím MS Windows spustiť pomocou dávkového súboru `XMLvisualizer.bat`, ktorý zväčšuje pamäť pridelenú aplikácii. V adresári `/uml` sa nachádza obrázok diagramu tried aplikácie, tento je možné nájsť aj v samotnom projekte po otvorení v prostredí Netbeans.