Ştefan Trăuşan-Matu

Prelucrarea documentelor folosind XML

2001

1. Prelucrarea textelor

1.1 Ipostaze ale unui text. Limbaje de descriere asociate

Un text sau un hipertext pot fi considerate ca având mai multe ipostaze [Tra00a]:

1. **Textul brut**, succesiunea de semne (cuvinte şi imagini), independente de forma de reprezentare. De exemplu, un fragment dintr-un curs de structuri de date ar putea fi:

```
Arbori_AVL

Arborii AVL, denumiti asa dupa cei care i-au imaginat pentru prima oara, sunt
arbori binari de cautare ce indeplinesc, suplimentar, un anumit criteriu de
echilibru. Acest criteriu impune ca inaltimile fiilor oricarui nod din arbore
sa difere prin maxim 1. Mai formal, trebuie indeplinita relatia,

|h(Sl(n)) - h(Sr(n))| <= 1, pentru oricare nod, n, din arbore.

Vom numi in cele ce urmeaza un arbore AVL, arbore echilibrat. Este evident ca
un arbore cu o asemenea proprietate este foarte aproape de idealul de
echilibru, si va avea timpi de cautare foarte buni.

. . . . . .
```

2. **Textul adnotat**, conform unor anumite limbaje, de exemplu, SGML, HTML, XML sau LaTex. De exemplu, o posibilă adnotare în XML a textului brut de mai sus s-ar putea face după cum se vede din fragmentul următor:

```
<?xml version="1.0" ?>
<!DOCTYPE lectie SYSTEM "avl.dtd">

<lectie nume="Arbori_AVL">
<titlu> Arbori_AVL </titlu>
<subiect nume="AVL">
<concept>Arborii AVL</concept>, denumiti asa dupa cei care i-au imaginat
pentru prima oara, sunt arbori binari de cautare ce indeplinesc, suplimentar,
un anumit criteriu de echilibru. Acest criteriu impune ca inaltimile fiilor
oricarui nod din arbore sa difere prin maxim 1. Mai formal, trebuie
indeplinita relatia,
<proprietate>|h(Sl(n)) - h(Sr(n))| &lt;= 1, pentru oricare nod, n, din arbore.
</proprietate>Vom numi in cele ce urmeaza un arbore AVL, arbore echilibrat.
</subiect>
<subiect nume="propravl">
Este evident ca un arbore cu o asemenea proprietate este foarte aproape de
idealul de echilibru, si va avea timpi de cautare foarte buni.

. . . . . .

<lectie>
```

3. **Limbajul în care se face adnotarea**. Acesta se poate defini, ca pentru orice limbaj artificial, printr-o gramatică formală. În SGML şi în XML, specificarea unui anumit limbaj de adnotare se face printr-un aşa numit **DTD** (“Document Type Definition”) sau printr-o declarație de schemă. Pentru exemplul anterior, un DTD posibil este:

```
<!ELEMENT lectie (titlu,(subiect)*) >
<!ATTLIST lectie
    nume ID #IMPLIED>
```

```
<!ELEMENT subiect ( concept | proprietate | fig | demonstratie | prog )*>
<!ATTLIST subiect
     nume ID #REQUIRED>
<!ELEMENT fig EMPTY>
<!ATTLIST fig
     nr ID #REQUIRED
     caption CDATA #REQUIRED>
<!ELEMENT titlu (#PCDATA)>
<!ELEMENT concept (#PCDATA)>
<!ELEMENT proprietate (#PCDATA)>
<!ELEMENT demonstratie (#PCDATA | concept | proprietate | fig)*>
<!ELEMENT prog (#PCDATA)>
<!ATTLIST proprietate
     nr ID #REQUIRED
     nume CDATA #IMPLIED>
```

4. **Arborele** care reprezintă imbricarea fragmentelor de text. De exemplu, pentru textul anterior, Internet Explorer 5 afişează arborele corespunzător fragmentului documentului de mai sus în forma următoare:

```
<?xml version="1.0" ?>
  <!DOCTYPE lectie (View Source for full doctype...)>
  - <!-- DOCTYPE lectie SYSTEM "c:/users/stefan/xml/ca/avl.dtd" -->
- <lectie nume="Arbori_AVL">
     <titlu>Arbori_AVL</titlu>
   - <subiect nume="AVL">
       <concept>Arborii AVL</concept>
       , denumiti asa dupa cei care i-au imaginat pentru prima oara, sunt
arbori binari de
       cautare ce indeplinesc, suplimentar, un anumit criteriu de echilibru.
Acest
       criteriu impune ca inaltimile fiilor oricarui nod din arbore sa difere
prin maxim
       1. Mai formal, trebuie indeplinita relatia,
       <proprietate nr="1">|h(Sl(n)) - h(Sr(n))| <= 1, pentru oricare nod, n,
din
       arbore.</proprietate>
       Vom numi in cele ce urmeaza un arbore AVL, arbore echilibrat.
     </subiect>
   - <subiect nume="propravl">
```

Un document poate fi reprezentat ca arbore în mai multe feluri. În programele care prelucrează texte adnotate în XML, se consideră un model al unui anumit document (“Document Object Model”, [DOM]), care include şi interfeţe pentru dezvoltarea de programe (API). DOM foloseşte tot o reprezentare arborescentă (vezi .).

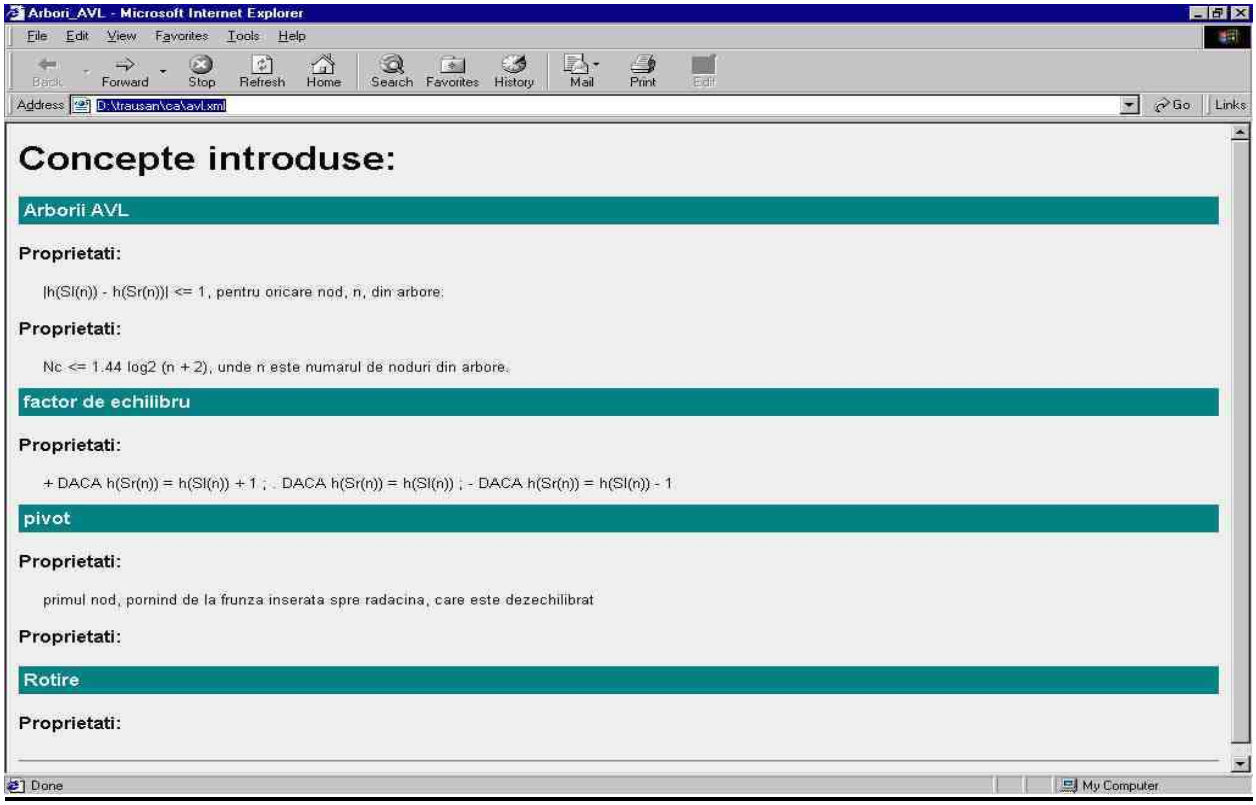
5. **Stilul** de afişare, implicit sau exprimat explicit, în CSS sau XSL [XSL]. De exemplu, un fişier XSL pentru documentul XML anterior ar putea fi:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <HTML>
      <HEAD>
        <TITLE><xsl:value-of select="lectie/titlu"/></TITLE>
      </HEAD>
      <BODY STYLE="font-family:Arial, helvetica, sans-serif; font-size:12pt;
            background-color:#EEEEEE">
       <H1> Concepte introduse: </H1>
        <xsl:for-each select="lectie/subiect">
          <DIV STYLE="background-color:teal; color:white;
                      padding:4px">
            <SPAN STYLE="font-weight:bold; color:white">
```

```
                <xsl:value-of select="concept"/></SPAN>
            </DIV>
            <H4>Proprietati:</H4>
            <DIV STYLE="margin-left:20px;
                        margin-bottom:1em; font-size:10pt">
              <xsl:value-of select="proprietate"/>
            </DIV>
          </xsl:for-each>
          <HR/>
        </BODY>
      </HTML>
    </xsl:template>
</xsl:stylesheet>
```

6. **Cunoştinţele** din text. O modalitate rudimentară de reprezentare a cunoştinţelor dintr-un text, adică o meta-descriere a unui text se poate face chiar şi în HTML folosind adnotările <META name="…" content="…">. Un limbaj derivat din XML pentru meta descrierea textelor pe web este RDF (vezi capitolul 10). Definirea de structuri RDF [RDF] este sprijinită de RDF-Schema [RDFS].

7. **Textul afişat** pe ecranul calculatorului, de exemplu, de un anumit browser, conform CSS sau XSL şi / sau modalităților implicite de afişare (de exemplu, în lipsa CSS).



8. **Textul tipărit pe hârtie**, de un anumit browser (uneori acesta este diferit de textul afişat pe ecran).

Concepte introduse:

Arborii AVL

Proprietati:

|h(Sl(n)) - h(Sr(n))| <= 1, pentru oricare nod, n, din arbore.

Proprietati:

Nc <= 1.44 log2 (n + 2), unde n este numarul de noduri din arbore. factor de echilibru

Proprietati:

+ DACA h(Sr(n)) = h(Sl(n)) + 1 ; . DACA h(Sr(n)) = h(Sl(n)) ; - DACA h(Sr(n)) = h(Sl(n)) - 1 pivot

Proprietati:

primul nod, pornind de la frunza inserata spre radacina, care este dezechilibrat

Proprietati:

Rotire

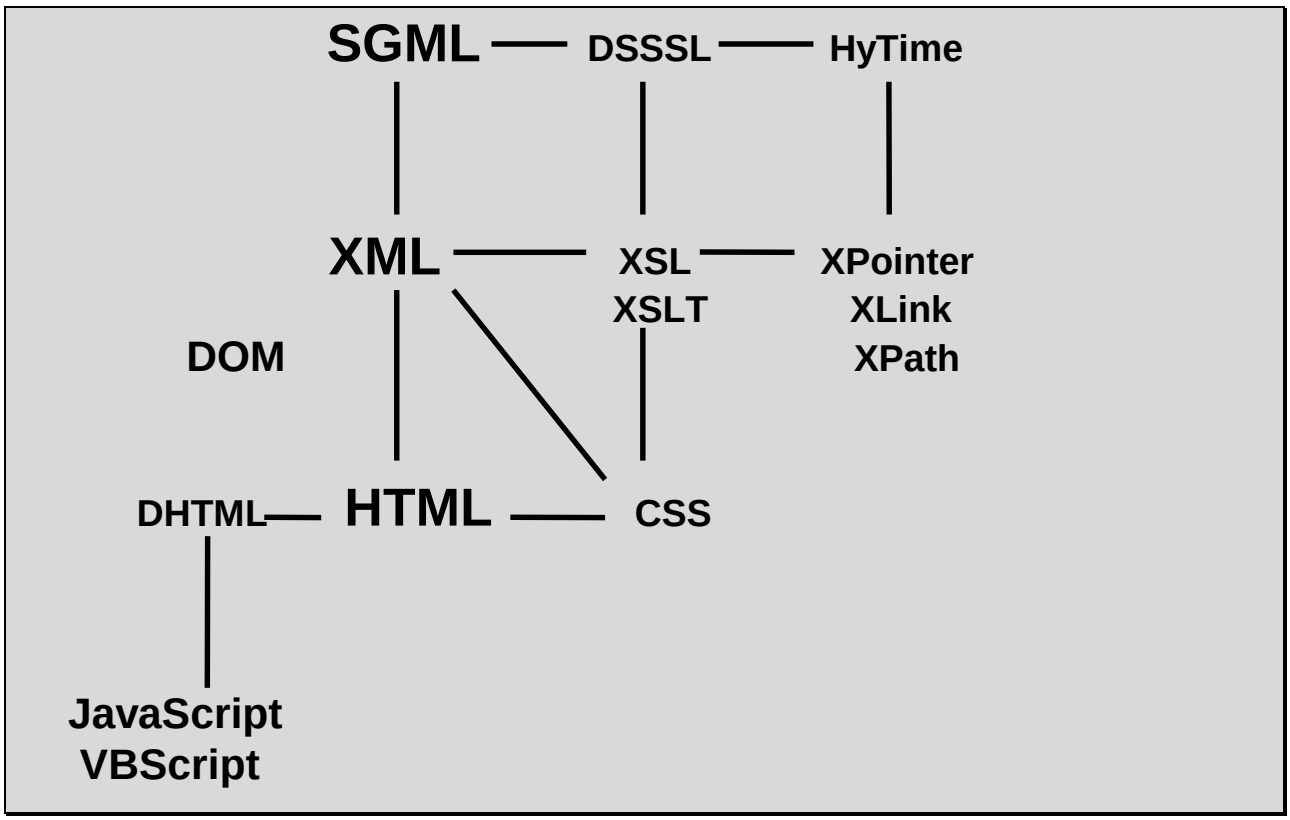
Proprietati:

9. **Structura de pagini** a textului, ancorele şi legăturile (structura de hipertext).
10. **Scopurile urmărite de autor**;
11. **Istoricul parcurgerii** hipertextului de către un cititor;
12. **Efectul** pe care textul îl are asupra cititorului.

În cartea de față sunt prezentate limbajele specifice ipostazelor 2-6. Prima parte a acestei cărți este dedicată reprezentării şi prelucrării textelor adnotate în XML. În acest scop sunt prezentate în detaliu limbajele XML, XPath, XPointer, XSLT şi XLink. Sunt prezentate, de asemenea, câteva elemente ale limbajelor RDF şi RDF-Schema şi ale procesoarelor XML.

Limbajele XSL, XSLT, XLink, RDF şi RDF-Schema sunt, de fapt, sublimbaje ale XML. XPath, XPointer şi convențiile de spații de nume sunt sublimbaje dezvoltate, de asemenea, în conexiune cu XML. În prezent există deja un număr considerabil de limbaje dezvoltate în XML pentru aplicații particulare.

În figura următoare este exemplificată relația care există între SGML, HTML, XML, limbajele de stilizare DSSSL, XSL, XSLT, CSS, limbajele de interconectare HyTime, Xlink, precum şi cu limbaje de tip scenariu (“script”):



1.2 Prelucrarea textelor

Textele pot fi prelucrate în regim de:

- **editare**, adică prin transformări punctuale în scopul de a le extinde, corecta sau modifica;
- **adnotare**, adică de delimitare şi asociere de informații unor zone de text;
- **transformare**, în scopul:
  - **formatării** (adăugare de informații pentru vizualizare pe ecran sau tipărire);
  - **modificării**;
- **extragere** de informații utile:
  - **sumarizare**;
  - **achiziția de cunoştințe** (**mineritul textelor**).

O modalitate unanim acceptată de adnotare a textelor este, după cum s-a precizat în secțiunea anterioară, cea folosind XML [XMLr]. Transformarea textelor adnotate cu XML se poate face cu XSLT, un limbaj dezvoltat tot în XML.

A doua parte a cărții este dedicată unei alte modalități de transformare a textelor: limbajul Perl. Prelucrarea textelor cu Perl nu exclude tehnologia centrată în jurul XML. Mai mult, au fost deja dezvoltate module care permit accesul din Perl la analizoare (parsere) de XML (de exemplu, Expat - vezi “http://jclark.com/XML”).

2. Caracteristicile XML

Limbajul HTML este, istoric, primul standard folosit în adnotarea documentelor de web. El a fost derivat din SGML (**S**tandard **G**eneralized **M**arkup **L**anguage [SGML]), standard ISO (nr. 8879) pentru adnotarea textelor în format electronic.

SGML este un limbaj conceput cu mai mulți ani înaintea apariției WWW. SGML pleacă de la ideea că orice document este format dintr-un număr de componente aflate într-o anumită structură.

SGML pleacă de la următoarele principii:

- permite reprezentarea textelor independent de un anumit sistem de operare, program sau maşină;
- este un metalimbaj, adică un limbaj de descriere formală a unui limbaj de adnotare;
- are un caracter declarativ, adică nu se indică ce se face ci doar se marchează zone de text, prelucrarea putând fi făcută de programe scrise de utilizatori;
- asigură independența datelor, oferind un mecanism general de substituire de şiruri.

Sfârşitul mileniului este caracterizat de o imensă efervescență în introducerea unui nou limbaj de adnotare pe web care să păstreze avantajele HTML, dar care să fie mai flexibil, să fie extensibil şi să permită şi prelucrări semantice. Nu s-a recurs în acest scop la folosirea SGML ca atare, deoarece acesta din urmă este prea puternic şi ar necesita procesoare prea sofisticate şi consumatoare de resurse. Ca urmare, s-a recurs la un nou limbaj, denumit XML [XMLr] ("e**X**tensible **M**arkup **L**anguage"), care păstrează circa 80% din SGML.

XML este destinat nu numai scrierii de pagini de web ci şi schimbului de informații pe Internet (de exemplu, între baze de date construite în standarde diferite, cu o importanță covârşitoare pentru afaceri - "B2B - Bussiness to Bussiness") şi, în general, adnotării documentelor stocate electronic.

XML păstrează avantajele HTML:

- este simplu de utilizat pe Internet;
- documentele XML sunt uşor de creat;
- documentele XML sunt uşor de prelucrat;
- textele pot fi citite relativ uşor şi în forma adnotată, fără a folosi un program special de vizualizare;
- este compatibil cu SGML;

XML are însă şi alte calități, pe care HTML nu le avea:

- Spre deosebire de HTML, XML este extensibil, permițând definirea de noi tipuri de documente (DTD-uri sau scheme) şi, în consecință, a unor noi adnotări, specifice unor aplicații din domenii particulare. A fost astfel eliminată una din principalele deficiențe ale HTML. Există deja, de exemplu, o multitudine de noi limbaje de adnotare derivate din XML, pentru diverse domenii.

- XML poate fi folosit ca un limbaj universal de reprezentare în diverse aplicații. De exemplu, în el se pot reprezenta cereri sau programe de conversie la baze de date în diverse standarde.

- O altă deficiență a HTML, eliminată în XML, este problema reprezentării conținutului documentelor, a semanticii textelor. Această problemă este deosebit de neplăcută astăzi, când există un mare număr de instrumente de căutare pe web dar acestea întorc de cele mai multe ori un număr mult prea mare de documente irelevante pentru subiectul căutat datorită imposibilității unei căutări după conținutul semantic şi nu după chei.

- Permite vizualizarea diferită a aceluiaşi document pentru mai mulți utilizatori.

- Permite transferarea unor prelucrări de la server la utilizator.

XML poate fi privit din mai multe puncte de vedere:

- este un limbaj universal de adnotare a documentelor (derivat din SGML);
- permite declararea unei gramatici pentru un limbaj de adnotări:
    - explicit, printr-un DTD (vezi capitolul 5) sau o schemă,
    - implicit, pe baza structurii de adnotări, chiar dacă nu există un DTD
- este o modalitate foarte comodă de transfer al informațiilor pe Internet între aplicații de categorii foarte diferite;
- este un limbaj adecvat aplicațiilor de baze de date federate;
- este o modalitate universală de a reprezenta liniar orice structură, oricât de complexă;
- este un limbaj cu tipuri manifeste, spre deosebire de formatul fix al fişierelor sau bazelor de date;
- este un limbaj de declarare formală a unui vocabular de elemente şi atribute [Hol00];
- permite restricționarea şi validarea documentelor care pot fi create [Hol00];
- este o “ontologie de nivel 0”, prin faptul că alegerea unui repertoriu de adnotări este echivalentă cu delimitarea entităților care există în domeniul în care se adnotează documentele respective;
- prin adnotare se face, de fapt, un prim pas către facilitarea achiziției cunoştințelor din document;

3. Structura documentelor XML

Limbajul XML (“Extensible Markup Language”) stă la baza clasei documentelor XML. Acest limbaj a fost definit în strânsă legătură cu faptul că documentele XML sunt destinate în primul rând prelucrării de către programe de calculator denumite **procesoare XML**.Un astfel de program citeşte documente XML (ceea ce presupune bineînțeles analizarea lor, motiv pentru care sunt denumite şi analizoare XML), în scopul facilitării accesului unei **aplicații** la conținutul şi structura lor.

3.1 Structura fizică a documentelor XML

Fiecare document XML are atât o structură logică cât şi una fizică. Fizic, documentul este compus din unități numite **entități**. O entitate are un nume şi un conținut. O entitate poate face referire la alte entități pentru a le include în document.

Entitățile pot fi, din punct de vedere al plasării în fişierul documentului:

- interne (în acelaşi fişier)
- externe (aflate în alt fişier).

Entitățile pot fi:

- prelucrate (analizate, “parsed”) de către procesoarele XML; acestea pot fi atât interne cât şi externe;
- neprelucrate (neanalizate, “unparsed”), aceste entități sunt întotdeauna externe.

O ultimă clasificare a entităților este în:

- entități generale, utilizabile fără restricții;
- entități parametru, utilizabile numai în cadrul unui DTD.

Un document începe printr-o **rădăcină** (“root”) sau **entitate document** (“document entity”).

O entitate analizată conține **text**, adică o secvență de caractere, ce pot reprezenta **adnotări** sau **date de tip caracter** (“caracter data”). Caracterele constituie atomii unui text, conform specificațiilor standardului ISO/IEC 10646. Caracterele legale sunt tab, “enter” (CR), sfârşit de linie LF şi caracterele legale ale Unicode şi ISO/IEC 10646.

3.2 Structura logică a documentelor XML

Logic, un document XML este compus din mai multe **elemente**, delimitate în text de adnotări (“**markup**”) de început şi de sfârşit. Adnotările sunt incluse între paranteze unghiulare (delimitate de caracterele “<” şi “>”, care nu pot fi folosite în alt scop în documentele XML; dacă este necesară utilizarea acestor caractere, se folosesc construcțiile: < şi >, adică referințe la entități special constitutite - vezi cap. 5.3). Adnotările apar în perechi, pentru a indica începutul respectiv sfârşitul elementului. Adnotarea de sfârşit se deosebeşte de cea de început printr-o bară oblică):

```
<adnotare>
```

```
 . . .

 </adnotare>
```

De exemplu, pagina de titlu a unei cărți poate avea titlul cărții, autorul, traducătorul şi cel care îngrijeşte volumul. O astfel de pagină de titlu [Tra00a] ar putea fi adnotată după cum urmează:

```
<pagtitlu>
<traducator>Andrei Cornea</traducator> a tradus <titlu>Republica</titlu> de
<autor>Platon</autor> <ingrijit>pentru volumul ingrijit de Constantin
Noica</ingrijit>.
</pagtitlu>
```

Există o infinitate de posibilități de adnotare a unui text. De exemplu, acelaşi text de mai sus ar putea fi adnotat şi cu informații gramaticale asupra substantivelor şi verbelor:

```
<pagtitlu>
<traducator><substantivPropriu>Andrei</substantivPropriu>
<substantivPropriu>Cornea</substantivPropriu></traducator> <verb>a
tradus</verb> <titlu>Republica</titlu> de
<autor><substantivPropriu>Platon</substantivPropriu></autor>
<ingrijit>
pentru <substantiv>volumul</substantiv> <verb>ingrijit</verb> de
<substantivPropriu>Constantin</substantivPropriu>
<substantivPropriu>Noica</substantivPropriu>.
</ingrijit>
</pagtitlu>
```

Pot exista şi elemente vide, fără conținut. Pentru aceste elemente există o adnotare specifică:

```
<adnotare/>
```

Adnotările pot fi atributate. Atributele sunt perechi nume-valoare asociate unui element. De exemplu, dacă informația referitoare la cine a îngrijit volumul nu se va afişa (este secretă), acest lucru se poate indica după cum urmează:

```
<pagtitlu>
<traducator>Andrei Cornea</traducator> a tradus <titlu>Republica<titlu> de
<autor>Platon</autor>
<ingrijit secret=”da”>
pentru volumul ingrijit de Constantin Noica.
</ingrijit>
</pagtitlu>
```

În concluzie, funcția unei adnotări într-un document XML este de a descrie structura sa logica şi de a asocia perechi de atribute-valori la această structură. XML furnizează un mecanism, opțional (spre deosebire de SGML, unde este obligatoriu) numit **DTD** (“Document Type Definition” - definirea tipului de document, vezi capitoolul 5), pentru a defini restricții asupra structurii logice.

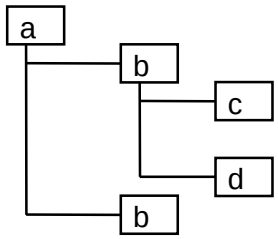
Fiecare document XML are o structură de arbore, cu un element rădăcină. Pentru fiecare element C care nu este radacina documentului, există un alt element P, **părintele lui C** (“parent”) în document, astfel încât C să fie conținut direct de acesta. Se spune că C este **copilul** lui P (“child”).

De exemplu, pentru documentul:

```
<a>
```

```
<b> text1 <c> text2 </c> <d> text3 </d> </b> text4
<b>
text5 </b></a>
```

structura de arbore este:



iar pentru documentul :

```
<bib>
<webdoc> DOM </webdoc>
<webdoc> KQML </webdoc>
<webdoc> RDF Specification
  <an>1999</an>
</webdoc>
<webdoc> RDF Schema Specification</webdoc>
<articol>
 <autor> Y. Shoham, Agent-oriented programming </autor>
 <revista>
   <titlu>Artificial Intelligence</titlu>
   <an>1993</an>
   <pp>51-92</pp>
 </revista>
</articol>
<webdoc> XML </webdoc>
<carte>
 <autor> Cagle, K. </autor><autor>Gibbons, D.</autor>
 <autor> Hunter, D.</autor><autor> Ozu, N.</autor>
 <autor>Pinnock, J.</autor><autor>Spencer, P.</autor>
 <titlu> Beginning XML </titlu>
 <editura>Wrox Press,</editura>
 <an> 2000</an>
</carte>
</bib>
```

structura de arbore este (ilustrată în modalitatea specifică InternetExplorer 5.x de la Microsoft):

```
- <bib>
    <webdoc>DOM</webdoc>
    <webdoc>KQML</webdoc>
  - <webdoc>
      RDF Specification
      <an>1999</an>
     </webdoc>
    <webdoc>RDF Schema Specification</webdoc>
  - <articol>
      <autor>Y. Shoham, Agent-oriented programming</autor>
    - <revista>
        <titlu>Artificial Intelligence</titlu>
        <an>1993</an>
        <pp>51-92</pp>
      </revista>
    </articol>
```

```
    <webdoc>XML</webdoc>
  - <carte>
      <autor>Cagle, K.</autor>
      <autor>Gibbons, D.</autor>
      <autor>Hunter, D.</autor>
      <autor>Ozu, N.</autor>
      <autor>Pinnock, J.</autor>
      <autor>Spencer, P.</autor>
      <titlu>Beginning XML</titlu>
      <editura>Wrox Press,</editura>
      <an>2000</an>
     </carte>
  </bib>
```

Zonele de text care nu fac parte dintr-o adnotare reprezintă **datele caracter** ("**character data**") ale documentului. În conținutul elementelor, datele caracter sunt orice şir de caractere ce nu conține delimitatorul de început al unei adnotări.

Un document XML poate conține în structura sa logică:

- elemente, conform precizărilor anterioare;
- comentarii;
- secțiuni CDATA;
- indicații (instructiuni) de procesare;
- referințe la caractere sau la entități;
- declarații de prolog, care cuprind declarația XML (vezi mai jos) şi, eventual definirea unui DTD).

Toate aceste tipuri de conținut sunt indicate in documentele XML prin adnotări specifice.

Comentariile pot apare practic oriunde în documentele XML, în afara adnotărilor. Comentariile sunt delimitate de "<!--" şi "-->":

```
<!-- Comentariu -->
```

Sectiunile CDATA conțin date caracter (CDATA), care se iau ca atare (nu sunt analizate de procesoarele XML). Ele pot apare oriunde pot apare datele caracter. Sunt utilizate pentru a include blocuri de text ce conțin caractere ce ar fi altfel recunoscute drept adnotări.

Secțiunile CDATA încep cu şirul "<![CDATA[" si se termina cu şirul "]]>":

<![CDATA[(Char* - (Char* '[[>' Char*))]]>

Un exemplu al unei sectiuni CDATA, in care "<adnotare>" si "</adnotare>" sunt considerate ca date caracter si nu ca adnotări este:

```
<![CDATA[<adnotare>
 Text
</adnotare>
]]>
```

Indicațiile de procesare ("processing instructions" - PI) sunt o modalitate de a include în documente indicații (instrucțiuni) pentru aplicațiile care prelucrează documentele XML. Ele nu sunt părți ale datelor caracter, dar sunt trimise aplicației.

Indicațiile de procesare sunt delimitate de “<?” şi ?>”, încep cu o țintă (“PITarget”) utilizată pentru a identifica aplicația căreia îi este destinată instrucțiunea, urmată de informații suplimentare de trimis aplicației. Numele de ținte de forma 'XML', 'xml' sau alte combinații posibile, sunt rezervate pentru standardizare în versiunea actuală [XMLr] sau pentru versiunile ulterioare.

Un exemplu de utilizare a indicațiilor de procesare este **declarația XML**, care specifică faptul că este vorba de un document XML, precum şi versiunea folosită. Documentele XML pot şi este indicat să înceapă cu ea.

```
<?xml version="1.0"?>
<adnotare> Text afisat </adnotare>
```

Într-un document XML se pot face **referințe la caractere** sau **referințe la entități interne**. Ele sunt secvențe de caractere delimitate de “&#” sau “&#x” şi “;”, pentru caractere, sau de “&” şi “;”, pentru entități. Aceste referințe țin locul, unui caracter, respectiv unui şir de caractere, cu care sunt înlocuite de procesoarele XML (vezi 5.3).

4. Documente XML bine formate

Un **document XML bine format** trebuie să respecte următoarele reguli:

- Documentul trebuie să aibă adnotările în perechi (nu pot lipsi adnotări de început sau sfârşit, ca în cazul SGML sau HTML), şi contează dacă o literă este mare sau mică (<adn> este diferit de <ADN>, XML este "case sensitive").

- Orice document XML trebuie să aibă un singur element superficial, numit **rădăcină** ("root"), sau element document, care nu apare în conținutul altui element. Altfel spus, un document nu trebuie să fie structurat ca mai jos:

```
<aaaa>
. . .
</aaaa>
<bbbb>
 . . .
</bbbb>
```

- Dacă sunt mai multe perechi de adnotări, ele trebuie să fie imbricate (să respecte o structură de "paranteze"), adică nu trebuie să avem situații de genul:

```
<aaaa>
     . . .
     <bbbb>
     . . .
     <cccc>
     . . .
     </bbbb>
     . . .
     </cccc>
     . . .
</aaaa>
```

- Numele elementelor trebuie să satisfacă anumite reguli, de exemplu, să înceapă cu o literă sau cu "_" (vezi 5.2).

- Valorile atributelor adnotărilor trebuie să fie puse între ghilimele (de exemplu, <adn val="2"> şi nu <and val=2>).

- Nu trebuie ca un atribut să apară de mai multe ori în aceeaşi adnotare.

- Pot exista adnotări care nu includ o zonă de text. Aceste adnotări, denumite **adnotări vide**, au caracterul "/" la sfârşit, de exemplu: <adnotare/>.

- Entitățile `amp, lt, gt, apos, quot` (pentru caracterelor "&", "<", ">", apostrof şi ghilimea) pot fi utilizate fără a fi declarate.

În plus, trebuie respectate nişte reguli pentru definirea şi utilizarea entităților [XMLr], cum ar fi faptul că entitățile interne nu pot fi recursive.

Mai jos este ilustrat un document XML bine format:

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

```
<bib>
<webdoc src="http://concept.cs.uah.edu/CG/cg-standard.html" abr="CG">
Conceptual Graphs </webdoc>
<webdoc abr="CYC" src="http://www.cyc.org"> CYC</webdoc>
<webdoc abr="DOM" src="http://www.w3.org/DOM"> DOM </webdoc>
<webdoc abr="KIF" src="http://logic.stanford.edu/kif/kif.html"> Knowledge
Interchange Format </webdoc>
<webdoc abr="KQML" src="http://www.cs.umbc.edu/kqml"> KQML </webdoc>
<webdoc abr="RDF" src="http://www.w3.org/RDF"> RDF Specification
<an>1999</an></webdoc>
<webdoc abr="RDFS" src="http://www.w3.org/TR/WD-rdf-schema"> RDF Schema
Specification</webdoc>
<articol abr="Sho93">
  <autor> Y. Shoham, Agent-oriented programming </autor>
  <revista nr="60">
    <titlu>Artificial Intelligence</titlu>
    <an>1993</an>
    <pp> 51-92</pp>
  </revista>
</articol>
<webdoc abr="XML" src="http://www.w3.org/XML"> XML </webdoc>
<carte abr="XMLc" isbn="1-861003-4-12">
  <autor> Cagle, K. </autor><autor>Gibbons, D.</autor><autor> Hunter,
D.</autor><autor> Ozu, N.</autor><autor>Pinnock, J.
 </autor><autor>Spencer, P.</autor>
 <titlu> Beginning XML </titlu>
 <editura>Wrox Press,</editura>
 <an> 2000</an>
</carte>
</bib>
```

şi structura sa de arbore, afişată de InternetExplorer:

```
- <bib>
    <webdoc src="http://concept.cs.uah.edu/CG/cg-standard.html"
abr="CG">Conceptual Graphs</webdoc>
    <webdoc abr="CYC" src="http://www.cyc.org">CYC</webdoc>
    <webdoc abr="DOM" src="http://www.w3.org/DOM">DOM</webdoc>
    <webdoc abr="KIF" src="http://logic.stanford.edu/kif/kif.html">Knowledge
Interchange Format</webdoc>
    <webdoc abr="KQML" src="http://www.cs.umbc.edu/kqml">KQML</webdoc>
  - <webdoc abr="RDF" src="http://www.w3.org/RDF">
      RDF Specification
      <an>1999</an>
    </webdoc>
    <webdoc abr="RDFS" src="http://www.w3.org/TR/WD-rdf-schema">RDF Schema
Specification</webdoc>
  - <articol abr="Sho93">
      <autor>Y. Shoham, Agent-oriented programming</autor>
    - <revista nr="60">
        <titlu>Artificial Intelligence</titlu>
        <an>1993</an>
        <pp>51-92</pp>
      </revista>
    </articol>
    <webdoc abr="XML" src="http://www.w3.org/XML">XML</webdoc>
```

```
  - <carte abr="XMLc" isbn="1-861003-4-12">
      <autor>Cagle, K.</autor>
      <autor>Gibbons, D.</autor>
      <autor>Hunter, D.</autor>
      <autor>Ozu, N.</autor>
      <autor>Pinnock, J.</autor>
      <autor>Spencer, P.</autor>
      <titlu>Beginning XML</titlu>
      <editura>Wrox Press,</editura>
      <an>2000</an>
    </carte>
  </bib>
```

Un exemplu remarcabil de documente XML bine formate sunt documentele XHTML [XHTML]. Acesta din urmă este un standard care permite să se scrie documente HTML care să fie, în acelaşi timp, şi documente XML bine formate. În acest scop, au fost introduse restricțiile de bună formare XML în scrierea de documente HTML. De exemplu, nu se mai permite ca adnotările de sfârşit să lipsească. Se pot folosi, în schimb elemente vide XML. Toate adnotările trebuie să fie scrise cu litere mici. Un exemplu de pagină de web XHTML este [XHTML] :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <title>Virtual Library</title>
 </head>
 <body>
  <p>Moved to <a href="http://vlib.org/">vlib.org</a>.
  </p>
 </body>
</html>
```

5. Tipizarea documentelor. Documente XML valide

Un document XML bine format devine un **document XML valid**, dacă, în plus față de restricțiile de bună formare, are asociată şi o definiție de tip de document (**Document Type Definition** - DTD), pe care o respectă. Un **tip de document** este constituit de un anumit repertoriu de adnotări împreună cu modalitatea lor de structurare. Această practică este preluată de la SGML, în acest mod introducându-se o tipizare a documentelor, fiecare document având ataşat un DTD.

Un DTD poate fi specificat separat de un anume document, fapt care are mai multe implicații importante:

- Acelaşi DTD poate fi folosit de mai multe documente. De exemplu, limbajul HTML este un subset al SGML, existând, bineînțeles, un DTD pentru el. Deoarece acest DTD al HTML este cunoscut de toate browserele de web este suficient să fie transmise numai documentele adnotate nu şi DTD-ul.

- Acelaşi document poate fi considerat conform mai multor DTD-uri (compatibile).

- Se poate verifica faptul că un document satisface un anumit DTD.

Un document are o structură restricționată de DTD-ul asociat. Bineînțeles că o structurare ierarhică a unui document se realizează trivial în SGML şi XML prin specificarea de adnotări imbricate una în alta. În SGML şi limbajele derivate din el se pot face însă şi specificări de documente structurate în formă de rețea prin adnotări care descriu legături între zone diferite din acelaşi document sau documente diferite. Aceste legături pot fi specificate prin mecanismul ID - IDREF sau prin limbajul HyTime, pentru SGML, respectiv XLink pentru XML (vezi capitolul 9).

Declarația tipului de document XML ("document type declaration") poate fi făcută atât în document (subset intern) cât şi într-o entitate externă, conform următoarei sintaxe:

<!DOCTYPE element_document [sursa_subset_extern] [subset_intern_al_DTD]>

unde **sursa_subset_extern** poate fi una din următoarele variante:

SYSTEM locație
PUBLIC locație1 locație2

De exemplu, DTD-ul "lectie", inclus în acelaşi document, se declară după cum urmează:

```
<?xml version="1.0"?>
<!DOCTYPE lectie
[
. . . continut declaratie DTD
]>

<lectie>
. . .
</lectie>
```

Dacă DTD-ul este într-un alt fişier se foloseşte următoarea declarație:

```
<?xml version="1.0"?>
<!DOCTYPE lectie SYSTEM "file:avl.dtd">

<lectie>
. . .
</lectie>
```

Un DTD conține mai multe **declarații de adnotări** (“markup declaration”). Acestea pot fi:

- declarație a unui tip de element (“ELEMENT”),
- declarație a unei liste de atribute (“ATTLIST”),
- declarație a unei entități (“ENTITY”),
- declarație de notatie (“NOTATION”).

În general, declarațiile de adnotări dintr-un DTD au forma:

<!cuvânt_cheie parametru*>

unde cuvânt_cheie poate lua und din cele patru valori de mai sus, corespunzătoare celor patru tipuri de dec;arații.

5.1 Declararea tipurilor de elemente

Declarația unui tip de element constă în:

- specificarea numelui elementului, nume care va fi folosit pentru adnotări;
- specificarea conținutului elementului:

<!ELEMENT nume conținut>

Dacă elementul are drept conținut un text luat ca atare, fără o structurare în alte elemente, se foloseşte notația:

```
(#PCDATA)
```

de exemplu:

```
<!ELEMENT titlu (#PCDATA)>
```

Dacă elementul poate avea o structură internă, aceasta este precizată printr-o gramatică simplă ce guvernează tipurile de elemente copil permise şi ordinea în care este permisă apariția lor.

Gramatica este specificată conform următoarelor reguli:

- parantezele sunt folosite pentru grupare sau delimitare;
- “,” separă elemente care apar în secvență;
- “|” separă două elemente alternative, din care se poate alege unul drept conținut;
- “+” indică apariția o data sau de mai multe ori a unui element;
- “*” indică lipsa sau apariția de mai multe ori a unui element;
- “?” specifică un element opțional;
- absența unui astfel de semn, înseamnă că elementul trebuie să apară exact o dată.

De exemplu, faptul că o carte are neapărat o pagină de titlu, urmată de unul sau mai multe capitole se precizează după cum urmează:

```
<!ELEMENT carte (pagtitlu, (capitol)+)>
```

Faptul că o pagină de titlu are un titlu, urmat de autor, traducător şi, opțional, de adnotarea care specifică cine a îngrijit cartea se specifică:

```
<!ELEMENT pagtitlu (titlu, autor, trad, ingrijit?)>
```

Declarația următoare specifică faptul că un subiect are în interior oricâte din elementele specificate:

```
<!ELEMENT subiect ( concept | proprietate | fig | demonstratie | prog )*>
```

Se poate ca un element să aibă şi un conținut mixt, adică atât text cât şi elemente, ca în exemplul următor:

```
<!ELEMENT demonstratie (#PCDATA | concept | proprietate | fig)*>
```

În această situație, este obligatoriu ca declarația `PCDATA` să apară la început.

Un exemplu de DTD este :

```
<!DOCTYPE carte [
 <!ELEMENT carte (pagtitlu, (capitol)+)>
 <!ELEMENT pagtitlu (titlu, autor, trad, ingrijit?)>
 <!ELEMENT titlu (#PCDATA)>
 <!ELEMENT autor (#PCDATA)>
 <!ELEMENT traducator (#PCDATA)>
 <!ELEMENT ingrijit (#PCDATA)>
 <!ELEMENT capitol (#PCDATA)>
]>
```

Acest DTD specifică faptul că o carte are o pagină de titlu urmată unul sau mai multe capitole. Pagina are un titlu, un autor, un traducător şi, opțional, numele celui care a îngrijit volumul.

5.2 Declararea listei de atribute a unui element

Atributele sunt folosite pentru a asocia elementelor perechi de nume - valoare. Declarațiile listelor de atribute precizează:

- ce atribute pot apare într-un tip de element;
- constrângerile legate de tipul acestor atribute;
- valorile implicite pentru atribute.

Declaratia listei de atribute ("**attribute-list declarations**") specifică numele, tipul şi valoarea implicită (dacă există) a fiecărui atribut asociat cu un tip de element, după cum urmează:

<!ATTLIST nume_element (nume_atribut tip_atribut valoare_implicita)* >

Tipurile de atribute XML sunt de trei feluri:

- de tip şir, definit prin `CDATA`;

- de tip nume: `ID, IDREF, IDREFS, NMTOKEN, NMTOKENS, ENTITY, ENTITIES`
- de tip enumerat.

Un **nume** (“**Name**”) este un cuvânt care începe cu o literă şi continuă cu litere, cifre, cratime, liniuțe de subliniere, două puncte sau cu puncte.

Un **fragment de nume** (“**Nmtoken**”) este un mixaj dintre mai multe caractere pentru nume.

NameChar ::= Letter | Digit | '.' | '-' | '_' | ':' | CombiningChar | Extender
Name ::= (Letter | '_' | ':') (NameChar)*
Names ::= Name (S Name)*
Nmtoken ::= (NameChar)+
Nmtokens ::= Nmtoken (S Nmtoken)*

unde S este o zonă de spațiu.

Atributele de tip nume pot avea asociate diverse restricții semantice şi lexicale:

- Tipul ID specifică faptul că atributul respectiv trebuie să aibă o valoare unică. ID nu poate avea ca declarație de valori implicite decât #REQUIRED şi #IMPLIED.
- IDREF este un tip care indică faptul că atributul este o referință la un element cu un ID.
- IDREFS este o mulțime de IDREF.
- NMTOKEN şi NMTOKENS se conformează producțiilor de mai sus.

Declarația valorii implicite furnizează informația dacă prezența atributului este necesară, iar dacă nu, cum trebuie să reacționeze procesorul XML dacă un atribut declarat lipseşte din document. Sintaxa acestei declarații este:

DefaultDecl ::='#REQUIRED' | '#IMPLIED'
| (('#FIXED' S)? AttValue)

În declararea unui atribut,

- #REQUIRED semnifică faptul că atributul şi valoarea sa vor trebui întotdeauna sa fie furnizate;
- #IMPLIED precizează că nu este furnizată nici o valoarea implicită;
- #FIXED stabileşte faptul că întotdeauna atributul va trebui să aibă valoarea implicită asociată.

De exemplu, atributul “nr” din elementul “proprietate” trebuie neapărat specificat şi valoarea acestuia trebuie să fie unică. Valoarea atributului “nume” va fi cea indicată, dacă este prezentă. Asupra valorii nu există nici o restricție.

```
<!ATTLIST proprietate
     nr ID #REQUIRED
     nume CDATA #IMPLIED>
```

Dacă atributul nu este declarat nici ca #REQUIRED nici ca #IMPLIED, AttValue este valoarea implicită.

Atributele enumerate specifică o listă de valori posibile. De exemplu, în declarația de mai jos, atributul “secret” poate lua una din valorile “da” sau “nu”. Implicit este luată valoarea “nu”.

```
<!ATTLIST ingrijit secret (da|nu) “nu”>
```

Un exemplu de DTD complet este:

```
<!DOCTYPE lectie [
<!ELEMENT lectie (titlu,(subiect)*) >
<!ATTLIST lectie
    nume ID #IMPLIED>
<!ELEMENT subiect ( concept | proprietate | fig | demonstratie | prog )*>
<!ATTLIST subiect
     nume ID #REQUIRED>
<!ELEMENT fig EMPTY>
<!ATTLIST fig
     nr ID #REQUIRED
     caption CDATA #REQUIRED>
<!ELEMENT titlu (#PCDATA)>
<!ELEMENT concept (#PCDATA)>
<!ELEMENT proprietate (#PCDATA)>
<!ELEMENT demonstratie (#PCDATA | concept | proprietate | fig)*>
<!ELEMENT prog (#PCDATA)>
<!ATTLIST proprietate
     nr ID #REQUIRED
     nume CDATA #IMPLIED>
]>
```

5.3 Declararea entităților

Un document XML poate consta dintr-una sau mai multe unități de stocare. Acestea poartă denumirea de **entități**. Toate entitățile au un **continut** şi pot fi identificate printr-un **nume**. Excepție face subsetul DTD extern şi **entitatea document**, care este rădăcina arborelui entităților, fiind un punct de plecare pentru procesorul XML.

Entitățile pot fi **analizate** sau **neanalizate**. Conținuturile entitatilor analizate sunt texte considerate părți integrante ale documentului.

O entitate neanalizată este o resursa al cărei conținut poate sau poate să nu fie text, iar dacă este text, poate să nu fie XML. Fiecare entitate neanalizată are asociată o notație, identificată prin nume. În spatele cerinței ca un procesor XML să-i facă disponibile unei aplicații, identificatorii entității şi ai notației, XML-ul nu pune nici o restricție asupra conținutului entităților neanalizate.

Declarația unei entități interne se face ca în exemplul următor:

```
<!ENTITY Pub "Universitatea Politehnica Bucuresti">
```

Ori de câte ori procesorul XML va întâlni &Pub; în document, se va rescrie această secvență cu “Universitatea Politehnica Bucuresti”.

Daca entitatea este externă, se declară după cum urmează:

```
<!ENTITY en SYSTEM “http://cs.pub.ro/~ceva/fisier.xml">
```

sau

```
<!ENTITY en PUBLIC “http://cs.pub.ro/~ceva/fisier.xml" “… URI alternativ …”>
```

La începutul unui fişier care este o entitate externă trebuie făcută o declarație de text (dacă nu se folosesc standardele UTF-16 sau UTF-8 pentru caractere). De exemplu, pentru un fişier folosind caractere pentru limba japoneză, se foloseşte declarația [XMLc}:

```
<?xml encoding=”Shift_JIS”?>
```

Entitățile neanalizate sunt întotdeauna externe şi trebuie neapărat să conțină o descriere “NDATA”, a notației folosite (vezi şi secțiunea următoare):

<!ENTITY nume SYSTEM “. . . locatie . . .” NDATA notatie>
<!ENTITY nume PUBLIC “. . . locatie1 . . .” “. . . locatie2 . . .” NDATA notatie>

Entitățile parametrizate sunt folosite exclusiv în interiorul unui DTD. Ele se declară prin:

<!ENTITY % nume “text inlocuitor”>

Referirea la aceste entități se face prin:

%nume;

5.4 Declararea notațiilor

Notațiile sunt folosite pentru a specifica ce tipuri de entități neanalizate sunt folosite în document, care este URI-ul unde se poate găsi un program care poate prelucra entitatea. Declarația de notație asociază un nume fiecărei notații. De exemplu, declarația următoare [XMLc] asociază numele “png” unui anumit tip de entitate neanalizată şi precizează programul care o prelucrează:

```
<!NOTATION png SYSTEM “http://www.wrox.com/Programs/PNG_Viewer.exe”>
```

6. Spații de nume pentru adnotarea documentelor

Fiecare document XML are un vocabular de nume de adnotări. O situație care poate apărea este folosirea în acelaşi document a mai multor vocabulare, situație care poate duce la ambiguități generate de prezența aceluiaşi nume în mai multe vocabulare. De exemplu, adnotarea `<set>` apare atât în vocabularul limbajului de adnotare Scalable Vector Graphics (SVG) cât şi în cel al Mathematical Markup Language (MathML). În SVG <set> defineşte un domeniu pentru o variabilă iar în MathML defineşte o mulțime [XSLT00].

Pentru a deosebi cele două vocabulare într-un acelaşi document XML, trebuie să se folosească adnotări prefixate:

<svg:set>
<math:set>

Prefixul, după cum se vede, este separat de adnotarea propriu-zisă prin caracterul ":". Fiecare prefix trebuie asociat cu un URI (Universal Resource Identifier) înainte de a fi utilizat. Acest lucru se face cu o declarație de spațiu de nume, care are forma:

xmlns:[*prefix*]=*URI*

Un URI poate fi un URL ("Universal Resource Locator") sau un URN ("Universal Resource Name"). Un URN este un şir de caractere (un nume) unic pe Internet, care, spre deosebire de un URL, nu specifică neapărat o locație pe web [XMLc], specificând doar un identificator unic.

De exemplu, pentru cele două limbaje de adnotare de mai sus, SVG şi MathML, declarațiile de spații de nume sunt:

xmlns:svg="http://www.w3.org/2000/svg-20000629"
xmlns:math="http://www.w3.org/1998/Math/MathML"

Dacă nu se specifică un prefix, se consideră spațiul respectiv de nume ca implicit. De exemplu, pentru declarațiile de spații de nume următoare:

xmlns="URI1"
xmlns:a="URI2"
xmlns:b="URI3"

adnotările

```
<ad> . . . <ad>

<a:ad> . . . <a:ad>

<b:ad> . . . <b:ad>
```

semnifică ceea ce este precizat în URI1, URI2 respectiv URI3.

Câteva declarații de spații nume folosite frecvent sunt:

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsl="http://www.w3.org/1999/XSL/Format"
xmlns:xsl="http://www.w3.org/TR/WD-xsl"

xmlns:xt=”http://www.jclark.com/xt”
xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:rdfp='http://www.w3.org/XML/2000/04/rdf-parse/#”
xmlns:xlink="http://www.w3.org/1999/xlink"

Primele trei URI (pentru prefixul xsl) se deosebesc prin faptul că primul se referă la vocabularul pentru XSLT (transformări), al doilea la XSL (pentru formatări) iar al treilea la vocabularul considerat de Internet Explorer 5.1. xt corespunde vocabularului procesorului XT, dezvoltat de James Clark (“http:/jclark.com/xml”).

7. Adresarea către interiorul documentelor XML

7.1 Limbajul XPath

XPath este un limbaj care permite adresarea către interiorul documentelor XML. El este folosit în XSL, XSLT şi XPointer. În XSLT, limbajul XPath permite şi definirea de şabloane.

XPath consideră un document XML ca un arbore format din şapte tipuri de noduri:

- **Rădăcina**. Există un singur astfel de nod într-un document. El are drept copii nodul element corespunzător documentului şi, eventual noduri pentru instrucțiuni de procesare şi noduri comentariu.

- **Element**. Pentru fiecare element din document există câte un nod element. Un astfel de nod are drept copii alte noduri element, noduri pentru instrucțiuni de procesare, comentarii şi noduri pentru text. Fiecare nod element poate avea un identificator unic, dacă acest lucru este specificat în DTD prin tipul ID (vezi secțiunea 5.2).

- **Atribut**. Fiecare nod element are o mulțime de noduri atribut, corespunzătoare atributelor sale. Aceste noduri atribut au ca părinte nodul element respectiv dar ele nu sunt copii ai elementului părinte.

- **Spațiu de nume**. Fiecare element poate avea asociate mai multe astfel de noduri.

- **Instrucțiuni de prelucrare**

- **Comentariu**; există câte un astfel de nod pentru fiecare comentariu din document;

- **Text** - secvențe maximale de date de tip caracter.

În general, nodurile pot avea nume formate dintr-o parte locală şi eventual un URI pentru un spațiu de nume.

Principala construcție XPath este expresia. În urma evaluării expresiilor XPath, se obțin obiecte de tipurile:

- un nod;
- mulțime de noduri;
- boolean;
- număr (în virgulă mobilă);
- şir de caractere.

Evaluarea fiecărei expresii XPath se face într-un context format din:

- un nod context;
- o pereche de numere întregi care reprezintă o poziție şi o mărime a contextului, prima fiind mai mică decât cea de-a doua;
- legări de variabile;
- o bibliotecă de funcții;
- mulțimea declarațiilor de spații de nume.

Principalul tip de expresie XPath este calea către o locație (“location path”). Ea include convențiile uzuale folosite pentru specificarea căilor către fişiere în sistemele de operare.

Căile către locații sunt o secvență de paşi separați de caracterul “/”. Ele pot fi:

- Relative - căi care încep de la nodul context curent.
- Absolute - căi care pleacă din rădăcina documentului, fapt indicat prin faptul că încep cu “/”.

Fiecare pas are trei părți: o axă, un tip de nod şi zero sau mai multe predicate.

axa::nod[predicat]*

Axa specifică direcția în care se face pasul curent. Ea poate fi:

- `child`, nodul copil, aceasta fiind axa implicită;
- `descendant`, adică copii sau copiii copiilor, pe oricâte nivele;
- `parent`
- `ancestor`, adică părinte sau părintele părintelui ş.a.m.d.;
- `following-sibling`
- `preceding-sibling`
- `following`
- `preceding`
- `attribute`, adică atributele nodului context;
- `namespace`
- `self`
- `descendent-or-self`
- `ancestor-or-self`

Tipul nodului este specific fiecărei axe în parte. De exemplu, el este un element pentru `child`, un atribut pentru `attribute` etc.

Predicatul este folosit pentru a filtra o mulțime de noduri.

În expresiile XPath pot fi folosite diverse funcții predefinite. Câteva din acestea sunt listate mai jos împreună cu ceea ce întorc şi ce argumente necesită:

number last() - ultimul element dintr-o mulțime
number position() - poziția unui element
number count(node-set) - numărul de elemente
node-set id(object) - nodul cu identificatorul unic dat ca argument
string name(node-set?)
string string(object?)
boolean starts-with(string, string)
boolean contains(string, string)
string substring-before(string, string)
number string-length(string?)

Pentru a simplifica scrierea expresiilor XPath sunt disponibile câteva abrevieri pentru cazurile cele mai frecvente:

Abrevieri	**Ce se consideră față de nodul context**
*	toți copii
text()	copiii de tip text

@nume	atributul denumit “nume”
@*	toate atributele
nume[i]	al i-lea copil “nume”
nume[last()]	ultimul copil “nume”
*/nume	nepoții “nume”
//nume	descendenții “nume”
.	nodul context
..	Părintele
../@nume	atributul “nume” al părintelui
nume[@atr=”val”]	copii “nume” care au atributul “atr” cu valoarea “val”

Pentru exemplificarea utilizării expresiilor XPath, să considerăm fişierul XML“b1.xml” de mai jos. În scopul urmăririi uşoare a structurii documentului, acesta a fost rescris după cum urmează:

```
<bib>

    <webdoc
        src="http://concept.cs.uah.edu/CG/cg-standard.html"
        abr="CG">
      Conceptual Graphs
    </webdoc>

    <webdoc
        abr="CYC"
        src="http://www.cyc.org">
      CYC
    </webdoc>

    <webdoc
        abr="DOM"
        src="http://www.w3.org/DOM">
      DOM
    </webdoc>

    <webdoc
        abr="KIF"
        src="http://logic.stanford.edu/kif/kif.html">
      Knowledge Interchange Format
    </webdoc>

    <webdoc
        abr="KQML"
        src="http://www.cs.umbc.edu/kqml">
      KQML
    </webdoc>

    <webdoc
        abr="RDF"
        src="http://www.w3.org/RDF">
      RDF Specification

      <an>
        1999
      </an>

    </webdoc>
```

```
<webdoc
    abr="RDFS"
    src="http://www.w3.org/TR/WD-rdf-schema">
  RDF Schema Specification
</webdoc>

<articol
      abr="Sho93">

  <autor>
    Y. Shoham, Agent-oriented programming
  </autor>

  <revista
      nr="60">

    <titlu>
      Artificial Intelligence
    </titlu>

    <an>
      1993
    </an>

    <pp>
      51-92
    </pp>

  </revista>
</articol>

<webdoc
    abr="XML"
    src="http://www.w3.org/XML">
  XML
</webdoc>

<carte
    abr="XMLc"
    isbn="1-861003-4-12">

  <autor>
    Cagle, K.
  </autor>

  <autor>
    Gibbons, D.
  </autor>

  <autor>
    Hunter, D.
  </autor>

  <autor>
    Ozu, N.
  </autor>

  <autor>
    Pinnock, J.
  </autor>

  <autor>
    Spencer, P.
  </autor>
```

```
      <titlu>
        Beginning XML
      </titlu>

      <editura>
        Wrox Press,
      </editura>

      <an>
        2000
      </an>

    </carte>
</bib>
```

Să vedem acum câteva expresii XPath pentru documentul anterior şi rezultatul obținut:

Expresie XPath	Rezultat obținut
/bib/webdoc	Conceptual Graphs
//an	1999
/bib/articol/*	Y. Shoham, Agent-oriented programming
/bib/webdoc/following-sibling::webdoc	CYC
/bib/webdoc/attribute::*	http://concept.cs.uah.edu/CG/cg-standard.html
//an/../@abr	RDF
/bib/webdoc[6]	RDF Specification 1999
/bib/webdoc[4]/following-sibling::*	KQML
/bib/webdoc[@*]	Conceptual Graphs
//*[@nr]	Artificial Intelligence 1993 51-92
/bib/webdoc[6][@abr]	RDF Specification 1999
/bib/webdoc/text()	Conceptual Graphs
/bib/webdoc[last()]	XML
//*[count(*)>2]/*[2]	CYC

7.2 Limbajul XPointer

În HTML se poate face o referire (de exemplu, o legătură) doar la un întreg document sau la un punct specificat prin #nume (şi definit în document prin <a name="nume">). XPointer [XPointer] este un limbaj pentru adresare în interiorul documentelor XML (pentru adresarea într-un tip oarecare de document, se poate folosi limbajul XLink - vezi capitolul 9). Expresiile XPointer sunt extensii ale XPath şi pot fi adăugate la un URI, după caracterul "#", ca în următorul exemplu:

```
http://.../b1.xml#xpointer(/bib/articol)
```

Această expresie nu presupune că ar exista o ancoră <a name="…"> în interiorul documentului, ca în HTML.

XPointer este un limbaj care extinde XPath în sensul că permite adresarea nu numai a unui nod sau a unei mulțimi de noduri (ca în XPath) ci şi a unui punct sau a unui fragment dintr-un nod al unui document.

Un **punct** dintr-un document este precizat printr-o pereche formată dintr-un nod container şi un index. Dacă nodul container are fii, indexul se referă la al câtelea fiu este considerat. Dacă

nodul container nu are fii, de exemplu, este un nod text, indexul este considerat față de şirul de caractere care formează nodul.

De exemplu, un punct situat la al 2-lea caracter al numelui autorului cărții din secțiunea anterioară poate fi referit cu XPointer după cum urmează:

```
. . .#xpointer(/carte/autor/text()[2]
```

Un **domeniu** este delimitat de două puncte. Un exemplu de utilizare este:

```
. . .#xpointer(/carte/autor to /carte/titlu)
```

În XPointer, pentru a simplifica scrierea, se pot folosi şi prescurtări, după cum urmează:

```
#nume
```

în loc de:

```
#xpointer(id(“nume”)
```

şi

```
#/1/1/2
```

în loc de:

```
#xpointer(/*[1]/*[1]/*[2])
```

8. Transformarea şi stilizarea documentelor; limbajele XSL şi XSLT

Este, sperăm, evident că documentele, în general, şi cele XML, în particular, sunt scrise pentru a fi citite sau prelucrate. Pentru a fi citite, ele trebuie, de cele mai multe ori, transformate într-o formă cât mai adecvată citirii sau, altfel spus, **formatate**.

Documentele pot fi folosite nu numai prin citire ci şi pentru alte scopuri, cum ar fi extragerea de informații utile, restructurarea lor, traducerea în alt limbaj (de fapt, formatarea poate fi văzută ca o traducere intr-un alt limbaj). Toate aceste prelucrări pot fi subsumate de termenul de **transformare**. Formatarea şi transformarea documentelor sunt prelucrări denumite **stilizare**.

XSL ("eXtended Stylesheet Language" – limbajul foilor de stil) şi XSLT (XSL Transformation) sunt limbaje de formatare şi transformare a documentelor XML. Un fişier XSL sau XSLT este, mai întâi de toate, un document XML bine format. Un fişier XSLT este denumit **foaie de stil** ("**stylesheet**") deoarece poate fi văzut ca o descriere a unei modalități de stilizare.

Transformarea documentelor XML folosind XSLT se poate face folosind un **procesor XSLT**. Majoritatea exemplelor din această secțiune au fost rulate cu procesorul denumit XT (vezi "http://jclark.com/XML"). InternetExplorer 5.x, de la Microsoft, are şi el înglobat un procesor XSLT astfel încât atunci când un document XML ce are ataşată o foie de stil (prin instrucțiunea de procesare "XML-stylesheet") este încărcat, browserul afişează automat rezultatul transformării.

De exemplu, o foaie de stil XSLT tipică este un document de tip "`xsl:stylesheet`" :

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

. . . . . .

</xsl:stylesheet>
```

Observație! Pentru InternetExplorer 5.1, spațiul de nume trebuie să fie:

xmlns:xsl="http://www.w3.org/TR/WD-xsl"

Pentru a prelucra un document XML este necesar să fie citit acel document şi să fie construită o reprezentare internă, un model al documentului (DOM - "Document Object Model", vezi capitolul 11), care pentru XML este un arbore. Acea reprezentare internă este punctul de plecare pentru prelucrările care transformă documentul în ceea ce se doreşte. Ultima activitate este generarea documentului rezultat.

Activitățile de mai sus se pot face în două feluri. O primă modalitate, cea mai uzuală în informatica actuală, este cea **procedurală**. Această abordare începe prin scrierea (sau refolosirea) unui procesor de XML, care generează DOM sau evenimente (vezi capitolul 11) şi dezvoltarea de rutine care efectuează prelucrările dorite. Bineînțeles că este necesară şi o rutină de generare a documentului rezultat.

Abordarea procedurală este foarte bună dar are şi dezavantaje. Un prim mare dezavantaj este că, chiar pentru transformări triviale trebuie scrise rutine sofisticate. Alt dezavantaj îl constituie dificultatea evoluției programelor scrise în limbaje procedurale.

O a doua modalitate de a programa transformările este cea **declarativă**, folosită şi în inteligența artificială. Această modalitate poate fi considerată ca o programare prin exemple [Hol00], cu mult mai simplă decât cea procedurală în cazul de față. Mai mult, gestiunea memoriei şi manipularea primară a documentelor (crearea DOM, traversarea sa şi crearea documentului rezultat) este făcută de procesorul XSLT, programatorului trebuind doar să declare care este transformarea dorită.

XSLT este un limbaj de specificare declarativă a transformărilor documentelor. Transformarea documentelor este realizată prin execuția unor reguli de rescriere care sunt aplicate pe baza unor şabloane care se aplică pe structura arborescentă a documentului inițial. Rezultatul aplicării regulilor este un fragment din arborele documentului rezultat. Se poate spune astfel că regulile XSLT transformă un arbore sursă într-un arbore rezultat. Din această perspectivă, XSLT poate fi văzut ca având o semantică de tip flux [Hol00]. El poate fi asemănat cu limbajele de programare funcțională [Tra01], cu atât mai mult cu cât XSLT nu are efecte laterale. Acest lucru nu este surprinzător dacă amintim că un precursor al XSLT a fost limbajul DSSSL [SGML], destinat formatării şi transformării documentelor SGML, limbaj care este scris în limbajul de programare funcțională Scheme.

Regulile XSLT au două părți:

I. O parte care conține un **şablon** pentru specificarea locului în care se aplică regula. Pentru referirea la anumite noduri din arborele unui document, se foloseşte limbajul XPath (vezi 7.1) pentru indicarea căilor de acces la acele noduri. Nodul astfel referit este **nodul context** pentru elementele din partea a doua a regulii, care descriu fragmentul de arbore rezultat.
II. Efectul aplicării regulii, adică fragmentul de arbore rezultat.

Declararea unei reguli se face cu elementul "`template`":

```
<xsl:template match=". . . sablon . . .">

. . . . . fragment de arbore rezultat

</xsl:template>
```

Fragmentul de arbore rezultat poate fi descris prin elemente XSL specifice, prin elemete HTML sau prin text. De exemplu, un element XSL frecvent folosit este "`value-of`", care precizează că în locul său va fi inserat fragmentul de arbore sursă selectat printr-o expresie XPath din nodul context, ca mai jos (în acest exemplu, fragmentul inserat este chiar nodul context al regulii):

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

Toate construcțiile existente în XSLT sunt specificate în anexa I.

Pentru exemplificările următoare să considerăm documentul "b1.xml" următor:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="b1.xsl" ?>

<bib>
<webdoc src="http://concept.cs.uah.edu/CG/cg-standard.html" abr="CG">
Conceptual Graphs </webdoc>
<webdoc abr="CYC" src="http://www.cyc.org"> CYC</webdoc>
<webdoc abr="DOM" src="http://www.w3.org/DOM"> DOM </webdoc>
<webdoc abr="KIF" src="http://logic.stanford.edu/kif/kif.html"> Knowledge
Interchange Format </webdoc>
```

```
<webdoc abr="KQML" src="http://www.cs.umbc.edu/kqml"> KQML </webdoc>
<webdoc abr="RDF" src="http://www.w3.org/RDF"> RDF Specification
<an>1999</an></webdoc>
<webdoc abr="RDFS" src="http://www.w3.org/TR/WD-rdf-schema"> RDF Schema
Specification</webdoc>
<articol abr="Sho93">
  <autor> Y. Shoham, Agent-oriented programming </autor>
  <revista nr="60">
    <titlu>Artificial Intelligence</titlu>
    <an>1993</an>
    <pp> 51-92</pp>
  </revista>
</articol>
<webdoc abr="XML" src="http://www.w3.org/XML"> XML </webdoc>
<carte abr="XMLc" isbn="1-861003-4-12">
  <autor> Cagle, K. </autor><autor>Gibbons, D.</autor><autor> Hunter,
D.</autor><autor> Ozu, N.</autor><autor>Pinnock, J.
 </autor><autor>Spencer, P.</autor>
 <titlu> Beginning XML </titlu>
 <editura>Wrox Press,</editura>
 <an> 2000</an>
</carte>
</bib>
```

Declarația:

```
<?xml-stylesheet type="text/xsl" href="b1.xsl" ?>
```

asociază documentului XML foaia de stil “b1.xsl”.

Un prim exemplu de foaie de stil (“b1.xsl”) va genera un document HTML care va include conținutul acelui element “webdoc” din documentul sursă care are atributul “abr” cu valoarea “RDF”. În acest scop vom folosi o singură regulă, care se aplică din rădăcina documentului (fapt indicat în XPath prin “/”, vezi secțiunea 7.1).

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

  <xsl:template match="/">
    <html>
     <body>
        <xsl:value-of select="/bib/webdoc[@abr='RDF']"/>
     </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

Rezultatul afişat de un browser de web este:

RDF Specification 1999

Al doilea exemplu de foaie de stil pentru acelaşi document XML, va genera pentru fiecare element “`webdoc`” din documentul “b1.xml” câte un element al unei liste ordonate HTML. Repetiția inserării unui fragment de arbore rezultat pentru o mulțime de noduri selectate printr-o expresie XPath se face cu elementul “`for-each`”:

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

  <xsl:template match="/">
    <html>
```

```
     <body>
      <h2>Paginile de web sunt :</h2>
       <ol>
        <xsl:for-each select="/bib/webdoc">
           <li>
            <xsl:value-of select="."/>
           </li>
        </xsl:for-each>
       </ol>
     </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

Rezultatul obținut este următorul document HTML:

```
<html>
<body>
<h2>Paginile de web sunt :</h2>
<ol>
<li> Conceptual Graphs </li>
<li> CYC</li>
<li> DOM </li>
<li> Knowledge Interchange Format </li>
<li> KQML </li>
<li> RDF Specification  1999</li>
<li> RDF Schema Specification</li>
<li> XML </li>
</ol>
</body>
</html>
```

care este afişat de un browser de web după cum urmează

Paginile de web sunt :

1. Conceptual Graphs
2. CYC
3. DOM
4. Knowledge Interchange Format
5. KQML
6. RDF Specification 1999
7. RDF Schema Specification
8. XML

Un alt exemplu, care transferă în arborele rezultat şi noduri de tip atribut este:

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

  <xsl:template match="/">
    <html>
     <body>
      <h2>Paginile de web sunt :</h2>
       <ol>
        <xsl:for-each select="/bib/webdoc">
           <li>
            -- <xsl:value-of select="@abr"/> --
            <xsl:value-of select="."/>
           </li>
```

```
        </xsl:for-each>
       </ol>
     </body>
    </html>
  </xsl:template>


</xsl:stylesheet>
```

cu documentul HTML rezultat :

```
<html>
<body>
<h2>Paginile de web sunt :</h2>
<ol>
<li>
             -- CG --
              Conceptual Graphs </li>
<li>
             -- CYC --
              CYC</li>
<li>
             -- DOM --
              DOM </li>
<li>
             -- KIF --
              Knowledge Interchange Format </li>
<li>
             -- KQML --
              KQML </li>
<li>
             -- RDF --
              RDF Specification  1999</li>
<li>
             -- RDFS --
              RDF Schema Specification</li>
<li>
             -- XML --
              XML </li>
</ol>
</body>
</html>
```

afişat de un browser astfel :

Paginile de web sunt :

1.-- CG -- Conceptual Graphs
2.-- CYC -- CYC
3.-- DOM -- DOM
4.-- KIF -- Knowledge Interchange Format
5.-- KQML -- KQML
6.-- RDF -- RDF Specification 1999
7.-- RDFS -- RDF Schema Specification
8.-- XML -- XML

Următorul exemplu ilustrează rezultatul aplicării unei foi de stil care conține o regulă care este aplicabilă pentru orice nod de tip text:

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

```
  <xsl:template match="text()">
    <html>
     <body>
       <xsl:value-of select="."/>
     </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Rezultatul este:

```
<html>
<body>
 Conceptual Graphs
 CYC
 DOM
 Knowledge Interchange Format
 KQML
 RDF Specification  1999
 RDF Schema Specification

   Y. Shoham, Agent-oriented programming

    Artificial Intelligence
    1993
     51-92


 XML

   Cagle, K. Gibbons, D. Hunter, D. Ozu, N.Pinnock, J.
 Spencer, P.
  Beginning XML
 Wrox Press,
  2000

</body>
</html>
```

Următorul exemplu a fost folosit pentru a ilustra utilizarea expresiilor XPath, în secțiunea 7.1:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
     <body>
       <p>/bib/webdoc ----->   <xsl:value-of select="/bib/webdoc"/>
       </p>
       <p>//an ----->   <xsl:value-of select="//an"/>
       </p>
       <p>*/*/an ----->   <xsl:value-of select="*/*/an"/>
       </p>
```

```
      <p>/bib/articol/* -----><xsl:value-of select="/bib/articol/*"/>
       </p>
      <p>/bib/webdoc/following-sibling::webdoc ----->
            <xsl:value-of select="/bib/webdoc/following-sibling::webdoc"/>
       </p>
      <p>/bib/webdoc/attribute::* ----->
            <xsl:value-of select="/bib/webdoc/attribute::*"/>
       </p>
      <p>//an/../@abr -----><xsl:value-of select="//an/../@abr"/>
       </p>
      <p>/bib/webdoc[6] -----><xsl:value-of select="/bib/webdoc[6]"/>
       </p>
      <p>/bib/webdoc[4]/following-sibling::* ----->
            <xsl:value-of select="/bib/webdoc[4]/following-sibling::*"/>
       </p>
      <p>/bib/webdoc[@*] -----><xsl:value-of select="/bib/webdoc[@*]"/>
       </p>
      <p>//*[@nr] -----><xsl:value-of select="//*[@nr]"/>
       </p>
      <p>webdoc[@abr='KIF'] -----><xsl:value-of
select="/bib/webdoc[@abr='KIF']"/>
       </p>
      <p>/bib/webdoc[6][@abr] -----><xsl:value-of
select="/bib/webdoc[6][@abr]"/>
       </p>
      <p>/bib/webdoc/text() -----><xsl:value-of
select="/bib/webdoc/text()"/>
       </p>
      <p>/bib/webdoc[last()] -----><xsl:value-of
select="/bib/webdoc[last()]"/>
       </p>
       <p>//*[count(*)>2]/*[2] -----><xsl:value-of
select="//*[count(*)>2]/*[2]"/>
       </p>
</body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Rezultatul este:

```
/bib/webdoc -----> Conceptual Graphs

//an -----> 1999

*/*/an -----> 1999

/bib/articol/* -----> Y. Shoham, Agent-oriented programming

/bib/webdoc/following-sibling::webdoc -----> CYC

/bib/webdoc/attribute::* -----> http://concept.cs.uah.edu/CG/cg-standard.html

//an/../@abr -----> RDF

/bib/webdoc[6] -----> RDF Specification 1999

/bib/webdoc[4]/following-sibling::* -----> KQML

/bib/webdoc[@*] -----> Conceptual Graphs

//*[@nr] -----> Artificial Intelligence 1993 51-92
```

```
webdoc[@abr='KIF'] -----> Knowledge Interchange Format

/bib/webdoc[6][@abr] -----> RDF Specification 1999

/bib/webdoc/text() -----> Conceptual Graphs

/bib/webdoc[last()] -----> XML

//*[count(*)>2]/*[2] -----> CYC
```

O variație a primului exemplu din această secțiune ilustrează faptul că în XSLT pot fi folosite şi variabile (trebuie făcută precizarea că termenul de “variabilă” nu este chiar cel uzual din limbajele de programare). Definirea unei variabile se face cu elementul “`xsl:variable`” iar accesul la valoarea ei se face prin numele ei precedat de caracterul “$”.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
     <body>
       <xsl:variable name="rdf">
        <xsl:value-of select="/bib/webdoc[@abr='RDF']"/>
       </xsl:variable>
       Variabila are valoarea : <xsl:value-of select='$rdf'/>
     </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Rezultatul este:

```
Variabila are valoarea : RDF Specification 1999
```

Bineînțeles că o pagină de stil poate avea mai multe reguli, fiecare fiind aplicabilă pentru un anumit nod din arborele sursă. De exemplu, în foaia de stil de mai jos sunt trei reguli, câte una pentru fiecare tip de element care poate apare în elementul “bib”:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/bib/webdoc">
    <xsl:value-of select="@src"/></xsl:template>
  <xsl:template match="/bib/articol">articol</xsl:template>
  <xsl:template match="/bib/carte">carte</xsl:template>
</xsl:stylesheet>
```

rezultatul fiind:

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

http://concept.cs.uah.edu/CG/cg-standard.html
http://www.cyc.org
http://www.w3.org/DOM
http://logic.stanford.edu/kif/kif.html
http://www.cs.umbc.edu/kqml
http://www.w3.org/RDF
http://www.w3.org/TR/WD-rdf-schema
articol
http://www.w3.org/XML
carte
```

Dacă există mai multe reguli, poate apare un **conflict** între regulile care pot fi aplicate pentru un acelaşi nod al arborelui sursă. Rezolvarea acestui conflict se face pe baza asignării explicite (de către programator) sau implicite (conform unei strategii standard) a unor priorități [XSLT] regulilor. De exemplu, implicit, o regulă care are un test în expresia XPath din şablon ("`/bib/webdoc[@src='KIF']`") are o prioritate mai mică decât una fără test ("`/bib/webdoc`"):

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/bib/webdoc">
    <xsl:value-of select="@src"/>
  </xsl:template>
  <xsl:template match="/bib/articol">articol</xsl:template>
  <xsl:template match="/bib/carte">carte</xsl:template>
  <xsl:template match="/bib/webdoc[@src='KIF']">
  document
  </xsl:template>
</xsl:stylesheet>
```

cu rezultatul:

```
http://concept.cs.uah.edu/CG/cg-standard.html
http://www.cyc.org
http://www.w3.org/DOM
http://logic.stanford.edu/kif/kif.html
http://www.cs.umbc.edu/kqml
http://www.w3.org/RDF
http://www.w3.org/TR/WD-rdf-schema
articol
http://www.w3.org/XML
carte
```

Un alt exemplu este următoarea pagină de stil, la care ultima regulă este preferată celorlalte reguli, deoarece ea nu are specificați şi nişte copii în expresia XPath din şablon:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/bib/webdoc">
    <xsl:value-of select="@src"/>
  </xsl:template>
  <xsl:template match="/bib/articol">articol</xsl:template>
  <xsl:template match="/bib/carte">carte</xsl:template>
  <xsl:template match="/bib">
  document
  </xsl:template>
</xsl:stylesheet>
```

Rezultatul este:

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

  document
```

Lansarea în execuție a unor alte reguli se poate face şi explicit cu "`apply-templates`", care lansează în execuție toate regulile aplicabile copiilor nodului context. De exemplu, mai jos, regula pentru rădăcină lansează regulile pentru copii:

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

  <xsl:template match="/">
    <html>
     <body>
       <h2>Situri de vizitat!</h2>
```

```
       <ul>
       <xsl:apply-templates/>
       </ul>
     </body>
    </html>
  </xsl:template>

  <xsl:template match="webdoc">
    <li>
      <xsl:value-of select="@src"/>
    </li>
  </xsl:template>

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

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

</xsl:stylesheet>
```

Rezultatul este:

```
<html>
<body>
<h2>Situri de vizitat!</h2>
<ul>
<li>http://concept.cs.uah.edu/CG/cg-standard.html</li>
<li>http://www.cyc.org</li>
<li>http://www.w3.org/DOM</li>
<li>http://logic.stanford.edu/kif/kif.html</li>
<li>http://www.cs.umbc.edu/kqml</li>
<li>http://www.w3.org/RDF</li>
<li>http://www.w3.org/TR/WD-rdf-schema</li>

<li>http://www.w3.org/XML</li>

</ul>
</body>
</html>
```

adică:

Situri de vizitat!

- http://concept.cs.uah.edu/CG/cg-standard.html
- http://www.cyc.org
- http://www.w3.org/DOM
- http://logic.stanford.edu/kif/kif.html
- http://www.cs.umbc.edu/kqml
- http://www.w3.org/RDF
- http://www.w3.org/TR/WD-rdf-schema
- http://www.w3.org/XML

Generarea fragmentului de arbore rezultat specificat într-o regulă poate necesita şi crearea de noi elemente folosind construcțiile xsl “`element`” şi “`attribute`”:

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

```
  <xsl:template match="/">
    <html>
     <body>
       <h2>Situri de vizitat!</h2>
       <ul>
       <xsl:apply-templates/>
       </ul>
     </body>
    </html>
  </xsl:template>

  <xsl:template match="webdoc">
    <li>
     <xsl:element name="a">
       <xsl:attribute name="href">
         <xsl:value-of select="@src"/>
       </xsl:attribute>
       <xsl:value-of select="."/>
     </xsl:element>
    </li>
  </xsl:template>

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

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

</xsl:stylesheet>
```

Rezultatul este:

```
<html>
<body>
<h2>Situri de vizitat!</h2>
<ul>
<li>
<a href="http://concept.cs.uah.edu/CG/cg-standard.html"> Conceptual Graphs
</a>
</li>
<li>
<a href="http://www.cyc.org"> CYC</a>
</li>
<li>
<a href="http://www.w3.org/DOM"> DOM </a>
</li>
<li>
<a href="http://logic.stanford.edu/kif/kif.html"> Knowledge Interchange Format
</a>
</li>
<li>
<a href="http://www.cs.umbc.edu/kqml"> KQML </a>
</li>
<li>
<a href="http://www.w3.org/RDF"> RDF Specification  1999</a>
</li>
<li>
<a href="http://www.w3.org/TR/WD-rdf-schema"> RDF Schema Specification</a>
</li>
```

```
<li>
<a href="http://www.w3.org/XML"> XML </a>
</li>

</ul>
</body>
</html>
```

adică:

Regulile vide pentru “articol” şi “carte”, din ultimele exemple, au rolul de a evita declanşarea unor reguli implicite, deoarece, dacă pentru anumite noduri ale arborelui sursă nu sunt precizate reguli, procesorul XSLT aplică astfel de **reguli implicite**. O primă astfel de regulă implicită este următoarea:

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

care este aplicată pe orice nod sau pe rădăcină, dacă nu există o altă regulă aplicabilă. Această regulă implicită asigură apelarea regulilor pentru nodurile descendente, chiar dacă nu există reguli pentru nodul tată.

O altă regulă implicită este aplicată nodurilor text:

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

De exemplu, în următoarea foaie de stil:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/bib/articol">articol</xsl:template>
  <xsl:template match="/bib/carte">carte</xsl:template>
</xsl:stylesheet>
```

chiar dacă nu se specifică o regulă pentru nodul rădăcină al arborelui sursă, se aplică (conform primei reguli implicite) regulile nodurilor descendente (în exemplul nostru, “webdoc”, “articol” şi “carte”). Deoarece, pentru webdoc nu există, de asemenea, reguli, se aplică din nou prima regulă implicită şi, apoi, a doua regulă implicită, care întoarce textul din conținutul nodurilor text, fii ai elementelor “webdoc” :

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

 Conceptual Graphs
 CYC
 DOM
 Knowledge Interchange Format
 KQML
 RDF Specification  1999
 RDF Schema Specification
articol
 XML
carte
```

Acelaşi lucru se întâmplă chiar dacă într-o foaie de stil nu se specifică nici o regulă :

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

adică se întoarce conținutul nodurilor text :

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

 Conceptual Graphs
 CYC
 DOM
 Knowledge Interchange Format
 KQML
 RDF Specification  1999
 RDF Schema Specification

   Y. Shoham, Agent-oriented programming

    Artificial Intelligence
    1993
     51-92


 XML

   Cagle, K. Gibbons, D. Hunter, D. Ozu, N.Pinnock, J.
 Spencer, P.
  Beginning XML
 Wrox Press,
  2000
```

Eliminarea regulilor vide se poate face şi prin direcționarea apelului regulilor din “`apply-templates`”, prin specificarea câmpului “`select`”, ca în exemplul următor:

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

```

```
  <xsl:template match="/">
    <html>
     <body>
       <h2>Situri de vizitat!</h2>
       <ul>
       <xsl:apply-templates select="/bib/webdoc"/>
       </ul>
     </body>
    </html>
  </xsl:template>

  <xsl:template match="webdoc">
    <li>
     <xsl:element name="a">
       <xsl:attribute name="href">
         <xsl:value-of select="@src"/>
       </xsl:attribute>
       <xsl:value-of select="."/>
     </xsl:element>
    </li>
  </xsl:template>

</xsl:stylesheet>
```

Rezultatul este:

```
<html>
<body>
<h2>Situri de vizitat!</h2>
<ul>
<li>
<a href="http://concept.cs.uah.edu/CG/cg-standard.html"> Conceptual Graphs
</a>
</li>
<li>
<a href="http://www.cyc.org"> CYC</a>
</li>
<li>
<a href="http://www.w3.org/DOM"> DOM </a>
</li>
<li>
<a href="http://logic.stanford.edu/kif/kif.html"> Knowledge Interchange Format
</a>
</li>
<li>
<a href="http://www.cs.umbc.edu/kqml"> KQML </a>
</li>
<li>
<a href="http://www.w3.org/RDF"> RDF Specification  1999</a>
</li>
<li>
<a href="http://www.w3.org/TR/WD-rdf-schema"> RDF Schema Specification</a>
</li>
<li>
<a href="http://www.w3.org/XML"> XML </a>
</li>
</ul>
</body>
</html>
```

O extindere a exemplului anterior, care listează şi articolele şi cărțile este:

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

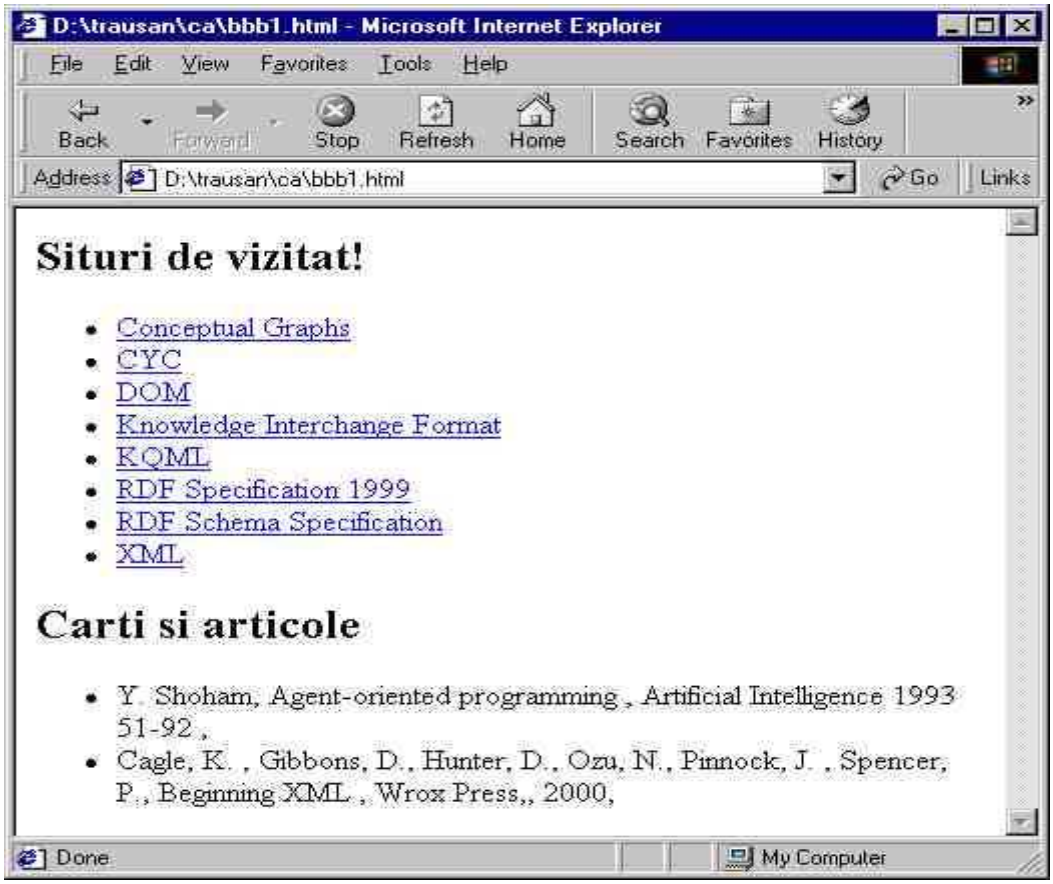
```
  <xsl:template match="/">
    <html>
     <body>
       <h2>Situri de vizitat!</h2>
       <ul>
         <xsl:apply-templates select="/bib/webdoc"/>
       </ul>
       <h2>Carti si articole</h2>
       <ul>
         <xsl:apply-templates select="/bib/articol|/bib/carte"/>
       </ul>
     </body>
    </html>
  </xsl:template>

  <xsl:template match="webdoc">
    <li>
     <xsl:element name="a">
       <xsl:attribute name="href">
         <xsl:value-of select="@src"/>
       </xsl:attribute>
       <xsl:value-of select="."/>
     </xsl:element>
    </li>
  </xsl:template>

  <xsl:template match="articol|carte">
    <li>
      <xsl:for-each select="./*">
         <xsl:value-of select="."/>,
       </xsl:for-each>
    </li>
  </xsl:template>

</xsl:stylesheet>
```

cu rezultatul:

```
<html>
<body>
<h2>Situri de vizitat!</h2>
<ul>
<li>
<a href="http://concept.cs.uah.edu/CG/cg-standard.html"> Conceptual Graphs
</a>
</li>
<li>
<a href="http://www.cyc.org"> CYC</a>
</li>
<li>
<a href="http://www.w3.org/DOM"> DOM </a>
</li>
<li>
<a href="http://logic.stanford.edu/kif/kif.html"> Knowledge Interchange Format
</a>
</li>
<li>
<a href="http://www.cs.umbc.edu/kqml"> KQML </a>
</li>
<li>
<a href="http://www.w3.org/RDF"> RDF Specification  1999</a>
</li>
```

```
<li>
<a href="http://www.w3.org/TR/WD-rdf-schema"> RDF Schema Specification</a>
</li>
<li>
<a href="http://www.w3.org/XML"> XML </a>
</li>
</ul>
<h2>Carti si articole</h2>
<ul>
<li> Y. Shoham, Agent-oriented programming ,

    Artificial Intelligence
    1993
     51-92
  ,
      </li>
<li> Cagle, K. ,
      Gibbons, D.,
       Hunter, D.,
       Ozu, N.,
      Pinnock, J.
 ,
      Spencer, P.,
       Beginning XML ,
      Wrox Press,,
       2000,
      </li>
</ul>
</body>
</html>
```

vizualizat după cum urmează:



Regulile pot fi invocate şi prin apel explicit, prin “call-template”. Apelul poate avea şi parametrii, specificați prin “with-param”. În această situație, regula trebuie să aibă un nume şi să conțină elemente “param” pentru parametrii. De exemplu, acelaşi rezultat ca mai sus se obține cu următoarea foaie de stil:

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

  <xsl:template match="/">
    <html>
     <body>
       <xsl:call-template name="1">
           <xsl:with-param name="titlu">
             Situri de vizitat!
           </xsl:with-param>
           <xsl:with-param name="adr" select="bib/webdoc"/>
       </xsl:call-template>
       <xsl:call-template name="1">
           <xsl:with-param name="titlu">
             Carti si articole
           </xsl:with-param>
           <xsl:with-param name="adr" select="/bib/articol|/bib/carte"/>
       </xsl:call-template>
     </body>
    </html>
  </xsl:template>

  <xsl:template name="1">
   <xsl:param name="titlu"/>
   <xsl:param name="adr"/>
    <h2>
     <xsl:value-of select="$titlu" />
    </h2>
    <ul>
      <xsl:apply-templates select="$adr"/>
    </ul>
  </xsl:template>

  <xsl:template match="webdoc">
    <li>
     <xsl:element name="a">
       <xsl:attribute name="href">
         <xsl:value-of select="@src"/>
       </xsl:attribute>
       <xsl:value-of select="."/>
     </xsl:element>
    </li>
  </xsl:template>

  <xsl:template match="articol|carte">
    <li>
      <xsl:for-each select="./*">
         <xsl:value-of select="."/>,
      </xsl:for-each>
    </li>
  </xsl:template>

</xsl:stylesheet>
```

Elementul “output” poate specifica prin atributul “method” că rezultatul transformării nu este un fişier HTML (cum a fost cazul până acum, fapt implicit datorită prezenței elementului “<html>”). Valoarea acestui atribut poate fi “text”, “HTML” sau “XML”. În cele ce urmează este dat un exemplu de foaie de stil care generează text:

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

<xsl:output method="text" indent="yes"/>

  <xsl:template match="webdoc">
      <xsl:value-of select="@src"/>
  </xsl:template>

</xsl:stylesheet>
```

cu rezultatul următor :

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

http://concept.cs.uah.edu/CG/cg-standard.html
http://www.cyc.org
http://www.w3.org/DOM
http://logic.stanford.edu/kif/kif.html
http://www.cs.umbc.edu/kqml
http://www.w3.org/RDF
http://www.w3.org/TR/WD-rdf-schema

   Y. Shoham, Agent-oriented programming

    Artificial Intelligence
    1993
     51-92


http://www.w3.org/XML

   Cagle, K. Gibbons, D. Hunter, D. Ozu, N.Pinnock, J.
 Spencer, P.
  Beginning XML
 Wrox Press,
  2000
```

În sfârşit, dăm un exemplu mai complex: Fie urmatorul fişier, “avl.xml” (dăm aici, din motive de spațiu, doar fragmente din el):

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="avl.xsl" ?>
<!DOCTYPE lectie SYSTEM "file:avl.dtd">

<!--DOCTYPE lectie SYSTEM "c:/users/stefan/xml/ca/avl.dtd" -->

<lectie nume="Arbori_AVL">
<titlu> Arbori_AVL </titlu>
<subiect nume="AVL">
<concept>Arborii AVL</concept>, denumiti asa dupa cei care i-au imaginat
pentru prima oara, sunt arbori binari de cautare ce indeplinesc, suplimentar,
un anumit criteriu de echilibru. Acest criteriu impune ca inaltimile fiilor
oricarui nod din arbore sa difere prin maxim 1. Mai formal, trebuie
indeplinita relatia,
<proprietate nr='1'>|h(Sl(n)) - h(Sr(n))| &lt;= 1, pentru oricare nod, n, din
arbore.
</proprietate>Vom numi in cele ce urmeaza un arbore AVL, arbore echilibrat.
</subiect>
```

```
<subiect nume="propravl">
Este evident ca un arbore cu o asemenea proprietate este foarte aproape de
idealul de echilibru, si va avea timpi de cautare foarte buni. S-a aratat ca
numarul de comparatii de chei in procesul de cautare indeplineste relatia,
<proprietate nr='2'>    Nc &lt;= 1.44 log2 (n + 2), unde n este numarul de
noduri din arbore.
</proprietate>
<demonstratie>
Demonstratia afirmatiei de mai sus se face remarcabil de simplu, si de aceea o
vom prezenta.
Este evident ca pentru a face procesul de cautare cit mai lung, numarul de
noduri fiind dat, n, trebuie ca arborele AVL sa aiba inaltimea cit mai mare
posibil. Sa vedem cum arata un astfel de arbore. Vom nota cu Ai arborele AVL
de inaltime i cu numar minim de noduri.
Pentru n = 0, A0 este in mod evident arborele vid.
Pentru n = 1, A1 este un arbore ce contine doar o cheie.
Pentru n = 2, A2 este un arbore cu 2 chei, care poate fi desenat
ca mai jos,

. . . . . .

si deci,

      Nc &lt;= 1.44 log2 (n+2)

</demonstratie>
</subiect>

<subiect nume="factech">

In reprezentarea arborilor AVL vom asocia fiecarui nod un <concept>factor de
echilibru</concept>, care pentru un arbore corect poate avea trei valori
posibile,
<proprietate nr='3'>
      +    DACA  h(Sr(n)) = h(Sl(n)) + 1 ;
      .    DACA  h(Sr(n)) = h(Sl(n)) ;
      -    DACA  h(Sr(n)) = h(Sl(n)) - 1
</proprietate>
</subiect>
<subiect nume="pivot">
O notiune extrem de importanta pentru acesti arbori, este aceea de
<concept>pivot</concept>. Sa presupunem ca inseram o cheie. Insertia o facem
ca la un arbore binar de cautare obisnuit. Este posibil ca in urma insertiei
arborele sa se dezechilibreze, adica intr-un anumit nod, criteriul de
echilibru sa nu mai fie respectat. In figura de mai jos este dat un astfel de
exemplu.

. . . . . .

</subiect>

<subiect nume="impl">
Implementare
<concept>Rotire</concept>
Cautarea intr-un arbore AVL se face la fel ca intr-un arbore binar de cautare
obisnuit, ignorind factorul de echilibru. Structura unui nod impreuna cu
functia de inserare in nod este prezentata mai jos,
<prog>
<![CDATA[/* avl-tree.h */

typedef unsigned char Balance;
typedef enum { Left, Right, Equal } Tree_balance;

typedef struct tag_AVL_Tree
{
  Tree_balance balance;
```

```
  int key;
  struct tag_AVL_Tree *l, *r;

} AVL_Tree;

. . . . . .

  a->l = c->r;
  b->r = c->l;
  c->balance = Equal;
  c->r = a;
  c->l = b;
  return c;
}
]]>
</prog>
</subiect>
</lectie>
```

cu DTD-ul:

```
<!-- XML DTD for LECTII -->
<!-- Last Mod: 3/11/2000 -->

<!ELEMENT lectie (titlu,(subiect)*) >
<!ATTLIST lectie
    nume ID #IMPLIED>
<!ELEMENT subiect ( concept | proprietate | fig | demonstratie | prog )*>
<!ATTLIST subiect
     nume ID #REQUIRED>
<!ELEMENT fig EMPTY>
<!ATTLIST fig
     nr ID #REQUIRED
     caption CDATA #REQUIRED>
<!ELEMENT titlu (#PCDATA)>
<!ELEMENT concept (#PCDATA)>
<!ELEMENT proprietate (#PCDATA)>
<!ELEMENT demonstratie (#PCDATA | concept | proprietate | fig)*>
<!ELEMENT prog (#PCDATA)>
<!ATTLIST proprietate
     nr ID #REQUIRED
     nume CDATA #IMPLIED>
```

Pentru foaia de stil:

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

  <xsl:template match="/">
    <html>
     <body>
      <h2>Conceptele prezentate sunt :</h2>
       <ol>
        <xsl:for-each select="/lectie/subiect/concept">
            <xsl:apply-templates select="." mode="1"/>
        </xsl:for-each>
       </ol>
      <h2>Proprietatile prezentate sunt :</h2>
       <ol>
        <xsl:for-each select="/lectie/subiect/proprietate">
            <xsl:apply-templates select="." mode="2"/>
        </xsl:for-each>
       </ol>
     </body>
```

```
    </html>
  </xsl:template>

  <xsl:template match="concept" mode="1">
    <li>
     <xsl:value-of select="."/>
    </li>
  </xsl:template>

  <xsl:template match="proprietate" mode="2">
    <li>
       <xsl:value-of select="."/>
    </li>
  </xsl:template>

</xsl:stylesheet>
```

rezultatul afişat va fi:

Conceptele prezentate sunt :

1.Arborii AVL
2.factor de echilibru
3.pivot
4.Rotire

Proprietatile prezentate sunt :

1.|h(Sl(n)) - h(Sr(n))| <= 1, pentru oricare nod, n, din arbore.
2.Nc <= 1.44 log2 (n + 2), unde n este numarul de noduri din arbore.
3.+ DACA h(Sr(n)) = h(Sl(n)) + 1 ; . DACA h(Sr(n)) = h(Sl(n)) ; - DACA h(Sr(n)) = h(Sl(n)) - 1
4.primul nod, pornind de la frunza inserata spre radacina, care este dezechilibrat

9. Conexiuni între documente; limbajul XLink

Limbajul XLink permite generalizarea posibilităților oferite de HTML pentru definirea de legături între documente. În HTML, legăturile erau uni-direcționale, între un document care conținea punctul de plecare al legăturii (definit prin “<a href= >”) şi un document țintă (ca un întreg sau un punct în document, definit prin “<a name= >”).

În XLink legăturile pot fi între mai mult de două resurse, resursele pot fi nu numai documente, direcțiile de traversare pot fi multiple (nu mai sunt uni-direcționale, ca la HTML) iar specificarea detaliilor legăturilor poate fi făcută într-un fişier separat.

Specificarea legăturilor în XLink se face folosind documente XML bine formate, pe baza unui repertoriu de atribute, după cum urmează:

- “type” este un atribut care specifică **tipul** legăturii. El poate avea ca valoare:
    - “simple”, pentru legăturile simple, între două resurse.
    - “extended”, pentru legături extinse. Pentru acest tip de legături se pot defini elemente pentru specificări suplimentare, conform următoarelor patru atribute:
    - “locator”, pentru resurse externe,
    - “arc”, pentru specificarea direcțiilor de traversare a legăturilor,
    - “resource”, pentru specificarea de resurse locale,
    - “title”, pentru a specifica un element pentru titlu (în locul atributului cu acelaşi nume, de care se va vorbi un pic mai jos).
- atribute referitoare la **semantica** legăturii:
    - “role”, etichetă care
    - “title”, numele care descrie legătura.
- atribute referitoare la **comportarea** legăturii:
    - “actuate”, precizează când va căutată noua resursă. Poate avea ca valoare:
        - “onLoad”, similar cu “<img>” din HTML,
        - “onRequest”, similar cu “<a href=. . .>”,
        - “undefined” - specific aplicației.
    - “show”, precizează cum va fi afişată noua resursă în urma parcurgerii unei legături. Poate avea ca valoare modalitatea de afişare:
        - “new”, într-o fereastră separată;
        - “replace”, prin înlocuire, ca la “<a href=. . .>” din HTML;
        - “embed”, ca la “<img>” din HTML;
        - “undefined”
- atribute care descriu **direcționalitatea**:
    - “from”, specifică direcția legăturii.
    - “to”, specifică direcția legăturii.

Pentru atributele Xlink există următoarele situații în care sunt opționale (O) sau obligatorii (R) [XLink] :

	Simple	extended	locator	arc	resource	title
Type	R	R	R	R	R	R
Href	O		R			
Role	O	O	O		O	
Arcrole	O			O		
Title	O	O	O	O	O	
Show	O			O		
Actuate	O			O		
Label			O		O	
From				O		
To				O		

Şi următoarele relații [XLink] :

Tipul părintelui	**Tipuri de copii**
Simple	Nu are
Extended	Locator, arc, resource, title
Locator	Title
Arc	Title
Resource	Nu are
Title	Nu are

Dăm în continuare un exemplu (adaptat din [XLink]) care ilustrează nişte legături extinse între studentul “Ion Ion” şi cursurile sale, indrumatorul proiectului de diploma, colegi etc.:

```
<cursuri xlink:title="cursurile lui Ion Ion">

    <persoana
      xlink:href="studenti/ionion231.xml"
      xlink:label="student231"
      xlink:role="http://www.example.com/linkprops/student"
      xlink:title="Ion Ion" />

    <persoana
      xlink:href="profi/dandan21.xml"
      xlink:label="prof21"
      xlink:role="http://www.example.com/linkprops/professor"
      xlink:title="Dr. Dan Dan" />

    <!-- alte resurse referitoare la profesori -->

    <curs
      xlink:href="cursuri/ie23.xml"
      xlink:label="IE-23"
      xlink:title="Interfete evoluate 23" />

    <!-- alte resurse despre cursuri -->

    <mergi
      xlink:from="student231"
      xlink:to="prof21"
      xlink:show="new"
      xlink:actuate="onRequest"
      xlink:title="Conducatorul proiect de diploma" />
```

```
    <mergi
      xlink:from="IE-23"
      xlink:arcrole="http://www.example.com/linkprops/auditor"
      xlink:to="student231"
      xlink:show="replace"
      xlink:actuate="onRequest"
      xlink:title="Scrie notele de curs" />
    <mergi
      xlink:from="student231"
      xlink:arcrole="http://www.example.com/linkprops/advisor"
      xlink:to="student228"
      xlink:show="replace"
      xlink:actuate="onRequest"
      xlink:title="Coleg proiect" />

  </cursuri>
```

10. Analiza documentelor XML

Pentru a prelucra documentele XML, adică pentru a le analiza şi pentru a da acces la structura şi continutul lor o **aplicaţie** trebuie să folosească un program numit **procesor XML**.

Procesoarele XML pot fi **cu sau fără validare**. Amândouă tipurile de procesoare trebuie să raporteze violările regulilor documentelor bine formate. Procesoarele cu validare trebuie să raporteze, în plus, violările constrângerilor date în DTD. Pentru a face acest lucru, procesoarele XML trebuie să citească şi să proceseze tot DTD-ul documentului şi toate entitățile externe spre care se fac referiri în document.

Procesoarele XML mai pot fi clasificate în două clase :

- procesoare XML care construiesc arborele documentului (modelul documentului) pentru a-l pasa aplicației beneficiar;
- procesoare XML bazate pe evenimente, care parcurg documentele XML şi generează evenimente la debutul unei adnotări, evenimente preluate de proceduri de tratare din aplicația beneficiar.

Un model al documentului independent de o anumită structurare este obținută prin intercalarea unui obiect “NodeList” între fiecare nod şi nodurile corespunzătoare elementelor sale fiu. Această idee este cea folosită de DOM (“Document Object Model” - [DOM]).

DOM furnizează, de fapt, interfața pentru aplicațiile care preiau informațiile generate de procesorul XML. De exemplu, o parte din DOM pentru XML [DOM], conform OMG IDL este:

```
interface Document : Node {
  readonly attribute  DocumentType         doctype;
  readonly attribute  DOMImplementation    implementation;
  readonly attribute  Element              documentElement;
  Element                   createElement(in DOMString tagName)
                                         raises(DOMException);
  DocumentFragment          createDocumentFragment();
  Text                      createTextNode(in DOMString data);
  Comment                   createComment(in DOMString data);
  CDATASection              createCDATASection(in DOMString data)
                                           raises(DOMException);
  ProcessingInstruction     createProcessingInstruction(in DOMString target,
                                                        in DOMString data)
                                                        raises(DOMException);
  Attr                      createAttribute(in DOMString name)
                                          raises(DOMException);
  EntityReference           createEntityReference(in DOMString name)
                                               raises(DOMException);
  NodeList                  getElementsByTagName(in DOMString tagname);
};

interface Node {
  // NodeType
  const unsigned short      ELEMENT_NODE       = 1;
  const unsigned short      ATTRIBUTE_NODE     = 2;
  const unsigned short      TEXT_NODE          = 3;
  const unsigned short      CDATA_SECTION_NODE = 4;
  const unsigned short      ENTITY_REFERENCE_NODE = 5;
  const unsigned short      ENTITY_NODE        = 6;
  const unsigned short      PROCESSING_INSTRUCTION_NODE = 7;
  const unsigned short      COMMENT_NODE       = 8;
  const unsigned short      DOCUMENT_NODE      = 9;
  const unsigned short      DOCUMENT_TYPE_NODE = 10;
  const unsigned short      DOCUMENT_FRAGMENT_NODE = 11;
  const unsigned short      NOTATION_NODE      = 12;
```

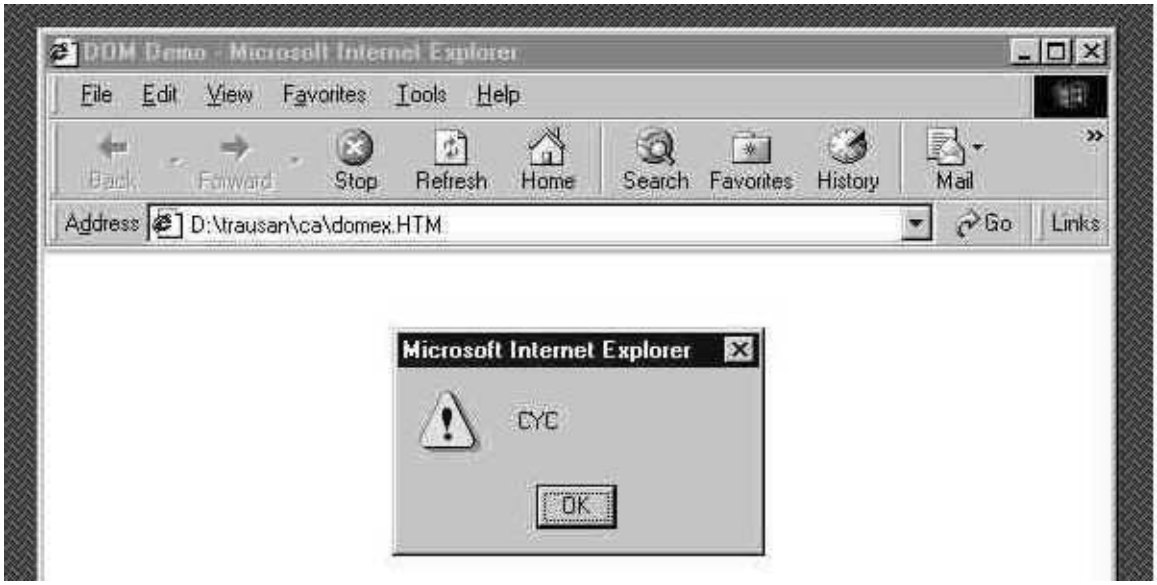
```
  readonly attribute  DOMString           nodeName;
           attribute  DOMString           nodeValue;
                                                 // raises(DOMException) on setting
                                                 // raises(DOMException) on retrieval
  readonly attribute  unsigned short      nodeType;
  readonly attribute  Node                parentNode;
  readonly attribute  NodeList            childNodes;
  readonly attribute  Node                firstChild;
  readonly attribute  Node                lastChild;
  readonly attribute  Node                previousSibling;
  readonly attribute  Node                nextSibling;
  readonly attribute  NamedNodeMap        attributes;
  readonly attribute  Document            ownerDocument;
  Node                      insertBefore(in Node newChild,
                                         in Node refChild)
                                         raises(DOMException);
  Node                      replaceChild(in Node newChild,
                                         in Node oldChild)
                                         raises(DOMException);
  Node                      removeChild(in Node oldChild)
                                        raises(DOMException);
  Node                      appendChild(in Node newChild)
                                        raises(DOMException);
  boolean                   hasChildNodes();
  Node                      cloneNode(in boolean deep);
};

interface NodeList {
  Node                      item(in unsigned long index);
  readonly attribute  unsigned long       length;
};
```

Mai jos este dat un exemplu de utilizare a DOM (adaptat din [XMLc]) construit de procesorul XML, folosit de Microsoft pentru InternetExplorer 5.x (“MSXML”). Accesul la DOM este obținut printr-un “ActiveX” ("MSXML.DOMDocument").

În exemplu, conținutul celui de-al doilea element de tip “webdoc” din documentul “b1.xml” este afişat într-o fereastră de atenționare (generată cu “alert”). Sunt utilizate atributele “firstChild” şi “nodeValue” din interfata pentru un nod. Primul din ele este cel corespunzător elementului al doilea (obținut cu metoda “item”) din lista nodurilor (“NodeList”) obținute cu “getElementsByTagName” (pentru “webdoc”) din interfata pentru document :

```
<HTML>
<HEAD><TITLE>DOM Demo</TITLE>

<SCRIPT language="JavaScript">

  var objDOM;
  objDOM = new ActiveXObject("MSXML.DOMDocument");
  objDOM.load("b1.xml");

  var objNodeList;
  objNodeList = objDOM.getElementsByTagName("webdoc");
  alert(objNodeList.item(1).firstChild.nodeValue);

</SCRIPT>

</HEAD>
<BODY>
  <P>Exemplu.</P>
</BODY>
</HTML>
```

Rezultatul încărcării paginii pe InternetExplorer este :



Detalii despre alte procesoare XML se pot obține la adresa http://jclark.com/XML.

11. Bibliografie

[BaT87] Barbuceanu, M., Trăuşan-Matu, Ştefan , Integrating Declarative Knowledge Programming Styles and Tools in a Structured Object Environment, in J. Mc.Dermott (ed.) Proceedings of 10-th International Joint Conference on Artificial Intelligence IJCAI'87, Italia, Morgan Kaufmann Publishers, Inc., 1987.

[CG] Conceptual Graphs, http://concept.cs.uah.edu/CG/cg-standard.html

[CYC] www.cyc.org

[DOM] http://www.w3.org/DOM

[Hol00] G. Ken Holman, The Context of XSL Transformations and the XML Path Language, http://www.xml.com/pub/2000/08/holman/s1.html

[KIF] Knowledge Interchange Format, http://logic.stanford.edu/kif/kif.html

[KQML] http://www.cs.umbc.edu/kqml

[Lar] LarFLaST, http://www-it.fmi.uni-sofia.bg/larflast/

[OML] Ontology Markup Language, http://wave.eecs.wsu.edu/CKRMI/OML.html, http://www.oasis-open.org/cover/oml9808.html

[RDF] RDF Specification, http://www.w3.org/RDF, 1999.

[RDFS] RDF Schema Specification, W3C, http://www.w3.org/TR/WD-rdf-schema, 1999.

[SGML] SGML web page, http://www.sil.org/sgml/

[Sho93] Y. Shoham, Agent-oriented programming, Artificial Intelligence 60, 1993, pp. 51-92.

[SHOE] SHOE http://www.cs.umd.edu/projects/plus/SHOE/

[Tra99a] Trăuşan-Matu, Ştefan, Web Page Generation Facilitating Conceptualization and Immersion for Learning Finance Terminology, *Proceedings of RILW99,* http://rilw.emp.paed.uni-muenchen.de/99/papers/Trăuşan.html

[Tra99b] Trăuşan-Matu, Ştefan, Produs program pentru generare dinamica de pagini de Web într-un sistem de instruire inteligent, RACAI Report RR-47.

[Tra00a] Trăuşan-Matu, Ştefan, Interfațarea evoluată om-calculator, Ed. MatrixRom, 2000.

[Tra00b] Trăuşan-Matu, Ştefan, Intelligent personalizing web pages and understanding facilities, in S.A.Cerri, D.Maraschi (eds.), *Proceedings of WITREC-2000*, Montpelier, France, pp. 59-68, http://www.lirmm.fr/WITREC

[Tra00c] Trăuşan-Matu, Ştefan, Metaphor Processing for Learning Terminology on the Web, in S.A.Cerri (ed.), Artificial Intelligence, Methodology, Systems, Applications 2000, Springer Verlag, 2000.

[Tra01] Trăuşan-Matu, Ştefan, Common-Lisp, limbajul inteligenței artificiale, în curs de apariție.

[XHTML] http://www.w3.org/TR/xhtml1

[XLink] http://www.w3.org/TR/xlink/

[XML] http://www.w3.org/XML

[XMLc] Cagle, K., Gibbons, D., Hunter, D., Ozu, N., Pinnock, J., Spencer, P., Beginning XML, Wrox Press, 2000

[XMLr] Recomandarea XML, http://www.w3.org/TR/2000/REC-xml-20001006

[XPath] http://www.w3.org/TR/xpath

[XPath99] http://www.w3.org/TR/1999/REC-xpath-19991116

[XPointer] http://www.w3.org/TR/xptr

[XSL] http://www.w3.org/XSL

[XSLT] http://www.w3.org/TR/xslt

[XSLT99] http://www.w3.org/TR/1999/REC-xslt-19991116

[XSLT00] http://www.xml.com/pub/2000/08/holman/s1_1_3.html