SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

Katedra informatizácie a riadenia procesov

Radlinského 9, 812 37 Bratislava

Bc. Radoslav Valo

INTERNETOVÝ DATABÁZOVÝ SYSTÉM SPRÁVY PREDMETU LCZA

Diplomová práca

vedúci práce

Doc. Dr. Ing. Miroslav Fikar

Bratislava, 2005

SLOVENSKÁ TECHNICKÁ UNIVERZITA

Fakulta chemickej a potravinárskej technológie

Radlinského 9, 812 37 Bratislava

Katedra: **informatizácie a riadenia procesov**

Číslo: 9/KIRP/2005

Vec: **Zadanie diplomovej práce**

Meno a priezvisko študenta: **Bc. Radoslav Valo**

Meno a priezvisko vedúceho diplomovej práce**: Doc. Dr. Ing. Miroslav Fikar**

Meno a priezvisko konzultanta diplomovej práce**:**

Názov diplomovej práce:

Internetový databázový systém správy predmetu LCZA

Generovanie zadaní a riešení problémov pre LCZA pomocou technológií XML, TeX a HTML v MATLABe. Automaticky generované zadania z MATLABu budú exportované do HTML, PDF, či RTF formátov.

Termín odovzdania diplomovej práce: **21. mája 2005**

Diplomová práca sa odovzdáva v 3 exemplároch vedúcemu katedry.

Bratislava **1. februára 2005**

Doc. Dr. Ing. Miroslav Fikar
vedúci katedry

Prof. Ing. Dušan Bakoš, DrSc.
dekan

Ďakujem vedúcemu diplomovej práce Doc. Dr. Ing. Miroslavovi Fikarovi za všestrannú pomoc pri získavaní informácii a cenných rád, ktoré mi ako vedúci diplomovej práce poskytol. Zároveň by som chcel poďakovať svojim rodičom za duševnú a materiálnu pomoc, ktorú mi poskytovali počas môjho štúdia.

Bratislava, 2005

Radoslav Valo

Abstrakt

Cieľom tejto diplomovej práce je vytvorenie internetového databázového systému správy predmetu laboratórne cvičenia z automatizácie. V diplomovej práci sa vytvára XML dokument, ktorý slúži ako databáza príkladov a z ktorého sa čerpajú informácie pre ďalšie spracovanie. Cieľom práce je tiež vytváranie elektronických dokumentov s podporou matematických výrazov a ich následovnou prezentáciou v elektronickej forme (HTML), v podobe pripravenej pre tlač (PDF) a podobe pripravenej pre inport (WebCT) do e-learningových programov. Celý projekt je založený na generovaní matematických zadaní a súčastne aj riešení problémov LCZA pomocou technológie XML. Výsledné formáty sa získavajú transformáciou vytvoreného XML dokumentu pomocou štandardu XSLT.

Abstract

The purpose of this thesis is to create an internet database system for the course laboratory excercise in process control fundamentals. In this thesis a XML document is created which serves as a database of problems and in which information is utilized for futher processing. The goal of the thesis is also the production of electronic documents with the support of mathematical expressions and its following presentation in electronic form (HTML), in a version created for print (PDF) and in a version made for import (WebCT) into e-learning programs. The entire project is based on generating of mathematical problems as well as solving problems with the support of XML technology. The output formats are obtained through tranformation of the created XML document with support of standard XSLT.

Obsah

Zoznam skratiek

API - Application Programmer Interface

ASCII - American Standard Code for Information Interchange

ASP – Active Server Pages

CSS – Cascading Style Sheet – kaskádové štýly

DOM – Document Object Model – objektový model dokumentu

DTD – Document Type Definition – definícia typu dokumentu

FOP - Formatting Objects Processor

HTML – Hypertext Markup Language – hypertextový značkovací jazyk

HTTP – Hypertext Transfer Protocol – protokol pre prenos hypertextu

JSP – Java Server Pages

MS IE – Microsoft Internet Explorer

PDF – Portable Document Format – prenosový formát dokumentu

SAX - Simple API for XML

SGML – Standard Generalized Markup Language – štandardný jazyk pre obecné značkovanie

SPDL – Standard Page Description Language – štandardný jazyk pre popis stránky

W3C – World Wide Web Consortium – konsorcium pre WWW

WWW – World Wide Web – sieť svetových rozmerov

WYSIWYG – What You See Is What You Get – máš čo vidíš

Xlink – XML Linking Language – jazyk pre prepojovanie v XML

XML – Extensible Markup Language – rozšíriteľný značkovací jazyk

XSL – XML Stylesheet Language – jazyk pre štýly v XML

XSLT – XSL Transformations – transformácia XSL

Úvod

Koniec deväťdesiatych rokov je právom považovaný za obdobie rozmachu informačných technológií a internetu. Súčasne s týmto rozvojom vzrastá potreba prezentovať a sprístupňovať dokumenty v elektronickej forme. Problémom je však existencia rôznych platforiem a formátov v ktorých sú dokumenty uložené. Jedno z možných riešení elektronickej publikácie a šírenia dokumentov môžeme nájsť vo využití štandardu XML. Tento štandard umožňuje nadefinovať vlastnú štruktúru dokumentu (DTD – Document Type Definition) a ďalej aj transformačné metódy pre prevod XML dokumentu do iných formátov.

Pre oblasť technickej dokumentácie je potrebná možnosť vkladania matematického aparátu. Súčastné systémy čiastočne umožňujú efektívne spracovanie matematiky v hypertextových dokumentoch pri zachovaní možnosti tlače. Štandard MathML vyvinutý konsorciom W3C, ktorý je založený na technológii XML, by tento nedostatok mohol vyriešiť. Keďže ale doteraz nie je uznaný ako oficiálny štandard, existuje jeho podpora iba v niektorých zo súčasných systémov. Práve implementácia podpory tohoto štandardu s jeho možnými náhradami a prevodom XML dokumentu do iných formátov, je predmetom tejto práce.

Vytváraný XML dokument má predpísanú vnútornú štruktúru vytváraných príkladov na testy a obsahuje zadania a ich možné odpovede. Vytváraný XML dokument slúži ako databáza príkladov, ktoré sa môžu ďalej využívať na transformáciu do iných formátov. Dokument sa vytvára v programe MATLAB a môžeme ho vytvoriť klasickým spôsobom zapisovania do súboru, alebo pomocou štandardu DOM. MATLAB obsahuje aj procesor XSLT, ktorý sa používa na transformáciu vytvoreného XML dokumentu na výsledné formáty. Pre elektronickú prezentáciu príkladov sa používa celosvetovo rozšírený štandard HTML, kde je treba vyriešiť problematiku zobrazovania matematických výrazov v elektronických dokumentoch a to použitím spomínaného štandardu MathML, nadstavby pre XML, aplikáciou JsMath, alebo aplikáciou MimeTEX. Pre tlačovú podobu je to systém LATEX, pomocou ktorého môžeme získať výstupný formát PDF. Posledným formátom je WebCT, ktorý sa ďalej využíva ako importný súbor pre e-learningový program Moodle.

1 Problematika elektronického publikovania

Prvá kapitola obsahuje opis problému elektronického publikovania a ďalej problému tvorby a tlače hypertextových dokumentov. Nachádza sa tu súhrn najpoužívanejších prehliadačov a najpoužívanejších prostriedkov v elektronických dokumentoch a súhrn ich výhod či nevýhod pri použití na stránkach WWW (World Wide Web).

1.1 Základné pojmy

1.1.1 Elektronické publikovanie

Pod pojmom elektronické publikovanie rozumieme vytváranie dokumentu pomocou výpočtovej techniky za pomoci špeciálnych programov a nástrojov pre editáciu textu. Nespornou výhodou tohto prístupu je možnosť kedykoľvek sa k takto napísanému dokumentu vrátiť a spätne ho upravovať či doplňovať o nové informácie. Vytvorený dokument možno ukladať na záznamové média a distribuovať na potrebné miesta, prípadne ich možno zdieľať prostredníctvom počítačovej siete.

S elektronickým dokumentom možno naviac okrem klasickej editácie textu, vykonávať aj iné operácie, napríklad ho možno rozšíriť o multimediálne prvky, ako sú napríklad obrázky, animácie či zvuky. Dokument tiež možno jednoduchým spôsobom formátovať podľa zadaných požiadaviek, alebo ho transformovať do inej formy prezentácie. Nezanedbateľnou výhodou je možnosť priameho prepojenia niekoľkých dokumentov medzi sebou pomocou takzvaných hypertextových odkazov.

1.1.2 Hypertextový dokument

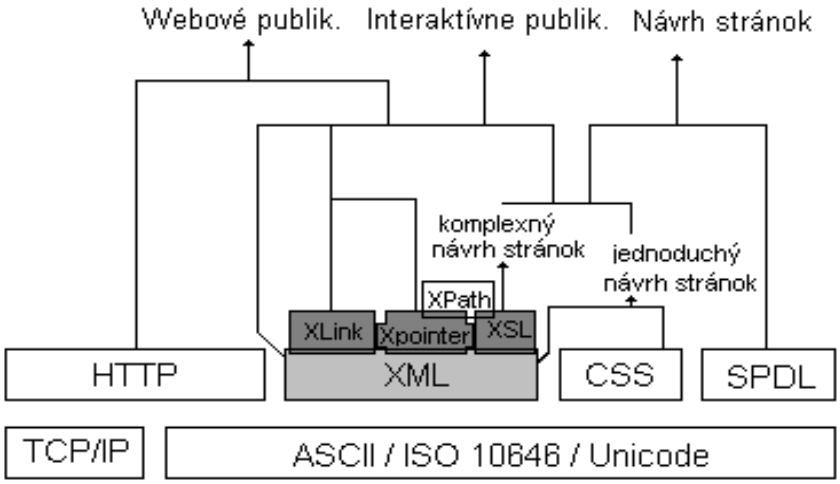
Pod pojmom hypertext rozumieme textový dokument obsahujúci elektronické odkazy na ďalšie dokumenty, ktoré sú uložené v počítačových systémoch. Vo fyzickom dokumente či knihe sú odkazy reprezentované zoznamom odkazov na konci textu. V elektronickej forme je dokument zobrazený na obrazovke počítača a užívateľ iba označí daný odkaz a automaticky je odkázaný na nový dokument, ktorý je so špecifikovaným odkazom zviazaný.

1.1.3 Značkovací jazyk

Značkovací jazyk definuje množinu špeciálnych symbolov, ktoré sa tiež nazývajú značky (markups). Značky sa vkladajú do pôvodného textu a rozširujú ho tak o ďalšie informácie.

Tieto informácie sú pre koncového užívateľa nepodstatné, slúžia predovšetkým softwaru, ktorý daný dokument spracováva. Značkami sa definuje štruktúra a formátovanie dokumentu.

1.2 Prostriedky pre elektronické publikovanie



Obr. 1.1 Štandardy pre elektronické publikovanie

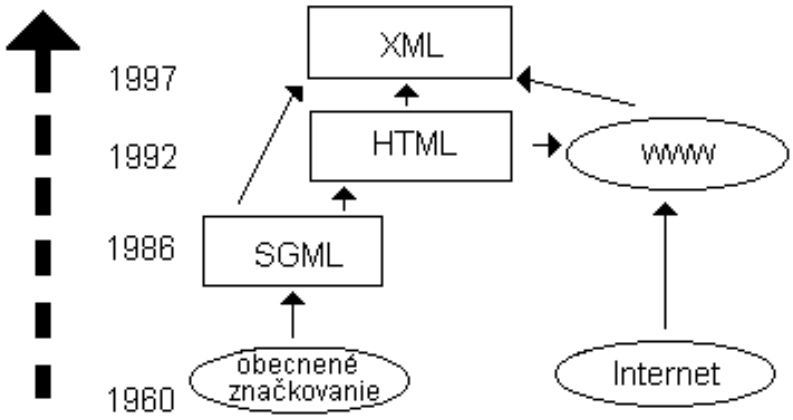
ASCII (American Standard Code for Information Interchange) je už dlhú dobu zavedený štandard pre elektronické ukladanie a prenos textu. **Unicode** a **ISO 10646** sú rozšírené verzie ASCII a sú základom pre ostatné štandardy, ako vidno na Obr. 1.1. HTTP (HyperText Transfer Protocol) je internetový protokol na ktorom je založený Web. **CSS** (*Cascading Style Sheet* – kaskádové štýly) predstavuje jednoduchý jazyk pre definíciu štýlu, zameraný najmä na formátovanie textu pre obrazovku. Pôvodne bol vyvinutý pre HTML (*Hypertext Markup Language*), ktorý je však možné v XML (*Extensible Markup Language*) používať bez ďalších modifikácií. **SPDL** (*Standard Page Description Language* – štandardný jazyk pre popis stránok) opisuje, ako presne majú byť informácie na stránke umiestnené. Viac informácii o prostriedkoch pre elektronické publikovanie v [3].

Internet sa v nedávnej dobe stal zavedeným médiom pre získavanie a výmenu informácií. Je prirodzenou platformou pre hypertextové služby, prepojuje dokumenty z celého sveta do webovej pavučiny informácií. Bolo však potreba vytvoriť značkovací

jazyk, ktorý by toto prepojenie umožnil a ponúkol tiež možnosť základného formátovania takto prepojených dokumentov.

Prenos dokumentov zo serveru, je uskutočňovaný pomocou vytvoreného protokolu HTTP (*Hypertext Transfer Protocol* – protokol pre prenos hypertextu), ktorý pracuje na princípe klient-server, kde HTTP server prijíma štandardné požiadavky jednotlivých klientov a splnené požiadavky, respektíve dokumenty zasiela späť.

Pre formátovanie prepojených dokumentov a pre ich nasledovnú prezentáciu na obrazovke bol vytvorený značkovací jazyk HTML (*Hypertext Markup Language* – jazyk pre značkovanie hypertextu). Jeho vlastnosti budú opísané nižšie.



Obr. 1.2 História vzniku hypertextových dokumentov

1.2.1 SGML

Koncepcia obecného značkovania (*generalized markup*) vznikla začiatkom 60, rokov minulého storočia, ale príliš sa nerozšírila, pokiaľ neprišlo k vytvoreniu SGML (*Standard Generalized Markup Language* – štandard pre obecný značkovací jazyk). SGML bolo schválené organizáciou ISO v roku 1986 a behom desiatich rokov sa jeho existencia prakticky nezmenila, čo vidno aj na Obr. 1.2. V skutočnosti bola možná špecifikácia SGML tak príliš pokročilá, že niektoré vlastnosti sa dodnes ani nepoužili. Kvôli zložitosti SGML a nasledovnej vtedy nákladnej implementácii systému záložného na tomto formáte, ho používalo pre ukladanie dokumentov a ich výmenu a publikovanie len relatívne málo organizácií. V počiatkoch bolo jeho použitie obmedzené len na vládne organizácie

a zameriavalo sa predovšetkým na technickú dokumentáciu amerického ministerstva obrany.

Všetky dokumenty SGML začínajú deklaráciou SGML, čo môžeme považovať za určitý druh konfiguračného súboru. Inštrukcie v tejto časti poskytujú hodnoty pre nižšie určené vlastnosti a obmedzenia.

1.2.2 HTML

História jazyka HTML je pevne zviazaná so vznikom Internetu ako média pre prenos a výmenu informácií. Tento jazyk prevzal od SGML základnú syntax, neprevzal však jeho princípy. Až neskôr bol tento jazyk úplne kompatibilný s SGML a mohol byť zapísaný ako jeho aplikácia (a teda definovaný vhodným DTD Document Type Definiton). Nevýhodou jazyka je však neposkytovanie autorom dokumentu možnosť rozšíriť jazyk pomocou vlastných zmysluplných tagov. Obsahuje však značky pre štýly ako napríklad element <i> (*italics* - kurzíva), kde je výstupný formát priamo určený, ale tiež obecnejšie objekty, ako (*emphasis* - zvýraznenie), kde je výstupný formát ponechaný na prehliadači.

Medzi najväčšie výhody však patrí schopnosť umožnovať čitateľom sledovať odkazy na iné dokumenty HTML po celom internete. Vybraním odkazu URL (*Uniform Resource Locator* – jednotný lokátor zdroja) iného súboru, prehliadač zasiela požiadavku na dokument HTML serveru a nahradí ňou pôvodný obsah, ak nie je v odkaze URL určené inak.

Členovia konsorcia W3C (World Wide Web Consortium) sa pokúšajú postupne sťahovať HTML a umožniť tak priebeh novej špecifikácie XHTML (*eXtensible HTML*). Je to vlastne revidovaný jazyk HTML odpovedajúci štandardom XML. Tieto dokumenty sú taktiež správne štrukturované a majú možnosť rozširovania o ďalšie štandardy (MathML, ...).

Základná štruktúra HTML dokumentu:

```
<html>
  <head>
  ...
  </head>
  <body>
    ...
    <p>text v <b>HTML <em>dokumentu</em></b> ...</p>
    ...
  </body>
</html>
```

Pokiaľ by sme sa rozhodli vytvárať XHTML dokument, tak na jeho začiatku musíme definovať špecifikáciu XHTML zapísaním DOCTYPE protokolu, v ktorej sa hovorí o tom, aké je použité kódovanie a verzia použitého HTML.

```
<?xml version=”1.0” encording=”Windows-1250”>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

1.2.3 XML

I keď sú dnes počítače schopné pracovať s elektronickými dokumentmi, ktoré obsahujú obrázky, hudbu čí video, mnohé dokumenty sú stále založené na texte. Ak chceme text a ďalšie média kombinovať, je nutné ich organizovať do určitého druhu infraštruktúry. Štandard XML takú platformu ponúka. Názov XML je vlastne skratka anglického termínu *Extensible Markup Language.* Bol vyvinutý konsorciom W3C a vytvorený podľa skúsenosti s predchádzajúcimi značkovacími jazykmi [3].

Jazyk XML je zjednodušenou formou jazyka SGML. Vychádza z jeho princípov a syntaxe, nezahŕňa však radu techník, kvôli ktorým bola práve implementácia SGML veľmi zložitá. Dokument XML má logickú fyzickú štruktúru, ktorá rozdeľuje dokument do pomenovaných jednotiek a podjednotiek, nazývaných **elementy**. Fyzická štruktúra umožňuje pomenovať a uložiť samostatné časti dokumentu, nazývané **entity**.

XML je v skutočnosti metajazyk, čo znamená, že je to jazyk, ktorý sa používa k opisu ďalších jazykov. Neexistuje preddefinovaný zoznam elementov. XML poskytuje uplnú slobodu pri využívaní prvkov, ktorých mená majú pre danú aplikáciu zmysel. Zmätku v pomenovaní elementov je však možné zamedziť vďaka mechanizmu, pomocou ktorého preddefinujeme elementy, ktoré môžu byť používané v danej triede dokumentov. **DTD** *(Document Type Definition)* definuje povolené prvky a validujúci parser (kontrolujúci analyzátor) porovnáva pravidla DTD s príslušným dokumentom, aby určil, či dokument týmto pravidlám neodporuje. Nasledujúci príklad definuje použitie elementov v dokumente XML (zadanie.xml):

```
<!ELEMENT priklad (podpriklad, zadanie, moznost)>
<!ATTLIST priklad name CDATA #REQUIRED>
<!ELEMENT podpriklad (zadanie, moznost)>
<!ELEMENT zadanie (#PCDATA)>
<!ATTLIST zadanie hodnotenie CDATA #REQUIRED>
<!ELEMENT moznost (#PCDATA)>
```

Príklad XML dokumentu:

```
<?xml version="1.0" encoding="windows-1250"?>
<!DOCTYPE priklady SYSTEM "Skuska.dtd">
<priklady>
   <priklad name="TRO">
      <zadanie hodnotenie="3">Uzavretý regulačný obvod tvorí  riadený  proces s
prenosom <span CLASS="math">G_p(s)={3\over{s^2+5s+7}}</span> ... </zadanie>
      <moznost ans="baditem">žiadna z ostatných odpovedí nie je ... </moznost>
      <moznost ans="baditem">0.29</moznost>
      <moznost ans="baditem">-1.15</moznost>
      <moznost ans="gooditem">2.06</moznost>
      <moznost ans="baditem">0.13</moznost>
   </priklad>
</priklady>
```

1.2.4 LaTeX

LaTeX je balík makier, ktorý umožňuje autorovi písať a následne tlačiť svoje práce v najvyššej typografickej kvalite, používajúc preddefinované profesionálne rozmiestnenie obsahu dokumentu. Pôvodným autorom LaTeXu je Leslie Lamport [16].

V roku 1994 bol balík LaTeX aktualizovaný pomocou tímu LaTeX3 vedeným Frankom Mittelbachom tak, aby obsahoval niektoré dlho požadované vylepšenia.

Značkovanie dokumentu je odlišné od SGML a jazyku z neho odvodených. Miesto klasických tagov je daný text uzavretý v zložených zátvorkách a tento celok je uvedený kľúčovým slovom, ktoré sa pre odlíšenie od okolitého textu rozlišuje spätným lomítkom na začiatku textu.

Príklad časť dokumentu v LaTeXu:

```
\documentclass[answers,nosep,scores]{exams1}
\usepackage[top=2cm,bottom=0.5cm,left=1cm,right=2.5cm]{geometry}
\begin{document}
\begin{exam}
\question{}{}{
\begin{problem}[\split]
\score{3}
Uzavretý regulačný obvod tvorí riadený proces s prenosom.....
\begin{choice}
\baditem{žiadna z ostatných odpovedí nie je správna}
\baditem{0.29}
\baditem{-1.15}
\gooditem{2.06}
\baditem{0.13}
\end{choice}
\end{problem}
}
\end{exam}
\end{document}
```

1.3 Editovanie elektronických publikácií

Existujú dva základné prístupy, ako vytvárať, upravovať a publikovať elektronický dokument, a to:

- **WYSIWYG** editory: sú editory pri ktorých sa kladie veľký dôraz na vizuálne možnosti editácie dokumentu. Užívateľ iba špecifikuje operácie, ktoré sa nad vybratým textom majú uskutočniť. Uskutočnené zmeny sa naviac ihneď prejavia a užívateľ tak má možnosť okamžite pozorovať výsledky svojej práce.

  Nevýhodou tohto prístupu je fakt, že užívateľ nemá kontrolu nad výsledným „zdrojovým" kódom. Ďalej sú do istej miery neefektívne, čo sa týka veľkosti výsledného súboru. Generujú totiž veľké množstvo nadbytočných značiek, ktoré nie sú potrebné pre správnu interpretáciu dokumentu.

  Medzi klasické WYSIWYG textové procesory patria: *MS Word*, *Word Perfect*, *AmiPro* a pre unixové platformy je to balík *Star Office.*
- **Nie - WYSIWYG** editory: sú také editory, kde užívateľ priamo edituje zdrojový text včítane prípadného značkovania. Programátorský prístup zaisťuje úplnú kontrolu nad zdrojovým kódom dokumentu. Výsledok je potom treba prehliadať externým programom, prípadne nejakým interne zabudovaným prehliadačom.

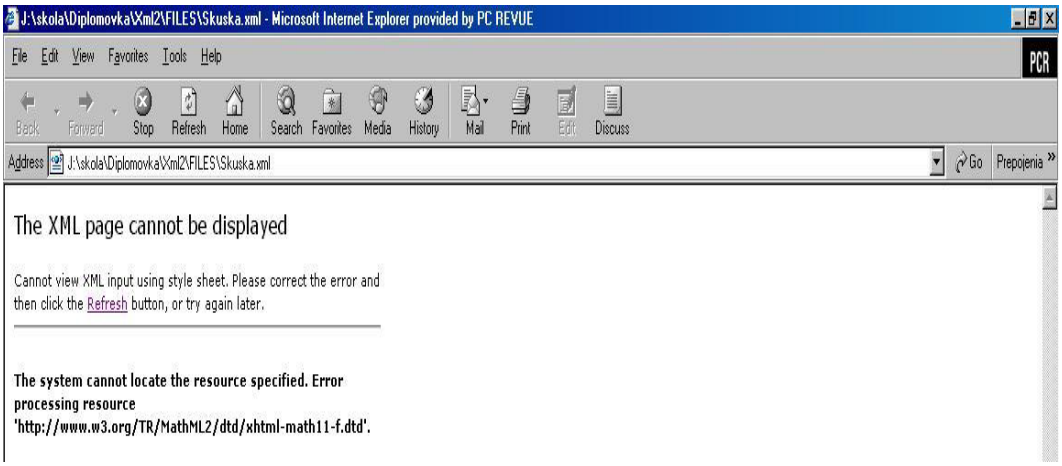
  Špecializované programy môžu prácu pri vytváraní dokumentu uľahčovať špeciálnymi panelmi nástrojov vkladajúcich bloky zdrojového textu, zvýraznením syntaxe, kontextovými odpoveďami, atď.

1.4 Prezeranie elektronických publikácií

Na prezeranie elektronických HTML publikácií možno v súčastnej dobe použiť mnoho prehliadačov, avšak medzi najrozšírenejšie patria: MS Internet Explorer, Mozilla, Netscape, Opera. Všetky z nich majú v najnovších verziách zabudovanú kompletnú podporu poslednej verzie HTML. Rozdiel môže nastať pri spracovaní niektorých HTML prvkov, ktorých funkcia nie je štandardne presne definovaná a ich interpretácia teda môže byť v každom iná.

1.4.1 MS Internet Explorer

V súčasnosti je najrozšírenejším prehliadačom HTML dokumentov MS Internet Explorer (IE), ktorý je k dispozícii iba v operačných systémoch MS Windows. Poslednou verziou je 6.0, ktorá okrem HTML sa vyznačuje dobrou podporou CSS (Cascading Style Sheets – kaskádové štýly) a podporou CSS2.

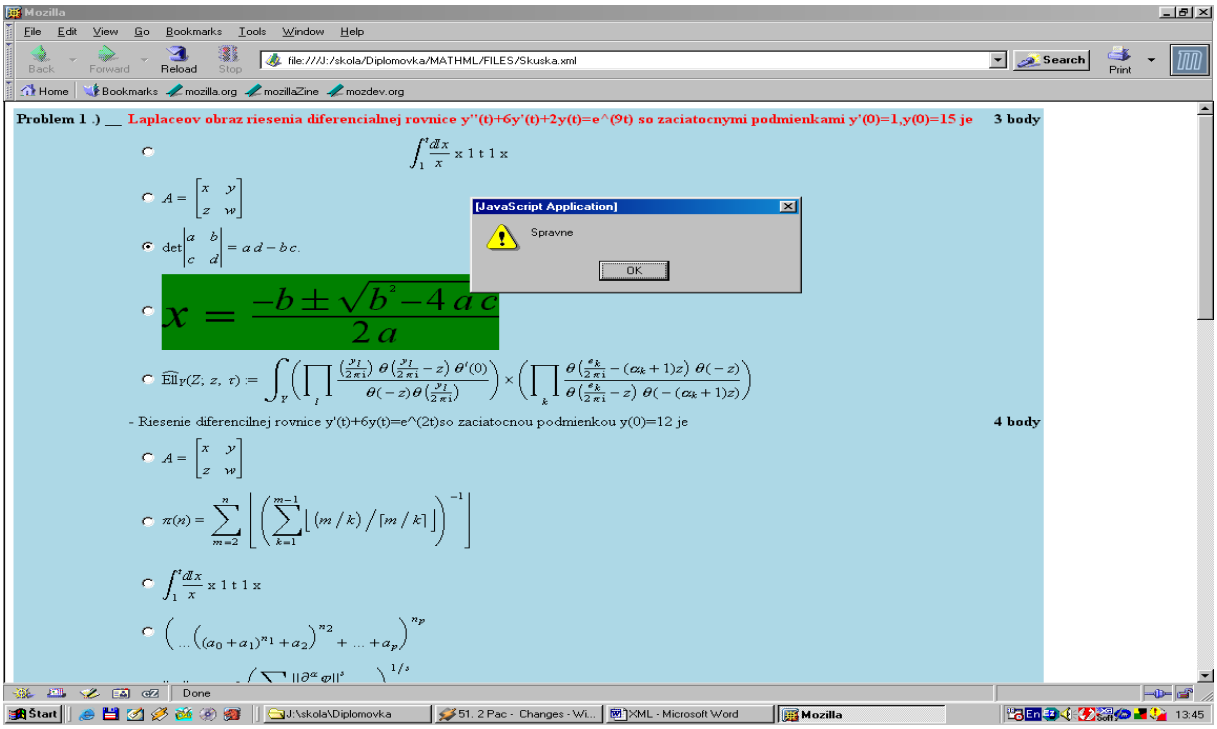


Obr. 1.3 Použitie MathML v prehliadači MS Internet Exploreri

V tejto verzii je zabudovaná aj podpora XML a to tak, že ak je XML dokument validný, zobrazí sa ako stromová štruktúra. v prípade že je v dokumente zadaný odkaz na XSL súbor, je IE schopný zobraziť transformovaný dokument. Nie je však zabudovaná podpora pre zobrazovanie jazyka MathML. Preto je potreba z domovskej stránky MS Windows stiahnuť a nainštalovať podporu (MathML Player), kedy IE nezobrazuje matematické vzorce napísané pomocou jazyka MathML tak dobre ako prehliadač Mozilla.

1.4.2 Mozilla

Mozilla je open-source program, ktorý sa v súčasnosti stále vyvíja a snaží sa podporovať najnovšie štandardy. Je dostupná pre všetky rozšírené operačné systémy a platformy.
Mozilla obsahuje dobrú implementáciu podpory CSS a CSS2. V súčastnej verzii 1.7.5 je podpora zobrazovania XML dokumentov, kde taktiež XML dokument je zobrazený ako stromová štruktúra pri validnom dokumente a pri odkaze na XSL súbor zobrazuje transformovaný dokument.
Výhodou Mozilly je, že ako jeden z mála prehliadačov má k dispozícii modul pre zobrazovanie jazyka MathML (*Mathematical Markup Language*).



Obr. 1.4 Použitie MathML v prehliadači Mozilla

1.4.3 Netscape Navigator

Tento prehliadač je k dispozícii na mnohých platformách. Staršie verzie majú veľmi zlú podporu CSS a podpora XML je na tom možno aj horšie. Keďže je staršieho vydania, dá sa tento nedostatok ospravedlniť. Novšie verzie ako 6.01 založené na zobrazovaciom jadre Gecko projektu Mozilla situáciu zlepšila. Podpora zobrazenia XML spočíva v zobrazení obsahu elementu, keď je XML dokument validný.

1.4.4 Opera

Opera 8.01 je najlepší prehliadač čo sa týka veľkosti a rýchlosti. Naviac existujú verzie pre viaceré platformy. Tento prehliadač má podľa dostupných informácií asi najlepšiu podporu obidvoch verzií kaskádových štýlov. Nepodporuje zobrazenie textu v kódovaní UTF-8. Nepodporuje ani MathML. Preto sa týmto prehliadačom nebudeme ďalej zaoberať.

2 Matematika v elektronických dokumentoch

Tato kapitola pojednáva o problematike použitia matematických výrazov v hypertextových dokumentoch a tiež prináša stručný pohľad na štandard MathML, ktorý je vážnym kandidátom pre prijatie ako oficiálneho štandardu. Taktiež sa zameriava na možné náhrady tohto štandardu, ktoré majú ešte stále prevahu pri použití v hypertextových dokumentoch.

2.1 Zobrazenie matematických výrazov

S rozvojom výpočtovej techniky vznikol problém, ako matematické výrazy prezentovať na obrazovke a ako prezentované výsledky poskytovať vo forme pre tlač. Stretáva sa tu problém zložitosti zápisu, zrozumiteľnosti a vzhľadu takého výrazu. I tak sa vyvinulo niekoľko foriem prezentácie. Matematika vyjadruje myšlienky pomocou vzorcov, ktoré sa viac podobajú grafike než bežnému textu. S masívnym rozširovaní prevažne textového Webu však vznikol problém ich zápisu. Väčšinou sa tento problém rieši bitmapovým obrázkom, čo má mnoho nedostatkov.

2.1.1 ASCII

Pri vzniku počítačov bol prijatý štandard, ktorým je ASCII (*American Standard Code for Information Interchange*). Definuje základnú sadu znakov, ktoré možno zobraziť na obrazovke počítača pracujúceho v textovom režime. Každý znak je reprezentovaný jedným bajtom, takže je definovaných 256 znakov (8 bitový kód ASCII). Pomocou týchto znakov sa vytvára výsledný matematický výraz.

$$\frac{x^2}{x+y}$$

Takýto matematický vzorec môžeme napísať pomocou dostupných znakov ako x^2/(x+y), alebo trochu názornejšie:

```
 x^2        <PRE>     x^2 <BR>
----                 -----<BR>
 x+y                   x+y <BR></PRE>
```

Ako vidieť, pre jednoduché výrazy sa dá použiť i táto metóda. Pokiaľ však požadujeme zložitejšie výrazy alebo špeciálne symboly, prestáva tento spôsob byť dostačujúci, a preto sa začali hľadať alternatívne riešenia.

2.1.2 Typografické prostriedky

Sú prostriedky určené predovšetkým pre tlač, ale za pomoci externých programov možno vytvorené dokumenty aj prehliadať. T_EX je nielen silný nástroj pre sadzbu textu, ale zahrňuje i skvelú podporu sadzby matematických výrazov. Pomocou jednoduchého systému značkovania možno vytvoriť ľubovoľne zložitý matematický výraz. Možno použiť všetky dostupné matematické symboly a pokiaľ neexistujú, tak ich je možné dodatočne naprogramovať.

Kvôli takejto podpore, ale aj konečným prezentovaným výsledkom sa T_EX a jeho nadstavby stali nepísaným štandardom na akademickej a vedeckej pôde.

Nevýhoda ale je, že generované dokumenty síce sú hypertextové, ale matematický kód nie je príliš prehľadný.

Výhodou je že vygenerované dokumenty je možné previesť na obrázky, ktoré sa potom pridajú do hypertextového dokumentu.

V minulosti bolo možno matematické výrazy prezentovať v HTML tromi spôsobmi:

- pomocou ASCII znakov
- využitím nových vlastností Html prehliadačov, respektíve nových štandardov ako sú CSS či nové typy kódovania **UTF** (*UCS Transformation Format* – transformačný formát pre UCS), čo je mechanizmus pre kompresiu dát UCS-2 a UCS-4 pre prenos dát medzi systémami, ktorý dokáže skomprimovať bežný textový súbor až o 75%.
- využitím externých programov. Ako príklad sa dá uviesť nasledujúci príklad: Matematický výraz napísaný v zdrojovom kóde T_EXu (možno použiť aj editor rovníc v MS Worde) sa vygeneruje do výsledného tvaru. Takto získaný objekt sa potom musí vhodný konvertorom previesť na grafický formát (GIF, JPEG) a tento obrázok sa vloží do hypertextového súboru a v hypertexte je naň vytvorený odkaz.

Nevýhodou spojenou so zápisom matematických vzorcov prostredníctvom bitmapových obrázkov je mnoho:

- dátová veľkosť nie je zrovna ideálna

- zložitá manipulácia s grafickým editorom
- nemožnosť interpretácie dát
- dynamická tvorba takýchto vzorcov je nemožná – nie je možné meniť vzorec priamo v kóde

V súčasnosti sú už vyvinuté nové možností, ako použiť matematické výrazy v hypertextových dokumentoch.

2.2 MathML

Okolo roku 1995 organizácia W3C uznala, že nemožnosť výmeny matematických vzorcov predstavuje vážny problém. Objavili sa návrhy na rozšírenie HTML (vtedy HTML 3.0) o nové elementy, umožňujúci formátovanie vzorcov. Neskôr sa ujasnilo, že by mal vzniknúť obecný mechanizmus (XML) a zápis vzorcov by mal byť jednou z jeho podmnožín. Nakoniec vznikol MathML (*Mathematical Markup Language* – matematický značkovací jazyk), podmnožina XML. Ide o značkovací jazyk podobný svojou štruktúrou HTML.

MathML bolo navrhnuté s niekoľkými základnými cieľmi:

- Možnosť konvertovať existujúce dokumenty do MathML.
- Možnosť pripojiť MathML k HTML a interpretovať ho prehliadačmi (to bolo dosiahnuté vďaka odvodenia z XML).
- Možnosti získavania vzorcov zo zdrojového zápisu, ktorý môže byť určeným programom interpretovaný a vyhodnotený.

Prezentácia MathML je do značnej miery založená na štýloch. Vďaka nim je síce možné zobraziť MathML v prehliadači, ale je treba rozšíriť schopnosti prehliadačov o elementy pre zobrazenie MathML. Sám jazyk MathML však prostredníctvom elementov a atribútov opisuje druh obsahu (číslo, premenná, operátor, ...) a v podstate aj požadovaný vzhľad, pretože väčšina matematických konštrukcií má kodifikovane určenú grafickú podobu.

2.2.1 Základný popis MathML

MathML je založené na XML, preto si pri tvorbe matematických vzorcov vystačíme len s textovým editorom a vhodným prehliadačom. Štruktúra MathML je tvorená, rovnako ako v prípade XML, elementmi, atribútmi a entitami pre zápis zvláštnych znakov. MathML prvky možno rozdeliť do niekoľko kategórií:

- prezentačné prvky opisujú štruktúru vzorca (horný index, dolný index),

- významové prvky priamo opisujú matematický objekt (napríklad “plus” alebo “vector”),
- prvky rozhrania sú tie, ktoré slúžia k zapojeniu MathML do dokumentu HTML, XML a ďalších.

V nasledujúcich tabuľkách uvádzame výber z prezentačných prvkov MathML:

Chápanie prvkov

<mtext>	uzatvára v sebe normálny text
<mspace/>	medzera (ekvivalent entity v (X)HTML)
<mi>	uzatvára identifikátory (premennej)
<mn>	uzatvára čísla
<mo>	uzatvára operátory (+, -, /, *) a zátvorky

Schémy

<mfrac>	element označujúci zlomok
<mrow>	horizontálna skupina prvkov
<msqrt>	druhá odmocnina
<mroot>	Odmocnina

Tabuľky

<mtable>	vyhradzuje obsah tabuľky
<mrow>	riadok tabulky
<mtd>	bunka tabulky

Vzájomná poloha prvkov

<msub>	dolný index
<msup>	horný index
<msubsup>	pridá dolný i horný index presne nad sebou
<munder>	obsah pod vybraným prvkom
<mover>	obsah nad vybraným prvkom
<munderover>	obsah pod i nad prvkom

Pre lepšie pochopenie využitie týchto prvkov je uvedená následujúca ukážka vzorca.

Vytvoríme XML dokument ktorý bude obsahovať zápis MathML (**Obr. 1.5**).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<math
xmlns="http://www.w3.org/1998/Math/MathML">
<msqrt>
  <mn>2</mn>
  <mo>+</mo>
  <mi>x</mi>
</msqrt>
<mo>-</mo>
<mfrac>
  <mn>2<mo>-</mo><mi>x</mi></mn>
  <mn>3</mn>
</mfrac>
<mo>+</mo>
<msubsup>
  <mn>X</mn>
  <mrow>
    <mn>i</mn>
  </mrow>
  <mrow>
    <mn>2</mn>
  </mrow>
</msubsup>
</math>
```

$$\sqrt{2+x}-\frac{2-x}{3}+X_i^2$$

Príklad zobrazenia vzorca v prehliadači Mozilla 1.7

Obr. 1.5 Zápis matematického výrazu v MathML

Môžeme si všimnúť chápanie zadaných hodnôt. Zlomok <*mfrac*> môže obsahovať dve čísla (dva prvky <*mn*>), pokiaľ ale chcete zapísať ako čitateľ či menovateľ ďalší vzorec, musí byť uzatvorený v prvku <*mn*>, inak by bola konštrukcia neplatná. Za zaujímavé možno bezpochyby pokladať tiež <*mrow*>, ktorý definuje horizontálne rozloženie prvkov. Dôležité je členenie do skupín pomocou elementov <*mn*> - číslo, <*mi*> - premenná a <*mo*> - operátor.

Samotné dokumenty MathML budeme ale potrebovať nejakým spôsobom integrovať do dokumentu (X)HTML. Ide to veľmi jednoducho, stačí vložiť požadovaný zápis na vybrané miesto a uzavrieť ho do elementu <*math*>. MathML však vychádza z XML, takže pre neho platia rôzne pravidla XML, napríklad odkazy vo vnútri vzorca je nutné prevádzať prostredníctvom technológie Xlink.

2.2.2 Podpora MathML v prehliadačoch a editoroch

Tak ako väčšina technológií založených na XML, je aj MathML určené pre internet a s nimi spojené prehliadače. MathML je možné spracovať v veľkom množstve matematických programov. Programy a aplikácie, ktorými sa budeme zaoberať, rozdelíme do niekoľkých kategórií:

- internetové prehliadače (MS IE, Mozilla, Opera...)
- plug-in pre prehliadače
- editory MathML (matematické programy ktorých výstup môže byť práve MathML)

2.2.2.1 Podpora v prehliadačoch

Podpora MathML v prehliadačoch je nadmieru nedostatočná. Najlepšie si vedie Amaya (testovací prehliadač W3C), ktorý umožňuje zobrazovanie a editáciu vzorcov. Pomerne slušne si stojí Mozilla (Obr. 1.4), ktorá MathML podporuje, ale pre plnú podporu je potrebné si stiahnuť z domovskej stránky fonty (v opačnom prípade sa sem tam objaví chybička). Internet Explorer (Obr. 1.3), ani Opera nepodporujú MathML vôbec, i keď táto neuspokojivá situácia by sa mala postupom času zmeniť k lepšiemu. MS IE sám popisuje riešenie podpory MathML v odpovediach na listy od užívateľov a doporučuje plug-in. Čo sa týka Opery, podpora asi vôbec neexistuje, aspoň na domovských stránkach Opery nenájdeme jediný náznak o MathML.

2.2.2.2 Plug-in pre prehliadače

Samotné prehliadače je možné doplniť o plug-in (podporné programy), ktoré umožnia prostredníctvom prehliadača vzorec konkrétne zobraziť. Asi najznámejším a najkvalitnejší je **TechExplorer Hyper Media Browser** [22], pôvodne od IBM, teraz vyvíjaný firmou Integre, ktorý umožňuje zobraziť MathML a T_EX v Microsoft Internet Exploreri a v Netscape Navigatore. Zobrazenie prebieha pomocou elementu *embed*. Pomocou MathML by zápis vzorca X^2 vyzeral nasledovne:

```
<embed type="text/mathml" mmldata="
<math>
  <semantics>
    <mrow>
      <msup>
        <mi>X</mi>
        <mn>2</mn>
      </msup>
    </mrow>
   <annotation encoding='MathType-MTEF'>
  </annotation>
  </semantics>
</math>">
```

Pomocou T_EXu by zápis rovnakého vzorca vyzeral takto:

```
<embed type="application/x-techexplorer" textdata="$$X^2 $$" >
```

Inou možnosťou môže byť **MathPlayer** [17], určený pre Microsoft Internet Explorer 5.5 a vyššie verzie. Veľkým plusom pre tento plug-in je cena – zadarmo. Umožňuje zobrazenie MathML 2.0. Lenže ani tento plug-in nepodporuje priame zobrazenie MathML, kedy v súbore HTML, XHTML alebo XML je časť kódu MathML ohraničená elementom *math.* V HTML dokumente je nutné doplniť element *html* nasledovne

```
<html xmlns:m="http://www.w3.org/1998/Math/MathML">
```

... do hlavičky vložiť objekt...

```
<object id="MathPlayer" classid="clsid:32F66A20-7614-11D4-BD11-
00104BD3F987" codebase="http://www.dessci.com/dl/mathplayer.cab">
</object><?import namespace="m" implementatio="#MathPlayer" >
```

... a potom už iba vložiť zdrojový kód MathML, v ktorom je nutné všetky značky doplniť o mennú priestorovú jednotku m (<m:mn>9<-m:mn>). Ako vidno, jednoduchému MathML sa veľmi rýchlo vzďaľujeme.

Do tretice je tu plug-in od HP, prezentovaný na stránkach W3C, HP **EzMath** [18], (Obr. 2.7) ktorého použitie je tiež značne neprirodzené. Súčasťou archívneho balíčku (asi 137 kB) je plug-in, použiteľný pre Microsoft Internet Explorer aj Mozillu a editor EzMath, ktorý je veľmi jednoduchý a práca s ním nie je príliš obtiažna. Výsledkom tohto editoru je opäť element *embed* a vzorec je prezentovaný ako text atribútu "alt".

```
<embed type="text/ezmath"
pluginspage="http://www.w3.org/People/Raggett/EzMath" alt="X^2">
```

Príbuznosť **EzMath** s MathML je dosť vzdialená, editor síce umožňuje export do MathML, ale už nezaisťuje podporu v prehliadači.

2.2.2.3 Editory MathML

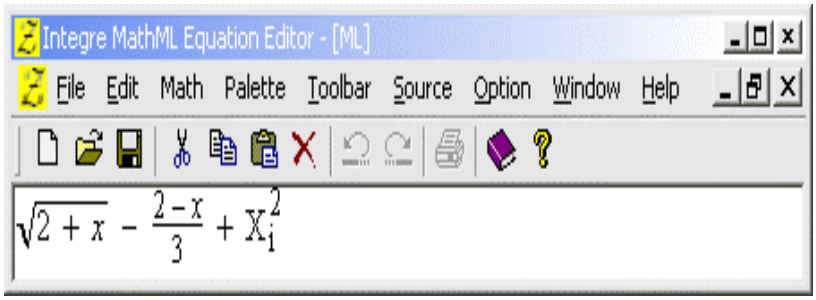
MathML je už podporované vo väčšine veľkých matematických systémov. V systéme **Matematica** [19] je podpora zaistená od verzie 4.0, táto podpora znamená možnosť importu MathML i exportu.

Veľmi dobrým editorom rovníc je **Math Type 5** [20] (Obr. 2.8), ktorý je naviac k dispozícii zadarmo na určitý časový interval. Verzia 3 je súčasťou sady Microsoft Office, kde vystupuje ako Equation Editor (Obr. 2.6). V Exceli a Worde umožňuje Equation Editor export do formátu gif alebo do VML (Vector Markup Language). Umožňuje vytvoriť asi akýkoľvek vzorec a ten následne uložiť vo formáte Encapsulated PostScript, Gif a Windows Metafile. Ďalej je možný preklad do TEXu a MathML. Tento preklad prebieha kopírovaním. Vzorec v MathType kopírujeme a následovne vkladáme do HTML editoru (už ako MathML alebo ako TEX).

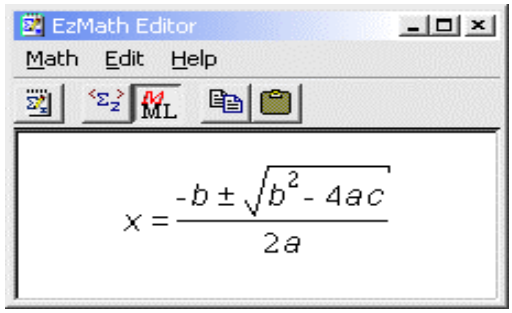
Nevýhodou je, že program nepodporuje opätovné vloženie kódu MathML a jeho preloženie do pôvodného vzorca.

Veľmi dobrým editorom je **WebEQ** [21], ktorý umožňuje vytvárať MathML vizuálne, aj priamo pomocou zdrojového kódu. Súčasťou balíku je aj prekladač z TEXu do MathML.

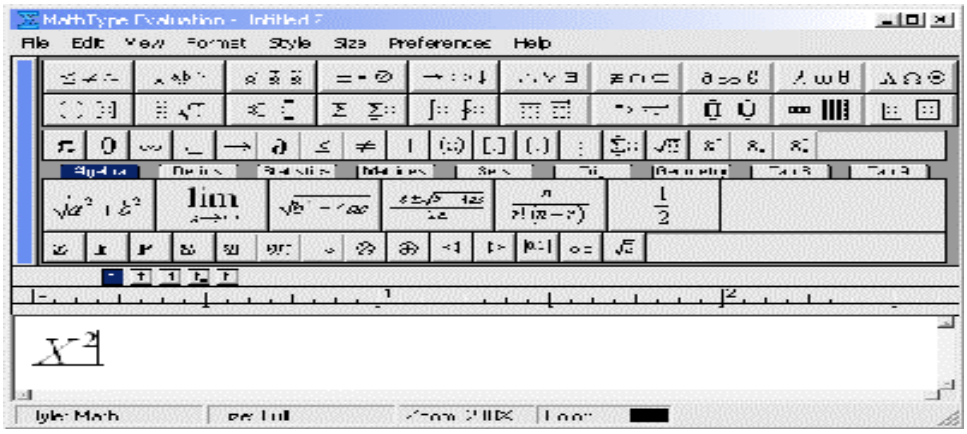
Ďalším už menej atraktívnym editorom je MathML Equation Editor od spoločnosti Integre, a ich free verzia 1.1.1. Práca s nim už však nie je tak jednoduchá ako s MathType. Výhodou tohto editora je možnosť otvárania a ukladania súborov MathML (.mml)



Obr. 2.6 MathML Equation Editor



Obr. 2.7 EzMath Editor



Obr. 2.8 Editor MathType5 (plná veľkosť, cca 10 kB)

2.3 MimeTEX

Ďalšou z možností ako publikovať prvky matematiky je aplikácia MimeTEX. Je to program podliehajúci licencii GPL. Pre inštaláciu je nutné iba umiestniť na server do cgi-bin adresára stiahnuté súbory z domovskej stránky MimeTEXu [8]. Tento program funguje tak, že rozoberie matematický výraz napísaný v LATEXu a vytvára buď obrázok vo formáte *.gif* , alebo vytvorí MIME xbitmapu, čo je lepšie ako zvyčajne používaná texovská *.dvi*. Tu si môžeme hneď všimnúť, že matematický zápis v texovskom formáte je priamo vložený do html kódu a jeho prevedenie je sa nachádza v tagu <img>, ako text atribútu "src".

```
<img src="/cgi-bin/mimetex.cgi? \frac{9}{s^2 +2 s +10}"
border="0" align="absmiddle">
```

To je výhoda oproti vkladaniu už vopred napísaných a uložených obrázkov do adresára. V prípade zmeny štruktúry vzorca sa iba prepíše matematický kód napísaný v hypertextovom dokumente a netreba zbytočne znovu generovať nový obrázok. Pri generovaní obrázku je potrebné opätovne napísať celý vzorec a pri veľkých vzorcoch to môže zabrať viac času. Šetrí sa takto teda aj čas. Ako vidno z ukážky, je možné používať aj atribúty *border*, ktorý určuje druh ohraničenia vkladaného matematického prvku, či *align* určujúci polohu v konečnom hypertextovom dokumente. Výhoda je tiež rýchlosť, ktorou sa zobrazujú matematické výrazy. Je to asi najrýchlejší spôsob transformácie takto napísaných vzorcov.

MimeTEX v súčastnej dobe obsahuje šesť fontov pre veľkosť zapisovaných znakov a to obyčajná deklarácia \tiny, \small, alebo \normalsize (táto veľkosť písma je štandardne nastavená), ďalej to je \large, \Large, \LARGE. \LARGE\sqrt{x^2+y^2} spraví \sqrt{x^2+y^2} len tri veľkosti nad \normalsize. Obsahuje taktiež výrazy ako je \Bigint, čo je

jeden z mnohých extra symbolov dostupných v mimeTEXu. Tento vytvára integrál. Problémom ostáva nastavenie farby textu. Pre túto možnosť neponúka mimeTEX žiadny zápis, ktorým by sa farba textu dala zmeniť. Ak teda použijeme mimeTEX na zobrazovanie matematických výrazov v hypertextových dokumentoch, je najlepšie písať čiernym písmom. Ďalšou nevýhodou je nemožnosť uloženia takto vytvorenej HTML stránky do počítača užívateľa. Lepšie povedané, celá stránka sa uloží bez problémov, ale takto vytvorená matematika sa stratí a užívateľ nemá možnosť ju nejakým spôsobom obnoviť. Táto možnosť zobrazovania matematických výrazov je teda použiteľná iba na internetových serveroch.

2.4 JsMath

JsMath je taktiež možný spôsob publikovania a používania matematických výrazov v elektronických dokumentoch. Tento spôsob však prekonáva nedostatok v tradičnom publikovaní, kedy jednotlivé metódy využívajú obrázky na reprezentáciu matematických výrazov. JsMath [9] využíva fonty, ktoré umožňujú zmenu veľkosti, ak sa zmení veľkosť textu používaného v prehliadačoch. Táto možnosť je zároveň aj použiteľná pre tlač a nemusíme pritom čakať, kým sa desiatky obrázkov stiahnu, aby boli viditeľné v texte. Rýchlosť zobrazenia stránky záleží od množstva použitých matematických výrazov. Pri veľkých počtoch však musíme počítať s väčším časovým intervalom zobrazovania ako pri predchádzajúcej metóde. Matematika je zapísaná v texovskom formáte, takže je jednoducho vytvoriteľná a použiteľná v internetových stránkach.

Pre použitie jsMath netreba nič inštalovať ani mať niečo špeciálne. Matematika sa zobrazuje bez pridania plug-inov, fontov alebo iných súborov.

Balíček jsMath je založený na JavaScripte, CSS (kaskádných štýloch) a unikódových fontoch. Jediné čo teda treba mať na prehliadavanie takto vytvorených stránok je prehliadač, ktorý podporuje takúto technológiu. K takým patria hlavne nové prehliadače, ktoré ho podporujú úplne všetky. Pre inštaláciu je treba stiahnuť jsMath adresár s potrebnými súbormi.

Potom už treba iba matematický výraz napísani v texovskom formáte vložiť do vhodných tagov <SPAN>, pre vloženie matematiky do textu, alebo <DIV> ktorý zarovná matematiku do stredu nového riadku. Ďalej je potrebné vložiť na začiatok dokumentu pred <html> tag kód

```
<script language="JavaScript" src="jsMath/jsMath.js">
```

ktorý volá uložený súbor jsMath.js, ten sa nachádza niekde na disku a my iba sprístupníme jeho URL adresu. Na koniec súboru treba tiež pripojiť kód,

```
<SCRIPT>jsMath.Process()</SCRIPT>
<SCRIPT> jsMath.ProcessBeforeShowing() </SCRIPT>
```

ktorým hovoríme, že je možné vykonať príkazy uvedené v súbore. Ak sa náhodou nachádza nejaký matematický výraz písaný v jsMath za týmto príkazom, nebude na výslednej HTML stránke zobrazený.

Prvý príkaz je vhodnejšie využiť, ak je v publikovanom dokumente veľa matematických výrazov, pretože jsMath nie je moc rýchly a postupné zobrazenie jednotlivých výrazov môže trvať dlhší čas. Druhý spôsob nezobrazí stránku dovtedy, kým nie je celá matematika správne transformovaná a vložená do textu.

Matematický vzorec napísaný v tomto formáte má nasledujúci tvar:

```
<SPAN CLASS="math"> f(x) = x+2 </SPAN>,
<DIV CLASS="math"> \sum_{i=1}^{n} i = {n(n+1)\over 2} </DIV>.
```

V takto písaných výrazoch je samozrejme možné zmeniť veľkosť zobrazovaných písmen v porovnaní z okolitým textom. Toto môžeme vykonať, ak niekde na začiatku tagu <body> napíšeme následovný kód spôsobujúci, že matematika je o 20% väčšia ako ostatný text. Samozrejme je možné voliť rôzne veľkosti písma.

```
<STYLE TYPE="text/css">
  .typeset {font-size: 120%}
  .normal  {font-size: 120%}
  .math  {font-size: 67%; color: #888888}
</STYLE>
```

Taktiež nie je vôbec ťažké meniť farbu zobrazovanej matematiky, prípadne sa dajú stiahnuť špeciálne fonty, kde si môžeme vybrať taký, v ktorom bude matematika zobrazovaná podľa našich požiadaviek najvhodnejšie.

3 DOM

Štandard DOM (*Document Object Model* – objektorý model dokumentov) sa používa pre vytváranie XML dokumentov a taktiež ho možno použiť aj na editovanie, či získavanie obsahu XML dokumentu. Program MATLAB využíva tento štandard a jednou z možností ako vytvoriť náš XML dokument je práve tento prístup. Väčšina parserov XML je schopná vytvoriť behom spracovania stromový model dokumentu a dovoliť aplikácii, aby pomocou API (Application Programmer Interface - špecifické rozhranie softwarového balíčku, pomocou ktorého môžu tento softwar používať ostatné aplikácie), k tomuto modelu pristupovať. DOM bol vyvinutý konsorciom W3C, predovšetkým pre určenie, akým spôsobom majú webové prehliadače a vložené skripty v budúcnosti pristupovať k dokumentom HTML a XML, ktorý sa zaoberá definíciou rozhrania pre prístup ku konštrukcii v oboch jazykoch. Pre DOM existuje objektový návrh, ktorý predpokladá použitie objektovo orientovaných programovacích a skriptovacích jazykov, ako je C++, Java, alebo JavaScript. DOM je tvorený mnohými rozhraniami [3].

Pre vytvorenie XML dokumentu v programe MATLAB sa používa metóda *CreateXmlDocument*. Táto metóda sa používa k spracovaniu súboru XML alebo dátového prúdu, vytvára strom dokumentu a vracia odkaz na uzol, ktorý reprezentuje celý dokument. Zadaný príkaz sa ukladá do premennej s názvom vytváraného dokumentu.

```
priklady = com.mathworks.xml.XMLUtils.createDocument('priklady');
dokument = priklady.getDocumentElement;
```

Pre rozšírenie vytváranej stromovej štruktúry dokumentu zapísaním nového elementu, sa používa metóda *CreateElement()*, pričom cestu k danému elementu je potrebné vypísať úplne presne a to pomocou metódy *AppendChild()*. Pred tieto metódy sa píše názov jednotlivých elementov, ktorými pri vytváraní nového elementu treba prejsť, čo je náročné na čas aj na písanie. Opäť netreba zabúdať vytváraný element vždy zapísať do rovnakej premennej v MATLABe, lebo inak je vytváraný element neplatný a procesor hlási chybu. Rovnako je na tom aj metóda pre zapísanie textu či čísla do vnútra elementu *CreateTextNode()*. Táto funkcia a taktiež funkcia *SetAttribute* môže obsahovať funkcie MATLABu, čo trochu zvýhodňuje takéto zapisovanie. Najväčšou výhodou takto pracne zapisovaných častí XML dokumentu je to, že parser sám ukončuje jednotlivé elementy, pokiaľ sa už nenachádzajú v ceste k vytváranému elementu. O niečo lepšie je na tom metóda pre zapisovanie atribútov

SetAttribute, ktorá obsahuje dve premenné a to meno vytváraného atribútu a jeho hodnotu. Táto metóda sa dá použiť aj na prepis už existujúceho atribútu, aj na vytvorenie nového atribútu.

```
zadanie = priklad.appendChild(priklady.createElement('zadanie'));

zadanie.setAttribute('hodnotenie','3');

zadanie.appendChild(priklad.appendChild(priklady.createTextNod
e('toto je zadanie prikladu')));
```

Pri ukončení dokumentu, ako bolo už spomenuté, sa ukončia všetky elementy, takže nemôže dôjsť ku chybe a teda takto vytváraný XML dokument bude vždy validný. Ku chybe môže dôjsť iba nesprávnym zápisom. Ako posledná sa používa metóda *XmlWrite*, ktorou zapíšeme celú vytvorenú stromovú štruktúru do plánovaného súboru, v našom prípade **zadanie.xml** a to vložením premennej **priklady** vytvorenej na začiatku súboru.

```
    xmlrado = ['zadanie.xml'];
    xmlwrite(xmlrado,priklady);
```

Použitím predošlých príkladov je možné vytvoriť validný XML súbor:

```
<?xml version="1.0" encoding="utf-8"?>
<priklady>
   <priklad name="TRO">
      <zadanie hodnotenie="3">Uzavretý regulačný obvod tvorý riadený proces
s prenosom <span CLASS="math"> ... </zadanie>
      <moznost ans="baditem">Žiadna z ostatných odpovedí nie je..</moznost>
      <moznost ans="baditem">-0.74</moznost>
      <moznost ans="baditem">1.08</moznost>
      <moznost ans="baditem">-0.13</moznost>
      <moznost ans="gooditem">0.00</moznost>
   </priklad>
</priklady>
```

4 SMARTY

Ďalším spôsobom ako vytvoriť XML dokument je použitie Smarty [23], balíčku pre PHP. Ak sa naň pozrieme viac konkrétne, tak môžeme povedať, že uľahčuje zvládnutie možnosti separácie logických aplikácii a napĺňať dané formy jednotlivými premennými. Najlepšie sa to dá opísať v situácii, kde programátor a šablónový dizajnér majú rozdielne role, ale vo väčšine prípadov sú tie isté osoby.

Jeden cieľ navrhovateľov Smarty je separácia obchodnej a prezentačnej logiky. To znamená, že šablóny môžu obsahovať logiku vo forme, ktorá je iba pre prezentáciu. Tá obsahuje ďalšie šablóny, ktoré obsahujú dáta, ako je farba tabuliek alebo pozadia a iné. Znamená to, že pokiaľ máme dokument, v ktorom sa mení iba časť textu, môžeme tento dokument separovať na dve časti, jednu tú čo sa mení a druhú čo sa nemení. Tá, čo sa nemení sa nazýva šablóna a do nej sa vkladajú na vopred určené miesta meniace sa elementy, ktoré sú definované v druhej časti takým spôsobom, aby bolo vopred jasné, kam patria.

Pri vytváraní dokumentu týmto spôsobom využívame funkcie MATLABu *fopen* pre otvorenie dokumentu **index.php,** do ktorého budeme zapisovať pomocou funkcie *fprintf*. Pri zapisovaní do tohto dokumentu, v ktorom je na začiatku napísaný kód pre PHP, sa uvádza cesta ku balíčku smarty, teda jeho jednotlivým súborom.

```
<?php
 require('/usr/local/lib/php/Smarty/Smarty.class.php');
 $smarty = new Smarty();
 $smarty->template_dir = '.';
 $smarty->compile_dir = '/home/websubory/smarty/templates_c';
 $smarty->cache_dir = '/home/websubory/smarty/cache';
 $smarty->config_dir = '/home/websubory/smarty/configs';

 $smarty->assign('b0','10');
 $smarty->assign('a1','+5');
  ....
  ....
 $smarty->display('tro.tpl');
  ....
  ....
?>
```

V príklade vidno, že deklarované premenné **b0** s hodnotou **10** a **a1** s hodnotou **+5** sa vložia pomocou zápisu *$smarty->assign('premenná','hodnota')* do šablóny **tro.tpl,**

```
<priklad>
<zadanie hodnotenie="4">Uzavretý regulačný obvod tvorí riadený
proces s prenosom <span CLASS="math">G_p(s) = {{$b0}\\over {s^2
{$a1} s {$a0}}}</span>a regulátor s prenosom <span
CLASS="math">G_R(s)= {{$pp} {$dd} s} {$ii}</span>.  Ak sa v čase t=0
žiadaná veličina zmení z hodnoty 0 na  {$ws}, je trvalá regulačná
odchýlka rovná</zadanie>
<moznost ans="{$ans1}">{$moznost1}</moznost>
<moznost ans="{$ans2}">{$moznost2}</moznost>
<moznost ans="{$ans3}">{$moznost3}</moznost>
<moznost ans="{$ans4}">{$moznost4}</moznost>
<moznost ans="{$ans5}">{$moznost5}</moznost>
</priklad>
```

ktorá vyzerá ako prostý text a jednotlivé hodnoty sa vložia na miesta vopred označené danou premennou so znakom $ na začiatku a uzavreté v zložených zátvorkách, teda pred b0 vyzerá následovne : **{$b0}**.

Do súboru s príponou php môžeme vkladať veľké množstvo premenných a taktiež ich vkladať do rôznych šablón príkazom *$smarty->display('*.tpl')*. Každá šablóna musí mať príponu .tpl, inak je neplatná a procesor hlási chybu. Uzavretie dokumentu vykonávame MATLABovskou funkciou *fclose.*

5 Prezentácia XML dokumentu

Pre definíciu vzhľadu XML dokumentu sa používajú takzvané štýlové jazyky. Medzi dva najznámejšie a najpoužívanejšie patria jazyky CSS a XSL. Jazyk XSL sa ďalej rozdeľuje na procesor XSLT a procesor XSL-FO.

5.1 CSS

CSS (*Cascading Style Sheets* – kaskádové štýly) je veľmi jednoduchý jazykový štýl, ktorý využíva textový formát ASCII, takže definíciu štýlov je možné jednoducho vytvárať a upravovať v ľubovolnom textovom editore či procesore. Neumožňuje meniť štruktúru dokumentu alebo napríklad generovať obsah dokumentu.

Pomocou jednoducho popísaných pravidiel možno určiť parametre, ktoré budú aplikované na obsah špecifikovaného elementu alebo skupinu elementov.

CSS pôvodne vzniklo pre potreby prezentácie HTML dokumentov, konkrétne pre použitie vo webových prehliadačoch, pretože autori požadovali väčšiu kontrolu nad prezentáciou webových stránok. Avšak v súčasnej dobe schválená verzia CSS2 rozširuje možnosti CSS i na XML dokumenty. Pri používaní CSS pre HTML mená elementov nerozlišujú malé a veľké písmená, takže body, Body a BODY identifikujú rovnaký element v HTML. V XML sú však v menách rozlišované malé a veľké písmená, takže aj štýly ich musia rozlišovať a je treba dbať na to, aby mená presne odpovedali menu uvedenému v deklarácii v DTD. Okrem nových typov zobrazenia obsahuje CSS2 rozšírené metódy výberu elementov, vylepšenú podporu tabuliek, podporu výstupu pre tlač a ďalšie.

Pravidla musia byť uložené vo zvláštnom súbore a v XML dokumente sa uvedie iba odkaz pomocou spracovania inštrukcie v hlavičke dokumentu:

```
<?xml-stylesheet href="styl.css" type="text/css"?>
```

5.2 XSL a XSLT

Je to v podstate mechanizmus pre formátovanie obsahu dokumentu XML a manipuláciou s ním. Jazyk XML umožňuje štruktúrovať dáta uložené v jednotlivých dokumentoch. XSLT (Extensible Stylesheet Language Transformations – transformácia v jazyku XSL) je v skutočnosti súčasťou jazyka XSL, čo je v podstate iba sada formátovacích objektov, ako sú

bloky, inline objekty a bunky tabuľky. Každý z nich má radu vlastností, ktoré môžu byť využité k formátovaniu obsahu profesionálnym spôsobom [2].

Medzi základné manipulácie s obsahom patrí roztriedenie databázových záznamov vo formáte XML alebo uloženie dát v dokumente HTML, textovom alebo znovu vo formáte XML. Táto technológia umožňuje užívateľom priamu manipuláciu s obsahom dokumentu.

Príklad XSLT dokumentu:

```
<?xml version="1.0" encoding="windows-1250"?>
<xsl:stylesheet version="1.0"
             xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output indent="yes" encoding="windows-1250"
                                                  method="html"/>
<xsl:template match=”/priklady”>
<html>
<head>
<title>Testy na skušky</title>
<table>
<tr><td><xsl:copy-of select=”zadanie”/></td><td><xsl:value-of
select=”bod”/></td></tr>
<xsl:for-each select=”moznost”/>
<tr><td></td><td><xsl:value-of select=”moznost”/></td></tr>
</xsl:for-each>
</table>
</head>
</html>
</xsl:template>
</xsl:stylesheet>
```

Využitím inštrukcie *<xsl:output>* sa prednostne určuje typ spracovania, a teda aj výsledného dokumentu. Tento typ určuje napríklad to, či procesor XSLT vloží do výsledného dokumentu inštrukciu pre spracovanie <?xml version = “1.0“?>. Umožňuje tiež určiť typ obsahu MIME (“text/xml”,”text/html”) dokumentov. Ak si zvolíme ako výstupný formát HTML, väčšina procesorov rozpozná, že určité elementy jazyka HTML nevyžadujú ukončovacie tagy.

Z množstva možných použiteľných atribútov sa najčastejšie používa *method*. Jeho prostredníctvom sa určuje typ výstupného stromu. Prednostne je nastavená výstupná metóda HTML. Vhodnejšie je však nastaviť vlastnú metódu výstupného formátu a to medzi tromi zvyčajnými hodnotami, “xml”, ”html” a ”text”.

5.2.1 Transformácia v XSLT

Procesor XSLT sa používa k tomu, aby bolo možné použiť jazyk XSLT k transformácii dokumentu XML na iné dokumenty (txt, html, rtf, ...). Jazyk XSLT môžeme v takej situácii využiť tromi spôsobmi:

- **Vo vnútri samotného programu nazývaných procesory XSLT**. Existuje niekoľko programov, obvykle založených na jazyku Java, ktoré samotný prevod spoľahlivo uskutočnia.
- **Vo vnútri klientského programu**. Klientský program (napríklad prehliadač) môže dokument transformovať iba vtedy, keď v ňom uvedieme inštrukciu <?xml-stylesheet?> Do určitej miery môže takto dokumenty XML spracovať aj prehliadač Internet Explorer.
- **Na serveri.** Serverový program, napríklad servlet jazyka Java, môže využívať štýl XSLT k automatickej transformácii dokumentu a k odoslaniu výsledku na klientsky počítač.

5.2.1.1 Použitie samostatných procesorov XSLT

Jedným z najbežnejších prostriedkov pre transformáciu dokumentu pomocou XSLT sú samotné procesory XSLT. Tých už v súčasnej dobe existuje skutočne veľmi mnoho. Spúšťajú sa obvykle z príkazového riadku (tj. Z okna MS-DOS v systéme Windows). Procesor očakáva prostredníctvom argumentu zdrojový dokument XML, dokument s definíciou štýlu XSLT a názov cieľového dokumentu, ktorý hodláme vytvoriť. Niektoré z mnohých sú: Saxon [13], Xalan [14], XT [15] ...

XT

Systém XT, ktorý vytvoril James Clark, môžeme získať zo serveru
www.jclark.com/xml/xt.html [15].
XT je taktiež implementáciou v jazyku Java. V stiahnutom balíčku je archív jazyka Java – xt.jar. Následujúci príklad ukazuje, ako možno rovnaké transformácie dosiahnuť v systéme Windows pomocou príkazu xt.exe:

```
C:\testy>xt zadanie.xml testy.xsl testy.html
```

5.2.1.2 Použitie prehliadačov k transformácii dokumentov XML

Aj Microsoft Internet Explorer, Mozilla, či Firefox obsahujú určitú podporu XSLT. Ale IE je ďaleko viac prepracovanejší a používanejší, a preto sa zameriame na jeho verziu 6.0. O podpore XSLT v aplikácii IE sa môžeme dočítať na domácej adrese MS IE. Obsahuje presnú syntax XSLT. Predchádzajúce verzii však túto syntax implicitne nepodporovali a preto sa museli súbory s príponou .xml a .xsl trochu pozmeniť. Pretože verzia 6.0 tento problém nemá a je voľne stiahnuteľná, nebudeme sa ďalej týmto problémom zaoberať.

Pre použitie takejto transformácie treba do dokumentov XML dopísať nasledujúci príkaz, ktorým hovoríme, kde sa má transformačný XSL súbor hľadať

```
<?xml-stylesheet href="Skuska.xsl" type="text/xsl"?>
```

5.2.1.3 Transformácia XSLT na serveri WWW

Transformácia XSLT sa naviac môže realizovať priamo na serveri WWW, takže zdrojové dokumenty budú spracované ešte skôr, než ich server WWW odošle klientskému prehliadaču. Najčastejšie sa takto prevádzajú dokumenty XML na stránky HTML.

Na rozdiel od ostatných prevodov XSLT, sa pri prevádzaní dokumentu na strane serveru nevyhneme programovaniu. Transformácia XSLT na strane serveru sa realizuje pomocou troch hlavných technológií WWW: Javy, stránok JSP (Java Server Pages) a stránok ASP (Active Server Pages).

5.2.2 Výstupná metóda HTML

Zvolením výstupnej metódy HTML (str. 5), sa predpokladá, že procesor XSLT vykoná určité rutinné operácie. Pri nastavení tejto metódy, by nemal chýbať atribút *version*, ktorý vyjadruje verziu jazyka HTML. Implicitná hodnotou je 4.0.

Použitím atribútu *ident* a jeho hodnoty „yes“ môže procesor XSLT pridať prázdny priestor tak, aby bol strom výstupného dokumentu odsadzovaný. Odsadenie však neovplyvňuje vzhľad výsledku zobrazeného v prehliadači WWW. Implicitná hodnotou je „yes“.

```
<xsl:output indent="yes" encoding="windows-1250" version=”4.0” 
method="html"/>
```

Výsledkom je HTML stránka napríklad tvarov a farieb ako je znázornená na Obr. 3.11.

5.2.3 Výstupný formát text

Tento typ výstupu zastupuje rýdzi text. Výstupný dokument obsahuje iba text stromu zdrojového dokumentu. Znamená to, že procesor XSLT vytvára výsledný strom tak, že na výstup odošle reťazové hodnoty všetkých textových uzlov bez akýchkoľvek zámien riadiacich znakov. Atribút *encording* určuje kódovanie použité procesorom XSLT k prevodu znakov na postupnosť bajtov. Pokiaľ teda bude výsledný dokument obsahovať znaky, ktoré nemožno pomocou nastaveného kódovania znázorniť, procesor bude generovať chybu.

```
<xsl:output indent="no" encoding="windows-1250" method="text"/>
```

Na druhej strane je zrejmé, že výstupná metóda text nie je určená výhradne pre tvorbu prostého textu. Používa sa pre tvorbu všetkých textových formátov, ktoré sú založené na XML alebo HTML. Napríklad môžeme urobiť prevod XML dokumentu na L^AT_EX formát. Formát L^AT_EX využíva vložené formátovacie značky a ten sa môže ďalej spracovať, čím získame PDF (Portable Document Format) Dokument (Obr. 3.13), prípadne PostScript Dokument. Alebo môžeme využiť výstupný formát text na tvorbu jazyka WebCT, ktorý sa ďalej využíva ako importný súbor s otázkami a odpoveďami pre e-learningový program Moodle (Obr. 3.12).

5.2.4 Výstupný formát XML

Použitím výstupnej metódy XML vytvára procesor XSLT správne vytvorenú externú všeobecne analyzovanú entitu. Správne vytvorená je i v prípade, že koreňový uzol výsledného stromu obsahuje jeden vnorený uzol typu element, ale žiadne vnorené textové uzly. Táto výstupná metóda obsahuje atribút *version* čo je číslo verzie XML, ktorý bude odoslaný na vstup. Implicitná hodnota je 1.0. Použitím atribútu *doctype-systém*, má procesor XSLT deklaráciu typu dokument <!DOCTYPE priklady SYSTEM "Skuska.dtd"> vložiť tesne pred prvý element nového XML dokumentu. Znamená to, že názov za značkou <? DOCTYPE je v skutočnosti názvom koreňového elementu. Ak však používame atribút *doctype-public*, mal by procesor XSLT zobraziť vo výstupe hodnotu “PUBLIC” nasledovanú verejným identifikátorom a následovne aj identifikátorom systémovým. Pri použití doctype-systém bude generovať reťazec “SYSTEM”.

```
<xsl:output doctype-system="Skuska.dtd" indent="no"
encoding="windows-1250" method="xml"/>
```

5.3 XSL-FO

Formát XSL-FO spolu s XSLT tvoria špecifikáciu XSL. Táto časť XSL umožňuje formátovať objekt presne do podoby výsledného dokumentu. Umožňuje nastaviť nielen typ a veľkosť písma, ale aj to, kde bude na strane zobrazené a aká bude jeho farba. Je to niečo ako tvorba dokumentov RTF. Skrýva však niečo viac. V dokumentoch XSL-FO možno vytvárať odkazy, ktoré umožňujú užívateľom navigáciu medzi jednotlivými dokumentmi. Tak, ako existuje mnoho procesorov XSLT, existujú aj procesory jazyka XSL-FO. Žiadny z nich sa však úplnému štandardu ani nepribližuje. Sú to napríklad:

- FOP – Jedná sa o aplikáciu v jazyku Java, ktorá načíta strom formátovacích objektov jazyka XSL a vytvorí dokument PDF [10].
- Passive TeX – Balíček TEX, ktorý formátuje výstup jazyka XSL-FO vo formáte PDF. K tomu využíva analyzátor xmltex od Davida Carlisla (XMLTEX) [11].
- TeXML – Prevádza dokumenty XML na dokumenty vo fomáte TEX [12].

Najviac používaným procesorom jazyka XSL-FO je FOP (formatting object processor). Písanie celého dokumentu pomocou formátovacích objektov XSL-FO však nie je vôbec jednoduché. Je vhodný iba pre krátke dokumenty. S takýmto vývinom sa samozrejme predpokladalo, a preto sa zaviedol jazyk pre transformáciu dokumentu XSLT. Inými slovami to znamená, že môžeme najprv vytvoriť štýl XSLT, ktorý použijeme k transformácii dokumentu XML tak, aby výsledný dokument používal formátovacie objekty XSL.
Pre nedostatok času sa tejto metóde ďalej nevenovalo, napriek tomu pre ukážku môžeme uviesť časť FOP dokumentu.

```
<xsl:template match="priklady">
   <fo:root xmlns:fo=http://www.w3.org/1999/XSL/Format>
      <fo:layout-master-set>
      <fo:simple-page-master master-name="page"
          page-height="400mm" page-width="300mm"
          margin-top="10mm" margin-bottom="10mm"
          margin-left="20mm" margin-right="20mm">

           <fo:region-body
             margin-top="0mm" margin-bottom="10mm"
             margin-left="0mm" margin-right="0mm"/>

           <fo:region-after extent="10mm"/>
       </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-name="page">
```

```
<fo:flow flow-name="xsl-region-body">
    <fo:block font-weight="bold" font-size="36pt"
        line-leight="48pt" font-family="Times"
        color="lightblue">
     <xsl:apply-templates/>
     </fo:block>
        .
        .
        .
        .
```

5.3.1 XMLTEX (Passive TeX)

Ak chceme XML dokument značkovať čisto prostriedkami TEXu bez nutnosti používania ďalších pomocných programov, je asi najvhodnejším nástrojom xmltex. Jedná sa o makro nadstavbu na LATEX, ktorá implementuje XML. Tá umožňuje načítanie dokumentu XML a podľa pravidiel definovaných v konfiguračnom súbore prevádzať jednotlivé elementy na sekvencie textového kódu, ktorý potom riadi značkovanie. Autorom LATEXu je David Carlisle [7], ktorý je uznávaným odborníkom ako na oblasť TEXu, tak aj XML.

V okamihu, keď XMLTEXu pridáme dokument XML k spracovaniu, snaží sa nájsť definičný súbor s príponou **.xmt**, opisujúci mapovanie XML na textovej sekvencii. Väzba medzi jednotlivými druhmi dokumentov XML na xmt - súbory je definovaná v katalógu. Vždy sa prehľadáva globálny katalóg, ktorý musí mať meno **xmltex.cfg**, a lokálny, ktorý sa volá rovnako ako dokument XML, ale má príponu **.cfg.**

Definičné xmt súbory:

```
\XMLelement{názov elementu}
           {zachytenie atributu}
           {kód pre začiatok elementu}
           {kód pre koniec elementu}
```

V definícii obsluhy začiatku elementu môžeme použiť aj špeciálne makro **\xmlgrab**. To načíta celý obsah elementu do parametru #1 a sprístupní ho v obsluhe koncového tagu. Napríklad môžeme zmeniť písmo obsahu na kurzíva, kde na začiatku aktivujeme *it*.

```
\XMLelement{zadanie}{}{\xmlgrab}{\it #1\/}
```

Taktiež je možné využívať všetky príkazy použiteľné v TEXu. Týmto spôsobom sa môžeme vyhnúť transformácii XML dokumentu do texovského zápisu a môžeme priamo z tohto dokumentu získať dokument pripravený pre tlač.

5.4 Validácia XML dokumentu

Väčšina prehliadačov overuje, či je XML dokument správne štruktúrovaný. Niektoré z nich však overujú aj skutočnosť, či je tento dokument validný (platný). Dokument XML je validný, ak je k nemu priradená deklarácia typu dokumentu (DTD). Pokiaľ XML dokument uvedenej štruktúre v DTD úplne neodpovedá, nie je validný a procesor bude hlásiť chybu. Znamená to, že DTD alebo schéma definujú množinu pravidiel, ktoré si vynucujú vnútornú konzistenciu príslušného dokumentu. Ak sa dá určiť, či sa dokument danými pravidlami riadi, je dokument validný (platný).

Existujú štandardizované rozhrania (API) pre prácu s týmito dokumentmi. Väčšina producentov validátorov tieto rozhrania používa.

- SAX (Simple API for XML) – je založené na riadení pomocou udatostí. Pomocou rozhrania sa vytvorí väzba medzi udalosťami, ktoré generuje parser a užívateľským kódom. V praxi to znamená, že sa nadefinujú funkcie, ktoré sa zavolajú v okamihu, kedy parser spracováva začiatok elementu, na obsah elementu, na koniec elementu, na komentár na inštrukcie pre zapracovanie, atď. Funkcii sú potom predané všetky potrebné parametre ako napríklad názov elementu [3].
- DOM (Document Object Model) – (str. 22).

6 Generovanie zadaní a riešení problémov LCZA

Ako už bolo spomenuté XML (str. 6) nadväzuje na princípy a konvencie dvoch existujúcich jazykov, HTML (str. 5) a SGML (str. 4) a vytvára jednoduchý a pritom veľmi účinný mechanizmus pre ukladanie spracovanie a šírenie informácií. Táto kapitola sa zaoberá vytvorením XML dokumentu v programe MATLAB [6], jeho validáciou a následovným transformovaním či prípadným spracovaním na konečný požadovaný formát. Obr. 3.9 znázorňuje metódy, akými sa dokument XML spracováva na výsledné formáty v tejto práci.

Okrem uvedených metód v Obr. 3.9 je možné použiť formát XSL-FO, ktorý priamo spracováva dokument XML a vytvára formát PDF, PS, RTF a iné. Formát XSL-FO tvorí spolu s XSLT špecifikáciu XSL [2]. Táto časť XSL umožňuje formátovať objekt presne do podoby požadovaného dokumentu.

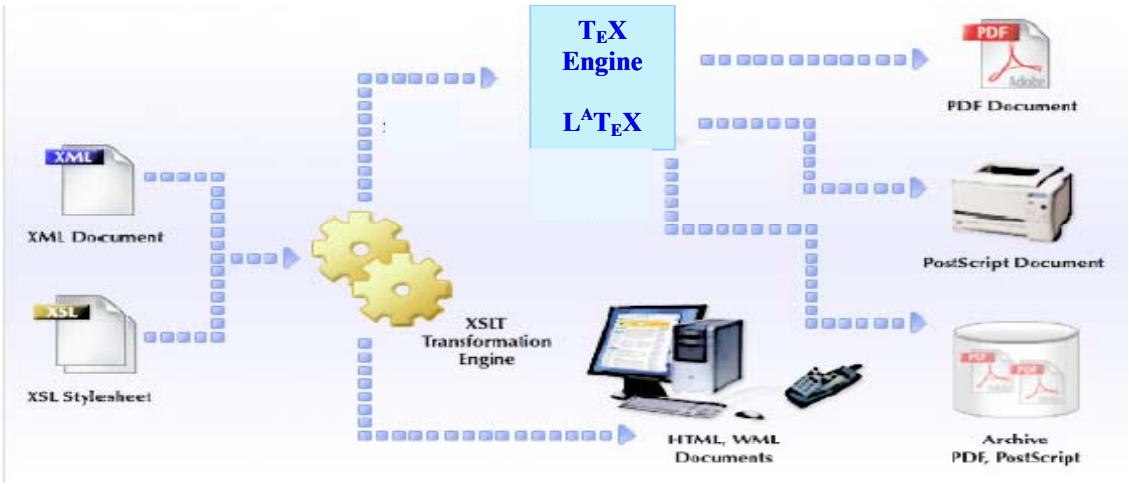
Celé generovanie a spracovanie XML dokumentu budeme ukazovať na príklade **TRO** (Trvalá regulačná odchýlka), aby sme upresnili jednotlivé kroky.

Ten je daný následovne:

Uzavretý regulačný obvod tvorí riadený proces s prenosom $G_R(s) = \frac{b_0}{s^2 + a_1 s + a_0}$ *a regulátor s prenosom* $G_P(s) = P + \frac{I}{s} + D.s$ *. Ak sa v čase t = 0 žiadaná veličina zmení z hodnoty 0 na w, je trvalá regulačná odchýlka rovná:*

$TRO = w.\left(1 - \frac{b_0 . P}{a_0 + b_0 . P}\right)$ - ak máme PD regulátor, teda I zložka je nulová,

TRO = 0 – ak máme PID regulátor.



Obr. 3.9 Spracovanie XML dokumentu využité v tejto práci

6.1 Formulácia problému

Našou úlohou je vytvorenie XML dokumentu, ktorý obsahuje zadania príkladov na skúšku a ich odpovedí, tak aby sa tento dokument mohol ďalej spracovávať do konečnej podoby.

Aplikáciou jazyka XSLT transformujeme vytvorený XML dokument na formáty HTML (str. 29), ktorý sa používa na publikovanie na internetových stránkach, a tak umožníme študentom, aby si podobný test ako dostanú na skúške, vopred pozreli a pokúsili sa odpovedať na kladené otázky. Druhou transformáciou získame PDF formát ktorý je vhodný okrem iného aj pre tlač, teda sa dá využiť pri tlačení testov a ich riešení na skúšky. Posledným formátom ktorý dostávame je WebCT formát, ktorý slúži pre export do vzdelávacieho programu Moodle. Tu je možné si uložiť všetky možné typy príkladov a vybrané príklady ďalej vkladať do testov.

Je potrebné podotknúť, že okrem generovania XML dokumentu a vytvorenia XSLT dokumentov pre jeho transformáciu je potrebné riešiť problém vkladania matematiky do textov. V súboroch na generovanie otázok a odpovedí sa nachádzajú texty zadania ktoré sa nemenia. Nemenia sa ani matematické výrazy, ale menia sa ich číselné hodnoty. Preto je potrebné vytvoriť taký systém, ktorý by vypočítal číselné hodnoty a dosadil ich do textu zadania a potom ich ako celok uložil do vytváraného XML dokumentu.

Preto je treba vytvoriť účinný systém ktorý by vytváral XML dokument a tiež vytvoriť XSLT dokumenty, ktoré by transformovali tento dokument na výsledné formáty HTML, PDF, WebCT.

6.2 Štruktúra vytváraného XML dokumentu

Vytváraný dokument musí spĺňať štruktúru predpísanú v DTD (str. 6) dokumente,

```
<!ELEMENT priklady (priklad+)>
<!ELEMENT priklad (podpriklad, zadanie, moznost)>
<!ATTLIST priklad name CDATA #REQUIRED>
<!ELEMENT podpriklad (zadanie, moznost)>
<!ELEMENT zadanie (span)>
<!ELEMENT zadanie (#PCDATA)>
<!ATTLIST zadanie hodnotenie CDATA #REQUIRED>
<!ELEMENT span (#PCDATA)>
<!ELEMENT moznost (span)>
<!ELEMENT moznost (#PCDATA)>
<!ELEMENT span (#PCDATA)>
<!ATTLIST moznost ans (gooditem, baditem) #REQUIRED>
```

v ktorom sa hovorí, ako presne má dokument vyzerať. Deklarácia *ELEMENT* sa používa na definíciu tagov a deklarácia *ATTLIST* hovorí o použití elementov v dokumente. V dokumente je možné používať nasledovné elementy:

<priklady> - hlavný element (Parent), všetky ostatné sú jeho podelementy (Childs)
<priklad> - element označujúci jeden problém a obsahuje atribút *name*, s hodnotou názvu príkladu.Príklad môže obsahovať elementy *zadanie*, *moznost* a taktiež *podpriklad* v prípade, že problém má svoje podotázky.
<podpriklad> - môže obsahovať iba elementy *zadanie* a *moznost*
<zadanie> - obsahuje text zadania
<moznost> - obsahuje text odpovede
 - označenie matematických výrazov

a ich ukončovacie elementy.

Elementy *zadanie* a *moznost* môžu taktiež obsahovať element *span* s atributom *CLASS* a jeho hodnotou *math,* pre označenie matematických prvkov.

Element zadanie obsahuje atribút *hodnotenie*, v ktorom sa uvádza bodová hodnota zadania príkladu. Element *moznost* obsahuje atribút *ans* ktorý môže nadobúdať dve hodnoty a to *gooditem* v prípade správnej odpovede, alebo *baditem* v prípade nesprávnej odpovede. Štruktúra dokumentu by teda mala vyzerať následovne:

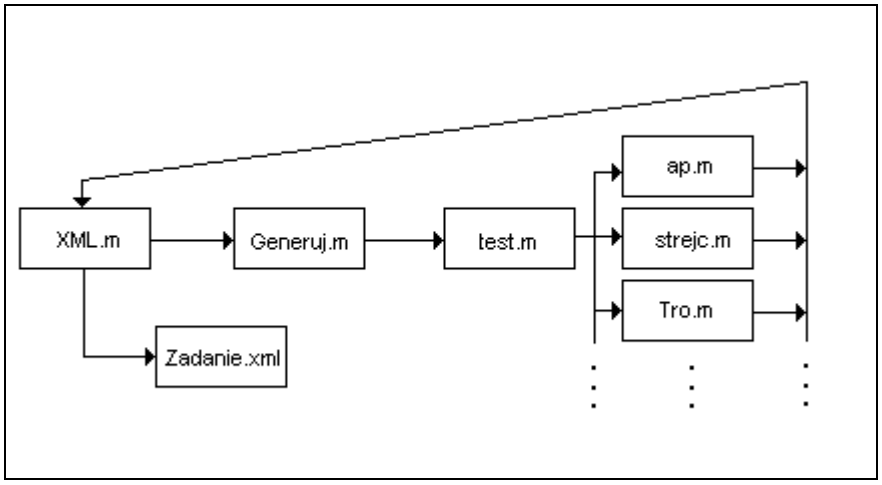
```
<priklady>
<priklad name="">
<zadanie hodnotenie="4"><span CLASS="math"></span></zadanie>
<moznost ans="gooditem"></moznost>
<moznost ans="baditem"></moznost>
<moznost ans="baditem"><span CLASS="math"></span></moznost>
<moznost ans="baditem"></moznost>
<moznost ans="baditem"></moznost>
</priklad>
</priklady>
```

6.3 Vytvorenie XML dokumentu v programe MATLAB

XML dokument je vytváraný v programe MATLAB [6]. Pri zapisovaní dokumentu takéhoto formátu môžeme v programovacom prostredí MATLABu použiť dve metódy vytvárania. Tieto metódy vytvárania XML dokumentu sa navzájom líšia štruktúrou, akou je dokument

zapísaný, a teda aj množstvom použitého zdrojového kódu. Jednou možnosťou je použitie zapisovania do súboru a druhou je metóda DOM (str. 22). Môžeme však použiť aj tzv. nepriamu metódu vytvárania XML dokumentu a tou je balíček SMARTY (str. 24).

V diplomovej práci sa budeme ďalej venovať metóde zapisovania do súboru, hlavne kvôli jej prehľadnosti. Ak sa pozrieme na postup vytvárania nášho XML dokumentu v programe MATLAB (Obr. 3.10) vidíme, že sa využívajú viaceré súbory, ktoré sú však rozdelené do jednotlivých typov skupín. Hlavný a zároveň spúšťací súbor **XML.m** volá ostatné m-súbory (súbory vytvorené v programe MATLAB) a taktiež otvára a na konci uzatvára vytváraný XML dokument. Do dokumentu sa zapisujú zadania s možnými odpoveďami na kladenú otázku. Hlavný program volá na začiatku súbor **Generuj.m**, ktorý výberie z databázy možných príkladov také, aby spĺňali podmienku počtu bodov celého testu a ich názvy uloží do súboru **test.m**. Ďalej sa postupne volajú vybrané príklady, obsahujúce zápis pre výpočet číselných hodnôt v zadaní či odpovediach a samotný text zadania a odpovede. Následne sa zapisujú do vytváraného súboru s príponou xml pomocou funkcie **moznost**. V našom prípade sa vytváraný XML dokument sa uloží pod názov **zadanie.xml** (Príloha D).



Obr. 3.10 Generovanie súboru Zadanie.xml v MATLABe

V jednotlivých súboroch s názvami príkladov (*ap.m*, *strejc.m*, *Tro.m*, ...) sa nachádzajú potrebné matematické výpočty pre číselné hodnoty uvedené v zadaní, ako aj hodnoty uvedené v odpovediach.

```
w=fix(rand(1,1)*10)+1;
[ws,e]=sprintf('%.0f ',w);
num=fix(rand(1,1)*10)+1; b0=num(1,1);
den=fix(rand(1,2)*10)+1;
a1 = den(1,2); a0 = den(1,1);
xx=fix(rand(1,3)*10)+1; pp=xx(1,1);ii=xx(1,2);dd=xx(1,3);
if (randn(1,1)<0)
  ii=0;
  tros = w*(1 - (b0 * pp)/ (a0+b0*pp) );
else
  tros = 0;
end
.
.
.

zadaniee=sprintf('<zadanie hodnotenie="3">Uzavretý regulačný obvod
tvorí riadený proces s prenosom <span CLASS="math">G_p(s) = { %.0f
\\over { s^2 %+.0f s %+.0f }}</span> a regulátor s prenosom <span
CLASS="math">G_R(s)= {%.0f %+.0f s} %s</span>.  Ak sa v čase t=0
žiadaná veličina zmení z hodnoty 0 na  %s, je trvalá regulačná
odchýlka rovná </zadanie>',b0,a1,a0,pp,dd,ii,ws);

choic = {zadaniee,{ss2,ss3,ss4,tross},'TRO'};
answers = moznost(choic,fw,0,1,0);
```

Po vykonaní výpočtu, (uskutoči sa ako prvý) sa vypočítané hodnoty vkladajú do textu vytváraného XML dokumentu. Možno teda povedať že vytváranie XML dokumentu je rozdelené na dva problémy. Prvým je problém výpočtu matematických hodnôt a tou druhou je vkladanie týchto hodnôt do textu zadania. Matematické hodnoty sú zapisované do elementov *span*. Názov tohto elementu nie je podstatný a mohol by sa volať úplne inak, pretože jazyk XML nám umožňuje vytvárať elementy ľubovolných názvov.

Takéto vytváranie zadaní nám umožnila funkcia *sprintf* (používa sa aj v programovacom jazyku C++ a iných). Jej úlohou je vloženie hodnoty premennej (napr. b0), ktorá obsahuje číselnú hodnotu, na vopred označené miesto v texte (%.0f). Získame teda premennú *zadaniee* obsahujúcu text zadania spolu s matematikou. Tá sa vkladá do vytvorenej štruktúry spolu s vypočítanými hodnotami odpovedí. Podobným spôsobom je možné vytvoriť tento dokument aj s použitím balička SMARTY (str. 24) pre jazyk PHP.

Pre náhodné usporiadanie odpovedí sa používa vytvorená funkcia **moznost**, ktorá má 5 vstupných argumentov a jeden výstupný, ktorým vykoná zapísanie zadania a odpovedí do vytváraného XML dokumentu (zadanie.xml) v novom, náhodnom poradí.

Výsledný generovaný XML súbor je potom naslednový:

```
<?xml version="1.0" encoding="windows-1250"?>
<!DOCTYPE priklady SYSTEM "Skuska.dtd">
<priklady>
   <priklad name="TRO">
     <zadanie hodnotenie="3">Uzavretý regulačný obvod tvorí  riadený
proces s prenosom <span CLASS="math">G_p(s) = { 3 \over { s^2 +5 s
+7 }}</span> a regulátor s prenosom <span CLASS="math">G_R(s)= {9 +5
s}  </span>.  Ak sa v čase t=0 žiadaná veličina zmení z hodnoty 0 na
10 , je trvalá regulačná odchýlka rovná      </zadanie>
      <moznost ans="baditem">žiadna z ostatných odpovedí nie je
      správna</moznost>
       <moznost ans="baditem">0.29</moznost>
       <moznost ans="baditem">-1.15</moznost>
       <moznost ans="gooditem">2.06</moznost>
       <moznost ans="baditem">0.13</moznost>
   </priklad>
</priklady>
```

Na vytvorenie XML dokumentu je možné použiť aj štandard DOM (str. 22), ktorý pristupuje k jednotlivým elementom pomocou vypísania všetkých elementov od elementu vytváraného až po koreňový element (Parent), pričom každý element musí byť zapísaný príkazom a*ppendChild*. Tento štandard je možné používať aj v programe MATLAB, ale pre našu prácu je nevýhodný, lebo jeho zápis je zdĺhavý a neprehľadný, preto sa používa vytvorená funkcia **moznost**.

6.3.1 Funkcia moznost

Pri zapisovaní informácii do vytváraného XML dokumentu sa využíva vytvorená funkcia **moznost** (Príloha E), ktorá slúži pre vytváranie začiatočných a konečných elementov a vkladanie zadania a jednotlivých odpovedí na tieto zadania, ktoré sú ukladané v náhodnom poradí. Táto funkcia má päť vstupných argumentov:

choic – premenná so zadaním a odpoveďami a názov príkladu,
fw – premenná s názvom súboru, do ktorého sa zapisovaného XML dokumentu,
obr – znamená, že je v odpovediach použitý text, alebo text a zároveň aj matematický výraz (0), matematické výrazy (1) alebo obrázok (2),
priklad – nadobúda hodnotu 1, ak je to hlavný príklad, inak 0, teda ak je to podpríklad,
podpriklad – nadobúda hodnotu 1, ak následovné zadanie je zadanie podpríkladu a 0, ak nasleduje zadanie príkladu.

Ako bolo spomenuté funkcia vykonáva taktiež vkladanie začiatočných a koncových elementov a to *<priklad name="">, </priklad>, <podpriklad>, </podpriklad>* .

Tento spôsob je najvýhodnejší z hľadiska prehľadnosti jednotlivých súborov slúžiacich na generovanie matematických hodnôt a zároveň prvkov slúžiacich na vytváranie XML dokumentu. Ako už bolo spomenuté, zadania a odpovede sa vkladajú pomocou vytvorenej funkcie *moznost*, ktorá je platná pre všetky príklady použiteľné pre vytváranie XML dokumentu (databázy). Takýmto spôsobom sme získali jeden súbor pre zapísanie príkladu do XML dokumentu a jednotlivé súbory s výpočtom matematiky a textom pre zadanie a odpovede.

Funkcia možnosť bola vytvorená aj pre pristupovanie k XML dokumentu pomocou štandardu DOM. Jej obsah sa však omnoho zväčšil a navyše sa museli riešiť problémy, ktoré sa pri zapisovaní do súboru ľahko odstránili, ako je otázka počtu matematických výrazov v zadaní, pre ktoré by musel byť vytváraný element *span.* Ďalším problémom bola aj štruktúra vytváraného XML dokumentu. Elementy boli síce úhľadne zarovnané a každý element bol na samostatnom riadku, ale práve to činilo problém pri transformovaní XML dokumentu pomocou procesora XSLT. Každý element na novom riadku či začiatočný, alebo konečný, znamená voľný riadok vo výslednom formáte. Formáty WebCT a T_EX by sa stávali kvôli prázdnym riadkom nevalidné. I keď aj takýto problém má určite svoje riešenie, pre nedostatok času nebol tento problém vyriešený.

6.4 Transformácia na prezentačné formuláre

V našej práci boli vytvorené tri, respektíve štyri XSLT dokumenty, ktorými sa ďalej uskutočňovala transformácia XML dokumentu na výsledný formát:

- jsMath.xsl
- mimetex.xsl
- WebCT.xsl
- Tex.xsl

6.4.1 jsMath.xsl a mimetex.xsl

Tieto dva dokumenty vytvorené v jazyku XSLT sa používajú na transformáciu generovaného dokumentu *zadanie.xml* na výsledný formát HTML. Dokumenty *JsMath.xsl* (Príloha A) a *mimetex.xsl* (Príloha A) obsahujú zápis založený na jazyku XSLT a zároveň využívajúceho

prvky štandardu HTML. Používajú sa elementy ako *<html>, <title>, <body>, <table>, <input>, , <img>* a iné. Rozdiel medzi XSLT dokumentmi je taký, že *jsMath.xsl* vkladá aplikáciu jsMath na zobrazovanie matematických výrazov pre publikovanie v elektronických dokumentoch a aplikáciu môžeme deklarovať v elemente *script*

```
<script language="JavaScript" src="jsMath/jsMath.js">
```

a elemente *style,* ktorý obsahuje nastavenie veľkosti písma vytváraného pomocou jsMath. Matematické výrazy sa vkladajú do elementov *span* pomocou šablóny (*xsl:template*) pre element span. V tomto prípade iba prepíšeme element span tým istým elementom. Táto šablóna v dokumente XSLT je veľmi dôležitá, pretože v opačnom prípade by sa tento element do výsledného dokumentu nepreniesol, ale jeho obsah áno. Ak by sme celé zadanie skopírovali, získali by sme HTML dokument ktorý by nebol validný, lebo by obsahoval aj element zadanie.

```
<xsl:template match="span"><span CLASS="math">
<xsl:value-of select="."/></span>
</xsl:template>
```

.

Pre overovanie správnosti odpovede v HTML formáte využívame aplikáciu JavaScript, zapísanú na začiatku hneď za úvodným elementom <html>.

```
<script language="JavaScript" src="file.js"> </script>
```

Toto overovanie spočíva v jednom vytvorenom súbore file.js, ktorý obsahuje následovný zdrojový kód funkcie:

```
function test(otazka, spravna)
  {
    if (otazka==spravna)
       window.alert("Spravne");
    else
       window.alert("Nespravne");
  }
```

Pri každej odpovedi sa vytvára tlačítko onclick v elemente *input*, ktoré testuje, či je odpoveď správna alebo nesprávna, v našom prípade porovnaním výrazu gooditem, ak je v teste pre onclik ako druhý element baditem (alebo akýkoľvek text odlišný od gooditem), vypíše sa “Nesprávne“, ak je však text rovnaký vypíše sa “Správne“.

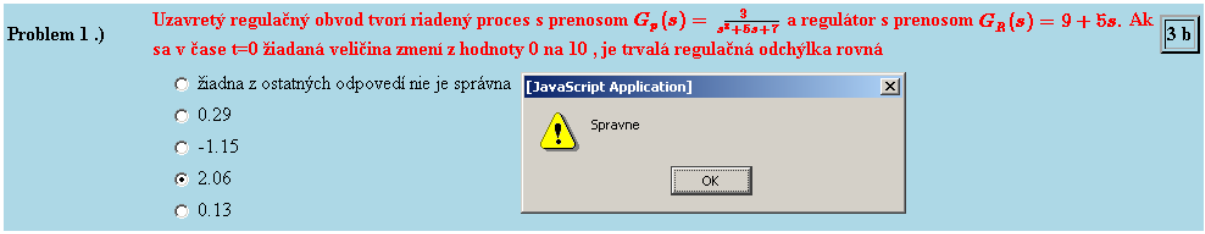
```
<input type="radio" name="tlacitko[1]" value="1"
onclick="test("gooditem", "gooditem")">
```

Dokument mimetex.xsl používa na zobrazovanie matematických výrazov aplikáciu mimeT$_E$X. Pri tejto aplikácii nie je potrebné volať vykonávací súbor na začiatku HTML dokumentu, ale priamo sa vkladá jej URL adresa ako atribút do elementu *img*. Hneď za ňou v tom istom atribúte ja zapísaný matematický výraz, ktorý sa má zobraziť. Pre vytvorenie elementu *img* sa používa následovná šablóna (*xsl:template*)

```
<xsl:template match="span"><IMG><xsl:attribute name="src">/cgi-
bin/mimetex.cgi? <xsl:value-of select="."/></xsl:attribute></IMG>
</xsl:template>
```

Ostatné funkcie sú rovnaké ako u dokumentu jsMath.xsl. Transformáciou XML dokumentu zadanie.xml pomocou XSLT dokumentu získame HTML formát ako na Obr. 3.11, pričom časť HTML kódu zodpovedajúcich obrázku je:

```
.
.
.
<tr>
   <td width="115"><B>Problem 1 .)</B></td>
   <td><B><font color="red">Uzavret&yacute; regula&#269;n&yacute; obvod
tvor&iacute; riaden&yacute; proces s prenosom <span CLASS="math">G_p(s) = { 3 \over
{ s^2 +10 s +7 }}</span> a regul&aacute;tor s prenosom <span CLASS="math">G_R(s)=
{4 +2 s}  +9/s </span>.  Ak sa v &#269;ase t=0 &#382;iadan&aacute; veli&#269;ina
zmen&iacute; z hodnoty 0 na  1 , je trval&aacute; regula&#269;n&aacute;
odch&yacute;lka rovn&aacute; </font></B></td>
   <td width="40">
   <P align="right">
      <table border="2">
          <tr>
              <td><B>3 b</B></td>
          </tr>
      </table>
   </P>
   </td>
</tr>
<tr>
   <td>
      <table>
         <tr>
             <td><input type="radio" name="tlacitko[1]" value="1"
onclick="test(&#34;gooditem&#34;,&#34;baditem&#34;)"></td>
             <td>-1.44</td>
         </tr>
         <tr>
             <td><input type="radio" name="tlacitko[1]" value="2"
onclick="test(&#34;gooditem&#34;,&#34;baditem&#34;)"></td>
             <td>-1.59</td>
         </tr>
         <tr>
           .
           .
```



Obr. 3.11 Výstupný formát HTML

6.4.2 WebCT.xsl

WebCT.xsl (Príloha C) je dokumentom jazyka XSLT a používa sa na transformáciu XML dokumentu zadanie.xml na výstupný formát WebCT, ktorý sa používa ako importný súbor do vzdelávacieho programu Moodle. Dokument vytvára výsledný formát nielen použitím informácii z XML dokumentu, ale vkladá aj prvky formátu WebCT potrebné na získanie validného výsledného dokumentu. Každý príklad začína označením *:TYPE:MC:*, ktorý hovorí aký typ otázky chceme vytvoriť: v našom prípade MC (Multiple Choice). Nasleduje označenie *:TITLE*:, teda názov príkladu (TRO – Trvalá regulačná odchýlka). Ak chceme použiť v príkladoch spätnú väzbu, musíme použiť ako ďalšie označenie *:FEEDBACK* . Nasleduje už iba text otázky označený príkazom *:QUESTION*: a ako posledné používame označenie *:ANSWER1:* , ktorý označuje prvú možnosť odpovede. Za každou odpoveďou môžeme nechať odkaz na spätnú väzbu pomocou príkazu *:REASON*:, ktorý umožňuje tvorcom príkladov nechať odkaz užívateľovi po označení odpovede, ktorá je podľa jeho výpočov správna.

V dokumente XSLT sa overuje hodnota správnej odpovede pomocou funkcie *xsl:if* a priraďuje sa hodnota v percentách nula (nesprávna odpoveď), alebo sto (správna odpoveď).

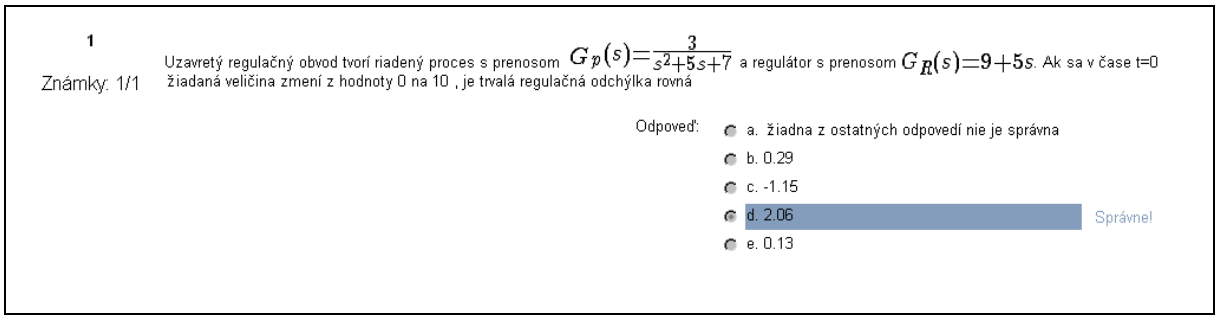
```
<xsl:if test="./@ans = 'gooditem'"><xsl:text>:ANSWER</xsl:text>
<xsl:number  format="1"/>:100:H <xsl:apply-templates/></xsl:if>
<xsl:if test="./@ans = 'baditem'"><xsl:text>:ANSWER</xsl:text>
<xsl:number  format="1"/>:0:H<xsl:apply-templates/></xsl:if>
```

Matematické výrazy písané vo formáte $\LaTeX$ sú vkladané do označenia dvoch dolárov *$$* pomocou šablóny (*xsl:template*).

```
<xsl:template match="span">
<xsl:text>$$</xsl:text><xsl:value-of select="."/><xsl:text>$$</xsl:text>
</xsl:template>
```

Takto označený matematický výraz je v programe Moodle automaticky zobrazovaný pomocou aplikácie mimeTEX. Otázka v teste je znázornená na Obr. 3.12. Výsledný WebCT formát pre problém TRO je potom:

```
:TYPE:MC:1:0:C
:TITLE:TRO
:FEEDBACK
:QUESTION:H
Uzavretý regulačný obvod tvorí riadený proces s prenosom $$G_p(s) = { 3 \over { s^2
+5 s +7 }}$$ a regulátor s prenosom $$G_R(s)= {9 +5 s}  $$.  Ak sa v čase t=0
žiadaná veličina zmení z hodnoty 0 na  10 , je trvalá regulačná odchýlka rovná
:IMAGE:
:LAYOUT:vertical
:ANSWER1:0:H
žiadna z ostatných odpovedí nie je správna
:REASON1:H
Sorry!
:ANSWER2:0:H
2.06
:REASON4:H
Správne!
.
.
.
:CAT:Default
```



Obr. 3.12 Importovaný test do programu Moodle pomocou formátu WebCT

Nevýhodou stále ostáva nemožnosť importovať bodovú hodnotu príkladu. Hodnotu potom musíme nastaviť ručne.

6.4.3 Tex.xsl

Úlohou tohto XSLT dokumentu je transformácia dokumentu **zadanie.xml** na TEXovský zápis. Obsahuje teda okrem funkcii XSLT aj príkazy jazyka TEX ktorými je potrebné značkovať výsledný dokument. Príkazy sa líšia od okolitého textu spätným lomítkom na začiatku a hodnotou vloženou do zložených zátvoriek. Pre spracovanie používame balík exams [1], ktorý bol upravený pre naše podmienky (slovenské texty, iba jeden vstupný súbor, ...). Z takéhoto formátu môžeme ľahko získať formáty:

DVI príkazom **cslatexw tex.tex,**

PS príkazom **dvips tex.dvi –o zadanie.ps**,

PDF príkazom **ps2pdf zadanie.ps**.

Nás však najviac zaujíma formát PDF, ktorý nám ďaľej bude slúžiť pre účel tlačenia testov na skúšky z LCZA a to zadanie (obr. 3.13) a riešenie (obr. 3.14).

Matematické výrazy sa vkladajú podobne ako pri dokumente WebCT.xsl medzi znaky dolára, ale tentokrát iba do jedného. Takáto zmena sa vykonáva znovu vytvorením predlohy (template) podľa ktorej sa každý obsah elementu *span* v dokumente **zadanie.xml** vloží medzi znaky dolára.

```
<xsl:template match="span">
<xsl:text>$</xsl:text><xsl:value-of select="."/><xsl:text>$</xsl:text>
</xsl:template>
```

Správne a nesprávne odpovede sú priamo značkované hodnotami \gooditem, alebo \baditem a text odpovede sa vkladá do zložených zátvoriek. Pre získanie takéhoto značkovania bola tiež vytvorená predloha, ktorá každý obsahu elementu možnosť výtvorí takéto značkovanie.

```
<xsl:template match="moznost">
<xsl:text>\</xsl:text><xsl:value-of select="./@ans"/><xsl:text>{</xsl:text>
<xsl:apply-templates/><xsl:text>}</xsl:text>
</xsl:template>
```

Bodová hodnota zadania má označenie \score. Celý test je vložený do označenia \begin{document} na začiatku a \end{document} na konci. Každý príklad je má začiatočné označenie \begin{problem} a konečné označenie \end{problem}. Rovnako sú všetky možnosti vkladané do označenia \begin{choice} na začiatku možností a \end{choice}na ich konci.

Výsledný L^AT_EX kód pre problém TRO je potom v tvare:

```
\question{}{}{
\begin{problem}[\split]
\score{3}
Uzavretý regulačný obvod tvorí riadený proces s prenosom $G_p(s) = { 3 \over { s^2
+5 s +7 }}$ a regulátor s prenosom $G_R(s)= {9 +5 s}  $.  Ak sa v čase t=0 žiadaná
veličina zmení z hodnoty 0 na  10 , je trvalá regulačná odchýlka rovná
\begin{choice}
\baditem{žiadna z ostatných odpovedí nie je správna}
```

```
\baditem{0.29}
\baditem{-1.15}
\gooditem{2.06}
\baditem{0.13}
\end{choice}
\end{problem}
}
```

Spracovaní do PDF formátu potom získame výsledky znázornené na Obr. 3.13 (Zadanie) a 3.14 (Riešenie).

Problém 1. Uzavretý regulačný obvod tvorí riadený proces s prenosom $G_p(s) = \frac{3}{s^2+5s+7}$ a regulátor s prenosom $G_R(s) = 9 + 5s$. Ak sa v čase t=0 žiadaná veličina zmení z hodnoty 0 na 10 , je trvalá regulačná odchýlka rovná

3

- ◯ 2.06
- ◯ -1.15
- ◯ 0.13
- ◯ 0.29
- ◯ žiadna z ostatných odpovedí nie je správna

Obr. 3.13 Výstupný formát písaný v LaTeXu a transformovaný do PDF (zadanie)

Problém 1. Uzavretý regulačný obvod tvorí riadený proces s prenosom $G_p(s) = \frac{3}{s^2+5s+7}$ a regulátor s prenosom $G_R(s) = 9 + 5s$. Ak sa v čase t=0 žiadaná veličina zmení z hodnoty 0 na 10 , je trvalá regulačná odchýlka rovná

3

- √ 2.06
- ◯ -1.15
- ◯ 0.13
- ◯ 0.29
- ◯ žiadna z ostatných odpovedí nie je správna

Obr. 3.14 Výstupný formát písaný v LaTeXu a transformovaný do PDF (riešenie)

Záver

XML v skutočnosti nadväzuje na princípy a konvencie jazykov, XHTML a SGML a vytvára jednoduchý a pritom veľmi účinný mechanizmus, o čom sme sa presvedčili pri jeho vytváraní v programe MATLAB. Pre vytvorenie dokumentu XML (zadanie.xml) bol použitý program MATLAB a jeho funkcia *fprintf*, i keď program umožňuje vytváranie XML dokumentu pomocou štandardu DOM. Jeho použitie je zdĺhavé a neprehľadné. V programe MATLAB tiež prebieha transformácia generovaného XML dokumentu pomocou procesoru XSLT do formátu HTML, WebCT či L^AT_EXovského zápisu, ktorý je možné ďalej spracovať na PostScript či PDF formát.

HTML formát slúži na publikovanie elektronických dokumentov na internete. Problém vkladania matematických výrazov do tohto dokumentu však nevyriešila nadstavba MathML pre štandard XML. I keď je obšírne rozpracovaná, nie je stále dostatočne podporovaná v prehliadačoch, najmä v celosvetovo najviac používaných. Ako náhrada bola použitá teda aplikácia JavaScriptu jsMath, ktorá je síce pri zobrazovaní matematických výrazov pomalšia ako aplikácia mimeT_EX, ale je výhodnejšia z hľadiska možností, ktorými môžeme vkladaný text upraviť. JsMath tiež umožňuje zobraziť matematické výrazy i po uložení na klientský disk ako formát HTML.

Formát WebCT sa využíva pre import otázok do e-learningového programu Moodle, kde sa jednotlivé príklady môžu vkladať do testov. Matematické výrazy sa vkladajú medzi dva znaky dolára a Moodle ich potom automaticky zobrazuje pomocou aplikácie mimeT_EX.

Posledným vytváraným formátom je formát L^AT_EXu, ktorý je často používaný pre vytváranie odborných textov. Jednoduchými príkazmi získame formát PDF. Tento formát je najvhodnejší pre tlačenie vytvorených testov, či už ako zadanie bez označenia alebo s označením správnej odpovede.

Literatúra

[1] Donald E. Knuth., The TEXbook, Volume A of Computers and Typesetting, Addison-Wesley, Reading, Massachusetts, druhé vydanie, 1984.

[2] Steven Holzner, XSLT príručka internetového vývojáře, Computer Press, Praha 2002.

[3] Neil Bradley, XML kompletní průvodce, Grada Publishing, Praha 2000.

[4] Kosek, J.:Články, dokumenty dostupné na URL http://www.kosek.cz/clanky/index.html (online, 2005).

[5] Dařílek, M :Použitie jazyka XML pro psaní hypertextových dokumentú, semestrálni projekt, FEI VUT Brno, 2000.

[6] MATLAB: The Language of Technical Computing, manual, (June) 2004.

[7] David Carlisle: Manuál pre XMLTEX dostupný na URL http://www.dcarlisle.demon.co.uk/xmltex/manual.html (online, 05. 03. 2005).

[8] John Forkosh: Použitie mimeTeXu pre písanie matematických výrazov http://www.forksoh.com/mimetex.html (online, 15. 03. 2005)

[9] Použitie jsMath pre písanie matematických výrazov: http://www.math.union.edu/~dpvc/jsMath/welcome.html (online, 18. 03. 2005)

[10] FOP (procesor XSL-FO) súčasť projektu Apache http://xml.apache.org/fop (online, 25. 04. 2005)

[11] David Carlisle: Passive TEX (procesor XSL-FO) http://www.tug.org/pipermail/pdftex/2003-March/003724.html (online, 05. 03. 2005)

[12] TeXML, transformácia XML do LATEXu použitím XSLT http://getfo.sourceforge.net/index.html (online, 03. 04. 2005)

[13] Michael Kay: Saxon, procesor XSLT http://users.iclway.co.uk/mhkay/saxon/ (online, 04. 02. 2005)

[14] Xalan (procesor XSLT) súčasť projektu Apache http://xml.apache.org/xalan-j/index.html (online, 03. 02. 2005)

[15] James Clark: systém XT (procesor XSLT) http://jclark.com/xml/xt.html (online, 02. 02. 2005)

[16] LATEX:A document preparation system http://www.latex-project.org (online, 02. 01. 2005)

[17] MathPlayer: plug-in pre prehliadač Microsoft Internet Explorer 5.5 a vyššie http://www.mathtype.com/ (online, 22. 03. 2005)

[18] EzMath: plug-in pre prehliadače MS IE a Mozilla
http://www.w3.org/People/Raggett/Ezmath/ (online, 22. 03. 2005)
[19] Matematica: MathML editor
http://www.wolfram.com/ (online, 24. 03. 2005)
[20] Math Type 5: MathML editor
http://www.mathtype.com/ (online, 24. 03. 2005)
[21] WebEQ: MathML editor
http://www.mathtype.com/ (online, 24. 03. 2005)
[22] TechExplorer Hyper Media Browser: plug-in pre MS IE
http://computing.unr.edu/FAGs/Labs/ITLaPC.html (online, 24. 03. 2005)
[23] Smarty: baliček pre PHP
http://smarty.php.net/ (online, 12. 03. 2005)

Prílohy

Príloha A – súbor jsMath.xsl

Dokument jsMath.xsl a mimetex.xsl pre transformáciu XML dokumentu na formát HTML

```
<?xml version="1.0" encoding="windows-1250"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output indent="yes" encoding="windows-1250" method="html"/>
<xsl:template match="/priklady">
<html>
      <body>
        <script language="JavaScript" src="file.js"> </script>
        <script language="JavaScript" src="jsMath/jsMath.js"></script>
        <style TYPE="text/css">
            .math    {visibility: hidden}
            .typeset {font-size: 120%}
            .normal  {font-size: 120%}
        </style>
      <table bgcolor="lightblue" width="1100">
      <xsl:for-each select="priklad">
      <xsl:variable name="cislo"> <xsl:number value="position()"
format="1"/></xsl:variable>
      <xsl:for-each select="zadanie">
        <tr><td width="115"><B>Problem <xsl:value-of select="$cislo"/> .)</B></td>
            <td><B><font color="red"><xsl:apply-templates/> </font></B></td>
              <td width="40"><P align="right">
              <table border="2">
                    <tr><td><B><xsl:value-of select="./@hodnotenie"/>
       </B></td></tr>
              </table></P></td></tr>
       <xsl:for-each select="img">
          <tr><td></td><td><xsl:copy-of select="."/></td><td></td><td></td></tr>
       </xsl:for-each></xsl:for-each>
        <tr><td></td>
            <td>
              <table>
        <xsl:for-each select="moznost">
        <tr><td></td>
             <td><input><xsl:attribute name="type"><xsl:text>radio</xsl:text>
</xsl:attribute>
                       <xsl:attribute name="name"><xsl:text>tlacitko</xsl:text>
[<xsl:value-of select="$cislo"/>]</xsl:attribute>
                       <xsl:attribute name="value"><xsl:number value="position()"
format="1"/></xsl:attribute>
                        <xsl:attribute name="onclick">test("gooditem","<xsl:value-
of select="./@ans"/>")</xsl:attribute></input></td>
                <td><xsl:copy-of select="."/><xsl:text></xsl:text></td>
                      <td></td></tr></xsl:for-each></table></td></tr>
        <xsl:for-each select="podpriklad">
         <xsl:variable name="podcislo"> <xsl:number value="position()" format="1"/>
</xsl:variable>
         <xsl:for-each select="zadanie">
              <tr><td></td><td> -
               <B><font color="red"><xsl:apply-templates/></font></B></td>
               <td width="40"><P align="right">
               <table border="2"><tr><td><B><xsl:value-of
select="./@hodnotenie"/>b</B></td>
               </tr></table></P></td></tr>
          <xsl:for-each select="img">
                          <tr><td></td><td><xsl:copy-of
select="."/></td><td></td><td></td></tr>
          </xsl:for-each></xsl:for-each>
```

```
          <xsl:for-each select="moznost">
                              <tr><td></td><td>
               <table>
              <tr><td></td><td><input>
                      <xsl:attribute
name="type"><xsl:text>radio</xsl:text></xsl:attribute>
                  <xsl:attribute
name="name"><xsl:text>tlacitko</xsl:text>[<xsl:value-of
select="$cislo"/>,<xsl:value-of select="$podcislo"/>]</xsl:attribute>
                      <xsl:attribute name="value"><xsl:number value="position()"
format="1"/>
</xsl:attribute>
                  <xsl:attribute name="onclick">test("gooditem","<xsl:value-of
select="./@ans"/>")</xsl:attribute></input></td>
                          <td><xsl:copy-of select="."/><xsl:text></xsl:text></td>
                     <td></td></tr>
                </table></td><td></td></tr>
          </xsl:for-each>
             </xsl:for-each>
                </xsl:for-each>
                </table><SCRIPT>jsMath.Process()</SCRIPT>
         </body>
 </html>

 </xsl:template>

 <xsl:template match="span"><span CLASS="math"><xsl:value-of select="."/></span>
</xsl:template>

 <xsl:template match="br"><br/>
</xsl:template>

 </xsl:stylesheet>
```

Ak chceme vytvoriť HTML stranu s použitím mimeT$_E$Xu, prepíšeme element *xsl:template* a vymažeme obsah elementu *script*, čo sa týka jsMath a obsah elementu *style*:

```
<xsl:template match="span"><IMG><xsl:attribute name="src">/cgi-bin/mimetex.cgi?
<xsl:value-of select="."/></xsl:attribute></IMG>
</xsl:template>
```

Príloha B – súbor tex.tex

Dokument tex.xsl pre transformáciu XML dokumentu do značkovacieho jazyka $LaTeX$

```
<?xml version="1.0" encoding="windows-1250"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output indent="no" encoding="windows-1250" method="text"/>
<xsl:strip-space elements="moznost"/>
<xsl:strip-space elements="zadanie"/>
<xsl:template match="/priklady">

\documentclass[answers,nosep,scores]{exams1}
\usepackage[top=2cm, bottom=0.5cm,left=1cm,right=2.5cm]{geometry}
\usepackage{slovak}
\usepackage{graphicx}
\usepackage{fancyhdr}
\usepackage{units}

\def\ve#1{\mathchoice{\mbox{\boldmath$\displaystyle#1$}}
{\mbox{\boldmath$\textstyle#1$}}
{\mbox{\boldmath$\scriptstyle#1$}}
{\mbox{\boldmath$\scriptscriptstyle#1$}}}
\renewcommand{\frac}[2]{{\begingroup\displaystyle#1\endgroup\over\displaystyle#2}}

\title{Základy automatizácie}
\author{M. Bakošová  a M. Fikar}
\target{FCPT STU}
\begin{document}
\newcommand{\datum}{22.5.2002}
\pagestyle{plain}
\renewcommand{\footrulewidth}{0.4pt}
\renewcommand{\headrulewidth}{0.4pt}
\lhead{Meno, ŠS:}
\rhead{}
\lfoot{KIRP FCHPT STU - ZA, \datum }
\cfoot{\thepage}
\begin{exam}[14785934]{\datum}
\rfoot{Test A}
\thispagestyle{fancyplain}

<xsl:for-each select="priklad">
\question{}{}{
\begin{problem}[\split]
<xsl:text>\score{</xsl:text><xsl:value-of select="zadanie/@hodnotenie"/>}
<xsl:if test="zadanie/img">\begin{minipage}{0.39\textwidth}
\includegraphics[width=\textwidth]{<xsl:value-of select="zadanie/img/@src"/>}
\end{minipage}
</xsl:if>
<xsl:apply-templates/>\end{choice}
\end{problem}
}
</xsl:for-each>
\end{exam}
\end{document}
</xsl:template>

<xsl:template match="podpriklad">\end{choice}
<xsl:text>\score{</xsl:text><xsl:value-of select="zadanie/@hodnotenie"/>}
<xsl:apply-templates/>
</xsl:template>

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

```
\begin{choice}</xsl:template>

<xsl:template match="moznost">
<xsl:text>\</xsl:text><xsl:value-of
select="./@ans"/><xsl:text>{</xsl:text><xsl:apply-templates/><xsl:text>}</xsl:text>
</xsl:template>

<xsl:template match="span"><xsl:text>$</xsl:text><xsl:value-of
select="."/><xsl:text>$</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

Príloha C – súbor WebCT.xsl

Dokument WebCT.xsl pre transformáciu XML dokumentu do formátu WebCT

```
<?xml version="1.0" encoding="windows-1250"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output indent="no" encoding="windows-1250" method="text"/>

<xsl:template match="/priklady">
<xsl:for-each select="priklad">
<xsl:variable name="cislo"> <xsl:number value="position()"
format="1"/></xsl:variable>
# Start of question: Multiple Choice Question
:TYPE:MC:1:0:C
:TITLE:<xsl:value-of select="$cislo"/>
:FEEDBACK
:QUESTION:H
<xsl:apply-templates/>
:CAT:Default
# End of question: Multiple Choice Question
</xsl:for-each>
</xsl:template>

<xsl:template match="podpriklad">
<xsl:variable name="podcislo"> <xsl:number value="position()" format="A"/>
</xsl:variable>
:TYPE:MC:1:0:C
:TITLE: .<xsl:value-of select="$podcislo"/>
:FEEDBACK
:QUESTION:H
<xsl:apply-templates/>
</xsl:template>

<xsl:template match="zadanie">
<xsl:apply-templates/>
:IMAGE:
:LAYOUT:vertical<xsl:if test="img">
<img><xsl:attribute name=”src“>http://www.kirp.chtf.stuba.sk/~valo/dip/<xsl:value-
of select="img/@src"/></xsl:attribute><xsl:attribute
name=“height“>100</xsl:attribute> <xsl:attribute
name=“align“>left"</xsl:attribute><xsl:attribute name=“width“>
100</xsl:attribute></xsl:if>
</xsl:template>

<xsl:template match="moznost"><xsl:if test="./@ans = 'gooditem'">
<xsl:text>:ANSWER</xsl:text><xsl:number  format="1"/>:100:H
<xsl:apply-templates/></xsl:if>
<xsl:if test="./@ans = 'baditem'"><xsl:text>:ANSWER</xsl:text><xsl:number
format="1"/>
:0:H
<xsl:apply-templates/></xsl:if>
:REASON
<xsl:number  format="1"/>:H
<xsl:if test="./@ans = 'gooditem'">Correct Answer!</xsl:if>
<xsl:if test="./@ans = 'baditem'">Sorry!</xsl:if></xsl:template>

<xsl:template match="span">
<xsl:text>$$</xsl:text><xsl:value-of select="."/><xsl:text>$$</xsl:text>
</xsl:template>

</xsl:stylesheet>
```

Priloha D – súbor zadanie.xml

Príklad vygenerovaného XML súboru

```
<?xml version="1.0" encoding="windows-1250"?>
<!DOCTYPE priklady SYSTEM "Skuska.dtd">
<priklady>
<priklad><zadanie hodnotenie="3">Uzavretý regulačný obvod tvorí riadený proces s
prenosom <span CLASS="math">G_R(s) = { 1 \over { s^2 +3 s +2 }}</span> a regulátor
s prenosom <span CLASS="math">G_p(s)= {2 +3 s}  +7/s </span>.  Ak sa v čase t=0
žiadaná veličina zmení z hodnoty 0 na  9 , je trvalá regulačná odchýlka rovná
</zadanie>
      <moznost ans="baditem">-0.04</moznost>
      <moznost ans="baditem">žiadna z ostatných odpovedí nie je správna</moznost>
      <moznost ans="baditem">0.17</moznost>
      <moznost ans="baditem">0.33</moznost>
      <moznost ans="gooditem">0.00</moznost>
</priklad><priklad><zadanie hodnotenie="2">Pri získavaní prechodovej
charakteristiky sa vykonal skok vstupnej veličiny z hodnoty 0.73 na hodnotu 0.31
.Výstupná veličina bola v počiatku v ustálenom stave 0.84 a po skončení
prechodových javov sa opäť ustálila na hodnote 0.57. Na nameranej prechodovej
charakteristike sa odčítal čas prieťahu 0.20 min a čas nábehu 1.40 min. Rád
náhradného prenosu <span CLASS="math">n</span> a časová konštanta <span
CLASS="math">T</span> sú</zadanie>
      <moznost ans="baditem">žiadna z ostatných odpovedí nie je správna</moznost>
      <moznost ans="baditem"><span CLASS="math">n=4, T=7.61 min</span></moznost>
      <moznost ans="gooditem"><span CLASS="math">n=2, T=0.52 min</span></moznost>
      <moznost ans="baditem"><span CLASS="math">n=6, T=6.41 min</span></moznost>
      <moznost ans="baditem"><span CLASS="math">n=2, T=5.30 min</span></moznost>
<podpriklad><zadanie hodnotenie="2">Navrhnite k identifikovanému prenosu pomocou
Strejcovej metódy najjednoduchší spätnoväzbový regulátor, ktorý odstráni TRO
(počítajte s presnosťou na 4 desatinné miesta)</zadanie>
      <moznost ans="gooditem"><span CLASS="math">1.556+2.264/s</span></moznost>
      <moznost ans="baditem"><span CLASS="math">1.867+2.491/s</span></moznost>
      <moznost ans="baditem">žiadna z ostatných odpovedí nie je správna</moznost>
      <moznost ans="baditem"><span CLASS="math">1.556</span></moznost>
      <moznost ans="baditem"><span CLASS="math">1.711+2.944/s+2.038
s</span></moznost>
</podpriklad></priklad><priklad><zadanie hodnotenie="2">Do nádrže s konštantným
objemom <span CLASS="math">V</span> priteká dvomi prúdmi voda. Prvý prúd s
objemovým prietokom <span CLASS="math">q_1</span> a teplotou <span
CLASS="math">\vartheta_1</span>, druhý prúd s objemovým prietokom <span
CLASS="math">q_2</span> a teplotou <span CLASS="math">\vartheta_2</span>. Parametre
sú <span CLASS="math">\rho=1000 kg {m}^{-3}, c_p=4.2 J.{kg}^{-1}.{K}^{-1}, V = 4.20
m^3, q_1 = 3.10 {m}^{3}.{h}^{-1},<br/> q_2=8.80 {m}^{3}.{h}^{-1}, \vartheta_1 =
280.80 K, \vartheta_2=368.40 K</span>. <br/> Matematický model nádrže sa dá opísať
rovnicou <span CLASS="math">\frac{d\vartheta}{dt}  = a_1 \vartheta_1 + a_2
\vartheta_2 + a_3 \vartheta </span>kde</zadanie>
      <moznost ans="baditem"><span CLASS="math">a_1 = q_1/V, a_2 = q_2/V, a_3 =
(q_1+q_2)/V</span></moznost>
      <moznost ans="gooditem"><span CLASS="math">a_1 = q_1/V, a_2 = q_2/V, a_3 = -
(q_1+q_2)/V</span></moznost>
      <moznost ans="baditem"><span CLASS="math">a_1 = -q_1/V, a_2 = -q_2/V, a_3 = -
(q_1+q_2)/V</span></moznost>
      <moznost ans="baditem">žiadna z ostatných odpovedí nie je správna</moznost>
      <moznost ans="baditem"><span CLASS="math">a_1 = -q_1/V, a_2 = -q_2/V, a_3 =
(q_1+q_2)/V</span></moznost>
<podpriklad><zadanie hodnotenie="2">Ustálená teplota vody v nádrži <span
CLASS="math">\vartheta</span> je približne</zadanie>
      <moznost ans="gooditem">345.58 K</moznost>
      <moznost ans="baditem">338.17 K</moznost>
      <moznost ans="baditem">300.54 K</moznost>
      <moznost ans="baditem">309.84 K</moznost>
      <moznost ans="baditem">žiadna z ostatných odpovedí nie je správna</moznost>
</podpriklad></priklad></priklady>
```

Príloha E – funkcia moznost

Funkcia *moznost* pre všetky typy príkladov

function [answers] = moznost(choic,fw,obr,priklad,podpriklad)

```
s = {'žiadna z ostatných odpovedí nie je správna',choic{2}{1},choic{2}{2},
choic{2}{3},choic{2}{4}};

i = randperm(5);
for r=1:5
if i(r)==5
answers{1}{r}=sprintf('gooditem');
else
answers{1}{r}=sprintf('baditem');
end
if (i(r)==1 | obr==0)
answers{2}{r}=sprintf('%s',s{i(r)});
else
answers{2}{r}=sprintf('<span CLASS="math">%s</span>',s{i(r)});
end
if obr==2
answers{2}{r}=sprintf('<img src="%s"/>',s{i(r)});
end
end

if priklad == 1
    fprintf(fw,'<priklad name="%s">\n',choic{3});
else
    fprintf(fw,'<podpriklad>\n');
end

fprintf(fw,'%s\n',choic{1});
fprintf(fw,'<moznost ans="%s">%s</moznost>\n',answers{1}{1},answers{2}{1});
fprintf(fw,'<moznost ans="%s">%s</moznost>\n',answers{1}{2},answers{2}{2});
fprintf(fw,'<moznost ans="%s">%s</moznost>\n',answers{1}{3},answers{2}{3});
fprintf(fw,'<moznost ans="%s">%s</moznost>\n',answers{1}{4},answers{2}{4});
fprintf(fw,'<moznost ans="%s">%s</moznost>\n',answers{1}{5},answers{2}{5});

if podpriklad == 0
    if priklad == 0
        fprintf(fw,'</podpriklad></priklad>\n');
    else
        fprintf(fw,'</priklad>\n');
    end
else
    if priklad == 0
        fprintf(fw,'</podpriklad>');
    else
        fprintf(fw,'');
    end
end
```