Sabin Corneliu Buraga

Semantic Web

fundamente și aplicații

Web, metadate, ontologii, RDF, OWL, servicii Web semantice, agenți Web

MATRIX ROM

Sabin Corneliu Buraga

Semantic Web

Fundamente şi aplicaţii

2004

În memoria bunicilor noştri.

Cuprins succint

Cuprins

Mulţumiri

Această lucrare nu ar fi ajuns la forma actuală fără suportul venit din partea *prof. dr. Dumitru Todoroi* – conducătorul ştiinţific al tezei de doctorat –, *prof. dr. Toader Jucan*, *prof. dr. Dan Grigoraş*, *prof. dr. Dan Cristea*, *prof. dr. Cristian Masalagiu* şi *conf. dr. Dorel Lucanu* de la Facultatea de Informatică a Universităţii "Al. I. Cuza" din Iaşi şi din partea *prof. dr. Ştefan Trăuşan-Matu* de la Universitatea Politehnică din Bucureşti.

Îi menţionăm şi pe colaboratorii şi prietenii apropiaţi *ing. Dragoş Acostăchioaie* (Biosfarm Iaşi S.R.L.), *cercet. drd. Lenuţa Alboaie* (Institutul de Informatică Teoretică al Academiei Române – filiala Iaşi), *cercet. Sînică Alboaie* (Institutul de Informatică Teoretică al Academiei Române – filiala Iaşi), *asist. drd. Mihaela Brut* (Facultatea de Informatică din Iaşi) şi *lect. drd. Marius Cioca* (Facultatea de Inginerie a Universităţii "L. Blaga" din Sibiu) pentru ajutorul acordat pe parcursul etapelor de elaborare a conţinutului acestui material.

Nu-i uităm nici pe absolvenţii *Petrică Găbureanu* şi *Victor Grigoriu*, mulţumindu-le pentru interesul acordat Web-ului semantic, dar mai ales pentru unele contribuţii la volumul de faţă. Exprimăm gratitudinea noastră studenţilor *Florin Bandas* şi *Adrian Mironescu* pentru ajutorul oferit la finalizarea acestei cărţi.

De asemenea, autorul este recunoscător profesorilor *Dr. Marcin Paprzycki* (Computer Science Department, Oklahoma State University, USA) şi *Dr. Ştefan Andrei* (Facultatea de Informatică din Iaşi) pentru facilitarea accesului la unele resurse bibliografice şi pentru comentariile deosebit de utile privitoare la conţinutul lucrării.

Listă de tabele

Listă de figuri

Preambul

Dacă omul ar putea călători fără întrerupere toată viaţa,
de la naştere până la moarte, timpul ar ajunge sinonim
cu spaţiul străbătut de paşii săi.
Bergson

Prezentare generală

TEMA principală a lucrării de faţă se înscrie pe coordonatele regăsirii informaţiilor hipermedia în spaţiul World-Wide Web. În vederea îndeplinirii acestui deziderat, pe parcursul materialului se prezintă un model original utilizat pentru specificarea relaţiilor spaţio-temporale dintre resursele multimedia ale unui (fragment de) sit Web, model circumscris problematicilor actuale ale Web-ului semantic.

La nivel abstract, s-a recurs la formalizări bazate pe logica temporală cu intervale – *ITL* (*Interval Temporal Logic*) [Allen, 1991]. Modelarea relaţiilor spaţiale are în vedere modul de stocare a acestora în cadrul unui sistem de fişiere distribuit, luându-se în consideraţie şi posibilele metadate care pot fi asociate resurselor (*e.g.*, drepturi de acces, tip, proprietar etc.).

Pentru a asigura independenţa de platformă şi alinierea la principalele standarde actuale ale Consorţiului Web, maniera de stocare a informaţiilor realizându-se prin crearea unor limbaje bazate pe metalimbajul *XML* (*Extensible Markup Language*) [Bray *et al.*, 2004]. Acestea sunt menite a fi integrate în aserţiuni *RDF* (*Resource Description*

Framework) [Beckett, 2004] – cadru oferind interoperabilitate aplicaţiilor distribuite care realizează schimb inteligent de informaţii, în sensul interpretării de către maşină a semanticii acestora. Din acest punct de vedere, problematica dezbătută poate fi considerată ca fiind aliniată *Web-ului semantic* [Berners-Lee *et al.*, 2001, Davies *et al.*, 2003].

Din moment ce resursele multimedia pot fi descrise şi pot fi interconectate prin relaţii spaţio-temporale, există posibilitatea căutării şi regăsirii lor, într-o manieră asemănătoare celei adoptate de motoarele de căutare actuale [Brin & Page, 1998, Chakrabarti *et al.*, 1999]. Lucrarea propune utilizarea descrierilor RDF ale metadatelor asociate resurselor, cu concursul relaţiilor stabilite între documentele multimedia.

Una dintre etapele importante ale procesului de căutare este cea a procesării interogărilor complexe formulate de utilizatori. Din acest punct de vedere, cercetările întreprinse se concentrează asupra exprimării interogărilor prin intermediul unui limbaj bazat pe XML, incluzând în plus unele informaţii privind structura, conţinutul şi relaţiile pe care le pot avea documentele multimedia căutate cu alte resurse.

Lucrarea propune o soluţie originală de implementare, concretizată în platforma *ITW*, bazată pe componente distribuite eterogene, reprezentate din agenţi [Bradshow, 1997, Luck, McBurney & Preist, 2003], servicii Web (semantice) [Curbera *et al.*, 2002] şi alte entităţi programabile. Aceste componente software, independente de platformă şi dezvoltate deschis, pot fi integrate în platforme de tip Grid [Buyya, 2002], cu utilizări în regăsirea informaţiilor multimedia în cadrul organizaţiilor virtuale existând în Internet.

Structura lucrării

Vom prezenta în continuare structura generală a cărţii:

Capitolul 1 descrie arhitectura spaţiului WWW, trecând în revistă principalele concepte pe care se bazează: *hipertextul* [Nielsen, 1990], *identificatorii uniformi de resurse* [Berners-Lee *et al.*, 1998] şi *limbajele de adnotare* [Buraga, 2001a, Trăuşan-Matu, 2001]. De asemenea,

se ilustrează evoluţia – mai ales din punctul de vedere al familiei de limbaje XML – a Web-ului din prisma sintaxei, stilului de prezentare, structurii şi semanticii conţinutului resurselor Web, urmându-se în principal liniile expuse în [Buraga, 2002a] şi [Alboaie & Buraga, 2002].

În capitolul 2 se realizează o prezentare generală a problematicilor legate de Web-ul semantic, insistându-se asupra RDF – conform cu cele detaliate de [Beckett, 2004], [Brickley & Guha, 2000] şi [Hayes, 2004] – şi a descrierii ontologiilor. Unele exemplificări recurg la o serie de contribuţii proprii legate de modelarea relaţiilor dintre resursele unui sistem de teleconferinţe [Buraga, 1998, Buraga, 2001b, Buraga, 2001d] sau ale componentelor unei platforme de tip *e-learning* [Buraga, 2001c, Buraga, 2003e].

Următorul capitol detaliază cercetările privitoare la descrierea şi regăsirea resurselor multimedia disponibile în Internet. Acest capitol este divizat în două mari părţi:

- Prima parte a capitolului prezintă o serie de modele teoretice utilizate pentru descrierea proprietăţilor temporale ale sistemelor distribuite, focalizându-se asupra logicii ITL (a se vedea secţiunea 3.2.3). Pe baza acestui formalism, se vor putea exprima relaţiile spaţio-temporale dintre resursele Web.

  În vederea descrierii resurselor distribuite în spaţiul WWW, se pleacă de la reprezentarea RDF a proprietăţilor părţilor componente ale unui sistem de fişiere distribuit, prin intermediul unui limbaj propriu bazat pe XML – *XFiles* – descris în [Buraga, 2000a], [Buraga, 2002b] şi [Buraga, 2003d]. Extinzând modelul la Web, se pot exprima relaţiile spaţiale stabilite între diverse resurse ale unui Web local (*e.g.*, intranetul unei organizaţii). Resursele temporale vor putea fi specificate cu concursul unui alt limbaj – *TRSL* (*Temporal Relation Specification Language*) [Buraga, 2002c] [Buraga & Ciobanu, 2002] – care rescrie în termenii XML relaţiile formale ale logicii ITL. Acest limbaj este suficient de flexibil pentru a da posibilitatea asocierii de acţiuni care vor putea fi executate la apariţia unui eveniment – de exemplu, iniţierea unei oglindiri (copieri) a unei resurse pe alt sit Web la un moment de timp.

- Subcapitolul secund se concentrează asupra problematicii căutării resurselor multimedia, lucrarea propunând o manieră de utilizare a descrierilor RDF ale metadatelor asociate resurselor şi exploatare a informaţiilor semantice asociate relaţiilor stabilite între resursele Web. După o succintă prezentare a arhitecturii motoarelor de căutare, se ilustrează modul de exprimare a interogărilor ce pot fi formulate de utilizator în *WQFL* (*Web Query Formulating Language*), conform seriei de cercetări întreprinse şi detaliate în [Buraga & Rusu, 2000], [Buraga & Brut, 2001] şi [Buraga & Brut, 2002]. Acest limbaj bazat pe XML va putea exprima interogările complexe date de utilizatori, dar – de asemenea – va fi capabil a desemna unele informaţii privind structura şi conţinutul resurselor hipermedia găsite.

  Strategia de interogare pleacă de la premisa că utilizatorii preferă să obţină documente (multimedia) având diverse structuri şi tipuri de conţinut, în acest sens trebuind specificate poziţiile şi numărul de apariţii ale unor elemente sintactice compunând un anumit document XML.

  Pentru a oferi o cât mai mare flexibilitate, limbajul WQFL a fost extins să faciliteze utilizarea expresiilor regulate Perl, recurgându-se la teoria limbajelor regulate [Jucan, 1999].

Capitolul 4 oferă o serie de soluţii de implementare a sistemului *ITW* [Buraga, 2003g, Buraga & Găbureanu, 2003], mediu eterogen şi distribuit utilizat la regăsirea informaţiilor multimedia, folosind agenţi şi servicii Web. După o prezentare a problematicii sistemelor multiagent [Bradshow, 1997, Mangina, 2002] şi a serviciilor Web bazate pe XML [Vasudevan, 2001], lucrarea continuă cu ilustrarea arhitecturii unei platforme distribuite compusă din agenţi, servicii Web şi alte entităţi programabile. Sistemul *ITW* foloseşte descrieri RDF/XML pentru căutarea resurselor multimedia, plecând de la interogări formulate în WQFL prin intermediul unei interfeţe Web flexibile [Buraga, 2002a, Buraga, 2003b], şi se bazează pe relaţiile spaţio-temporale adnotate în limbajele *XFiles* şi TRSL definite în cadrul capitolului 3.

Sistemul de agenţi este bazat pe formalismul *BDI* (*Belief-Desire-Intention*) [Rao *et al.*, 1995], iar pentru implementarea efectivă se uti-

lizează o platformă specifică – *Omega* [Alboaie & Buraga, 2002] – permiţând interschimb de informaţii XML între agenţi prin intermediul serializării [Buraga & Alboaie, 2004, Alboaie & Buraga, 2003a].
Se oferă şi două posibile utilizări:

- în domeniul *e-learning* – conform cercetărilor concretizate în lucrări precum [Buraga, 2001c], [Buraga, 2003c], [Buraga, 2003e] şi [Buraga & Brut, 2003];
- în modelarea fluxului de informaţii în cadrul întreprinderilor virtuale (*e-enterprise*) – rezultatele cercetărilor efectuate fiind detaliate în [Cioca & Buraga, 2003a] şi [Cioca & Buraga, 2003b].

Lucrarea se încheie cu prezentarea concluziilor generale şi direcţiilor de cercetare viitoare.

Prima anexă ilustrează modul de generare şi de utilizare a metadatelor asociate unor resurse Web prin intermediul aserţiunile RDF în contextul unui sit de comerţ electronic. Următoarele trei anexe detaliază definiţiile formale ale sintaxei limbajelor bazate pe XML specificate în capitolul 3 – *XFiles*, TRSL şi WQFL, respectiv –, pentru aceasta folosindu-se o abordare orientată-obiect facilitată de schemele XML. Ultima anexă enumeră acronimele utilizate în cadrul acestui material.

Contribuţii

Din cele aproximativ 300 de referinţe bibliografice ale lucrării, peste 40 sunt contribuţii originale ale autorului, ca unic autor sau co-autor, concretizate în cărţi tipărite, articole recenzate şi publicate în reviste internaţionale de prestigiu sau în *proceeding*-urile unor conferinţe internaţionale, editate pe plan mondial de IEEE Computer Society Press, Springer-Verlag (LNCS) sau IOS Press şi pe plan naţional de Polirom sau de editurile unor instituţii academice din Bucureşti, Craiova, Galaţi, Iaşi ori Timişoara.

Lucrarea de faţă se bazează pe o serie de cercetări efectuate în cadrul unui număr de 5 contracte finanţate de Academia Română, ANSTI şi CNCSIS în perioada 1999–2002, dintre care se pot menţiona:

- grantul CNCSIS 283/2002 – *Tehnici avansate de căutare a documentelor hipermedia pe Web*, director: Sabin Buraga;
- grantul CNCSIS 966/2001 – *Metodologii generative pentru proiectarea maşinilor abstracte*, director: Gheorghe Grigoraş.

De asemenea, o parte din conţinutul lucrării de faţă se bazează pe experienţa acumulată de autor în cadrul comitetelor ştiinţifice ale unor evenimente internaţionale precum *International Symposium on Parallel and Distributed Computing – ISPDC 2003*, Ljubljana, 2003, *Agent-Based Computing*, sesiune specială în cadrul *World Multiconference on Systemics, Cybernetics and Informatics – SCI 2003*, Orlando, 2003 şi *International Symposium on Parallel and Distributed Computing – ISPDC 2004*, Cork, Ireland, 2004.

Destinatari

Cartea se adresează tuturor celor interesaţi de problematicile actuale şi de continua dinamicitate a Web-ului semantic, putând fi consultată de studenţii din anii terminali, masteranzi sau doctoranzi, de specialiştii în domeniul ştiinţei calculatoarelor şi de toţi cei care doresc să îşi formeze o privire de ansamblu asupra temelor principale de cercetare referitoare la Web.

Părţi ale materialului de faţă – redactat integral folosind instrumente *open-source* (LaTeX şi GIMP) rulate pe platforme Linux (Red Hat şi Mandrake) – se bazează pe cursurile *Tehnologii Web* şi *Tehnologii Web II*, predate de autor studenţilor anului IV ai Facultăţii de Informatică a Universităţii "Al. I. Cuza" din Iaşi.

Aşteptăm reacţiile dumneavostră prin poşta electronică la adresa `busaco@infoiasi.ro`. Situl Web dedicat acestei lucrări este disponibil la `http://www.infoiasi.ro/~sweb/`.

Autorul
martie 2004, Iaşi

Capitolul 1

Arhitectura spaţiului WWW

Acest capitol descrie succint conceptele de bază ale spaţiului World-Wide Web, trecând în revistă evoluţia – din mai multe perspective – a limbajelor de marcare pentru Web.

1.1 Prezentare generală

UNUL dintre cele mai importante şi de succes servicii ale Internetului, **World-Wide Web**-ul – mai pe scurt, **Web** sau spaţiul **WWW** –, a fost instituit la CERN (Centrul European de Cercetări Nucleare de la Geneva) în anul 1989, graţie fizicienilor Tim Berners-Lee, Robert Caillau şi echipei acestora, scopul principal urmărit fiind facilitarea accesului rapid la informaţiile tehnice cuprinse în manualele de utilizare a calculatoarelor [Berners-Lee, 1989]. Web-ul reprezintă un *sistem de distribuţie locală sau globală a informaţiilor hipermedia* [Berners-Lee, 1999].

Spaţiul Web pune la dispoziţie un sistem global şi standardizat de comunicare multimedia, informaţiile fiind organizate asociativ, Web-ul

funcţionând conform modelului client/server şi beneficiind de facilităţile oferite de structurarea sub formă de hipertext a resurselor. Cu toată dezvoltarea lui spectaculoasă, Web-ul nu trebuie confundat cu Internetul, ci poate fi considerat drept cea mai dinamică şi spectaculoasă componentă software a acestuia.

Cantitatea de informaţii disponibile în spaţiul WWW, în oricare domeniu, este copleşitoare şi în continuă creştere. Conceptul iniţial al Web-ului a fost tocmai integrarea unor sisteme informaţionale disparate (ca, de exemplu, sistemele de gestiune a bazelor de date) într-un mod unitar, formându-se un spaţiu abstract, în care diferenţele dintre diversele surse de date să nu mai existe. Actualmente, Web-ul cumulează orice tip de informaţie, indiferent de platforma pe care există fizic.

Ideea de a agrega şi de a asocia resurse de informaţii disponibile în manieră distribuită provine din cercetările întreprinse în domeniul *hipertextului*[1], modalitatea de a adresa resursele – text, imagini statice, multimedia (audio, animaţii, video) etc. – realizându-se prin intermediul *identificatorilor uniformi de resurse* (*URI – Uniform Resource Identifiers*), prezentaţi în cadrul secţiunii 1.2.2. Aceşti identificatori reprezintă o modalitate flexibilă şi eficientă de accesare a oricărei resurse Internet, prin oricare protocol de comunicare – cel mai utilizat fiind *HTTP* (*HyperText Transfer Protocol*) [Fielding *et al.*, 1997].

Limbajul prin care sunt structurate şi prezentate informaţiile şi, de asemenea, sunt specificate legăturile dintre diverse resurse hipertext este popularul limbaj de marcare – sau de adnotare[2] – *HTML* (*HyperText Markup Language*) [Raggett *et al.*, 1999]. În prezent, limbajul HTML a fost rescris în termenii XML, apărând noul limbaj de marcare a hipertextului XHTML [Pemberton *et al.*, 2002].

Identificatorii uniformi de resurse URI, protocolul HTTP şi limba-

[1] A se consulta şi [Nielsen, 1990], [Balasubramanian, 1994], [Trăuşan-Matu, 2000] sau [Buraga, 2001a].

[2] Acţiunea de adnotare se realizează prin intermediul unui *limbaj de adnotare* (sau de specificare), care reprezintă un set de convenţii de marcare utilizate pentru codificarea datelor, specificând mulţimea de marcaje obligatorii, permise, maniera de identificare a marcajelor şi semantica fiecărui marcaj disponibil [Buraga, 2002a].

jul HTML au reuşit să îndeplinească practic scopurile principale ale spaţiului World-Wide Web [Berners-Lee, 1999, Berners-Lee, 2002]:

- *independenţa de dispozitiv* – aceleaşi informaţii pot fi accesate via o multitudine de dispozitive, precum *mainframe*-urile, calculatoarele personale etc.;

- *independenţa de software* – forme diferite de software – clienţi (navigatoare, roboţi şi agenţi Web etc.) şi servere Web – oferă şi extrag informaţii într-o manieră universală, fără ca nici un produs-program să reprezinte o componentă critică pentru spaţiul WWW, deoarece Web-ul nu reprezintă un program, ci un set de protocoale şi specificaţii standardizate, deschise, redactate de *Consorţiul Web* [W3C];

- *scalabilitatea* – dezvoltarea exponenţială a Web-ului este un exemplu interesant al efortului intens depus de comunitatea utilizatorilor Internet-ului, independent de resursele hardware şi software disponibile;

- *caracterul multimedia* – documentele disponibile pe Web, regăsite şi sub denumirea de *pagini*, pot integra surse de informaţie multiple, în diverse forme, de la date discrete – text, imagini statice, precum fotografii, scheme, diagrame etc. – până la cele continue – animaţii, audio şi video.

Uşurinţa creării şi publicării paginilor Web de către orice utilizator având cunoştinţe minime de marcare a datelor şi de design – folosind, eventual, multitudinea de editoare HTML disponibile de cele mai multe ori gratuit – a condus la apariţia *siturilor Web*. Un sit Web reprezintă o colecţie de documente orientate uzual către informaţii unitare sau scopuri comune [Buraga, 2002a]. Virtual, oricine – de la o persoană particulară până la o organizaţie guvernamentală, academică ori comercială – îşi poate dezvolta propria interfaţă (sit) Web, publicând-o şi integrând-o în spaţiul World-Wide Web.

1.2 Componente de bază ale spaţiului WWW

Spaţiul World-Wide Web funcţionează în practică datorită[3]:

- unei scheme consistente de identificare a resurselor, prin intermediul identificatorilor uniformi de resurse [Berners-Lee *et al.*, 1998] (a se vedea secţiunea 1.2.2);
- unui mecanism flexibil de transfer de date, reprezentat de protocolul HTTP [Fielding *et al.*, 1997];
- unei descrieri logice a structurii documentelor hipertext, prin concursul unor limbaje de adnotare bazate pe meta-limbajele SGML (Standard Generalized Markup Language) [Goldfarb, 1990] – ne referim aici mai ales la HTML [Raggett *et al.*, 1999] – sau, mai recent, la XML (Extensible Markup Language) [Bray *et al.*, 2004][4].

În continuare, vom prezenta caracteristicile esenţiale ale acestor componente de bază ale spaţiului WWW (a se vedea şi figura 1.1).

1.2.1 Hipertextul

1.2.1.1 Definiţii

Conceptul de *hipertext* reprezintă o manieră particulară de organizare versatilă a informaţiilor. Termenul *hipertext* (text non-linear) are o multitudine de definiţii, dintre care se pot menţiona[5]:

- Hipertextul reprezintă o formă nelineară de document electronic.
- Hipertextul este un mod de organizare complexă a informaţiilor în care datele sunt memorate într-o reţea (graf) de noduri şi legături (a se vedea secţiunea 1.2.1.2).

[3] Pentru detalii, a se consulta [Berners-Lee, 1999] sau [Buraga, 2001a].

[4] Datele marcate în XML pot fi vizualizate sau transformate în alte formate prin utilizarea specificaţiei *XSL* (*Extensible Stylesheet Language*) [Adler *et al.*, 2001].

[5] Pentru detalii, a se consulta lucrările [Balasubramanian, 1994], [Buraga, 2001a], [Nielsen, 1990], [Louka, 1994] sau [Trăuşan-Matu, 2000].

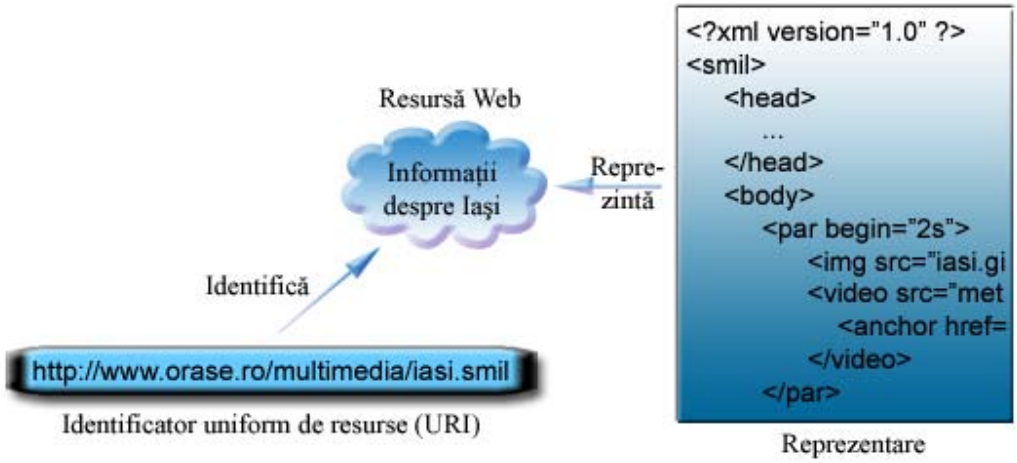

Figura 1.1: Relaţiile dintre o resursă multimedia, adresa ei – via URI – şi reprezentarea structurată a resursei (adaptare după [Jacobs, 2003])

- Hipertextul reprezintă o formă de comunicare dintre om şi calculator, interfaţa fiind chiar hipertextul.

Documentele hipertext se numesc *hipermedia* în cazul în care locul informaţiilor text este luat de cele multimedia.

1.2.1.2 Concepte

Componentele centrale ale hipertextului sunt *nodurile* şi *legăturile*.

Un nod reprezintă un concept unic, având în componenţă informaţii (discrete ori continue) sau programe generând un anumit conţinut. Un nod poate avea asociat un tip care înglobează o informaţie semantică. Nodurile sunt conectate la alte noduri prin intermediul legăturilor. Nodul sursă al unei legături poartă numele de *referinţă*, iar nodul destinaţie se numeşte *referent*. Nodurile referinţă şi referent sunt denumite şi *ancore*.

Legăturile reprezintă conexiuni între noduri (sau concepte) dependente unul de altul, putând fi stabilite în cadrul aceluiaşi document şi/sau între documente diferite, stocate în manieră distribuită, permiţându-se astfel o organizare nelineară a informaţiilor.

Legăturile, văzute drept arce ale (di)grafului hipertext, sunt *bidirecţionale* sau *unidirecţionale*. Legăturile pot fi de două tipuri, conform naturii relaţiei dintre noduri [Louka, 1994]:

- *referenţiale* – non-ierarhice, utile pentru realizarea referinţelor încrucişate, fiind cele care deosebesc cel mai bine hipermedia de celelalte forme de stocare a informaţiei;
- *organizaţionale* (denumite ierarhice sau structurale) – desemnează relaţiile părinte-copil dintre noduri, fiind folosite la organizarea nodurilor în manieră ierarhică într-o structură strictă; acest tip de legături este esenţial pentru linearizarea hipertextului şi permite autorilor să verifice coerenţa structurii hipertext.

De asemenea, legăturile pot fi *statice* (stabilite *a-priori* de autorul documentului, la momentul proiectării reţelei hipertext) sau *dinamice* (create în momentul parcurgerii structurii hipertext, în funcţie de context – *e.g.*, restricţii de acces – sau de cerinţele/experienţa utilizatorilor).

1.2.1.3 Documente hipertext

Înţelegerea unui document hipertext şi navigarea prin acesta depind de abilitatea utilizatorului de a proiecta şi construi o reprezentare mentală coerentă a structurii hipertextului, creatorul acelui document fiind responsabil să asigure această coerenţă [Trăuşan-Matu, 2000].

Un document hipertext considerat coerent este constituit din trei componente, prezentate pe scurt în continuare:

1. *partea de conţinut*

   Nodurilor şi legăturilor le pot fi asociate proprietăţi (semantici), în vederea asigurării coerenţei informaţiei. Partea de conţinut stochează obiecte purtătoare de informaţie: noduri de conţinut – care memorează efectiv date – şi legături de conţinut – care conectează nodurile de conţinut pe baza unor relaţii semantice,

i.e. folosindu-se diverse ontologii[6]. Nodurile de conţinut pot fi atomice sau pot fi compuse din alte noduri.

Modalitatea de memorare a informaţiilor în cadrul nodurilor de tip conţinut variază de la un sistem hipertext la altul, în prezent adoptându-se meta-limbajul de marcare XML, pentru Web pretându-se – mai ales din punctul de vedere al manierei de prezentare – limbajul HTML. Actualmente, se utilizează un număr mare de limbaje bazate pe XML pentru marcarea diferitelor informaţii. Pentru a oferi doar câteva exemple[7], meta-limbajul XML este folosit la modelarea şi adnotarea de:

- prezentări multimedia sincronizate prin *SMIL* (*Synchronized Multimedia Integration Language*) [Ayars *et al.*, 2001];
- grafică vectorială pentru Web folosind *SVG* (*Scalable Vector Graphics*) [Ferraiolo *et al.*, 2003];
- limbaje de interogare (*query languages*) pentru Web (a se vedea [DeRose, 1998], [Shanmugasundaram *et al.*, 1999], [Oliboni & Tanca, 2000] sau [Malhotra *et al.*, 2003]);
- documente exprimând construcţii sintactice ale unor limbaje de programare funcţională [Boley, 2000];
- ontologii, prin intermediul limbajului *OWL* (*Web Ontology Language*) [Dean & Schreiber, 2004].

În cadrul acestui context, menţionăm şi cercetările proprii întreprinse în proiectarea unor limbaje bazate pe XML pentru reprezentarea sistemelor Lindenmayer (*L-systems*) [Buraga, 2000b, Buraga *et al.*, 2002b] în vederea vizualizării 3D în *VRML* (*Virtual Reality Modeling Language*) sau pentru exprimarea fluxului informaţiilor hipertext în cadrul întreprinderilor virtuale – *e-enterprise* [Cioca & Buraga, 2003a] [Cioca & Buraga, 2003b] [Cioca & Buraga, 2003c].

[6]Ontologiile reprezintă specificări ale unor conceptualizări [Gruber, 1993]. A se vedea şi cele discutate în capitolul 2.

[7]A se consulta şi [Trăuşan-Matu, 2001], [Oasis] sau [W3C].

2. *partea de organizare*

   Nodurile şi legăturile de structură stocate în cadrul acestei părţi asigură documentului o coerenţă sporită deoarece prin intermediul lor autorul îşi structurează reţeaua hipertext din perspectiva cititorului.

   Nodurile de structură pot fi clasificate în *noduri de secvenţă* – prin care autorul defineşte o anumită secvenţă de parcurgere a conţinutului hipertextului – şi *noduri de explorare* – care oferă utilizatorului posibilităţi de explorare complexă, non-secvenţială a reţelei hipertext.

   Nodurile de secvenţă împreună cu legăturile de secvenţă pot oferi diverse secvenţe (scenarii) de prezentare a conţinutului hipertext (ca de exemplu căi de vizitare secvenţiale, arborescente sau condiţionale).

3. *partea de prezentare*

   Această parte pregăteşte vizualizarea structurii şi conţinutului hipertextului, oferind diverse mecanisme de navigare (a se vedea şi secţiunea 3.3.2.2 a capitolului 3). Autorii pot adopta diferite stiluri de prezentare a informaţiilor:

   - *textual* – nu există o vizualizare grafică a structurii, prezentarea fiind limitată la afişarea conţinutului unuia sau mai multor noduri (utilizatorul nu este conştient că traversează o structură hipertext);
   - *grafic* – există o vizualizare grafică a structurii hipertext (*e.g.*, harta legăturilor dintre noduri, arborele de navigare etc. – a se vedea modul de vizualizare a informaţiilor găsite de metamotorul de căutare Kartoo [Kartoo]);
   - *combinat* – oferă ambele posibilităţi de prezentare.

Documentele hipertext sunt structurate asemenea reţelelor (grafurilor), fără a se impune restricţii în ceea ce priveşte mărimea nodurilor

sau modul de realizare a legăturilor dintre ele. Pentru realizarea unei structuri hipertext valide şi facil de parcurs, pot fi adoptate diverse principii de structurare, ca de exemplu metafora cărţii tipărite, structura lineară cu salturi sau structura ierarhică [Balasubramanian, 1994, Louka, 1994].

De asemenea, trebuie menţionat faptul că din punct de vedere formal, structurile hipertext pot fi modelate folosind teoria grafurilor.

1.2.2 Localizarea resurselor Web

1.2.2.1 Identificatori uniformi de resurse

Localizarea resurselor Web se realizează prin intermediul unor *identificatori uniformi de resurse – URI* [Berners-Lee *et al.*, 1998].

Este considerată *resursă* orice entitate având identitate (ca de exemplu: un document electronic, o imagine, un serviciu – i.e. serviciul de poştă electronică –, o colecţie de alte resurse). Sunt considerate resurse şi cele care nu pot fi accesate direct via Internet (*e.g.*, fiinţele umane, organizaţiile etc.). O resursă se poate menţine constantă în timp, deşi conţinutul ei – entităţile cărora le corespunde – se poate modifica.

În vederea unei organizări facile şi unitare, resursele sunt desemnate printr-un *identificator*. Pentru ca diverse tipuri de identificatori de resurse să poată fi utilizate în aceeaşi manieră, se recurge la folosirea unor scheme *uniforme* de identificatori. Acest principiu adoptat asigură independenţa de mecanismul (protocolul) folosit pentru accesarea resurselor, permiţându-se astfel interpretarea în mod uniform a mai multor convenţii sintactice desemnând identificatori ai unor resurse eterogene. De asemenea, se pot introduce noi tipuri de identificatori de resurse fără a fi modificat modul de adresare a vechilor tipuri.

Aşadar, pentru ca o resursă să poată fi numită, partajată sau interconectată cu alta/altele în cadrul spaţiului WWW, ea trebuie să aibă ataşat un identificator uniform de resurse.

Mulţimea URI este divizată în:

- *localizatori uniformi de resurse – URL* (*Uniform Resource Locator*) care identifică resursele printr-o reprezentare a mecanismu-

lui de accesare a lor (*e.g.*, localizarea unor resurse prin intermediul adresei IP), nu prin nume sau alte atribute;

- *nume uniforme de resurse – URN* (*Uniform Resource Name*) care permit referirea unei resurse chiar dacă resursa a dispărut ori a devenit inaccesibilă, prin intermediul unui nume persistent şi unic. URN-ul se utilizează mai ales pentru a desemna entităţi (componente, tipuri de date, servicii etc.) folosite de anumite aplicaţii Web.

1.2.2.2 Sintaxa URI

Un identificator uniform de resurse poate fi reprezentat ca şir de caractere aparţinând unui alfabet limitat (compus din literele alfabetului latin, cifrele şi diverse caractere de punctuaţie). Un URI poate include, de asemenea, o serie de *caractere rezervate* [Berners-Lee *et al.*, 1998].

Un identificator generic este compus din următoarele componente:

```
schema "://" authority path "?" query
```

Existând mai multe metode de a accesa resursele, vor fi disponibile mai multe *scheme* pentru a le identifica (i.e. `http`, `mailto`, `ldap`, `urn` etc.) – a se vedea şi [Jacobs, 2003].

Componenta *autoritate* (*authority*) este definită de o locaţie de server disponibil la nivelul Internetului sau de o secvenţă specială de înregistrare. Serverul poate fi specificat fie prin adresa IP (de exemplu `193.231.30.225`), fie prin adresa simbolică (*e.g.* `thor.infoiasi.ro`) – via *DNS* (*Domain Name System*) [Naik, 1998] –, eventual fiind urmat de un număr de port.

Componenta *cale* (*path*) conţine date menite a identifica o resursă localizată pe serverul desemnat de componenta *autoritate* descrisă mai sus. Calea poate conţine secvenţe de segmente de cale separate prin "`/`", fiecare secvenţă putând include o secvenţă de parametri.

Ultima componentă este cea de *interogare* (*query*), reprezentată de un şir de informaţii ce vor fi interpretate de resursă.

Ca exemplu de identificator uniform de resurse poate fi dată următoarea adresă Web, în care sunt prezente primele trei componente ale unui URI:

```
http://www.infoiasi.ro/~busaco/books.html
```

Pentru sintaxa completă (în forma EBNF [Jucan & Andrei, 2002]), recomandăm consultarea lucrărilor [Berners-Lee *et al.*, 1998] sau [Buraga, 2001a].

Deoarece majoritatea documentelor (resurselor) Web sunt stocate în manieră arborescentă (ierarhică), în locul adresării absolute se poate folosi o adresare relativă. Aceasta permite o independenţă parţială a locaţiei şi a schemei de acces, fiind desemnată sintactic prin intermediul *identificatorilor uniformi de resurse relativi*.

În plus, un URI poate avea inclus un *identificator de fragment* (*fragment identifier*) [Berners-Lee *et al.*, 1998] pentru a se permite identificarea indirectă a unei resurse secundare prin intermediul referinţei la resursa primară şi al informaţiilor suplimentare. Mai precis, dacă identificatorul uniform de resurse U identifică resursa R şi reprezentarea resursei R este în formatul F, iar conform specificaţiilor formatului F se ştie că identificatorii de fragment identifică resurse secundare în cadrul instanţelor lui F, atunci identificatorul pentru resursa secundară, identificată în interiorul unei instanţe a lui F de un identificator de fragment fid, este reprezentat de $U\#fid$. [Jacobs, 2003, Berners-Lee *et al.*, 1998]

Drept exemplu, poate fi menţionată următoarea adresă, sufixul ei desemnând identificatorul de fragment `public`:

```
http://www.infoiasi.ro/~busaco/cv.html#public
```

Identificatorii uniformi de resurse oferă suport pentru realizarea de legături între diverse noduri (resurse) ale spaţiului WWW: când reprezentarea unei resurse referă o altă resursă prin intermediul unui identificator URI, atunci acesta reprezintă o legătură – văzută în termenii hipertextului – între cele două resurse [Jacobs, 2003].

După cum am amintit în cadrul secţiunii 1.1, accesul la reprezentarea resurselor se realizează prin intermediul unui protocol de comunicaţie, în cazul Web-ului utilizându-se protocolul HTTP.[8]

[8] Pentru detalii, se pot consulta lucrările [Fielding *et al.*, 1997], [Buraga, 2001a], [Buraga *et al.*, 2002a] sau [Naik, 1998].

1.3 Evoluţia spaţiului WWW

În prezent, Web-ul a devenit fără îndoială "universul informaţiilor accesibile în reţea", conform definiţiei anticipative dată de creatorul acestuia, Tim Berners-Lee [Berners-Lee, 1989]. Spaţiul WWW s-a achitat de această promisiune într-o manieră spectaculoasă, din două motive. Primul dintre ele este reprezentat de flexibilitatea şi independenţa faţă de conţinut a protocolului HTTP – care se poate adapta pentru a transfera orice format de document – şi, în plus, de existenţa identificatorilor universali de resurse (URI) capabili de a reprezenta legături către orice format de document. Un al doilea motiv este faptul că selecţia naturală pare să fi favorizat câteva formate care au fost dezvoltate pentru Web în mod explicit. În peisajul de o prolifică varietate a *sintaxei*, *stilului*, *structurii* şi *semanticii* documentelor Web, observăm adoptarea preferenţială a limbajelor SGML, HTML, CSS şi XML. Fiecare dintre acestea aduce o strategie evoluţionară lentă, favorizând codările declarative faţă de formatele binare (de cele mai multe ori proprietare), stilurile de afişare separabile de conţinutul propriu-zis faţă de formatarea directă, marcarea declarativă faţă de cea procedurală şi semantica bine definită faţă de comportamentul operaţional.

Evoluţia Web-ului [Berners-Lee, 1999, Berners-Lee, 2002] se poate remarca mai ales în următoarele direcţii:

- *evoluţia sintaxei spre formate declarative*;
- *evoluţia stilului (prezentării): de la formatare la foi de stiluri*;
- *evoluţia structurii*;
- *evoluţia semanticii*;
- *evoluţia elementelor programabile*.

Fiecare dintre acestea vor fi detaliate în continuare, urmând liniile prezentate în [Buraga, 2002a] şi [Alboaie & Buraga, 2002].

1.3.1 Evoluţia sintaxei spre formate declarative

Primul compromis se face în privinţa optării între limbajul maşină (binar) şi limbajul natural (textual). La o primă vedere, codificarea specifică maşinii pare mai puţin costisitoare datorită faptului că ea reflectă direct datele în memorie. Totuşi, transformarea datelor în format binar poate fi prea fragilă pentru a fi folosită pe platforme eterogene (de exemplu, reprezentarea numerelor) şi în versiuni de software multiple. Codificarea binară poate, de asemenea, necesita mai mult spaţiu şi timp pentru împachetare şi despachetare. Fişierele text pot fi la fel de eficiente, în plus formatele textuale (chiar şi cele în forme parţial lizibile) sunt mai uşor de editat, corectat şi extins.

Al doilea compromis este legat de amalgamul dintre formatele declarative şi cele procedurale. Poate fi mai uşor de elaborat un analizor pentru un program care calculează numărul π decât de transmis un miliard de cifre ale sale. Limbajul TEX [Knuth, 1984], bazat pe marcaje, este un exemplu edificator privind puterea oferită de formatele procedurale pentru conceperea şi procesarea documentelor, faţă de editoarele de texte folosind formate binare proprietare (*e.g.*, Word).

În fine, există un compromis între formatele specifice şi cele generice. În măsura în care refolosirea informaţiei este o preocupare majoră, o importanţă chiar mai mare decât aceasta o are promovarea unei familii de gramatici înrudite. Puterea meta-limbajelor SGML şi XML este reflectată de flexibilitatea în gestionarea documentelor de toate tipurile, de la manuale şi comunicate de presă până la contracte legale şi specificaţii de proiecte, şi de posibilitatea de reutilizare a acestora în vederea producerii de cărţi, rapoarte şi ediţii electronice (pentru CD-ROM, Web, dispozitive mobile etc.) pornind de la acelaşi (aceleaşi) fişier(e) sursă, folosind marcaje declarative similare.

Există o tensiune fundamentală între performanţa, costul şi capacitatea de utilizare a strategiilor de codificare uşor lizibile de către maşină şi, respectiv, de către om. Abordarea XML găseşte un echilibru rezonabil între cele două. Facila lizibilitate pentru om implică robusteţe, iar descifrarea rapidă de către maşină implică validitate; ambele calităţi adaugă valoare informaţiei şi facilitează evoluţia documentelor în timp.

1.3.2 Evoluţia stilului (prezentării): de la formatare la foi de stiluri

De când au apărut documentele, există autori şi proiectanţi care şi-au concentrat efortul de perfecţionare tehnică şi stilistică pentru fiecare atingere de peniţă, fiecare fragment de corp de literă, fiecare imagine plasată. În măsura în care stilul generic al unei prezentări poate fi captat în vederea reutilizării ulterioare, cu atât mai multă valoare capătă design-ul şi documentul în sine. Istoria evoluţiei formatelor de documente pentru Web favorizează formatarea externă în detrimentul directivelor încapsulate tocmai cu scopul ca informaţia să poată fi reprezentată independent de stil şi viceversa.

Pe parcursul istoriei procesării computerizate a datelor, au existat numeroase abordări ale reprezentărilor orientate spre formatări interne, de la comenzile `troff` sau `groff` din mediile UNIX până la directivele formatului *RTF* (*Rich Text Format*), şi la marcajele pentru fonturi din HTML (a se vedea, de exemplu, marcatorul `<font>`). Aproape inevitabil, acestea au fost completate cu "scurtături" de formatare reutilizabile: pachete de macro-uri, rigle şi parametri pentru afişarea în navigator etc. Clădite pe această experienţă, formatele actuale de documente Web – HTML şi XML – permit ca formatarea datelor să poată fi realizată extern, prin foi de stiluri. Foile de stiluri în cascadă – *CSS* (*Cascading Style Sheets*) [Bos *et al.*, 1998] – merg mai departe şi permit compunerea unor stiluri separate, cum ar fi proprietăţile de culoare şi font, seturile de caractere, grafica şi prezentarea. Mai mult, controlul formatării este divizat între autor şi cititor, acesta din urmă putându-şi interpune propria sa înlănţuire de foi de stiluri, deci contribuind la îmbunătăţirea prezentării datelor, în funcţie de dorinţele sale individuale [Buraga, 2001a, Buraga, 2002a].

Stilurile nu sunt limitate la domeniul vizual – ele pot controla redarea conţinutului pentru monitoare, hârtie, audio, medii Braille, terminale mobile şi multe altele. Acest lucru este preţios îndeosebi dacă se doreşte adaptarea prezentării documentului pentru utilizatorii cu handicapuri fizice, dislexie sau pentru analfabeţi – ca şi în cazul a diverse privaţiuni de circumstanţă (utilizatori care vorbesc la telefon sau lucrează într-un mediu zgomotos). Fluxurile audio pot fi transcrise pe

Web ca text pentru persoanele cu deficienţe de auz; navigatoarele audio (precum Web Galaxy, Vox Portal sau WIRE) oferă facilitatea de a citi paginile Web pentru utilizatorii care nu le pot parcurge în mod direct, în conformitate cu foile de stiluri aurale.

Cu toate că tehnologia Web actuală suportă atât formatarea internă, cât şi pe cea prin intermediul foilor de stiluri, sistemele pentru gestionarea unor cantităţi mari de informaţie de tip hipertext necesită de cele mai multe ori formatare externă pentru ca informaţia să rămână uşor navigabilă şi facil de prelucrat. Se recomandă, aşadar, separarea datelor de modul lor de prezentare [Buraga, 2002a].

1.3.3 Evoluţia structurii

Să considerăm următorul exemplu:

Exemplul 1 În activitatea de documentare, un cercetător realizează o listă de referinţe bibliografice cuprinzând informaţii despre publicaţiile (electronice sau nu) regăsite. Structura fiecărui articol ştiinţific include un titlu, numele autorului, corpul articolului şi subsolul.

Diversele formate punând la dispoziţie structuri concurente (*e.g.*, SGML) de documente încearcă să surprindă această structură în cadrul reprezentării lor. Unele descriu bucăţi din document în termeni legaţi de *prezentarea* datelor: proprietăţi precum *italic*, *indentat*, stabilirea unui anumit corp de literă (font) şi aşa mai departe. La cealaltă extremă, alte formate folosesc termeni *declarativi*: titlu, adresă, intrare de la tastatură etc. Multe alte formate se află undeva între aceste două extreme, precum marcajele XHTML [Pemberton *et al.*, 2002] `<em>` şi `<font>`, în comparaţie cu `<address>` sau `<abbr>`.

Alt tip de structuri declarative pentru aplicaţiile SGML şi XML sunt definiţiile de tipuri de documente – *DTD* (*Document Type Definition*)[9] – care impun documentelor valide să includă mai multe elemente (marcaje) într-o ordine precizată (de exemplu, “fiecare apariţie

[9] Regulile sintactice de specificare a definiţiilor de tipuri de documente [Goldfarb, 1990, Bray *et al.*, 2004] urmează teoria limbajelor formale (a se vedea [Jucan, 1999]).

a elementului <oraş> trebuie să fie precedată de marcajul <adresă> şi urmată de <cod_poştal>"). Actualmente, declaraţiile DTD tind a fi înlocuite de schemele XML [Fallside, 2001].

Luarea unei decizii finale de-a lungul acestei axe aduce după sine compromisuri între precizie şi comprehensibilitate: semanticile orientate spre prezentare, mai lejere, sunt în mod universal mai bine înţelese decât cele declarative. Marcajul <address> a devenit parte din repertoriul HTML, însă <abstract> (desemnând marcarea rezumatului unei lucrări ştiinţifice) nu. Marcarea declarativă, indicând în mod clar rolul diverselor părţi de document, are avantajul de a putea fi refolosită mai târziu. De exemplu, motoarele de căutare (ca de exemplu, Google [Google]) pot atribui într-un mod mai semnificativ ponderi termenilor dintr-un rezumat sau pot extrage automat numele reporterilor dintr-un set de "tăieturi din ziare" folosind un instrument de captare a informaţiei.

Vom opta pentru descrierea structurii unui document după funcţia sa mai degrabă decât după formă. Aceasta conduce la asigurarea unui suport pentru un set extensibil de marcaje, ceea ce HTML şi CSS nu pot oferi. Evoluţia centralizată a HTML împiedică întocmirea unei liste exhaustive de marcaje răspunzând tuturor idiomurilor dorite de potenţialii autori de pagini Web. Un marcaj nou are în mod potenţial o sintaxă ambiguă, o semantică ambiguă şi o prezentare ambiguă (mai ales fără adăugiri de foi de stil).

Comunităţile de interese de pe Web au nevoie să-şi publice propriile definiţii uşor, proces facilitat de folosirea meta-limbajului XML. Aceste noi definiţii pot chiar să meargă mai departe de specificarea rolului fiecărui marcaj, pentru a include interpretări şi comportamente, i.e. pentru a suporta noi *semantici*. De altfel, navigatoarele Web actuale oferă un tot mai bun suport pentru documentele XML.

1.3.4 Evoluţia semanticii

Testul suprem pentru ca un format de document să "supravieţuiască" pe Web este măsura în care conţinutul său suportă diversele tipuri de utilizări. Documentele există ca artefacte ale unor procese mai largi, precum achiziţionarea, raportarea sau dezvoltarea software-ului, iar

aceste utilizări ataşează *componente semantice* diferitelor părţi ale documentului.

Suportul pentru semantică al unui anumit format se referă începând de la comportamentul *operaţional* codificat *hard* în procesele care manipulează conţinutul şi ajungând până la a fi *bine definit*, adică a da definiţii larg disponibile şi documentate ale conţinutului.

Să considerăm un fişier text constând dintr-o listă cu acţiuni planificate pentru perioada de timp imediat următoare. Acest fişier poate fi folosit drept mini-agendă de activităţi descrise în limbaj natural de către un utilizator. O versiune HTML cuprinzând termene-limită pentru fiecare activitate ar putea fi analizată de un program care procesează conţinutul şi alertează utilizatorul (eventual via un mesaj pe telefonul mobil) în preajma acestor termene. Mai mult, o a treia variantă a acestui fişier, în format XML, ar putea declara o clasă de documente care să definească în mod explicit un marcator <agenda> pentru datele dorite. Doar ultimul dintre formate poate pretinde că are o semantică bine definită, legată de documentul însuşi prin intermediul oricărei aplicaţii separate. Această semantică poate fi încapsulată în interiorul altor documente XML şi interschimbată cu alte comunităţi de utilizatori, păstrându-şi totuşi neambiguitatea.

Limbajul XML suportă eficient ontologii parţiale[10] prin intermediul definiţiilor tipurilor de documente (DTD) sau al schemelor care pot fi compuse în mod dinamic şi referite prin identificatori uniformi de resurse, în acest fel descentralizând, şi prin urmare accelerând ciclul publicării şi adoptării noilor formate de documente.

În mod fundamental, ontologiile – reprezentând specificări ale unor conceptualizări, conform [Gruber, 1993] – întruchipează principiul peren care afirmă faptul că sistemele autoreprezentative sau autodescriptive reduc costul unui articol. Prin stabilirea unei adrese Web (dată sintactic printr-un URI – a se vedea secţiunea 1.2.2) pentru un fragment de cunoştinţe conţinând programe şi/sau informaţii, şi apoi prin *partajarea* acelei adrese cu alţii, autorii pun în practică procesul democratic următor: orice persoană aparţinând comunităţii Web poate arăta

[10] A se vedea, pentru detalii, lucrările [Davies *et al.*, 2003], [Trăuşan-Matu, 2000] sau [Trăuşan-Matu, 2001].

că anumite informaţii sunt interesante (“bune”) prin crearea unei legături către resursa al cărei conţinut include respectivele informaţii, pentru o eventuală utilizare ulterioară. Utilizatorii nu copiază documentul iniţial, eventual operând mici modificări, pentru că ar fi prea costisitor; cel mai ieftin mod de a accesa şi propaga ideile conţinute de respectivul document este prin intermediul adresei Web, realizându-se o legătură hipertext către acel fragment de informaţie. Aceasta alimentează ciclul selecţiei naturale în reprezentarea cunoştinţelor: utilizarea determină comunitatea, care rafinează la rândul ei, permanent, ontologia comună.

Exemplul 2 Un alt exemplu, relativ mai complex, este următorul. Să presupunem că în timp ce parcurge această lucrare cititorul întâlneşte următoarea referinţă bibliografică, scrisă astfel:

> “A Model for Accessing Resources of the Distributed File Systems”, Sabin Buraga, *Lecture Notes in Computer Science* – LNCS **2326**, Springer-Verlag: 224–230 (2002)

Folosindu-şi intuiţia, cititorul ar putea stabili înţelesul acestei referinţe, pe când un analizor digital e posibil să nu fie capabil să o facă. Mai mult, din cauza diverselor convenţii de publicare, alt editor ar putea formata referinţa în maniera următoare:

> S. Buraga, “A Model for Accessing Resources of the Distributed File Systems”, *Lecture Notes in Computer Science – LNCS 2326*, Springer-Verlag, 2002, pp. 224–230.

Până şi diferenţele minore de punctuaţie şi de notaţie pot încurca un program de calculator care încearcă să analizeze referinţa; drept rezultat, automatizarea conversiilor între diversele formate reprezentând acelaşi tip de cunoştinţe este tentantă. Utilizarea unei sintaxe fragile, având doar o sintaxă de formatare şi una operaţională, deşi are destulă semnificaţie pentru un utilizator uman, furnizează puţine informaţii pentru maşină.

Prin reformatarea referinţei utilizând marcarea prezentaţională a limbajului XHTML [Pemberton *et al.*, 2002], referinţa devine mai accesibilă, cu toate că regulile reale, tacite (“cuvintele scrise cursiv reprezintă publicaţia”) sunt invizibile:

```
<p>"A Model for Accessing Resources of
the Distributed File System", Sabin Buraga,
<i>Lecture Notes in Computer Science</i> --
LNCS <b>2326</b>, Springer-Verlag: 224--230 (2002)</p>
```

Deşi construcţiile XHTML permit autorului o oarecare precizare a structurii, impun în acelaşi timp atât autorului, cât şi cititorului să cadă de acord asupra semanticii atributelor, a valorilor şi a modului lor de marcare. Această structură (ambiguă) bazată pe elementele de prezentare a datelor permite autorului şi cititorului să scoată în evidenţă ceea ce este important, dar reprezintă doar un stadiu incipient pe drumul către o marcare structurală mai semnificativă.

Prin reformatarea referinţei utilizând marcarea structurală (destul de precară şi ea) a limbajului XHTML, regulile reale de interpretare devin ceva mai clare (“caracterele incluse într-un element `<cite>` reprezintă un titlu de referinţă” şi “conţinutul elementului `<h3>` referă numele unei publicaţii”):

```
<cite>A Model for Accessing Resources of
      the Distributed File Systems</cite>
<h3>Lecture Notes in Computer Science</h3>
<h4>Sabin Buraga</h4>
<ul>
<li>LNCS 2326</li>
<li>2002</li>
<li>224--230</li>
</ul>
```

Utilizarea structurii de mai sus în referinţele bibliografice poate îmbunătăţi şi modul de formatare a informaţiilor. Putem aminti aici şi pe unul dintre mecanismele precursoare, reprezentat de sistemul BibTEX [Knuth, 1984, Goossens *et al.*, 1994].

Mergem mai departe prin reformatarea referinţelor bibliografice utilizând un document XML particularizat pentru astfel de referinţe. Astfel, regulile reale de interpretare devin precise (mai ales dacă utilizăm spaţii de nume XML [Bray *et al.*, 1999]):

```
<bibliografie>
  <titlu>A Model for Accessing Resources of
         the Distributed File Systems</titlu>
  <publicatie numar="2326" abreviere="LNCS">
     Lectures Notes in Computer Science
  </publicatie>
  <autor>
     <prenume>Sabin</prenume>
     <nume>Buraga</nume>
  </autor>
  <an>2002</an>
  <pagini prima="224" ultima="230" />
</bibliografie>
```

Documentul din acest exemplu este bine formatat chiar în lipsa unei definiţii DTD sau a unei scheme XML. Cititorul uman poate înţelege semantica, iar maşinile pot manevra informaţiile marcate în XML (de exemplu, se pot afişa toate publicaţiile prin parcurgerea marcajelor şi selectarea caracterelor aflate în interiorul fiecărei instanţe a elementului `<publicatie>`).

Acest tip de marcare permite modelului informaţional să fie mai descriptiv, astfel încât o maşină să poată analiza lucrurile pe care o comunitate umană le înţelege de la sine. Problemele sintactice – precum codificarea caracterelor şi punctuaţia – sunt definite utilizând adnotări structurate; manipularea documentelor – de exemplu restructurarea şi filtrarea – pot fi automatizate, iar fiecare componentă a unui document poate fi identificată în mod precis [Buraga, 2001a].

Putem continua rafinarea marcării datelor, stabilind un spaţiu de nume pentru marcajele definind referinţe bibliografice (în vederea evitării ambiguităţilor), plus ataşând metadate – recurgând eventual la aserţiuni RDF (a se vedea şi capitolul 2) – fiecărei informaţii (de exemplu, specificând faptul că Sabin Buraga este autorul acelui articol şi

traducătorul articolului într-o altă limbă, iar drepturile de proprietate intelectuală asupra publicaţiei sunt deţinute de altcineva etc.):

```
<rdf:Description
  rdf:about="http://www.springer.de/LNCS2326/224-230.pdf">
  <bib:titlu>A Model for Accessing Resources of
         the Distributed File Systems</bib:titlu>
  <bib:autor>
     <rdf:Description
       rdf:about="mailto:busaco@infoiasi.ro">
       <bib:prenume>Sabin</bib:prenume>
       <bib:nume>Buraga</bib:nume>
     </rdf:Description>
  </bib:autor>
</rdf:Description>
```

Conform [Berners-Lee *et al.*, 2001, Davies *et al.*, 2003], acest tip de marcare face posibilă *reprezentarea cunoştinţelor* pentru:

- descoperirea resurselor prin furnizarea unor indicii utile motoarelor de căutare;
- catalogarea computerizată a conţinutului şi relaţiilor sale cu alte date, în vederea constituirii bibliotecilor digitale;
- evaluarea conţinutului pentru agregare şi filtrare;
- codificarea, partajarea şi schimbul de cunoştinţe, într-o manieră inteligentă.

Meta-limbajul XML reprezintă principala speranţă şi totodată componenta de bază pentru *Web-ul semantic* [Berners-Lee *et al.*, 2001] – obiectul studiului de faţă. Prin intermediul unor mecanisme de descriere în XML a resurselor şi relaţiilor dintre acestea, se va oferi cadrul necesar pentru interoperabilitatea între diverse aplicaţii care realizează schimburi inteligente de informaţii.

Conform creatorului spaţiului WWW, Tim Berners-Lee, Web-ul semantic reprezintă o pânză consistentă şi logică a tuturor resurselor de pe Web, accentul punându-se pe interpretarea datelor de către maşină şi nu pe reprezentarea lor (a se vedea capitolul 2).

1.3.5 Evoluţia elementelor programabile

Evoluţia către formatele declarative (vezi tabelul 1.1)[11] poate fi constatată nu numai în domeniul documentelor, ci şi în tendinţele comunităţii programatorilor. Aceleaşi forţe par să acţioneze şi în dezvoltarea succesivă a limbajelor de programare de tipul mai mult declarativ şi mai puţin operaţional: de exemplu, de la cod-maşină evoluţia a condus la limbajul de asamblare, apoi spre Pascal, C, C++, Perl, Java, Python, Pike şi C#.

De asemenea, Web-ul – prin evoluţia continuă a limbajelor bazate pe XML – poate fi considerat un veritabil *sistem hipermedia distribuit* utilizând ca infrastructură Internetul. Prin popularea acestei infrastructuri cu seturi de componente orientate-obiect şi găsirea de modalităţi (semantice) de integrare a acestora, Web-ul poate fi văzut ca un mediu eterogen pentru dezvoltarea şi exploatarea de sisteme obiectuale distribuite menite a manipula componente multimedia [WOI].

Iniţial, spaţiul WWW era compus din pagini (documente) statice (incluzând text şi imagini, apoi elemente multimedia), interconectate prin intermediul legăturilor hipertext (a se vedea secţiunea 1.2.1). Aplicaţii de tip client (precum navigatoare Web) erau folosite pentru accesarea – via adrese (URI) – a reprezentării acestor resurse, stocate pe servere Web. Programe suplimentare (*plug-in*-uri), incluse în navigatoarele Web, erau menite să redea tipuri de conţinuturi multimedia nestandardizate, ca fişiere Word, PostScript, *PDF* (*Portable Document Format*), Flash etc. Pentru a oferi conţinut dinamic utilizatorilor, sunt adoptate diverse modalităţi programatice[12], reprezentate pe partea server de programe *CGI* (*Common Gateway Interface*), servere de aplicaţii precum *PHP* (*PHP: Hypertext Preprocessor*) sau *JSP* (*Java Server Pages*), iar pe partea client de programe JavaScript sau applet-uri Java. De remarcat faptul că o serie de elemente programabile funcţionează drept componente *middleware* [Bakken, 2001] (o arhitectură *3-tier*, în fapt), reprezentând interfeţe pentru accesarea unor servicii la distanţă

[11] Se pot consulta, de asemenea, şi [Naik, 1998], [Knuth, 1984], [Goossens *et al.*, 1994], [Lamport, 1994], [Buraga, 2001a] sau [W3C].

[12] A se vedea lucrările [Buraga, 2001a], [Buraga *et al.*, 2002a], [Buraga, 2003a], [Tanasă *et al.*, 2003] şi [Trăuşan-Matu, 2001].

(*e.g.*, sisteme relaţionale de baze de date).

Modelul client/server pe care se bazează Web-ul nu oferă suport suficient pentru dezvoltarea unor aplicaţii distribuite pe scară largă, lipsind o serie de facilităţi precum scalabilitatea, serviciile de bază (i.e. servicii de nume, servicii de regăsire a resurselor etc.) sau securitatea. Programele CGI nu sunt scalabile deoarece fiecare instanţă a acestora necesită un proces separat, executat pe serverul Web pentru a procesa cererile solicitate de un client particular. Serviciile sunt limitate deseori la accesarea unuia sau mai multor sisteme de gestiune a bazelor de date, uzual tranzacţiile efectuându-se necriptat. Aceeaşi situaţie persistă chiar şi în cazul unor soluţii recente, orientate-obiect, bazate pe infrastructuri precum CORBA/IIOP [OMG] sau OLE/DCOM [MSDN] ori pe medii de programare Java [Java].

Continua dezvoltare a unor metodologii şi instrumente obiectuale orientate spre Web prezintă următoarele avantaje principale, conform cu [Tanenbaum, 2001, Tanenbaum, 2003, Alboaie & Buraga, 2002]:

- extensibilitatea;
- interoperabilitatea pe diverse platforme;
- analiza, proiectarea şi dezvoltarea de componente programabile;
- reutilizarea componentelor software;
- oferirea de servicii distribuite (ca de exemplu, servicii privind numirea, transparenţa, replicarea, regăsirea etc.);
- mai buna utilizare a resurselor de sistem.

În strânsă legătură cu familia de limbaje XML şi cu metodologiile obiectuale, unul dintre paşii importanţi ai evoluţiei spaţiului WWW îl reprezintă *serviciile Web* bazate pe XML. Serviciile Web [Gorman, 2001, Vasudevan, 2001] permit ca elemente programabile să fie plasate pe siturile Web, alţi programatori putând accesa funcţionalităţile acestor programe, în manieră distribuită şi directă, pentru orice număr potenţial de sisteme independente, prin folosirea unor standarde precum XML şi HTTP. Aceasta va conduce la adoptarea unei alte viziuni

privind programarea Web şi la construirea unei infrastructuri complexe pe baza căreia să se dezvolte diverse aplicaţii specifice, precum servicii Web semantice [Davies *et al.*, 2003] şi agenţi [Bradshow, 1997, Wooldridge & Ciancarini, 2000] – conform celor detaliate în capitolul 4.

O altă direcţie în continuă dezvoltare este Grid-ul, colecţie eterogenă de informaţii şi aplicaţii distribuite (*e.g.*, baze de date, biblioteci digitale, instrumente, servicii de control al fluxului de date etc.), considerată drept resursă unică, procesarea datelor decurgând independent de platformă [Buyya, 2002, Foster & Kesselan, 1999]. Configuraţiile realizate de componentele unui Grid sunt dinamice, realizate *ad-hoc*, în funcţie de necesităţi sau atunci când sunt disponibile în Internet.

1.4 Concluzii

Consorţiul Web [W3C] – forţa motrice din spatele meta-limbajului XML – consideră că misiunea sa este aceea de a dirija *evoluţia* spaţiului World-Wide Web.

Deşi a împărtăşit mai multe idei comune Internetului [ISOC] – ca "software-ul gratuit se răspândeşte mai repede", "protocoalele proaste imită; protocoalele bune fură" şi "sistemele ASCII proliferează mai rapid decât cele binare" – Web-ul a impus o strategie inovatoare: "autodescrierea". *Spaţiul WWW poate fi construit peste el însuşi*.

În esenţă, apariţia meta-limbajului XML în spectrul formatelor de date pentru Web pune capăt disputei legate de modalităţile de structurare a documentelor în vederea realizării viziunii originare a creatorilor Web-ului [Berners-Lee, 1999]. La ora actuală, XML reprezintă unica modalitate existentă pentru adnotarea complexă a datelor şi pentru oferirea unui suport independent de platforma hardware/software, deschis, în vederea realizării de aplicaţii Web de manipulare a resurselor multimedia.

Format de document	Sintaxă	Stil	Structură	Semantică
ASN.1	Binară	–	Tip-lungime-valoare	Dependentă de aplicaţie
Text	ASCII, Unicode,...	Linii	–	Limbaj natural
`troff`	Text lizibil	Directive interne	Secţiuni, pagini	Compunere
TEX	Program lizibil	LATEX	Secţiuni, pagini	Compunere
PostScript	Limbaj de comenzi	–	Pagini	Desenare
Rich Text Format	Text opac	Directive extensibile	Caractere, paragrafe	–
Formatare HTML	Text lizibil	Directive imbricate	Prezentaţională	–
Structură HTML	Text lizibil	CSS	Declarativă	Fixă (<address>)
XML	Text lizibil	CSS, XSL	Declarativă	Extensibilă
RDF	Text XML	–	Declarativă	Scheme metadate, descrieri semantice

Tabela 1.1: Comparaţie între diverse formate de documente în evoluţia spaţiului World-Wide Web ([Buraga, 2002a])

Capitolul 2

Web-ul semantic

Capitolul de faţă prezintă Web-ul semantic şi arhitectura sa conceptuală, insistând asupra specificării metadatelor şi ontologiilor prin intermediul limbajelor RDF, OIL, DAML+OIL şi OWL.

2.1 Prezentare generală

2.1.1 Preambul

ÎN cadrul acestui capitol, vom face o trecere în revistă a principalelor aspecte referitoare la Web-ul semantic, insistând asupra modalităţilor de exprimare atât a relaţiilor stabilite între resursele disponibile pe Web, cât şi a metadatelor privitoare la aceste resurse.

Astfel, ne vom concentra asupra ilustrării trăsăturilor definitorii ale *RDF* (*Resource Description Framework*) – componentă indispensabilă a Web-ului semantic, pe baza căreia vom putea modela relaţiile spaţio-temporale dintre resursele multimedia ale unui sit Web. Acest model va fi prezentat în cadrul capitolului 3, secţiunea 3.2.3.3. De asemenea, vom trece în revistă manierele de asociere a metadatelor şi limbajele de

exprimare via XML a ontologiilor, descriindu-se în special *OWL* (*Web Ontology Language*).

2.1.2 Caracterizare şi direcţii de interes

Scopurile originare principale ale spaţiului WWW au fost acelea de a oferi o modalitate de comunicare inter-umană prin partajarea cunoştinţelor şi posibilitatea exploatării în manieră distribuită a puterii de calcul a computerelor conectate la Internet [Berners-Lee, 1989].

Spaţiul cibernetic a fost iniţial conceput pentru a uşura regăsirea de către calculator a oricărei date indiferent de localizarea ei, fără a se pune problema înţelegerii semnificaţiei acesteia de către maşină. Din cauza volumului tot mai mare de informaţii prezent pe Web este dificil de a automatiza regăsirea lor inteligentă, cu atât mai puţin de către operatorul uman. Una dintre priorităţile Consorţiului Web este aceea de a pune la dispoziţie o modalitate, bazată pe actualele tehnologii XML, de prelucrare de către calculator a informaţiilor. Soluţia este prezentată în [Berners-Lee *et al.*, 2001] prin intermediul unui scenariu despre ceea ce ar trebui să fie cunoscut sub numele de *Web semantic*. În cadrul acestui scenariu vizionar, dispozitive inteligente partajează cunoştinţe privitoare la propriile funcţionalităţi şi la contextul în care îşi desfăşoară activitatea, utilizând reguli de inferenţă şi metadate pentru a (re)găsi informaţii soliticitate de utilizatorii umani.

Ideea de bază este aceea ca spaţiul World-Wide Web să reprezinte o pânză consistentă de relaţii stabilite între obiecte identificabile (de exemplu, în prezent prin identificatori uniformi de resurse, iar ulterior prin nume stabilite de utilizatori), dezvoltarea viitoare a Web-ului fiind focalizată asupra maşinilor, nu numai asupra oamenilor.

Astfel, printre deziderateле Web-ului semantic se pot enumera (conform [Berners-Lee *et al.*, 2001] şi [Davies *et al.*, 2003]):

- asocierea de semantici legăturilor dintre resurse, cu posibilitatea extinderii acestor semantici;
- resursele Web să poată fi extinse şi clasificate, pentru aceasta trebuind a fi adoptate specificaţii conceptuale;

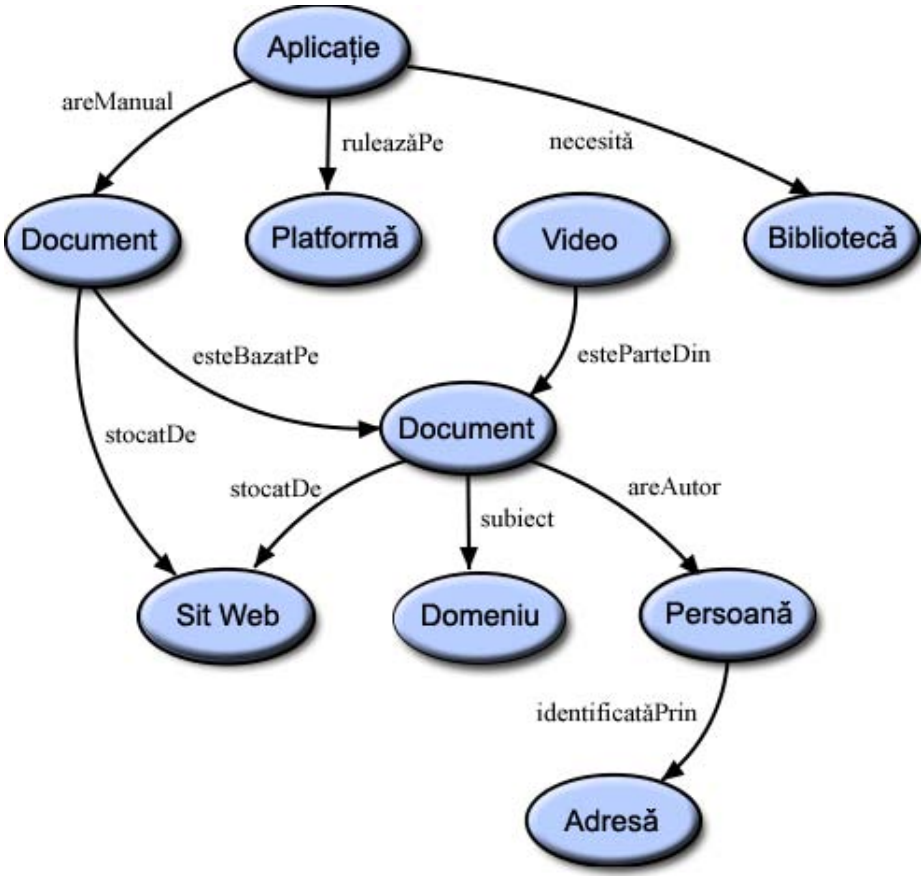

Figura 2.1: În cadrul Web-ului semantic, resursele şi legăturile au asociate descrieri semantice

- la nivel programatic, să existe entităţi capabile să proceseze în manieră inteligentă informaţiile şi să poată raţiona, oferind maşinilor şi oamenilor servicii complexe (*e.g.*, căutarea datelor, regăsirea unor tipuri de resurse, monitorizarea activităţii aplicaţiilor, filtrarea informaţiilor etc.);

- partajarea de către utilizatori a cunoştinţelor, indiferent de modul de stocare şi de reprezentare a acestora.

După cum se va putea observa din cele prezentate în capitolele ur-

mătoare, tema lucrării de faţă se înscrie în sfera de interes a Web-ului semantic, cercetările efectuate având în vedere oferirea unor soluţii – atât formale, cât şi de proiectare şi implementare – flexibile, bazate pe tehnologiile Web actuale, pentru descrierea şi regăsirea resurselor multimedia localizate pe Web.

Conform celor descrise în lucrările [Baumgartner & Furbach, 2002], [Davies *et al.*, 2003] şi [Strang *et al.*, 2003], una dintre actualele direcţii de interes ale comunităţilor academice şi industriale circumscrise Web-ului semantic este aceea a *managementului cunoştinţelor* (*knowledge management*), mai ales în contextul intranet-urilor organizaţiilor sau în domeniul învăţământului virtual (*e-learning*). Problemele cu care se confruntă cercetătorii sunt cele legate de:

- *căutarea informaţiilor* – metodele de căutare bazată pe cuvinte-cheie conduc la rezultate nerelevante, neluându-se în calcul structura conţinutului resurselor dorite sau relaţiile acestora cu alte resurse înrudite[1];
- *extragerea informaţiilor* – instrumentele de extragere automată a informaţiilor provenind din surse multiple nu sunt încă suficient de flexibile să poată prelua informaţii din surse non-textuale sau din domenii de cunoaştere diferite;
- *mentenanţa depozitelor de cunoştinţe* – pentru ca informaţiile (organizate uzual în colecţii de documente XML) să fie consistente, corecte şi facil de actualizat trebuie găsite reprezentări semantice procesabile de către calculator. Acest aspect va conduce la verificarea formală a consistenţei cunoştinţelor şi la instituirea unor relaţii de încredere în resursele disponibile pe Web – aşa-numitul *Web of trust* [Berners-Lee *et al.*, 2001];
- *generarea automată de documente* – acest aspect va conduce la reconfigurarea automată a modalităţilor de prezentare a conţinutului siturilor Web, pe baza profilului cognitiv al vizitatorilor (*e.g.*, preferinţe, moduri de operare, experienţă etc.) sau al altor factori relevanţi.

[1] A se vedea şi cele discutate în cadrul secţiunii 3.3.2.2.

2.1.3 Structura Web-ului semantic

În vederea soluţionării aspectelor prezentate mai sus, Web-ul semantic adoptă o viziune stratificată (a se vedea figura 2.2), prezentată iniţial în [Berners-Lee, 2000].

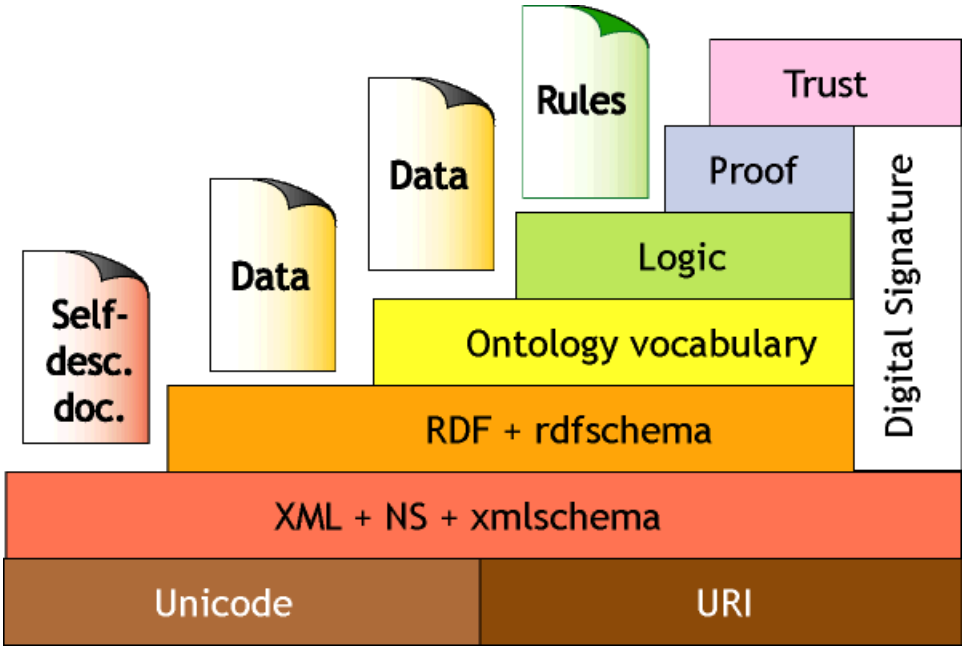

Figura 2.2: Nivelurile de specificare a Web-ului semantic [Berners-Lee, 2000]

Pentru fiecare nivel al arhitecturii stratificate a Web-ului semantic, Consorţiul Web a standardizat sau urmează să standardizeze diferite limbaje bazate pe familia XML. Vom discuta pe scurt fiecare dintre acestea.

2.1.3.1 Interschimbul "inteligent" de date via XML

Meta-limbajul XML [Bray *et al.*, 2004] reprezintă nivelul de bază pentru realizarea schimbului de date în manieră structurată sau semi-structurată, cu concursul identificatorilor uniformi de resurse şi a spaţiilor de nume. După cum am putut vedea şi în capitolul 1, folosind

definiţii ale tipurilor de documente (DTD) sau scheme XML, datele marcate cu ajutorul elementelor şi atributelor XML pot fi validate formal.

În ceea ce priveşte Web-ul semantic, XML poate avea utilizări multiple, cele mai importante fiind prezentate în continuare, conform [Davies *et al.*, 2003, Patel-Schneider & Simeon, 2002]:

- XML poate oferi o sintaxă serializată pentru ale limbaje de marcare, reprezentând astfel *lingua franca* a Web-ului semantic;
- separarea formei de conţinut – folosind transformări XSL, un anumit vocabular XML poate fi tradus fie în XHTML în vederea reprezentării la nivelul clientului (navigatorului), fie în alte limbaje XML (*e.g.*, pentru translatarea ontologiilor);
- XML poate fi considerat ca fiind un format universal de schimb de date între aplicaţii distribuite, aliniate Web-ului semantic, precum agenţii software; o astfel de soluţie este prezentată în secţiunea 4.4.2.1 şi publicată în lucrările [Alboaie & Buraga, 2003a] şi [Buraga & Alboaie, 2004].

Meta-limbajul XML nu poate exprima semantici, ci doar reflectă o manieră uniformă sintactică de structurare a datelor. În fapt, schemele XML nu oferă decât o descriere formală a convenţiilor sintactice de modului de apariţie a componentelor unui document XML.

2.1.3.2 Exprimarea metadatelor

Din moment ce, folosind numai marcatori XML, nu putem descrie resursele, un prim pas este asocierea de *metadate*, adică date privitoare la date. Forma de exprimare a metadatelor trebuie să aibă în vedere citirea şi procesarea acestora de către maşină, într-o manieră independentă de aplicaţie sau de platformă.

De asemenea, trebuie să se pună la dispoziţie o modalitate de asociere (de dorit via unor mecanisme externe) a metadatelor resurselor considerate. Una dintre soluţii este aceea de a exprima diferite aserţiuni privitoare la resurse, metadatele fiind considerate proprietăţi ale resurselor în cauză.

Drept soluţie, Consorţiul Web propune un cadru de descriere a resurselor, concretizat în specificaţia *RDF* (*Resource Description Framework*) [Lassila & Swick, 1999], în prezent recomandarea oficială fiind [Beckett, 2004]. A se vedea şi [Manola & Miller, 2004].

Astfel, la nivel general RDF reprezintă un cadru menit să ofere suport pentru procesarea în manieră inteligentă a metadatelor, oferind interoperabilitatea între diverse aplicaţii care realizează schimb de informaţii, în sensul înţelegerii de către maşină a semanticii acestora.

Bază pentru limbaje mai complexe de reprezentare a datelor, RDF îşi găseşte loc în utilizări precum [Berners-Lee *et al.*, 2001]:

- *inspectarea resurselor*, oferind noi capabilităţi motoarelor de căutare, mai ales în ceea ce priveşte clasificarea siturilor Web;
- *catalogarea datelor* pentru descrierea şi/sau evaluarea conţinutului şi relaţiilor între diverse informaţii stocate de o bibliotecă electronică, sit Web etc.;
- *agenţi inteligenţi*, facilitând schimbul şi partajarea cunoştinţelor;
- *descrierea drepturilor de proprietate intelectuală* a resurselor Web;
- *securitate personală sau generală* a datelor (oferind suport pentru ataşarea de *semnături digitale* utile în comerţul electronic, tranzacţii economice şi juridice etc.).

RDF foloseşte limbajul XML pentru reprezentarea sintactică a metadatelor. Unul dintre scopurile cadrului este de a face posibilă specificarea semantică a datelor, bazată pe XML [Bray *et al.*, 2004], printr-o metodă standardizată, independentă de maşină, extensibilă. RDF şi XML sunt complementare în acest sens.

Influenţa RDF se poate întrevedea în *structurarea inteligentă a documentelor* (realizată prin intermediul limbajelor bazate pe XML), în *reprezentarea cunoştinţelor* (*knowledge representation*), în *standardizarea Web-ului*. Alte arii de interes ar fi limbajele orientate-obiect şi de modelare a cunoştinţelor sau bazele de date distribuite.

Vom detalia limbajul RDF în cadrul subcapitolului 2.2.

Trebuie menţionat faptul că pentru domenii restrânse pot fi asociate metadate prin intermediul unor construcţii XML de genul celor exprimate de *DCMI* (*Dublin Core Metadata Initiative*) [Purl], care pot fi integrate în cadrul construcţiilor RDF[2]. DCMI defineşte o schemă având 15 proprietăţi de bază care pot fi utilizate pentru descrierea resurselor Web.

Exemplul 3 Metadatele asociate unui document *PDF* (*Portable Document Format*) reprezentând un articol disponibil *on-line* pot fi exprimate astfel, folosind şi contrucţii DCMI:

```
<rdf:RDF
   xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
   xmlns:iX='http://ns.adobe.com/iX/1.0/'>

 <!-- descrieri ale metadatelor Adobe (pentru PDF) -->
 <rdf:Description
     about='http://www.infoiasi.ro/~busaco/
             publications/articles/regexp.pdf'
     xmlns='http://ns.adobe.com/pdf/1.3/'
     xmlns:pdf='http://ns.adobe.com/pdf/1.3/'>
  <pdf:CreationDate>
     2002-07-16T13:13:03Z
  </pdf:CreationDate>
  <pdf:ModDate>
     2002-09-18T15:10:22+04:00
  </pdf:ModDate>
  <pdf:Producer>
     Acrobat Distiller
  </pdf:Producer>
  <pdf:Author>
     Sabin Corneliu Buraga,
     Victor Tarhon-Onu
  </pdf:Author>
  <pdf:Creator>
     QuarkXPress(tm) 5.01
```

[2] Pentru amănunte se poate consulta [Kokkelink & Schwänzl, 2002].

```
 </pdf:Creator>
 <pdf:Title>
    Expresii regulate in Perl
 </pdf:Title>
 <pdf:Keywords>
    Perl, Linux, programare,
    expresii regulate, limbaje formale
 </pdf:Keywords>
</rdf:Description>

<!-- descrieri ale metadatelor Adobe (pentru XAP) -->
<rdf:Description
    about='http://www.infoiasi.ro/~busaco/
            publications/articles/regexp.pdf'
    xmlns='http://ns.adobe.com/xap/1.0/'
    xmlns:xap='http://ns.adobe.com/xap/1.0/'>
 <xap:CreateDate>
    2002-07-16T13:13:03Z
 </xap:CreateDate>
 <xap:ModifyDate>
    2002-09-18T15:10:22+04:00
 </xap:ModifyDate>
 <xap:Author>
    Sabin Corneliu Buraga,
    Victor Tarhon-Onu
 </xap:Author>
 <xap:MetadataDate>
    2002-09-18T15:10:22+04:00
 </xap:MetadataDate>
 <!-- titluri alternative -->
 <xap:Title>
  <rdf:Alt>
    <rdf:li xml:lang='x-default'>
       Expresii regulate in Perl
    </rdf:li>
    <rdf:li xml:lang='en-US'>
       Perl regular expressions
    </rdf:li>
```

```
   </rdf:Alt>
  </xap:Title>
 </rdf:Description>

 <!-- descrieri folosind DCMI -->
 <rdf:Description
     about='http://www.infoiasi.ro/~busaco/
              publications/articles/regexp.pdf'
     xmlns='http://purl.org/dc/elements/1.1/'
     xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <dc:creator>
     Sabin Corneliu Buraga,
     Victor Tarhon-Onu
  </dc:creator>
  <dc:title>
     Expresii regulate in Perl
  </dc:title>
 </rdf:Description>

</rdf:RDF>
```

De menţionat faptul că un alt exemplu de astfel de descriere este disponibil în lucrarea [Buraga, 2001a].

O specificaţie care permite asocierea de descrieri siturilor Web, prin intermediul aserţiunilor RDF, este *RSS* (*Rich Site Summary*) [RSS]. Specificaţiile RSS sunt disponibile în mai multe versiuni (0.9x, 1.0 şi 2.0).

Exemplul 4 Noutăţile apărute pe situl *Romanian Web Developers* – `http://rowd.org/` – sunt stocate într-un fişier RSS având următoarea structură:

```
<?xml version="1.0" ?>
<rss version="0.91">
   <channel>
      <!-- informatii despre sit -->
```

```
      <title>Noutati pe situl ROWD</title>
      <link>http://rowd.org</link>
      <description>Ultimele noutati aparute
         pe situl Romanian Web Developers.
      </description>
      <language>ro</language>
      <pubDate>10/25/03</pubDate>
      <image>
         <title>Romanian Web Developers</title>
         <url>http://rowd.zuavra.net/img/logo/1a.gif</url>
         <link>http://rowd.org/</link>
         <width>100</width>
         <height>31</height>
      </image>

      <!-- stirile -->
      <item>
         <title>Aplicatii Web la cheie.
            Studii de caz implementate in PHP</title>
         <link>http://www.infoiasi.ro/~phpapps/</link>
         <description>...</description>
      </item>
      <!-- eventual, alte elemente 'item' -->
   </channel>
</rss>
```

Una dintre probleme în folosirea metadatelor o reprezintă reutilizarea metadatelor în contexte multiple. O soluţie – prezentată în lucrarea [Ohia, 2001] – este aceea de a modulariza seturile de construcţii XML folosite pentru exprimarea metadatelor şi de a defini diverse reguli pentru refolosirea acestora.

2.1.3.3 Exprimarea ontologiilor

O definiţie a conceptului de ontologie este următoarea (conform lucrărilor [Gruber, 1993] şi [Guarino & Giaretta, 1995]):

Definiţia 1 *O* ontologie *reprezintă conceptualizarea unui domeniu de cunoaştere într-un format destinat a fi procesat de calculator, format modelând entităţi, atribute, relaţii şi axiome.*

Din punct de vedere formal, conceptualizările se pot defini în maniera prezentată în continuare, conform [Guarino, 1998].

Definiţia 2 *O* conceptualizare *este exprimată printr-o structură* $\langle D, \mathcal{R} \rangle$*, unde* D *reprezintă un domeniu, iar* $\mathcal{R}$ *desemnează o mulţime de relaţii peste* D*.*

Dacă relaţiile ordinare sunt definite pe un domeniu particular, relaţiile conceptuale sunt specificate peste un *spaţiu de domenii* (*domain space*).

Definiţia 3 *Un* spaţiu de domenii *este o structură de forma* $\langle D, W \rangle$*, unde* D *este un domeniu şi* W *o mulţime de stări maximale ale domeniului.* W *se numeşte mulţimea* lumilor posibile.

De exemplu, dacă D reprezintă piesele dispuse pe o tablă de şah, atunci W poate desemna mulţimea tuturor configuraţiilor valide ale acestor piese pe tabla considerată.

Definiţia 4 *Dat un spaţiu de domenii* $\langle D, W \rangle$*,* relaţia conceptuală ρ^n *de aritate* n *peste* $\langle D, W \rangle$ *se defineşte ca o funcţie totală* $\rho^n : W \rightarrow 2^{D^n}$ *de la mulţimea lumilor posibile* W *la o mulţime de relaţii* n*-are ordinare peste domeniul* D*.*

Definiţia 5 *Pentru o relaţie conceptuală oarecare* ρ*, mulţimea* $\mathbf{E}_\rho = \{\rho(w), w \in W\}$ *va conţine* extensiile admisibile *ale lui* ρ*.*

Definiţia 6 *O* conceptualizare *pentru domeniul* D *este un triplu* $\mathbf{C} = \langle D, W, \mathfrak{R} \rangle$*, unde* $\mathfrak{R}$ *reprezintă o mulţime de relaţii conceptuale peste spaţiul de domenii* $\langle D, W \rangle$*.*

Ontologiile pot pune la dispoziţie conceptualizări complexe ale domeniului de lucru a unei organizaţii, reprezentând conceptele de bază şi relaţiile dintre diversele activităţi desfăşurate de entităţile acelei organizaţii.

Ca exemple de ontologii putem enumera ontologia reprezentând domeniul academic – menită a modela persoane, proiecte, publicaţii, evenimente ştiinţifice şi subiecte de cercetare –, dezvoltată de grupul AKT (Advanced Knowledge Technologies) [O'Hara *et al.*, 2001]. Un alt exemplu provine din domeniul *e-travel*, dezvoltându-se o ontologie care să exprime conceptele vehiculate în cadrul unui sistem *on-line* oferind formule de petrecere a zilelor de vacanţă [Wright *et al.*, 2003].

Ontologiile pot fi de mai multe tipuri, în funcţie de domeniul de interes al comunităţii cercetătorilor care definesc şi utilizează o anumită ontologie. Un prim aspect important este cel al *nivelului de descriere* pe care îl poate oferi o ontologie (plecând de la simple lexicoane şi ajungând la tezaure organizate pe categorii de termeni sau taxonomii complexe). Ontologiile pot diferi şi în ceea ce priveşte scopul şi contextul utilizării conţinutului pe care-l modelează. De exemplu, se pot crea ontologii specializate, folosite într-un domeniu de interes precum chirurgia, dar şi ontologii generale în care informaţiile provin din orice domeniu. Acest din urmă caz poate fi întâlnit la procesarea limbajului natural.

În ceea ce priveşte maniera de exprimare a unei ontologii, există definite diverse *limbaje de specificare*[3], plecând de la limbaje de programare logică precum Prolog sau derivate din Prolog (*e.g.*, Golog, ConGolog) şi mergând până la limbaje specializate, cum ar fi *KIF* (*Knowledge Interchange Format*) [Genesereth & Fikes, 1992] sau succesorul său *CL* (*Common Logic*). Alte limbaje sunt bazate pe logici descriptive – este cazul limbajelor prezentate mai jos: OIL, DAML+OIL şi OWL.

[3] Pentru un studiu comparativ, se pot consulta [Corcho & Gomez-Perez, 2000], [Grosof *et al.*, 2003] şi [Horrocks & Patel-Schneider, 2003].

Aspectele formale privitoare la ontologii şi la conceptualizări sunt detaliate în [Guarino, 1998].

RDF Schema

Specificaţia *RDFS* (*RDF Schema*) [Brickley & Guha, 2000] extinde modelul RDF pentru a oferi posibilitatea interpretării la nivel semantic a proprietăţilor exprimate prin intermediul modelului RDF, fără a impune constrângeri în ceea ce priveşte modul de exprimare sintactică a unei descrieri RDF asociate unei resurse.

Se permit aşadar definirea unui vocabular particular privitor la datele exprimate în RDF şi specificarea tipurilor de obiecte care constituie aserţiunile RDF/XML. În fapt, RDFS oferă un sistem de tipuri de bază pentru modelele exprimate în RDF [Davies *et al.*, 2003], adică oferă premisele specificării de *ontologii* simple via primitive de modelare a cunoştinţelor. Principalele contribuţii ale RDFS sunt cele legate de punerea la dispoziţie a unei sintaxe standardizate pentru specificarea de ontologii şi a unui set standard de primitive de modelare, precum relaţii de tipul *subclasă* (*subclass-of*) sau *instanţă* (*instance-of*).

Mai multe detalii vor fi prezentate în subcapitolul 2.2 al capitolului de faţă.

Anumite construcţii complicate pentru exprimarea unor tipuri complexe de cunoştinţe nu pot fi modelate folosind RDF şi/sau RDFS. De exemplu, este dificil a se exprima exact ce tipuri de valori pot lua diverse instanţe de resurse (*e.g.*, titlurile cărţilor stocate de o bibliotecă digitală sunt şiruri de caractere, iar cantitatea de volume dintr-un anumit domeniu este desemnată de un număr întreg pozitiv). Printre alte critici aduse modelului RDFS se pot enumera imposibilitatea exprimării proprietăţilor algebrice relaţionale (precum tranzitivitatea ori simetria) sau lipsa unui mecanism de procesare a interogărilor referitoare la aserţiunile RDF [Anutaryia *et al.*, 2001].

OIL (Ontology Inference Layer)

Un prim pas în rezolvarea unor astfel de probleme l-a constituit *OIL* (*Ontology Inference Layer*) [Fensen *et al.*, 2000], construit pe baza lim-

bajului *XOL* (*XML Ontology Language*). Scopurile proiectării limbajului OIL au avut în vedere, printre altele:

- modelarea unei cât mai largi varietăţi de ontologii;
- oferirea unei semantici formale pentru a facilita interpretarea de către maşină a acestei semantici;
- oferirea unor servicii de raţionament corect (*sound*) şi complet;
- maximizarea compatibilităţii cu standardele deja existente (XML şi RDF).

Modelul formal de bază l-a constituit logica cu predicate, relaţiile dintre resurse fiind modelate prin intermediul predicatelor. Conceptul central se axează asupra *claselor* (*classes* sau *frames*) având asociate diverse proprietăţi denumite *atribute*. Atributele pot fi aplicate numai asupra claselor pentru care au fost definite. Putem asocia acelaşi nume de atribut unor clase diferite, impunând anumite *restricţii* ale valorilor atributelor considerate (*range restrictions*). O clasă oferă un context pentru modelarea unui aspect specific al unui *domeniu* (*domain*) [Horrocks *et al.*, 2000, Davies *et al.*, 2003].

OIL se bazează pe noţiunea de *concept* şi pe definirea de superclase şi atribute. Relaţiile pot fi definite atât între atribute şi clasa din care fac parte, cât şi între entităţi independente având domenii şi valori specifice. Ca şi clasele, relaţiile constituite pot fi grupate în ierarhii. În acest mod, OIL permite specificarea de ontologii, fiecare ontologie exprimată în OIL putând fi adnotată cu ajutorul metadatelor, folosind eventual DCMI. Sintaxa de bază utilizează aserţiuni RDF(S).

Modelul OIL este organizat la rândul său pe niveluri:

- nivelul de bază (*Core OIL*) coincide cu RDFS, pentru a facilita agenţilor care procesează construcţii RDF interpretarea ontologiilor exprimate în OIL;
- nivelul standard (*Standard OIL*) intenţionează să ofere un set de primitive indispensabile în modelarea cunoştinţelor, permiţând

formalizarea semanticilor ontologiilor considerate prin intermediul unor mecanisme logice reprezentate de diverse *logici descriptive* (*Description Logics*)[4], dintre care se poate menţiona logica $\mathcal{SHIQ}$ [Horrocks *et al.*, 2001];

- nivelul de instanţiere (*Instance OIL*) oferă posibilitatea integrării diferitelor instanţe de ontologii în alte sisteme, precum sistemele de management al bazelor de date;

- nivelul superior (*Heavy OIL*) adaugă facilităţi pentru reprezentarea cunoştinţelor folosind limbaje bazate pe reguli.

O ontologie este exprimată în OIL prin intermediul unui container de ontologii (*ontology container*) şi a unei definiţii. La nivel sintactic, containerul este exprimat prin elemente DCMI. Partea care defineşte ontologia constă dintr-o directivă de import I opţională, o regulă de bază R opţională, o serie de definiţii de sloturi (proprietăţi) S şi o serie de definiţii de axiome A.

O definiţie de clasă (*class-def*) asociază unui nume de clasă o descriere a acelei clase. În RDFS, acest lucru se specifică prin elementul `<rdfs:Class>`. Tipul descrierii poate fi unul *primitiv* (condiţiile specificate pentru membrii clasei sunt necesare, dar nu şi suficiente) sau *definit* (condiţiile sunt necesare şi suficiente). În plus, trebuie specificate:

- o directivă *subclass-of* – defineşte o listă de expresii de clasă (nume de clase, constrângeri de proprietăţi, expresii logice).
  Construcţia RDFS analoagă este `<rdfs:subClassOf>`.

  O restricţie de slot (*slot-constraint*) reprezintă o listă de una sau mai multe restricţii care se vor aplica proprietăţilor (sloturilor). Ca exemple de constrângeri se pot menţiona *has-value*, *value-type* sau *max-cardinality*.

- o definiţie de slot *slot-def* – asociază unui nume de slot o descriere a slotului care specifică diverse restricţii globale (domeniu, valori,

[4] A se vedea şi [Baader *et al.*, 2002].

proprietăţi de tranzitivitate sau simetrie etc.). Un slot la rândul său poate include sub-sloturi (sub-constrângeri).

- o mulţime de axiome, fiecare axiomă specificând fapte suplimentare despre clasele ontologiei (de exemplu, clasele `Imagine` şi `Sunet` sunt disjuncte în cadrul unei ontologii privitoare la multimedia). Printre axiomele valide care pot fi utilizate se enumeră *disjoint* şi *equivalent*.

  Axiomele pot fi considerate obiecte, putând fi astfel modelate în RDF[5].

Lucrările [Horrocks *et al.*, 2000] şi [Fensen *et al.*, 2000] prezintă descrierea la nivel formal a modelului OIL. Alte consideraţii referitoare la formalizarea limbajului sunt prezentate în [Kamps & Marx, 2002].

Exemplul 5 Prezentăm un fragment de ontologie modelând regnul animal (după [Broekstra *et al.*, 2001]):

```
(class-def defined ierbivor
   subclass-of animal
     slot-constraint
       value-type (planta or
         (slot-constraint este-fragment-din
            has-value planta))
class-def elefant
   sub-class-of ierbivor mamifer
   slot-constraint consuma
       value-type planta
   slot-constraint culoare
       has-filler "gri"
```

Folosind XML şi RDFS, cele de mai sus se exprimă astfel:

```
<rdfs:Class rdf:ID="elefant">
   <rdfs:subClassOf rdf:resource="#mamifer" />
   <rdfs:subClassOf rdf:resource="#ierbivor" />
```

[5] Detalii în [Staab & Mädche, 2000] şi [Broekstra *et al.*, 2001].

```
   <oil:hasPropertyRestriction>
       <oil:ValueType>
           <oil:onProperty
              rdf:resource="#consuma" />
           <oil:toClass rdf:resource="#planta" />
       </oil:ValueType>
       <oil:hasFiller>
           <oil:onProperty
              rdf:resource="#culoare" />
           <oil:stringFiller>
              gri
           </oil:stringFiller>
       </oil:hasFiller>
   </oil:hasPropertyRestriction>
</rdfs:Class>
```

DAML+OIL

Succesor al OIL, *DAML+OIL* [Broekstra *et al.*, 2001, DAML+OIL] reprezintă o inițiativă de a exprima ontologiile direct prin mecanismele oferite de specificațiile RDFS, dar plecând de la *DAML* (*DARPA Agent Markup Language*) [Hendler & McGuinness, 2001], limbaj menit a desemna o manieră standard de interschimb de date între agenții inteligenți. Semantica DAML+OIL este definită folosind un model bazat pe *KIF* (*Knowledge Interchange Format*) [Genesereth & Fikes, 1992]. Construcțiile DAML+OIL pot exprima un subset al clauzelor propoziționale Horn, aceste clauze permițând specificarea constrângerilor și regulilor de raționament care pot fi manipulate de aplicațiile Web-ului semantic.

Limbajul DAML+OIL descrie aserțiunile privitoare la tipul (clasa) unei entități sau la relațiile stabilite între diverși membri ai unei clase prin intermediul exclusiv al contrucțiilor RDF. În DAML+OIL se poate specifica, în manieră explicită, faptul că două componente ale unei ontologii sunt diferite sau similare, iar în plus nu e obligatoriu să se definească o anumită relație între ele.

Tipurile de date care pot fi prezente într-un document DAML+OIL sunt analoage celor din XML Schema [Fallside, 2001] şi pot fi folosite în specificarea restricţiilor atributelor (sloturilor) şi a mulţimilor de valori permise.

Limbajului DAML+OIL i-a fost asociată o semantică axiomatică, formalismele implicate fiind detaliate în [Fikes & McGuinness, 2001]. Cercetările s-au concentrat şi asupra utilizării ontologiilor exprimate prin DAML+OIL în contextul ingineriei software [Davies *et al.*, 2003].

Exemplul 6 Folosind fragmentul de ontologie descrisă în exemplul 5, construcţiile DAML+OIL sunt:

```
<daml:Ontology rdf:about="">
   <daml:versionInfo>1.0</daml:versionInfo>
</daml:Ontology>
<daml:Class rdf:about="#elefant">
   <rdfs:subClassOf>
      <!-- subclasa, vazuta ca intersectie
           dintre clasele 'mamifer' si 'ierbivor' -->
      <daml:Class>
         <daml:intersectionOf
            rdf:parsetype="daml:Collection">
            <daml:Class rdf:about="#mamifer" />
            <daml:Class rdf:about="#ierbivor" />
         </daml:intersectionOf>
      </daml:Class>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty
            rdf:resource="#consuma" />
         <daml:hasClass rdf:resource="#planta" />
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty
            rdf:resource="#culoare" />
```

```
        <daml:hasValue>gri</daml:hasValue>
      </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>
```

OWL (Web Ontology Language)

Pe baza DAML+OIL, Consorţiul Web dezvoltă limbajul *OWL* (*Web Ontology Language*) [Dean & Schreiber, 2004], actualmente recomandare oficială.

Ca şi precedentele limbaje, OWL este structurat pe niveluri:

- *OWL Lite* oferă posibilitatea descrierii unor ierarhii de clasificare care nu posedă restricţii complicate. *OWL Lite* va putea fi folosit pentru specificarea dicţionarelor de termeni şi a altor taxonomii.
- *OWL DL* are drept scop principal oferirea unei expresivităţi crescute, fără a se pierde completitudinea computaţională (se garantează astfel că toate declaraţiilor vor putea fi procesate de către maşină). Modelul teoretic al acestui nivel este bazat pe logica descriptivă [Baader *et al.*, 2002].
- *OWL Full* oferă un maximum de expresivitate şi libertatea sintactică a RDF, fără a se pune problema computabilităţii.

Remarca 1 Fiecare dintre sublimbajele prezentate este o extensie a celui aflat pe nivelul imediat inferior. Au loc relaţiile următoare (relaţiile inverse nu sunt satisfăcute):

- Orice ontologie exprimată în *OWL Lite* poate fi exprimată şi în *OWL DL*.
- Orice ontologie exprimată în *OWL DL* poate fi exprimată şi în *OWL Full*.
- Orice concluzie validă deductibilă în *OWL Lite* este validă şi în *OWL DL*.

- Orice concluzie validă deductibilă în *OWL DL* este validă şi în *OWL Full*.

O ontologie OWL este compusă din clase, instanţe de clase şi relaţiile dintre aceste instanţe. Succesul unei ontologii depinde în mare măsură de abilitatea de a raţiona despre indivizii unei clase. OWL oferă posibilitatea descrierii, via construcţii XML, a claselor din care fac parte indivizi specifici şi a proprietăţilor care-i caracterizează. În *OWL Full* o clasă reprezintă un obiect din lumea reprezentată de ontologia modelată.

Filosofia limbajului este diferită de aceea a unui sistem de baze de date, neimpunându-se *a-priori* o cunoaştere completă a domeniului de interes. Clasele şi proprietăţile pot fi specificate în forme diferite, iar declaraţiile privitoare la indivizi nu trebuie obligatoriu să fie incluse în acelaşi document în care s-a specificat o anumită ontologie[6].

O clasă particulară poate fi derivată din una mai generală prin intermediul unei construcţii `rdfs:subClassOf`. Dacă X este subclasă a clasei Y, atunci orice instanţă a lui X este, de asemenea, instanţă a lui Y. Relaţia `rdfs:subClassOf` este tranzitivă.

Exemplul 7 Pentru a specifica faptul că *Audio* este subclasă a clasei *Multimedia* vom scrie:

```
<owl:Class rdf:ID="Audio">
   <rdfs:subClassOf rdf:resource="#Multimedia" />
</owl:Class>
```

În vederea formării claselor, *OWL DL* pune la dispoziţie construcţii menite să creeze *expresii de clasă* [Dean & Schreiber, 2004], operatorii permişi fiind cei din teoria mulţimilor: reuniune (`owl:unionOf`), intersecţie (`owl:intersectionOf`), complementară (`owl:complementOf`). Clasele disjuncte pot fi specificate prin `owl:disjointWith`.

[6] A se vedea [Costello & Jacobs, 2003].

Exemplul 8 Pentru a defini faptul că documentele video nu sunt lumi virtuale şi nici aplicaţii executabile, vom scrie:

```
<owl:Class rdf:ID="Video">
   <rdf:subClassOf rdf:resource="#Multimedia" />
   <owl:disjointWith rdf:resource="#LumiVirtuale" />
   <owl:disjointWith rdf:resource="#Executabile" />
</owl:Class>
```

Clasele pot fi specificate şi prin intermediul enumerării membrilor, pentru aceasta folosindu-se elementul `owl:oneOf`.

Suplimentar, alături de numirea unei clasei sau a referinţelor la alte clase (eventual prin intermediul relaţiei `<rdfs:subClassOf>`), se poate specifica o listă de restricţii pe care trebuie să le safisfacă indivizii clasei definite.

O construcţie majoră este reprezentată de *proprietăţi*, relaţii binare care pot specifica fapte privitoare la membrii unei clase sau la invidizi. Proprietăţile se pot referi la tipurile de date (relaţii între instanţele de clase şi literalii RDF sau tipurile de date XML Schema) ori la obiecte (relaţii între instanţele a două clase). Definind o proprietate, în fapt se impune o restricţie. O proprietate poate fi definită ca fiind o subproprietate (specializare) a unei proprietăţi deja specificate. Domeniul şi codomeniul relaţiei (proprietăţii) pot fi şi ele specificate via construcţiile `domain` şi `range`.

Proprietăţile pot fi ierarhizate, similar claselor.

Exemplul 9 Pentru a specifica o proprietate de tip obiect, privitoare la faptul că un document multimedia poate fi stocat pe un sit Web multimedia, vom defini următoarele:

```
<owl:ObjectProperty rdf:ID="stocat">
   <rdfs:domain rdf:resource="#Multimedia" />
   <rdfs:range rdf:resource="#sitMultimedia" />
</owl:Class>
```

Pentru a descrie faptul că un document audio poate fi stocat de măcar un sit Web multimedia, vom putea scrie:

```
<owl:Class rdf:ID="Audio">
   <rdf:subClassOf rdf:resource="#Multimedia" />
   <rdf:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#stocat">
         <!-- indicam cardinalitatea minima -->
         <owl:minCardinality
            rdf:datatype="&xsd;nonNegativeInteger">
            1
         </owl:minCardinality>
      </owl:Restriction>
   </rdf:subClassOf>
</owl:Class>
```

Fiecare instanţă de document audio trebuie să apară măcar în cel puţin o relaţie `stocat`. În acest exemplu se observă şi folosirea unei proprietăţi referitoare la tipuri de date (s-a utilizat un tip de dată XML Schema).

Proprietăţile posedă diverse caracteristici principale (conform specificaţiei [Dean & Schreiber, 2004]):

- *tranzitivitate*:
  pentru o proprietate P are loc $P(x,y) \wedge P(y,z) \Rightarrow P(x,z), \forall x,y,z$;
- *simetrie*:
  pentru o proprietate P are loc $P(x,y) \Rightarrow P(y,x), \forall x,y$;
- *inversă*:
  dacă proprietatea P_1 este etichetată cu `owl:inverseOf` P_2, atunci $P_1(x,y) \iff P_2(y,x)$;
- *funcţională*:
  pentru o proprietate P etichetată ca fiind funcţională are loc $P(x,y) \wedge P(x,z) \Rightarrow y = z$.

Asupra proprietăţilor pot fi impuse restricţii, i.e. diverse constrângeri privitoare la intervalul de valori ale unei proprietăţi în cadrul unui

context specific. Aceste restricţii trebuie să apară definite în cadrul elementelor `owl:Restriction` sau `owl:onProperty`.
Ca exemple de restricţii impuse proprietăţilor se pot menţiona următoarele: `allValuesFrom`, `someValuesFrom` şi `hasValue`. Se pot specifica şi restricţii legate de cardinalitate (*e.g.*, `minCardinality` sau `maxCardinality`), după cum s-a putut observa în exemplul anterior.

Echivalenţa dintre clase se defineşte prin `owl:equivalentClass`, iar cea între proprietăţi via `owl:equivalentProperty`. Aceste construcţii sunt utile pentru a conecta ontologii independente. Un mecanism similar se aplică şi în cazul indivizilor, prin intermediul elementului `owl:sameAs`.

De asemenea, OWL oferă posibilitatea de a defini colecţii de indivizi distincţi via `owl:distinctMembers`.

Exemplul 10 Pentru a specifica, în termeni *fuzzy*, diverse localizări – la nivel local, în intranet-ul sau în extranet-ul organizaţiei – ale unei resurse multimedia, vom punea scrie:

```
<owl:AllDifferent>
   <owl:distinctMembers rdf:parseType="Collection">
      <xf:Location rdf:about="#local" />
      <xf:Location rdf:about="#intranet" />
      <xf:Location rdf:about="#extranet" />
   </owl:distinctMembers>
</owl:AllDifferent>
```

Mai multe detalii privitoare la limbajul OWL sunt disponibile în [Dean & Schreiber, 2004], [Guha & Hayes, 2003] şi [W3C].

2.2 RDF – cadru de descriere a resurselor

2.2.1 Prezentare generală

RDF (*Resource Description Framework*) [Beckett, 2004] defineşte un mecanism de descriere a resurselor independent de domeniul de folosire a datelor, fără a specifica *a priori* vreo semantică. Modalităţile

de descriere a resurselor trebuie să se realizeze într-o manieră neutră, dar generală.

Limbajul RDF are drept scop principal reprezentarea informaţiilor privitoare la resursele Web. RDF poate fi utilizat, de asemenea, pentru reprezentarea informaţiilor care pot fi *identificate* pe Web, chiar dacă în mod direct nu pot fi *accesate* via WWW.

Prima specificaţie se regăseşte în [Lassila & Swick, 1999], actualmente Consorţiul Web punând la dispoziţie, ca recomandări oficiale, o serie de documente care să ofere suportul necesar folosirii RDF ca fundaţie pentru Web-ul semantic (ghid de utilizare, concepte de bază, sintaxă abstractă, semantică, diverse studii de caz şi altele) [Beckett, 2004, Manola & Miller, 2004].

Pentru a facilita definirea datelor RDF, va fi necesar un sistem de clase similar celui din programarea orientată-obiect. O colecţie de clase (dezvoltată pentru un anumit scop specific) se numeşte *schemă*. Clasele sunt organizate ierarhic oferind extensibilitatea prin rafinarea subclaselor. Astfel, pentru crearea unei noi scheme, putem pleca de la o schemă de bază (privită ca o clasă abstractă în termenii programării obiectuale). Se asigură în acest mod şi reutilizarea definiţiilor de metadate. Datorită caracterului extensibil, agenţii software care procesează metadatele vor fi capabili de versatilitate în prelucrarea schemelor. Moştenirea multiplă permite exploatarea, în maniere diverse, a aceleaşi informaţii. Este posibil să se creeze instanţe de date RDF bazate pe scheme provenind din surse diferite.

2.2.2 Modelul de bază al RDF

Modelul de bază se construieşte cu ajutorul următoarelor tipuri de obiecte:

- *resurse*

  Datele descrise de expresiile RDF sunt denumite *resurse*. O resursă poate fi o pagină Web completă – de exemplu, un document XHTML desemnat de un URI precum o adresă de genul `http://www.infoiasi.ro/~busaco/index.html` –, o parte a unei pagini Web (un element specific HTML sau XML prezent

în sursa documentului, *i.e.* o imagine) sau un obiect care nu este direct accesibil via Web (*e.g.* o carte tipărită). Resursele sunt specificate de URI-uri plus un identificator de legătură, opţional.

- *proprietăţi*

  O *proprietate* reprezintă un aspect specific, o caracteristică, un atribut sau o relaţie pentru a descrie o resursă. Fiecare proprietate posedă o semantică, un set de valori permise, o mulţime de tipuri de resurse pe care le descrie şi un set de relaţii (interdependente) cu alte proprietăţi.

- *declaraţii*

  O anumită resursă împreună cu o proprietate a sa având asociată o valoare formează o *declaraţie*. Putem privi declaraţia ca un triplu $\{subiect, predicat, obiect\}$. Obiectul declaraţiei (valoarea proprietăţii) poate desemna o altă resursă (specificată de un URI) sau un *literal* (tip primitiv de dată sau şir de caractere, conform specificaţiilor XML). În modelul RDF, un *literal* poate conţine marcaje XML care însă nu vor fi evaluate (analizate) de procesorul RDF.

2.2.3 Modul de reprezentare

Declaraţiile se pot reprezenta astfel:

- *graf orientat* – nodurile sunt fie subiecte, fie obiecte, iar arcele semnifică un predicat (a se vedea figura 2.3);
- *marcaje* – `<subject> HAS <predicate> <object>`
- *RDF/XML* – după cum vom vedea în continuare.

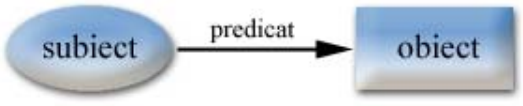

Figura 2.3: Reprezentarea prin grafuri a declaraţiilor RDF

O proprietate poate avea drept valoare o *entitate structurată*. De exemplu: "*Individul Sabin Corneliu Buraga, având adresa de e-mail* `busaco@infoiasi.ro`*, este autorul resursei (paginii) Web* `http://www.infoiasi.ro/~busaco/pub.html`".

În acest caz, obiectul nu va fi un şir de caractere, ci o colecţie formată din trei literali:
⟨ *individ, Sabin Corneliu Buraga, busaco@infoiasi.ro* ⟩

Pentru triplul $\{p, s, o\}$ aparţinând mulţimii de declaraţii RDF, putem găsi interpretarea: "s are proprietatea p cu valoarea o".

Interpretările date unor declaraţii RDF sunt relative la o mulţime de referinţe de identificatori uniformi de resurse (notaţi cu *uriref*), numită *vocabularul* interpretării considerate [Manola & Miller, 2004, Hayes, 2004].

Definiţia 7 *Formal, o* interpretare *se defineşte recursiv astfel:*

- *dacă* E *este un literal simplu, atunci* $I(E) = E$*;*
- *dacă* E *este un literal cu tip, atunci* $I(E) = IL(E)$*;*
- *dacă* E *este un* uriref*, atunci* $I(E) = IL(E)$*;*
- *dacă* E *este un triplu* $\{s, p, o\}$*, atunci* $I(E) = \text{true}$ *dacă* $(I(s), I(o)) \in I_{EXT}(I(p))$ *şi* $I(E) = \text{false}$ *în caz contrar;*
- *dacă* E *este un graf RDF, atunci* $I(E) = \text{false}$ *dacă există un triplu* E' *în* E *astfel încât* $I(E') = \text{false}$*; altfel,* $I(E) = \text{true}$.

Notăm cu LV *mulţimea tuturor literalilor, iar cu* IR *universul (domeniul) lui* I *– mulţimea tuturor resurselor care sunt implicate în* I *(*$IR \supset LV$*). Mulţimea proprietăţilor este desemnată prin* IP *(*$IP \subset IR$*).*

Fiecărei proprietăţi îi este asociată mulţimea perechilor de forma (subiect, obiect) *care formează o relaţie cu proprietatea considerată. Această asociere se realizează prin* $I_{EXT} : IP \rightarrow 2^{IR \times IR}$ *(*$IR(x)$ *este extensia unei entităţi* x*).*

Mai sunt specificate funcţia $IS : URI \rightarrow IR$ *care asociază fiecărui* uriref $\in URI$ *o resursă din univers şi funcţia* $IL : LVT \rightarrow IR$ *care asociază o resursă fiecărui literal cu tip din mulţimea* $LVT \subset LV$.

Pentru a putea procesa pe calculator astfel de aserţiuni va trebui să se recurgă la un sistem de identificare de către maşină a resurselor (via URI) şi la un limbaj uşor de procesat în manieră automată care să faciliteze reprezentarea şi prelucrarea aserţiunilor. Soluţia este oferită de XML.

2.2.4 Sintaxa de bază RDF

Vom furniza în continuare descrierile sintactice în notaţia *EBNF* (*Extended Backus-Naur Form*) [Jucan & Andrei, 2002]. Toate facilităţile sintactice din XML sunt suportate.

Mai multe definiţii pentru o anumită resursă pot fi grupate în cadrul elementului `Description` (suport pentru încapsulare).

Regulile sintactice sunt următoarele [Beckett, 2004]:

```
[1]   RDF       ::= ['<rdf>:RDF>']
                      descript*
                    ['</rdf>:RDF>']
[2]   descript  ::= '<rdf:Description' idAboutAttr? '>'
                      propElt*
                    '</rdf:Description>'
[3]   idAboutAttr ::= idAttr | aboutAttr
[4]   idAttr    ::= 'ID="' Idsymbol '"'
[5]   aboutAttr ::= 'about="' URI-ref '"'
[6]   propElt   ::= '<' propName '>' value '</' propName '>'
                  | '<' propName resAttr '/>'
[7]   propName  ::= Qname
[8]   value     ::= descript | string
[9]   resAttr   ::= 'resource="' URI-ref '"'
[10]  Qname     ::= [ NSprefix ':' ] name
[11]  URI-ref   ::= string
[12]  IDsymbol  ::= (orice simbol legal XML)
[13]  name      ::= (orice simbol legal XML)
[14]  NSprefix  ::= (orice prefix al unui spatiu de nume)
[15]  string    ::= (orice text XML)
```

Exemplul 11 Pentru cele specificate în secţiunea anterioară, avem:

```
<rdf:RDF>
  <rdf:Description
      rdf:about="http://www.infoiasi.ro/~busaco/pub.html">
    <s:Autor>Sabin Corneliu Buraga</s:Autor>
  </rdf:Description>
</rdf:RDF>
```

Aici prefixul *s* se referă la un prefix specific ales de autorul expresiei RDF şi definit într-o declaraţie a unui spaţiu de nume XML, conform unei scheme XML:

```
xmlns:s="http://description.org/schema.xsd"
```

Spaţiul de nume *rdf* trebuie să apară în cadrul oricărei declaraţii RDF.

Exemplul 12 Un alt exemplu este următorul, în care vom reprezenta în RDF o aserţiune particulară referitoare la proprietarul unui fişier stocat de un anumit sistem de fişiere [Buraga, 2002b]:

```
<rdf:RDF>
  <rdf:Description
    rdf:about="file:///home/busaco/teza.tex">
    <xf:Owner>
      <rdf:Description
         rdf:about="http://www.infoiasi.ro/~busaco/">
        <xf:Login>busaco</xf:Login>
        <xf:Group>profs</xf:Group>
        <xf:Name>Sabin Corneliu Buraga</xf:Name>
      </rdf:Description>
    </xf:Owner>
  </rdf:Description>
</rdf:RDF>
```

S-au exprimat următoarele: “Individul referit de URI-ul specificat se numeşte Sabin Corneliu Buraga, având numele de cont busaco şi

aparţinând grupului de utilizatori `profs`. Resursa (fişierul) identificată local prin `/home/busaco/teza.tex` îl are ca proprietar pe acest individ".

Spaţiul de nume *xf* corespunde limbajului *XFiles* prezentat în secţiunea 3.2.5.1.

2.2.5 Scheme şi spaţii de nume

Atunci când scriem o afirmaţie în limbaj natural, utilizăm cuvinte care au un anumit înţeles pentru noi şi pentru cel căruia îi este adresată. Înţelegerea semanticii propoziţiei este crucială în stabilirea cu exactitate a procesării care trebuie urmată. Este extrem de important ca atât scriitorul, cât şi cititorul enunţului să recepteze *acelaşi* înţeles al termenilor utilizaţi, altfel s-ar crea confuzii. În mediul global reprezentat de spaţiul WWW nu este suficient a ne ghida după înţelegerea culturală comună a conceptelor.

Înţelesul unui termen (lingvistic sau nu) în RDF este exprimat printr-o referinţă la o *schemă*. Putem privi schema ca întruchipare a unei ontologii simple (a se vedea secţiunea 2.1.3.3), definind termenii pe care îi vom utiliza în declaraţiile RDF şi asociindu-le o semantică precisă. Se pot folosi o varietate de scheme, specificate sau nu ca documente separate.

O schemă conţine definiţii şi restricţii de utilizare a proprietăţilor. Pentru evitarea confuziilor dintre definiţiile independente a unui acelaşi obiect, RDF se bazează pe facilitatea spaţiilor de nume din XML. După cum am văzut, spaţiile de nume oferă o modalitate simplă de a folosi la un moment dat o unică definiţie a unui termen. Fiecare predicat al unei declaraţii RDF trebuie identificat de o schemă unică. Un element `Description` poate însă conţine declaraţii având predicate aparţinând mai multor scheme.

2.2.5.1 Schemele în detaliu

Declararea proprietăţilor (atributelor) unor resurse şi semantica asociată lor se realizează prin intermediul schemelor. RDF poate fi văzut

astfel şi ca *limbaj de specificare a schemelor*. Schemele RDF au la bază idei preluate din reprezentarea cunoştinţelor (reţele semantice, logica predicatelor) ori din limbajele de specificare a bazelor de date. Un mecanism mai flexibil decât schemele RDF este cel bazat pe limbajul OWL.

Schemele RDF oferă un sistem de tipuri de date pentru modelul RDF [Manola & Miller, 2004], punând la dispoziţie mecanisme utile pentru specificarea claselor şi proprietăţilor obiectelor în contextul unui vocabular (a se vedea şi figura 2.4). Prin intermediul schemelor RDF se poate indica – printr-o sintaxă XML care poate fi procesată de maşină – ce clase şi proprietăţi pot fi asociate (de exemplu, o proprietate desemnată de `<xf:Name>` să fie utilizată în descrierea proprietarului unei resurse Web).

Sunt definite o serie de clase şi de proprietăţi fundamentale. Pentru schemele RDF se specifică spaţiul de nume `rdfs`.

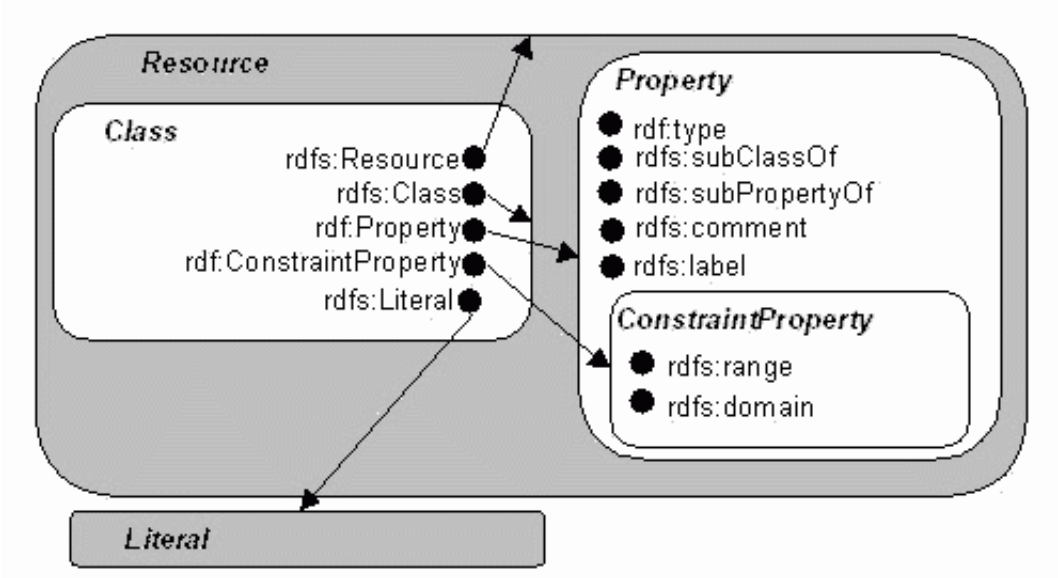

Figura 2.4: Mulţimile de clase şi de proprietăţi

2.2.5.2 Clase fundamentale

Specificaţia RDF pune la dispoziţie următoarele clase fundamentale:

- `rdfs:Resource` – defineşte clasa resurselor, corespunzând într-o anumită măsură conceptului de obiect al paradigmei de programare orientate-obiect [Brickley & Guha, 2000].
- `rdf:Property` – reprezintă clasa proprietăţilor resurselor.
- `rdfs:Class` – corespunde conceptului general de tip sau categorie. Când o schemă defineşte o nouă clasă, resursa reprezentând acea clasă trebuie să aibă o proprietate `rdfs:type` a cărei valoare este resursa `rdfs:Class`.

Clasele RDF pot specifica, de exemplu, pagini Web, tipuri de documente, baze de date, persoane etc.

De exemplu, pentru a oferi informaţii privitoare la diverse tipuri de documente multimedia va trebui să specificăm o clasă care să reprezinte categoria obiectelor (resurselor) multimedia considerate. Resursele care aparţin acestei clase se vor numi *instanţe*. Instanţele clasei de faţă vor fi resurse multimedia. Asemenea aserţiune va trebui să fie înţeleasă şi procesată de către calculator.

Având în vedere cele discutate mai sus, pentru clasa descriind categoria resurselor multimedia va trebui să asociem (o referinţă către) un identificator (*e.g.*, un fragment de resursă `#multimedia` sau o adresă `http://www.infoiasi.ro/multimedia.xsd`).

În plus, trebuie să specificăm faptul că resursele vor avea asociată proprietatea `rdf:type` a cărei valori va fi resursa definită printr-un element `rdfs:Class`. Prin aceasta, am enunţat declaraţia RDF de forma {ResursaMultimedia, `rdf:type`, `rdfs:Class`}.

2.2.5.3 Proprietăţi fundamentale

Fiecare model RDF care utilizează un mecanism de scheme include implicit proprietăţile de mai jos, instanţe ale clasei `rdf:Property`, oferind o modalitate de a exprima relaţiile dintre clase şi instanţele lor sau supraclase.

- `rdf:type`

  Indică faptul că o resursă este membră a unei clase. Atunci când o resursă are o proprietate `rdf:type` a cărei valoare reprezintă

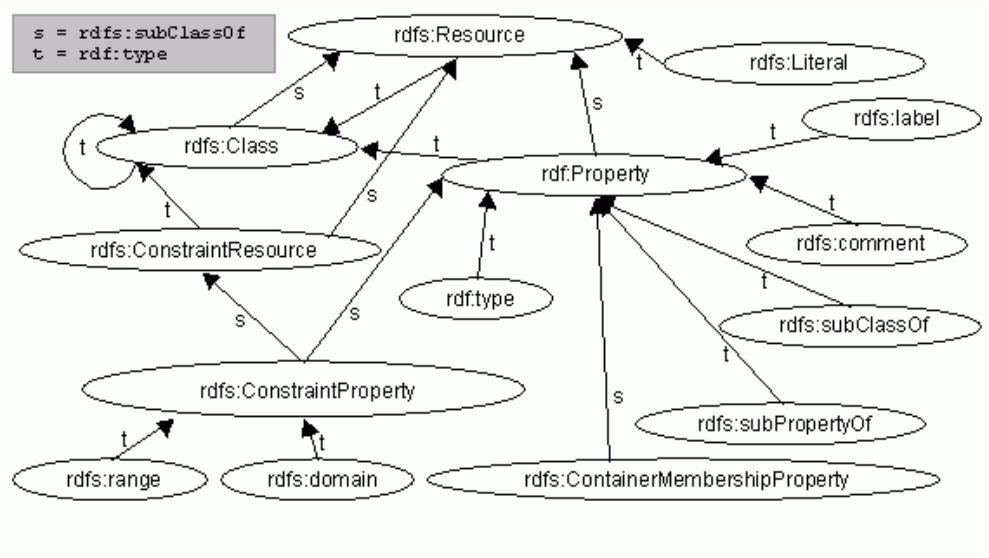

Figura 2.5: Ierarhiile de clase RDF

o anumită clasă, vom spune că resursa este o *instanţă* a acelei clase. Valoarea lui `rdf:type` pentru o resursă este o altă resursă, instanţă a lui `rdfs:Class`. Clasele individuale întotdeauna vor avea `rdf:type` asignată cu valoarea `rdfs:Class` (ori o subclasă a lui `rdfs:Class`). O resursă poate fi instanţă a mai multor clase, desigur.

- `rdfs:subClassOf`

  Desemnează relaţia de moştenire a claselor, fiind o relaţie tranzitivă. Numai instanţe ale unui element `rdfs:Class` pot avea proprietatea `rfds:subClassOf`, iar valoarea ei este întotdeauna `rdf:type rdfs:Class`. O clasă poate fi subclasă a mai multor clase. O clasă niciodată nu poate fi declarată ca subclasă a ei însăşi sau drept subclasă a subclaselor sale.

- `rdfs:subPropertyOf`

  O proprietate poate avea zero, una sau mai multe proprietăţi, specializări ale ei.

Dacă o anumită proprietate P_1 este o subproprietate a unei proprietăţi mai generale P_2 şi dacă o resursă A are proprietatea P_2 având asignată valoarea B, atunci aceasta implică faptul că resursa A are, de asemenea, proprietatea P_1 cu valoarea B.

Exemplul 13 Pentru a defini clasa "Multimedia" conţinând subclasele "Audio" şi "Video", vom scrie următoarele:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description rdf:ID="Multimedia">
    <rdf:type rdf:resource=
       "http://www.w3.org/2000/01/rdf-schema#Class" />
  </rdf:Description>

  <rdf:Description rdf:ID="Video">
    <rdf:type rdf:resource=
       "http://www.w3.org/2000/01/rdf-schema#Class" />
    <rdfs:subClassOf rdf:resource="#Multimedia" />
  </rdf:Description>

  <rdf:Description rdf:ID="Audio">
    <rdf:type rdf:resource=
       "http://www.w3.org/2000/01/rdf-schema#Class" />
    <rdfs:subClassOf rdf:resource="#Multimedia" />
  </rdf:Description>
</rdf:RDF>
```

2.2.5.4 Restricţii

O schemă poate declara anumite restricţii – a se vedea figura 2.6 – asociate claselor şi proprietăţilor. În jargonul RDF, vor fi folosite conceptele de *domeniu* (*domain*) şi *interval* (*range*).

Un model RDF care nu respectă o restricţie este un *model inconsistent*. Diverse aplicaţii pot avea comportamente eronate în cadrul unui model inconsistent.

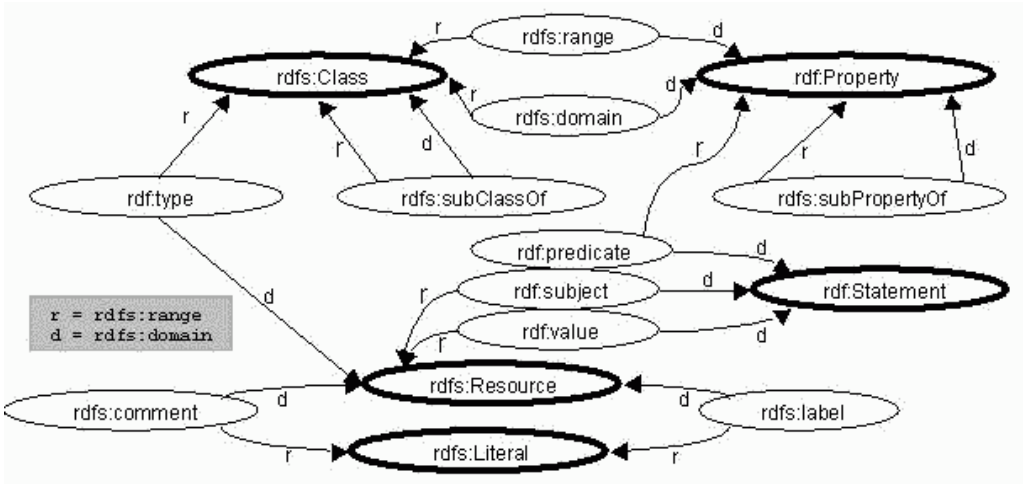

Figura 2.6: Restricţiile în RDF

Exemplul 14 Exemple de restricţii:

- valoarea unei proprietăţi trebuie să fie o resursă ori o clasă definită de proiectantul unei scheme. Această restricţie este exprimată de proprietatea `range` (de exemplu, restricţia aplicată proprietăţii "`autor`": valoarea ei trebuie să fie o resursă, instanţă a clasei "`Persoană`").
- o proprietate poate fi utilizată doar de resursele unei anumite clase (de exemplu, proprietatea "`autor`" poate fi folosită numai dacă valoarea ei este o resursă a cărei instanţă este clasa "`Teza`"). Acest lucru se exprimă prin proprietatea `domain`.

2.2.5.5 Restricţii fundamentale

Recomandarea RDFS [Brickley & Guha, 2000] specifică două restricţii fundamentale de bază:

- `rdfs:range` este folosită pentru a restricţiona valorile unei proprietăţi. Valoarea lui `range` este întotdeauna o clasă. Valoarea

unei proprietăţi al cărei interval este A va fi constrânsă să fie instanţă a clasei A. Putem avea cel mult o proprietate `range`.

- `rdfs:domain` este utilizată să specifice o clasă care poate fi asignată ca valoare a unei proprietăţi. O proprietate poate avea valori din zero, una sau mai multe clase. Dacă nu există definit nici un domeniu, poate fi folosită oricare resursă.

Remarca 2 Proprietăţile `rdfs:domain` şi `rdfs:Range` precizează domeniul şi, respectiv, codomeniul unei proprietăţi, adică tocmai clasele cărora trebuie să le aparţină subiectul, respectiv obiectul.

Mai multe detalii privitoare la schemele RDF sunt disponibile în lucrările [Brickley & Guha, 2000] şi [Manola & Miller, 2004].

2.2.6 Colecţii de resurse

Este necesar deseori să utilizăm colecţii de resurse, pentru aceasta în RDF definindu-se trei tipuri de obiecte [Beckett, 2004]:

bag – desemnează o listă neordonată de resurse sau de literali, valorile duplicate fiind permise (multi-set);

secvenţă – specifică o listă ordonată de resurse sau literali. Ca mai sus, valorile pot fi duplicate;

alternativă – desemnează o listă de resurse/literali care reprezintă alternative pentru o singură valoare a unei proprietăţi.

Tipul mulţime nu este definit de specificaţiile RDF.

Pentru a se putea crea o colecţie de resurse, RDF utilizează o resursă suplimentară care reprezintă o colecţie specifică (o *instanţă* a colecţiei), această resursă declarându-se ca instanţă a unui tip din cele anterioare, prin proprietatea `type`. Relaţia dintre colecţia de resurse şi resursele ce aparţin acesteia este dată de un set de proprietăţi denumite `"_1"`, `"_2"`, `"_3"` etc. Colecţiile de resurse pot avea şi alte proprietăţi, desigur.

Sintaxa formală este următoarea:

```
[16]  contain   ::= seq | bag | alt
[17]  seq       ::= '<rdf:Seq' idAttr? '>'
                      member* '</rdf:Seq>'
[18]  bag       ::= '<rdf:Bag' idAttr? '>'
                      member* '</rdf:Bag>'
[19]  alt       ::= '<rdf:Alt' idAttr? '>'
                      member* '</rdf:Alt>'
[20]  member    ::= referItem | inlineItem
[21]  referItem ::= '<rdf:li resourceAttr '/>'
[22]  inlineItem::= '<rdf:li>' value '</rdf:li>'
```

Colecţiile pot apărea oriunde este permis un element `Description`. Regulile sintactice se modifică astfel:

```
[1a]  RDF   ::= '<rdf:RDF>' obj* '</rdf:RDF>'
[8a]  value ::= obj | string
[23]  obj   ::= descript | contain
```

Semantica formală a colecţiilor de resurse este definită în lucrările [Conen & Klapsing, 2000] şi [Hayes, 2004].

Exemplul 15 Pentru a specifica proprietarul şi o metodă de acces bazată pe autentificare prin parolă asupra unui set de fişiere stocate pe o maşină locală sau pe un calculator aflat la distanţă, putem defini următoarele aserţiuni RDF [Buraga, 2000a, Buraga, 2002b]:

```
<rdf:RDF>
  <!-- o colectie de resurse eterogene, distribuite -->
  <rdf:Bag rdf:ID="myfiles">
    <rdf:li
       rdf:resource="file:///tmp/article.tex" />
    <rdf:li
       rdf:resource="ftp://ftp.tuiasi.ro/pub/src/gaen" />
    <rdf:li
       rdf:resource="http://www.infoiasi.ro/fcs.jpg" />
  </rdf:Bag>

  <!-- specificarea proprietatilor despre resurse -->
  <rdf:Description rdf:about="#myfiles">
```

```
    <xf:Properties>
      <!-- modul de autentificare -->
      <xf:Auth>Basic</xf:Auth>
      <!-- proprietarul resurselor -->
      <xf:Owner>
        <rdf:Description
          rdf:about="http://www.infoiasi.ro/~busaco">
          <xf:Login uid="714">busaco</xf:Login>
          <xf:Password>NU74b33cs</xf:Password>
        </rdf:Description>
      </xf:Owner>
      <!-- permisiunile de acces -->
      <xf:Permissions>
        <xf:Permission>User-Read</xf:Permission>
        <xf:Permission>User-Write</xf:Permission>
        <xf:Permission>Group-Read</xf:Permission>
      </xf:Permissions>
    </xf:Properties>
  </rdf:Description>
</rdf:RDF>
```

S-a exprimat următorul fapt: “Pentru colecţia de resurse (fişiere) dată, proprietarul acesteia este utilizatorul busaco. Fişierele vor fi accesate autentificat prin intermediul unei parole şi numai proprietarul va putea să le acceseze pentru citire şi scriere. Membrii grupului proprietarului vor putea numai să le citească”.

Exemplul 16 Modelul pentru enunţul:

> *Codul sursă pentru aplicaţia* GAEN *poate fi găsit la una dintre locaţiile* `ftp.infoiasi.ro`, `ftp.uaic.ro` *sau* `ftp.tuiasi.ro`.

în RDF se scrie astfel:

```
<rdf:RDF>
  <rdf:Description
```

```
    rdf:about="http://www.infoiasi.ro/~busaco/gaen">
    <s:DistributionSite>
      <!-- alternativa a locatiilor resursei -->
      <rdf:Alt>
        <rdf:li
           rdf:resource="ftp://ftp.infoiasi.ro" />
        <rdf:li
           rdf:resource="ftp://ftp.uaic.ro/pub/gaen" />
        <rdf:li
           rdf:resource="ftp://ftp.tuiasi.ro/src/Linux" />
      </rdf:Alt>
    </s:DistributionSite>
  </rdf:Description>
</rdf:RDF>
```

2.2.7 Referenţi distributivi

Obiectul descris de o declaraţie RDF (indicat de atributul about) este numit *referent*.

Exemplul 17 Considerăm fragmentul de aserţiuni [Buraga, 2001a]:

```
<rdf:Bag rdf:ID="pages">
  <rdf:li
     rdf:resource="http://www.infoiasi.ro/web/web1.html" />
  <rdf:li
     rdf:resource="http://www.infoiasi.ro/web/web2.html" />
  <rdf:li
     rdf:resource="http://www.infoiasi.ro/web/web3.html" />
    ...
</rdf:Bag>

<rdf:Description rdf:about="#pages">
  <s:Autor>Sabin Corneliu Buraga</s:Autor>
</rdf:Description>
```

În acest exemplu, s-a exprimat faptul că “Sabin Corneliu Buraga” este autorul colecţiei de pagini identificate prin `pages`. Referentul elementului `Description` este o colecţie (de tip *Bag*), nu membrii ei.

Pentru a specifica faptul că “Sabin Corneliu Buraga” este autorul fiecărei pagini, se foloseşte un alt mod de referire, via `aboutEach` – acest tip de referent fiind numit *referent distributiv*.

```
<rdf:Description rdf:aboutEach="#pages">
  <s:Autor>Sabin Corneliu Buraga</s:Autor>
</rdf:Description>
```

Un alt exemplu, în care folosim de asemenea atributul `aboutEach` este următorul.

Exemplul 18 Vom enunţa o serie de aserţiuni privitoare la resursele unui sistem de teleconferinţe – GAEN [Buraga, 1998]. În acest caz, resursele vor aparţine unei colecţii de tip *Bag* care va desemna utilizatorii cu drept de administrare a unei instanţe a sistemului, identificată prin adresa maşinii pe care rulează:

```
<?xml version="1.0" ?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:DC="http://purl.org/dc/elements/1.0/"
   xmlns:g="http://www.infoiasi.ro/~busaco/gaen/gaen.xml">

   <!-- o colectie de resurse; in acest caz utilizatori
        desemnati de adresele siturilor lor Web -->
   <rdf:Bag rdf:id="UsersOfSuperiorLevel">
      <rdf:li
          rdf:resource="http://www.infoiasi.ro/~busaco" />
      <rdf:li
          rdf:resource="http://www.ac.tuiasi.ro/mituc" />
      <rdf:li
          rdf:resource="mailto:skocsis@uaic.ro" />
   </rdf:Bag>
```

```
   <!-- o asertiune RDF privitoare la utilizatori -->
   <rdf:Description rdf:aboutEach="#UsersOfSuperiorLevel">
      <g:gaen g:host="delta.ac.tuiasi.ro"
              g:mainport="7777">
          <g:user>
              <g:info>
                 <g:level g:number="7">
                    Administrator
                 </g:level>
                 <g:comment>
                    Lista utilizatorilor
                    cu rang de administrator
                 </g:comment>
              </g:info>
              <DC:date>
                 2003-09-20
              </DC:date>
          </g:user>
      </g:gaen>
   </rdf:Description>
</rdf:RDF>
```

În cadrul documentului de mai sus, sunt folosite trei spaţii de nume şi anume: rdf este spaţiul de nume specific documentelor RDF[7], DC desemnează spaţiul de nume al setului de elemente de identificare a metadatelor – *Dublin Core Metadata Initiative* [Purl] –, iar g reprezintă spaţiul de nume al limbajului bazat pe XML care modelează diverse informaţii privitoare la sistemul de teleconferinţe [Buraga, 2001b].

2.2.8 Colecţii referite de un URI

Una dintre utilizările metadatelor este cea de a face declaraţii despre “toate paginile Web disponibile pe un anumit server” sau “toate paginile Web descriind un anumit aspect, aflate la o adresă specifică”.

[7] A se consulta şi [Beckett, 2004] sau [Manola & Miller, 2004].

În multe cazuri este dificil ori neimportant să încercăm să enumerăm fiecare resursă în mod explicit şi să o identificăm ca membru al unei colecţii. Folosind o a doua formă de referenţi distributivi putem crea o colecţie de tip *Bag* pentru a defini toate resursele care ne interesează în aplicaţiile noastre:

```
[3a]  idAboutAttr ::= idAttr | aboutAttr | aboutEachAttr
[24]  aboutEachAttr ::= 'aboutEach="' URI-ref '"' |
                        'aboutEachPrefix="' string '"'
```

Atributul `aboutEachPrefix` declară o colecţie ai cărei membri sunt toate resursele corespunzătoare identificatorilor care încep cu şirul de caractere al valorii atributului.

Exemplul 19 Pentru toate paginile Web prezente pe serverul Facultăţii de Informatică, putem declara o proprietate de *copyright* scriind (se indică şi intervalul de timp pentru care drepturile de autor sunt în vigoare, prin intermediul atributelor `start` şi `end`) [Buraga, 2001a]:

```
<rdf:Description
      rdf:aboutEachPrefix="http://www.infoiasi.ro">
  <s:Copyright s:start="1997" s:end="2004">
      Sabin Corneliu Buraga
  </s:Copyright>
</rdf:Description>
```

În acest exemplu, nu ne interesează câte documente există stocate pe serverul Web specificat de adresa `http://www.infoiasi.ro` şi nici interdependenţa lor (structura hipertext pe care o pot genera este necunoscută).

2.2.9 Modelarea declaraţiilor

În cadrul RDF, se pot construi declaraţii privitoare la alte declaraţii. Vom numi acest tip de declaraţii *declaraţii de nivel înalt*.

Considerând afirmaţia:

> *Sabin Corneliu Buraga este autorul resursei desemnate de identificatorul uniform de resurse* `http://www.infoiasi.ro`.

o putem vedea ca un *fapt* (asemănător clauzelor Prolog[8]).

Dacă însă avem enunţul:

> *Dumitru Todoroi afirmă faptul că Sabin Corneliu Buraga este autorul resursei desemnate de identificatorul uniform de resurse* `http://www.infoiasi.ro`.

exprimăm un fapt despre o afirmaţie enunţată de altcineva.

În această manieră, sunt modelate metadate privitoare la metadate [Kokkelink & Schwänzl, 2002].

Termenul tehnic utilizat de specificaţiile RDF este cel de *reificare* (*reification*), termen preluat din domeniul reprezentării cunoştinţelor. Putem modela declaraţia originală ca o resursă având patru proprietăţi.

Din punct de vedere formal, un model al datelor RDF se defineşte în maniera de mai jos [Beckett, 2004]:

Definiţia 8 *Fie mulţimile denumite* $Resurse$, $Literali$ *şi* $Declaratii$ *şi o submulţime numită* $Proprietati$ *a mulţimii* $Resurse$, $Proprietati \subset Resurse$.

Elementele mulţimii $Declaratii$ *sunt triple de forma* $\{pred, sub, obj\}$, *în care fiecare* $pred$ *este o proprietate aparţinând mulţimii* $Proprietati$, $pred \in Proprietati$, sub *este o resursă (element al mulţimii* $Resurse$*) şi* obj *este fie o resursă, fie un literal (un literal este element al mulţimii* $Literali$*),* $obj \in Resurse \cup Literali$.

Exemplul 20 Afirmaţia:

> *Sabin Corneliu Buraga este autorul resursei desemnate de identificatorul uniform de resurse* `http://www.infoiasi.ro`.

[8] A se vedea, de exemplu, şi [Metakides & Nerode, 1998].

are asociat triplul d (element al mulţimii $Declaratii$):

(d) {autor, [http://www.infoiasi.ro], "Sabin Corneliu Buraga"} (2.1)

în care am notat [I] resursa identificată prin URI-ul I, iar ghilimelele încadrează un literal (în cazul de faţă, numele autorului resursei).

Prin intermediul triplelor este relativ uşor să se poate specifica reificarea declaraţiilor. Considerând declaraţia (d) de mai sus, exprimăm reificarea acesteia ca o nouă resursă X după cum urmează:

```
{type,      [X],    [rdf:Statement]}
{predicate, [X],    [autor]}
{subject,   [X],    [http://www.infoiasi.ro]}
{object,    [X],    "Sabin Corneliu Buraga"}
```

Cele patru triple specifică proprietăţile declaraţiei reificate.

Definiţia 9 Reificarea *unui triplu* $\{pred, sub, obj\} \in Declaratii$ *constă dintr-un element* $r \in Resurse$ *şi din elementele* s_1, s_2, s_3, s_4 *ale mulţimii* $Declaratii$ *astfel încât:*

- $s_1 : \{rdf : predicate, r, pred\}$
- $s_2 : \{rdf : subject, r, sub\}$
- $s_3 : \{rdf : object, r, obj\}$
- $s_4 : \{rdf : type, r, [rdf : Statement]\}$

unde:

1. *elementul* `rdf:type` *este un element al mulţimii* $Proprietati$, *cu* $\{rdf : Type, sub, obj\}$, $sub, obj \in Resurse$;
2. r *este un element al mulţimii* $Resurse$, *astfel încât* $r \notin Proprietati$;

3. *elementele* `rdf:predicate`, `rdf:subject` *şi* `rdf:object` *aparţin mulţimii* $Proprietati$.

Regulile de mai sus formalizează procesul de reificare, *iar resursa* r *se numeşte* declaraţie reificată.

Specificaţiile RDF oferă două versiuni ale teoriei semantice pe care se bazează modelul RDF: prima foloseşte o manieră neutră din punct de vedere metafizic şi ontologic, fiind exprimată în [Hayes, 2004], iar cea secundă utilizează o semantică axiomatică, transformând construcţiile sintactice din RDF şi RDFS în cele ale unui limbaj formal nou – L_{base} [Guha & Hayes, 2003]. Scopul principal limbajului L_{base} este cel de a oferi o bază teoretică pentru oricare dintre limbajele de specificare a Web-ului semantic (*e.g.*, OWL sau orice limbaj derivat din OWL).

Utilizările RDF vor fi prezentate în secţiunea 2.3. De asemenea, modelul RDF va juca un rol central în ceea ce priveşte descrierea resurselor şi specificarea relaţiilor dintre resursele multimedia (a se vedea capitolul 3).

2.3 Utilizări şi aplicaţii

În cadrul acestei secţiuni vom prezenta o serie de utilizări şi aplicaţii ale metadatelor, ontologiilor şi limbajelor specifice Web-ului semantic.

Folosirea metadatelor

În ceea ce priveşte metadatele, una dintre utilizările notabile este referitoare la managementul şi distribuirea de conţinut multimedia. Ca exemplu, poate fi dat *Metia* [Sticker, 2001] – un cadru de lucru folosind metadate pentru descrierea caracteristicilor importante ale mediilor electronice de comunicare. Printre caracteristicile importante se numără scalabilitatea, portabilitatea, modularitatea şi extensibilitatea componentelor mediului. Se recurge la mai multe module funcţionale pentru a asocia atribute semantice conţinutului multimedia, pentru a structura şi arhiva informaţiile stocate şi pentru a oferi posibilităţi de

interogare. Propunerea noastră de implementare, descrisă în cadrul capitolului 4 pune la dispoziţie acelaşi set de funcţionalităţi, adăgând însă dimensiunea temporală.

Pentru a permite identificarea şi regăsirea facilă a metadatelor, au fost propuse servicii de catalogare (*metadata registry*), precum *DCMI Registry* [Heery & Wagner, 2002]. O serie de prototipuri utilizează pentru stocare sisteme de baze de date (PostgreSQL sau Berkeley DB), pentru funcţionalitate limbajul Java, iar interfaţa foloseşte transformări XSLT. Procesarea metadatelor (stocate ca documente RDF) se realizează prin intermediul mai multor interfeţe de programare a aplicaţiilor (*API – Application Programmer Interface*), dintre care se pot menţiona:

- *EOR* (*Extensible Open RDF Toolkit*) [EOR];
- *Jena Semantic Web Toolkit* [McBride, 2002];
- *RDF NetAPI* [Seaborne, 2002];
- *RAP* (*RDF API for RDF*) [RAP].

Un exemplu de generare şi de utilizare a metadatelor în contextul comerţului electronic este ilustrat în cadrul anexei A.

Una dintre aplicaţiile majore care folosesc metadate exprimate în RDF este *Edutella* [Edutella], o arhitectură *peer-to-peer* al cărei scop major este să pună la dispoziţie servicii de metadate permiţând interoperabilitatea între componente distribuite care fac schimb de resurse educaţionale. Serviciile principale oferite sunt de interogare şi accesare a metadatelor, de replicare a datelor, de mediere în vederea evitării conflictelor la nivel sintactic/semantic, de adnotare a materialelor stocate de o reţea Edutella. Structura operaţională a aplicaţiei recurge la JXTA [JXTA].

Acţiunea de interogare a metadatelor recurge la limbaje de interogare pentru RDF, bazate sau nu pe XML: *SquishQL* [Miller *et al.*, 2002], *TRIPLE* [Sintek & Decker, 2002] sau *QuizRDF* [Davies *et al.*, 2003].

O altă platformă destinată stocării şi interogării documentelor RDF şi RDF Schema este *Sesame* [Broekstra *et al.*, 2002]. Schemele RDF sunt procesate prin intermediul unei interfeţe de programare denumite SAIL.

În vederea unei indexări şi căutări de calitate a informaţiilor, resursele Web pot avea asociate diverse metadate, extensii ale DCMI, folosindu-se diverse standarde de catalogare a documentelor, precum UNIMARC sau Dewey [Beckett, 2002]. Subiectul căutării resurselor Web va fi dezvoltat în cadrul subcapitolului 3.3 al capitolului următor.

Printre contribuţiile proprii, se pot menţiona cercetările privind asocierea de metadate resurselor sau componentelor unor aplicaţii, precum un server de teleconferinţe [Buraga, 2001b] [Buraga, 2001d] [Buraga *et al.*, 2001], un sistem de agenţi [Buraga & Alboaie, 2004] sau un mediu de *e-learning* [Buraga, 2001c, Buraga, 2003c].

Folosirea ontologiilor

Procesul de dezvoltare şi exploatare a ontologiilor prezintă încă dificultăţi. Pentru facilitarea creării şi regăsirii de ontologii stocate în manieră distribuită au fost propuse diverse soluţii. Una dintre ele este descrisă în [Maedche *et al.*, 2003].

O altă direcţie de cercetare este aceea a proiectării şi implementării unei platforme care să ofere modalităţi de management al ontologiilor necesare dezvoltării ulterioare de aplicaţii destinate Web-ului semantic. O abordare este concretizată în *KAON* (*the Karlsruhe Ontology and Semantic Web Tool Suite*) [Bozsak *et al.*, 2002]. KAON pune la dispoziţie şi instrumente destinate proiectanţilor de ontologii, precum un generator de portaluri bazate pe ontologii sau un cadru de specificare a ontologiilor – *OntoMat*. Pe partea de server, sunt disponibile o interfaţă de programare (*KAON API*) şi *KAON Server*.

Ontologiile sunt utilizate şi în cadrul sistemelor distribuite, pentru interschimb de informaţii semantice între componente, ca agenţi Web ori servicii Web semantice. Detalii sunt furnizate de lucrări precum [Davies *et al.*, 2003], [Scott-Cost *et al.*, 2002] şi [Strang *et al.*, 2003]. De cele mai multe ori, pentru descrierea ontologiilor se folosesc limbajele DAML+OIL sau OIL.

În ceea ce priveşte modelele de specificare a aplicaţiilor destinate Web-ului semantic, unele cercetări se focalizează asupra adoptării limbajului RDF sau a unor limbaje derivate [Klapsing & Neumann, 2000]. Altă direcţie vizează integrarea limbajului UML [OMG] sau crearea unor limbaje specializate de modelare ca *WebML* (*Web Modeling Language*) [Ceri *et al.*, 2000].

Pentru verificarea consistenţei ontologiilor se pot folosi diverse instrumente software precum *ConsVISor* [Baclawski *et al.*, 2002], oferind suport pentru diagrame UML, aserţiuni RDF/RDFS sau DAML+OIL.

Pentru alte amănunte, se poate consulta [Davies *et al.*, 2003].

2.4 Concluzii

În cadrul acestui capitol, am prezentat principalele componente ale Web-ului semantic, definit de Tim Berners-Lee în [Davies *et al.*, 2003]. Arhitectura Web-ului semantic este una funcţională, deoarece modul de constituire a acesteia se bazează pe specificarea incrementală a unor limbaje, pornind de la nivelul inferior (i.e. nivelul metadatelor) şi ajungând la niveluri superioare (*e.g.*, nivelul logic). Limbajele disponibile pe fiecare nivel pot satisface cerinţe impuse de diferite tipuri (sau niveluri) de aplicaţii:

1. *nivelul metadatelor* (*metadata layer*) pune la dispoziţie cadrul general pentru exprimarea unor aserţiuni semantice simple. Modelul oferit conţine concepte precum *resursă* şi *proprietate* care sunt utilizate să exprime meta-informaţii. Acest model – constituit din specificaţiile RDF [Beckett, 2004, Manola & Miller, 2004] – va fi utilizat de limbajele aflate pe nivelurile superioare şi a fost prezentat în cadrul secţiunii 2.2.

2. *nivelul schemelor* (*schema layer*) oferă posibilitatea specificării de ontologii simple pentru a se putea defini o descriere ierarhică a conceptelor şi proprietăţilor. Limbajele de specificare a schemelor – ca de exemplu RDF Schema [Brickley & Guha, 2000] – utilizează modelul oferit de nivelul de metadate pentru a institui o arhitectură de bază pentru meta-modelarea unor limbaje Web

de specificare a ontologiilor. Principalele trăsături ale schemelor RDF au fost prezentate în secţiunea 2.2.5.

3. *nivelul logic* (*logical layer*) introduce limbaje ontologice mai complexe, capabile să modeleze ontologii sofisticate. Aceste limbaje sunt bazate pe nivelul schemelor, cadrul teoretic fiind asigurat de logici descriptive [Baader *et al.*, 2002]. Se doreşte astfel constituirea unor servicii (*reasoning services*) pentru Web-ul semantic. Ca limbaje reprezentative în secţiunea 2.1.3.3 s-au menţionat OIL [Horrocks *et al.*, 2000], DAML+OIL [Broekstra *et al.*, 2001] şi OWL [Dean & Schreiber, 2004].

În cadrul secţiunii 2.3 a capitolului de faţă s-a urmărit prezentarea unora dintre cele mai reprezentative aplicaţii din domeniul Web-ului semantic, atât din perspectiva procesării unor tipuri de limbaje, cât şi din cea a unor arhitecturi conceptuale şi platforme software de modelare şi exploatare.

Cercetările efectuate, detaliate în următoarele capitole, s-au concentrat mai ales asupra oferirii suportului specificării relaţiilor spaţio-temporale dintre resurse, pe baza nivelurilor metadatelor şi schemelor, în vederea descrierii şi regăsirii informaţiilor pe Web.

Capitolul 3

Descrierea şi regăsirea resurselor multimedia

Acest capitol realizează o descriere a manierelor de exprimare a metadatelor asociate resurselor Web şi a modului de specificare a relaţiilor spaţio-temporale stabilite între resurse. Pentru îndeplinirea acestor deziderate sunt folosite anumite limbaje bazate pe XML.

3.1 Preliminarii

3.1.1 Motivaţie

DUPĂ cum s-a putut vedea şi din cele discutate în capitolele anterioare, principalul scop al lucrării de faţă este acela de a găsi modalităţile corespunzătoare regăsirii informaţiilor multimedia relevante disponibile pe Web, prin intermediul unor mecanisme specifice Web-ului semantic.

Descoperirea resurselor este strâns legată de descoperirea relaţiilor care se pot stabili între acestea, un rol central având-l relaţiile spaţio-temporale. Eforturile în ceea ce priveşte Web-ul semantic sunt acelea de a asocia adnotări ontologice, atât resurselor disponibile pe Web, cât şi relaţiilor stabilite între acestea [Sheth, Arpinar, Kashyap, 2002, Davies *et al.*, 2003]. Identificarea, descoperirea, validarea şi utilizarea relaţiilor dintre resurse reprezintă unele dintre principalele direcţii de cercetare ale Web-ului semantic şi, implicit, ale acestui material.

3.1.2 Prezentare generală

În prima parte a acestui capitol, vom prezenta o manieră originală bazată pe construcţii XML şi RDF pentru modelarea relaţiilor spaţio-temporale care pot fi stabilite între resursele (multimedia) ale unui (fragment de) Web. Pentru aceasta se vor defini două limbaje: primul – detaliat în secţiunea 3.2.5.1 şi denumit *XFiles* – va putea fi folosit pentru a descrie anumite proprietăţi specifice ale resurselor via metadate, iar al doilea – purtând numele *TRSL* (*Temporal Relation Specification Language*) şi fiind descris în cadrul secţiunii 3.2.7.1 – va ajuta la specificarea relaţiilor spaţio-temporale dintre resurse, adoptându-se ca model teoretic logica *ITL* (*Interval Temporal Logic*) – a se vedea secţiunea 3.2.3. Prin intermediul acestor limbaje vor fi formulate diverse aserţiuni RDF, ilustrându-se astfel alinierea cercetărilor noastre la problematicile Web-ului semantic. Principalele rezultate obţinute sunt descrise în [Buraga, 2000a], [Buraga, 2001d], [Buraga, 2002b], [Buraga, 2002c], [Buraga, 2003d] şi [Buraga & Ciobanu, 2002].

Cea de-a doua parte se concentrează asupra exprimării prin intermediul unui limbaj bazat pe XML a interogărilor complexe formulate de utilizatori, în vederea realizării regăsirii de documente multimedia structurate sau semi-structurate. Acest limbaj – prezentat în secţiunea 3.3.3 şi denumit *WQFL* (*Web Query Formulating Language*) – va putea desemna diverse informaţii privind structura şi conţinutul resurselor hipermedia găsite. Strategia de interogare pleacă de la premisa că utilizatorii preferă să obţină documente având diverse structuri şi tipuri de conţinut, în acest sens trebuind specificate poziţiile şi numărul de apariţii ale unor elemente sintactice compunând un anu-

mit document XML. O parte dintre rezultatele obţinute sunt publicate în [Buraga & Rusu, 2000], [Buraga, 2001a], [Buraga & Rusu, 2001], [Buraga, 2002a], [Buraga & Brut, 2001] şi [Buraga & Brut, 2002].

3.2 Modelarea relaţiilor dintre resurse

În cadrul acestui subcapitol, vom prezenta o serie de modele teoretice utilizate la descrierea proprietăţilor temporale ale sistemelor distribuite, insistând asupra logicii ITL pe care o vom folosi la exprimarea relaţiilor temporale dintre resursele multimedia disponibile pe Web (o serie dintre cercetările efectuate se pot regăsi în [Buraga, 2002b], [Buraga, 2002c] şi [Buraga & Ciobanu, 2002]).

De asemenea, vom realiza o clasificare a relaţiilor care pot fi stabilite între resursele Web.

3.2.1 Clasificare a relaţiilor dintre resurse

În definirea şi descoperirea unei relaţii stabilite între două resurse Web, un factor important este dacă acea relaţie se bazează pe o cunoaştere explicită, precisă sau exactă ori pe cunoaşteri imprecise, aproximative (i.e. bazate pe măsurători statistice sau probabilistice). Mai ales în contextul spaţiului WWW, cel mai frecvent caz este cel de-al doilea.

Lucrarea [Sheth, Arpinar, Kashyap, 2002] propune trei criterii care trebuie luate în calcul atunci când se doreşte caracterizarea unei relaţii între resursele Web:

- conţinutul informaţional capturat de relaţia în cauză;
- modalităţile de reprezentare a relaţiei;
- metodologiile folosite pentru “calcularea” (*e.g.*, identificarea, descoperirea, validarea etc.) şi exploatarea relaţiilor.

Astfel, o clasificare ar putea împărţi relaţiile în:

- *relaţii independente sau dependente de conţinut*

  Relaţiile independente pot fi considerate ca fiind artifacte ale organizării conţinutului în contextul unui sistem de calcul pe baza unor raţiuni legate de performanţă ori de scalabilitate. De exemplu, două documente multimedia sunt legate printr-o relaţie pentru că sunt stocate pe acelaşi server Web, independent de conţinutul acestora.

  Din prisma Web-ului semantic, relaţiile care prezintă interes sunt relaţiile care se bazează pe conţinutul unor entităţi care modelează obiecte ale lumii reale. Astfel de relaţii sunt întâlnite în cazul ontologiilor şi pot fi asociate unor entităţi aparţinând unui acelaşi domeniu (*intra-domain relationships*) sau din domenii diferite (*inter-domain relationships*). Ca exemple de astfel de relaţii pot fi date relaţiile între entităţi provenind din domeniile diferite ale unor ontologii multiple (*inter-ontology relationships*).

  Două entităţi (indivizi) pot prezenta similarităţi semantice (aflându-se într-o vecinătate semantică), fără a se putea stabili o relaţie exactă (*crisp*) între ele. Un exemplu ar fi acela în care se utilizează un predicat *fuzzy*, i.e. `destul_de_aproape` (*close enough*), ca în cazul a două evenimente aflate într-o vecinătate temporală (a se vedea şi cele discutate în secţiunea 3.2.2).

- *relaţii clasificate în funcţie de reprezentare*

  Reprezentarea fundamentală a relaţiei dintre două concepte este reflectată de structura matematică folosită. În acest context, relaţiile se pot clasifica în funcţie de aritate, de cardinalitate, de structură etc.

  Pentru Web, de o importanţă majoră sunt relaţiile bazate pe hiperlegături[1], acestora putându-li-se asocia semantici via metadate, adoptându-se diverse tehnici precum cele bazate pe DCMI [Purl] sau XLink [DeRose *et al.*, 2001]. Limbajele de interogare, prezen-

[1] A se vedea [Botafogo *et al.*, 1991], [Buraga, 2001a], [Chakrabarti *et al.*, 1999] şi [Trăuşan-Matu, 2000].

tate în secţiunea 3.3.3.5, sunt principalele beneficiare ale structurii hipertextuale a relaţiilor dintre resursele căutate.

- *relaţii clasificate în funcţie de operaţiile efectuate asupra lor*

  Printre principalele operaţii realizate în vederea managementului şi exploatării relaţiilor se numără[2]:

  – *identificarea* – reprezintă procesul prin care poate fi determinată o relaţie a cărei semantică este cunoscută şi înţeleasă (*e.g.*, via reprezentarea ei);

  – *descoperirea* – este procesul prin care se caută – folosind diferite tehnici (*e.g.*, *pattern matching*) – relaţii noi, în cadrul unui model semantic sau al unei ontologii (de exemplu, referitoare la timp);

  – *validarea* – reprezintă procesul prin care este validată informaţia colectată şi analizată, folosind un ansamblu de relaţii complexe şi alţi operatori, pe baza unui model formal;

  – *evaluarea* – este procesul prin care o anumită relaţie poate fi "calculată", în funcţie de context şi de entităţile implicate.

3.2.2 Modele temporale

3.2.2.1 Introducere

Una dintre cele mai importante probleme privitoare la modelarea proprietăţilor dinamice ale sistemelor distribuite, în general, şi a spaţiului WWW, în special, este reprezentarea timpului. În funcţie de perspectivele de reprezentare a informaţiilor temporale, se pot utiliza diferite formalisme referitoare la timp [Allen, 1991].

Astfel, pentru a capta diferite proprietăţi ale sistemelor în timp-real distribuite (precum sisteme de management al bazelor de date, sisteme multimedia sincronizate sau spaţiul WWW), au fost dezvoltate o serie de modele formale temporale. În contextul unui cadru de lucru formal pentru sistemele Web distribuite implicând noţiunea de timp,

[2]Mai multe detalii sunt disponibile în [Sheth, Arpinar, Kashyap, 2002].

dintre problematicile care trebuie luate în consideraţie, se pot enumera următoarele [Chomicki, 1994]:

- alegerea *domeniilor temporale*: punctuale sau intervale, lineare sau ramificate, dense sau discrete, mărginite sau nemărginite;
- definirea unui sistem temporal *abstract* şi a unuia *real*;
- proiectarea şi utilizarea unor *limbaje de interogare temporală* pentru extragerea informaţiilor stocate pe Web (aici insistându-se asupra semanticii formale a limbajului, asupra expresivităţii interogărilor care pot fi formulate şi asupra implementării efective).

Lucrarea [Pnueli, 1977] prezintă o logică temporală simplă şi elegantă cu scopul de a surprinde dinamicitatea unor proprietăţi ale programelor concurente. S-au propus şi alte modele formale, dintre care se pot menţiona: logica temporală liniară (*Linear-time Temporal Logic*) [Lamport, 1980, Owicki & Lamport, 1982], logica temporală cu intervale (*Interval Temporal Logic*) [Allen & Hayes, 1989], logica temporală propoziţională (*Timed Propositional Temporal Logic*) [Alur & Henzinger, 1992, Alur & Henzinger, 1994], logica descriptivă temporală (*Temporal Description Logics*) [Artale & Franconi, 1999], logica temporală a acţiunilor (*Temporal Logic of Actions*) [Lamport, 2003] sau logica temporală ramificată (*branching time temporal logic CTL – Computation Tree Logic*) [Emerson, 1990].

Pentru o taxonomie a modelelor temporale, se poate consulta lucrarea [Knight & Ma, 1994].

Toate aceste propuneri s-au direcţionat mai ales asupra modelării din punct de vedere formal a sistemelor distribuite şi a componentelor acestora. Teoriile logice de mai sus au luat mai puţin în consideraţie aspectele exprimării într-o manieră cât mai apropiată de calculator a modelelor formale temporale, în vederea implementării diverselor aplicaţii (precum agenţi Web inteligenţi, mediatori Web sau servicii Web – a se vedea şi capitolul 4) pentru realizarea de activităţi de descoperire a resurselor sau de formulare de interogări implicând timpul.

3.2.3 Logica temporală cu intervale (*ITL – Interval Temporal Logic*)

3.2.3.1 Structuri temporale în logica ITL

Structura temporală introdusă de logica temporală cu intervale desemnează un model liniar al timpului. Noţiuni precum posibilitatea – folosită şi în modelele de timp ramificate – vor putea fi exprimate prin introducerea unui operator modal [Allen & Ferguson, 1997] care nu este intrinsec inclus în cadrul structurii temporale.

Date două intervale temporale, putem stabili 13 relaţii mutual exclusive între aceste intervale [Allen, 1983, Allen & Hayes, 1989]. Pentru a putea modela aceste relaţii, este introdusă primitiva $Meets$. Intuitiv, două perioade m şi n se pot întâlni (*meet*) dacă şi numai dacă m precede n, nu există o altă perioadă de timp între m şi n şi, în plus, m şi n nu se suprapun. Axiomatizarea perioadelor de timp este ilustrată în figura 3.1, unde i, j, k, l şi m sunt variabile desemnând perioade de timp. Pentru început, modelul nu ia în calcul perioade infinite sau semi-infinite ori puncte de început/sfârşit ale intervalului de timp considerat.

3.2.3.2 Sistemul axiomatic al logicii ITL

Modelul axiomatic al logicii ITL este prezentat în continuare:

- Pentru fiecare perioadă de timp, există o perioadă pe care o întâlneşte şi o alta cu care se întâlneşte. Formal, aceasta se poate scrie astfel:

$$\forall i \, \exists j, k : Meets(j, i) \wedge Meets(i, k) \tag{3.1}$$

- Perioadele de timp pot fi compuse (prin juxtapunere) pentru a produce o perioadă de timp mai mare. Pentru oricare două perioade care se întâlnesc, există o perioadă de timp care reprezintă "concatenarea" acestora:

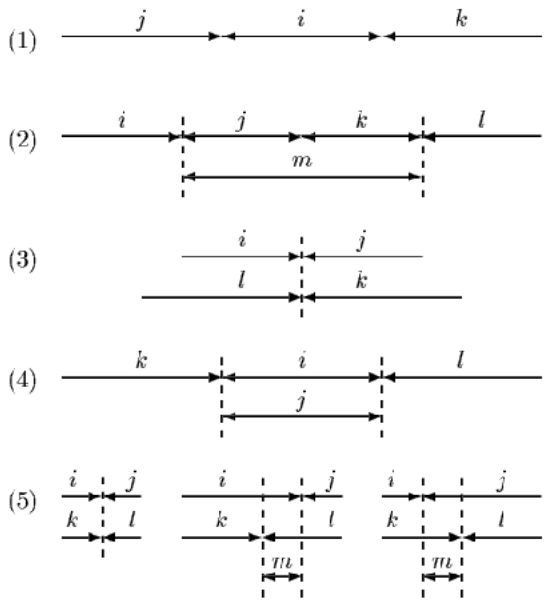

Figura 3.1: Axiomatizarea perioadelor de timp

$$\begin{aligned} \forall i,j,k,l \quad : \quad & Meets(i,j) \wedge Meets(j,k) \\ & \wedge Meets(k,l) \supset \\ & \exists m : Meets(i,m) \wedge Meets(m,l) \end{aligned} \tag{3.2}$$

Notaţia 1 *Vom nota $j+k$ pentru a desemna intervalul care reprezintă concatenarea intervalelor j şi k.*

Această notaţie este justificată deoarece se poate demonstra că rezultatul $j + k$ este unic [Allen & Hayes, 1989].

- Perioadele de timp definesc o clasă de echivalenţă a perioadelor care le întâlnesc. În particular, dacă i întâlneşte j şi i întâlneşte

k, atunci orice perioadă l care întâlneşte j trebuie de asemenea să întâlnească k. Formalizând, acest lucru se poate exprima în modul următor:

$$\begin{aligned} \forall i,j,k,l \quad : \quad & Meets(i,j) \wedge Meets(i,k) \\ & \wedge Meets(l,j) \supset Meets(l,k) \end{aligned} \tag{3.3}$$

Aceste clase de echivalenţă definesc în mod unic perioadele. Dacă două perioade întâlnesc simultan o aceeaşi perioadă de timp, iar o altă perioadă le întâlneşte pe primele două, atunci perioadele sunt egale:

$$\begin{aligned} \forall i,j,k,l \quad : \quad & Meets(k,i) \wedge Meets(k,j) \\ & \wedge Meets(i,l) \wedge Meets(j,l) \supset i = j \end{aligned} \tag{3.4}$$

- Vom introduce o *axiomă de ordonare*. Intuitiv, această axiomă exprimă faptul că dacă oricare ar fi două perechi de perioade, astfel încât i întâlneşte j şi k întâlneşte l, atunci fie ambele se întâlnesc în acelaşi "loc", fie locul unde i întâlneşte j precede locul unde k întâlneşte l sau vice-versa. În termenii relaţiei $Meets$, această situaţie poate fi axiomatizată după cum urmează (simbolul $\otimes$ desemnează operatorul logic "sau exclusiv" – *exclusive-or*):

$$\begin{aligned} \forall i,j,k,l \quad : \quad & (Meets(i,j) \wedge Meets(k,l)) \supset \\ & Meets(i,l) \otimes (\exists m \, : Meets(k,m) \wedge \\ & Meets(m,j)) \otimes \\ & (\exists m \, : Meets(i,m) \vee Meets(m,l)) \end{aligned} \tag{3.5}$$

Remarca 3 Multe dintre proprietăţile care în mod intuitiv sunt necesare şi care nu au fost încă menţionate pot fi considerate teoreme ale

acestei axiomatizări. Astfel, se poate demonstra că o perioadă nu se poate întâlni cu ea însăşi. La fel, dacă o perioadă i întâlneşte o alta j, atunci j nu poate întâlni i (modele de timp circulare finite nu sunt posibile).

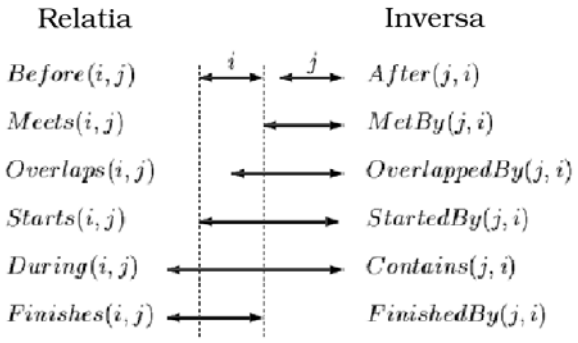

Figura 3.2: Relaţiile posibile între perioadele de timp (relaţia de egalitate nu este figurată) [Allen & Ferguson, 1997]

3.2.3.3 Specificarea relaţiilor temporale între perioade

Putem defini intuitiv cele treisprezece relaţii care se pot stabili între perioadele de timp. De exemplu, o perioadă are loc înaintea alteia dacă există o altă perioadă care "umple" timpul dintre acestea:

$$Before(i,j) \equiv \exists m \ : Meets(i,m) \wedge Meets(m,j) \tag{3.6}$$

În figura 3.2, ilustrăm grafic fiecare dintre relaţiile care se pot stabili între perioadele de timp (relaţia de egalitate nu este figurată).

Notaţia 2 *Simbolurile de mai jos vor putea fi utilizate pentru a exprima relaţiile uzuale şi disjuncţiile acestor relaţii:*

$$\begin{aligned} Meets(i,j) &\quad i:j \\ Before(i,j) &\quad i \prec j \\ During(i,j) &\quad i \sqsubset j \\ Before(i,j) \vee Meets(i,j) &\quad i \prec: j \\ During(i,j) \vee i = j &\quad i \sqsubseteq j \end{aligned}$$

Definiţia 10 *Între două perioade de timp, poate fi stabilită şi relaţia* $Disjoint$.
Două intervale sunt disjuncte *dacă acestea nu se suprapun în nici un mod. Vom nota aceasta prin "*$i \bowtie j$*" şi o vom defini formal astfel:*

$$i \bowtie j \equiv i \prec: j \vee j \prec: i \tag{3.7}$$

3.2.3.4 Caracterizarea perioadelor de timp

O perioadă poate fi clasificată în funcţie de relaţiile pe care le poate avea cu alte perioade de timp. O perioadă care nu are sub-perioade va fi denumită *moment*, iar una care are sub-perioade de timp va putea purta numele de *interval*. De asemenea, putem defini noţiunea de *punct* prin intermediul unei construcţii care specifică începutul şi sfârşitul unei perioade.

Astfel, fiecărei perioade t îi putem asocia punctele t.`start` şi t.`end`.

Remarca 4 De remarcat faptul că momentele şi punctele sunt considerate distincte şi nu pot fi colapsate.

În particular, momentele pot fi văzute drept perioade de timp şi pot întâlni alte perioade: dacă $Meets(i,m) \wedge Meets(m,j), \forall m$, atunci $i < j$.

Punctele nu sunt perioade şi nu pot întâlni perioade.[3]

[3] Pentru detalii, a se vedea [Allen & Hayes, 1989].

Logica ITL specifică un model de timp discret, în care perioadele sunt asociate unor perechi de numere întregi $\langle I, J \rangle$, unde $I < J$. Momentele corespund perechilor de forma $\langle I, I + 1 \rangle$, iar punctele sunt desemnate de aceşti întregi. Un model similar, luând în consideraţie perechi de numere reale în locul celor întregi nu va permite momente.

Relaţiile temporale ITL acoperă toate posibilele relaţii stabile între două intervale de timp, putând descrie ce s-a întâmplat în trecut între două intervale temporale. Relaţiile nu pot fi folosite în cazul în care durata unui anumit eveniment nu este cunoscută (de exemplu, punctul de sfârşit al unui interval este necunoscut).

Există 12 relaţii, prezentate în tabelul 3.1, stabilite între cele patru puncte ale celor două intervale t_1 şi t_2 temporale considerate (t_1.`start`, t_1.end, t_2.`start` şi t_2.end). Se poate remarca faptul că există două relaţii implicite care, evident, sunt îndeplinite întotdeauna: t_1.`start` $<$ t_2.end şi t_2.`start` $<$ t_2.end.

t_1.end < t_2.`start`	t_1.end = t_2.`start`	t_1.end > t_2.`start`
t_1.`start` < t_2.end	t_1.`start` = t_2.end	t_1.`start` > t_2.end
t_1.`start` < t_2.`start`	t_1.`start` = t_2.`start`	t_1.`start` > t_2.`start`
t_1.end < t_2.end	t_1.end = t_2.end	t_1.end > t_2.end

Tabela 3.1: Relaţiile stabilite între punctele de început şi de sfârşit ale două intervale temporale [Yu, 1997]

Relaţiile ilustrate în tabelul 3.1 pot fi restrânse la zece relaţii care nu mai sunt însă mutual exclusive. Aceste relaţii sunt prezentate în tabelul 3.2 şi pot fi folosite la modelarea comportamentului unei prezentări hipermedia sincronizate [Yu, 1997, Buraga, 2001a].

Proprietăţile computaţionale ale calculului cu intervale şi algoritmii pentru menţinerea reţelelor de restricţii temporale sunt prezentate în [Allen, 1991, Allen, 1983, Vilain *et al.*, 1990]. De asemenea, reprezentarea formală a evenimentelor şi acţiunilor bazată pe logica temporală cu intervale se poate găsi în [Allen & Ferguson, 1997], iar o teorie

t_1.end $\leqslant t_2$.start	t_1.end > t_2.start
t_1.start < t_2.end	t_1.start = t_2.end
t_1.start < t_2.start	t_1.start = t_2.start
t_1.start > t_2.start	t_1.end < t_2.end
t_1.end = t_2.end	t_1.end > t_2.end

Tabela 3.2: Relaţiile restrânse între punctele de început şi de sfârşit ale două intervale temporale [Yu, 1997]

cantitativă a mişcării bazată pe primitive spaţio-temporale este descrisă în [Vieu, 1991]. Au fost formulate şi o serie de extinderi ale modelului ITL, dintre care se poate menţiona [Knight & Ma, 1993].

Logica ITL poate fi îmbogăţită cu construcţii exprimând evenimente şi acţiuni pentru a captura diferite proprietăţi referitoare la dinamica sistemelor distribuite.[4]

Printre utilizările logicii ITL se enumeră cele legate de structura scenariilor temporale, fiind modelul formal de bază folosit în cadrul unor limbaje precum *SMIL* (*Synchronized Multimedia Integration Language*) [Ayars *et al.*, 2001], *TimeML* [Pustejovsky *et al.*, 2002] sau *DAML+TIME* [DAML+TIME]. În cadrul secţiunii 3.2.7 vom vedea cum putem utiliza logica ITL în exprimarea relaţiilor temporale care pot fi stabilite între diverse resurse Web.

3.2.4 Sisteme de fişiere distribuite

Pentru început, vom arăta cum putem modela şi specifica – într-un limbaj bazat pe XML şi folosind aserţiuni RDF – proprietăţile sistemelor de fişiere distribuite, pentru ca apoi modelul să fie extins la resurse multimedia disponibile pe Web.

[4] Una dintre aceste abordări este prezentată în [Allen & Ferguson, 1997].

3.2.4.1 Caracterizare

Un *sistem distribuit* reprezintă o colecţie de calculatoare interconectate prin intermediul unei reţele de comunicaţie. Din punctul de vedere al unui computer făcând parte dintr-un sistem distribuit, restul maşinilor şi resursele acestora sunt considerate a fi *resurse aflate la distanţă* (*remote*), pe când resursele sale proprii sunt numite *resurse locale*. Un calculator care are ataşată o adresă şi care face parte dintr-un sistem distribuit se va numi *gazdă* (*host*).

Parte importantă a unui sistem de operare distribuit, *sistemul de fişiere* pune la dispoziţie clienţilor săi servicii specifice. Interfaţa prin intermediul căreia se pot accesa asemenea servicii constă dintr-un set de primitive denumite *operaţii cu fişiere* (*file operations*). Cele mai frecvent folosite operaţii sunt: deschiderea unui fişier, ştergerea unui fişier, citirea dintr-un fişier sau scrierea într-un fişier.

Definiţia 11 *Un* sistem de fişiere distribuit *este un sistem de fişiere ai cărui clienţi, servere şi dispozitive de stocare (*e.g.*, discuri) sunt dispersate în cadrul calculatoarelor interconectate aflate în componenţa unui sistem distribuit* [Silberschatz *et al.*, 2001].

Un sistem de fişiere constă din două părţi distincte: o colecţie de *fişiere* (*files*) existente, fiecare conţinând informaţii înrudite, şi o *structură de directoare* (*directory structure*) care oferă informaţii referitoare la toate fişierele prezente în sistem.

Sistemele de operare actuale adoptă ca schemă de stocare logică a fişierelor un sistem de directoare bazat pe grafuri aciclice, conform [Silberschatz *et al.*, 2001] şi [Tanenbaum, 2001].

În practică, configuraţiile şi implementările concrete ale sistemelor de fişiere distribuite sunt multiple, fiind dificil de determinat care este implementarea optimă. Un sistem de fişiere distribuit poate fi implementat ca parte componentă a unui sistem de operare distribuit sau poate fi regăsit ca fiind plasat pe un nivel software distinct, având ca funcţie primară pe aceea de a realiza managementul comunicaţiei dintre sistemele de operare convenţionale şi sistemele de fişiere existente.

3.2.4.2 Fişierele ca tipuri abstracte de date

Putem considera un fişier ca fiind un tip abstract de date. În vederea manipulării acestui tip de date, vom defini un set minimal de operaţii cu fişiere (primitive) în modul următor [5]:

Definiţia 12 Operaţiunile cu fişiere *se definesc în maniera de mai jos:*

- open( ) *se utilizează pentru deschiderea unui fişier. Această primitivă returnează o valoare specială denumită* descriptor (handler) *prin intermediul căruia se va accesa fişierul via celelalte primitive:*

$$h = open(f), f \in \text{Filenames}, h \in \text{Handlers}$$

- close( ) *este folosită la închiderea unui fişier şi la eliberarea resurselor alocate acestuia (precum descriptorul asociat):*

$$close(h), h \in \text{Handlers}$$

- seek( ) *se utilizează pentru a stabili pointerul de fişier în vederea efectuării următoarei operaţii de intrare/ieşire; această primitivă oferă posibilitatea de a accesa un fişier în manieră secvenţială sau directă:*

$$seek(h, p), h \in \text{Handlers}, p \in \mathbb{Z}$$

- read( ) *este folosită pentru a citi dintr-un fişier anumite informaţii ce vor fi stocate într-o zonă de memorie internă de lungime fixă, zonă numită* buffer*:*

[5] Conform celor prezentate în [Buraga, 2000a] şi [Buraga & Ciobanu, 2001, capitolul 2].

$$read(h, m, s), h \in \mathsf{Handlers}, m \in \mathsf{Memory}, s \in \mathbb{N}$$

- write( ) *este utilizată la scrierea într-un fişier a unor date stocate într-o zonă de memorie internă:*

$$write(h, m, s), h \in \mathsf{Handlers}, m \in \mathsf{Memory}, s \in \mathbb{N}$$

Formal, mulţimile Handlers, FileNames şi Memory pot fi considerate tipuri abstracte de date.

În practică, Handlers $\subset \mathbb{N}$, FileNames $\subset \mathsf{Chars}^+$ şi Memory $\subset \mathbb{N}$ (adrese de locaţii de memorie), unde Chars este o mulţime de caractere valide care pot fi incluse în numele unui fişier (submulţime a codurilor ASCII sau Unicode).

Programatorii au la dispoziţie multe alte primitive folositoare (a se vedea [Buraga & Ciobanu, 2001]). Aceste primitive sunt implementate frecvent de nucleul sistemului de operare.

3.2.4.3 Interfaţa de programare

Printre cele mai răspândite sisteme de fişiere actuale se numără sistemele de fişiere *ext2*, *ext3* sau *proc* prezente în Linux, sistemul de fişiere *HPFS* (*High Performance File System*) adoptat de IBM OS/2 sau sistemele de fişiere *VFAT* (*Virtual File Allocation Table*) şi *NTFS* (*Windows NT File System*) [Tanenbaum, 2001]. Fiecare sistem de fişiere particular prezintă aceeaşi interfaţă de programare sau utilizare. Un fişier este reprezentat sub forma unei structuri abstracte de date numită *vnode* (*virtual information node*). Aceste *vnodes* sunt structuri de date utilizate de un sistem de fişiere virtual. Pentru fiecare sistem de fişiere sau pentru fiecare tip de fişier vom putea deriva din clasa *vnode* o clasă specializată pentru a specifica primitivele (metodele) particulare corespunzătoare managementului informaţiilor pentru acel tip de sistem. În cadrul clasei *vnode*, fiecare operaţie poate fi considerată ca fiind virtuală pură.

Pentru uniformitate, clasa *vnode* va putea fi folosită să reprezinte în manieră uniformă şi alte entităţi (abstracţiuni) utile, precum conducte (*pipes*), dispozitive, pseudo-dispozitive, procese sau socluri (*sockets*). Astfel, fiecare dispozitiv special (i.e. scaner, cameră digitală, terminal virtual) sau linie de comunicaţie pot fi considerate drept fişiere şi aceleaşi primitive pot fi utilizate pentru accesul la informaţii particulare deţinute de un fişier anume.

Acest mecanism este similar celui adoptat de identificatorii uniformi de resurse (URI) [Berners-Lee *et al.*, 1998], unde fiecare protocol Internet se poate specifica printr-o schemă URI diferită, fără însă a modifica forma generală a identificatorului asociat unei anumite resurse.

3.2.4.4 Proprietăţi

Proprietăţile generale ale unui sistem de fişiere distribuit, oferind suport pentru tehnologiile Internet actuale, sunt cele referitoare la scalabilitate, suport pentru modelul de programare client/server, organizare globală a fişierelor independentă de localizarea efectivă, administrare *on-line*, recuperare în caz de dezastru bazată pe fişiere-jurnal (*log-files*), replicare sigură şi securitate [Kramer, 1996].

În afară de alte caracteristici, o atenţie specială trebuie acordată numirii şi transparenţei fişierelor din cadrul unui sistem de fişiere distribuit, mai ales în contextul oferit de Internet:

- *Numirea* reprezintă o funcţie de punere în corespondenţă a obiectelor fizice ale reţelei la cele logice (cele disponibile utilizatorului final sau anumitor servicii de nivel înalt). Problema importantă este cea de a adopta o schemă de numire judicioasă, astfel încât să fie evitate ambiguităţile, duplicările de nume sau inconsistenţa grafurilor de directoare. Una dintre soluţii o reprezintă ataşarea de directoare aflate la distanţă (corespunzătoare unor sisteme de fişiere externe) la structura de directoare locală, păstrându-se în acest mod o structură arborescentă coerentă de organizare logică a fişierelor. Această ataşare se realizează prin intermediul operaţiei de *montare* (*mount*) a unui sistem de fişiere, recurgându-se la

un protocol special [Tanenbaum, 2001, Silberschatz *et al.*, 2001, Comer & Stevens, 1993].

O altă abordare va fi prezentată în cadrul următoarei secţiuni.

- *Transparenţa* conferă iluzia utilizatorului final că localizarea fizică a fişierelor nu este importantă. Aceasta conduce la posibilitatea *replicării* fişierelor, un tip special de redundanţă folosită la creşterea disponibilităţii informaţiilor. Metadatele privitoare la un fişier particular (de exemplu, nume, proprietar, tip, dimensiune, adresă etc.) vor putea fi cunoscute de toţi membrii reţelei, indiferent de localizarea acestui fişier. Acest aspect devine cu mult mai important în contextul Web-ului şi al Internet-ului fără fir (*wireless*), în cazul unor activităţi precum descoperirea şi regăsirea resurselor sau utilizarea *ad-hoc* a capacităţilor de calcul ale unui grup de computere [Grigoraş, 2004]. Procesul de replicare poate fi utilizat pe orice sistem de operare actual – *e.g.*, folosind serviciile *NIS* (*Network Information Service*) disponibile în Linux.

În prezent şi în viitorul apropiat, sistemele de operare (mai ales cele distribuite) trebuie să ofere suport pentru integrarea diverselor servicii Internet (Web), în special în ceea ce priveşte punerea la dispoziţie a unui mecanism flexibil pentru accesarea de la distanţă a resurselor, mecanism având la bază funcţionalităţile expuse de sistemele de fişiere tradiţionale.

Deşi maniera de stocare a resurselor va putea rămâne aceeaşi, sistemele de operare vor putea utiliza tehnologiile Web – mai ales, cele bazate pe XML – în vederea facilitării schimbului la distanţă de informaţii privitoare la componentele sistemelor de fişiere (*e.g.*, ierarhii de directoare, resurse, metadate etc.).

Ca exemple actuale de sisteme de fişiere distribuite, pot fi menţionate:

- *NFS* (*Network File System*) [Comer & Stevens, 1993, NFS] – oferă un mecanism bazat pe paradigma *RPC* (*Remote Procedure Call*) pentru accesarea la distanţă a fişierelor. Sistemul NFS permite unui calculator să ruleze ca server pentru a permite unui grup de

fişiere să fie disponibil în exterior aplicaţiilor de pe alte computere. NFS utilizează multe facilităţi puse la dispoziţie de sistemul de fişiere tradiţional din UNIX şi presupune un sistem de numire bazat pe ierarhii de directoare. O altă abordare este *NIS* (*Network Information Service*) [NIS], implementat pe sistemele Linux. Modelul RDF propus în această lucrare poate fi utilizat pentru descrierea diferitelor ierarhii ale fişierelor aflate la distanţă, folosind convenţiile adoptate de serverele NFS.

- *Active Directory* [MSDN] – rulează ca serviciu special pe platformele Windows 2000/XP pentru a oferi diferite funcţionalităţi în vederea instituirii unui catalog logic arborescent de resurse (directoare, fişiere, utilizatori, echipamente etc.) în cadrul unei reţele de calculatoare conectate la Internet (de exemplu, intranetul sau extranetul unei organizaţii). Active Directory, ca şi NIS, face topologia fizică a reţelei şi protocoalele de comunicaţie folosite să fie transparente pentru utilizator şi oferă suport pentru managementul PC-urilor distribuite, serviciilor de reţea şi aplicaţiilor conexe. Propunerea noastră bazată pe XML/RDF poate modela diverse informaţii referitoare la structura logică a unui Active Directory (*e.g.*, domenii, unităţi organizaţionale etc.).

- *Prospero* [Prospero] – model de sistem virtual compatibil cu tehnologiile Internet, bazat pe identificatori uniformi de resurse; o propunere similară este şi *Coda* [Kramer, 1996] – un sistem de fişiere experimental dezvoltat la Universitatea Carnegie Mellon.

La nivel de Internet, pentru facilitarea serviciilor de numire există diverse standarde, dintre care enumerăm *INS* (*Intentional Naming System*) [Balazinska *et al.*, 2002], *JNDI* (*Java Naming and Directory Interface*) [Lee & Selgman, 2000] şi *UDDI* (*Universal Description, Discovery and Integration*) [UDDI].

3.2.5 Un model de descriere XML/RDF a sistemelor de fişiere distribuite

În cadrul acestei secţiuni, vom vedea cum pot fi formulate aserţiuni RDF privitoare la relaţiile spaţiale dintre diverse fişiere stocate de sisteme de fişiere eterogene. De asemenea, metadatele asociate fişierelor vor putea fi reprezentate prin intermediul elementelor şi atributelor unui limbaj bazat pe XML. Asimilând noţiunea de fişier cu cea de resursă şi desemnând resursele prin identificatori uniformi de resurse, prin extindere putem oferi un model RDF de descriere a relaţiilor spaţiale care pot fi stabilite între resurse Web multimedia.

Pentru a putea formula construcţii RDF despre diferite caracteristici ale fişierelor, pentru început vom defini un limbaj bazat pe XML care va fi folosit la stocarea proprietăţilor referitoare la fişiere, în special a metadatelor. Acest limbaj a fost denumit *Extensible File Properties Markup Language* (*XFiles*), variantele preliminare fiind descrise în lucrările [Buraga, 2000a], [Buraga, 2002b], [Buraga, 2002c] şi [Buraga, 2003d].

Printre cerinţele importante pe care trebuie să le satisfacă un limbaj de modelare a metadatelor putem menţiona: *acurateţea*, *completitudinea*, *consistenţa*, *persistenţa* şi *inteligibilitatea* [Rotenberg, 1996]. În proiectarea limbajului *XFiles* a fost urmărită îndeplinirea tuturor acestor deziderate.

3.2.5.1 Limbajul *XFiles*

Vom defini *xf* ca fiind spaţiul de nume XML [Bray *et al.*, 1999] pentru toate elementele şi atributele limbajului *XFiles* în vederea evitării ambiguităţilor apariţiei acestor construcţii sintactice în alt context. Pentru fiecare proprietate a unei resurse (fişier), limbajul *XFiles* oferă un element care corespunde acelei proprietăţi (de exemplu, elementul `<Location>` pentru proprietatea *Location* desemnând locaţia resursei).

Pentru validarea şi procesarea informaţiilor adnotate în XML, vom crea o schemă XML. În locul unei abordări bazate pe definirea tipului de document (*DTD – Document Type Definition*), vom folosi standardul

XML Schema [Fallside, 2001], deoarece schemele sunt mai flexibile şi pot fi extinse facil, iar în plus – datorită faptului că gramatica lor se bazează pe construcţii XML – pot fi uşor de procesat de către analizoarele XML. O schemă oferă o specificaţie formală a gramaticii[6] asociate unui limbaj bazat pe XML, utilizându-se o sintaxă XML.

Schema XML care validează sintactic limbajul *XFiles* este prezentată în anexa B.

Elementul rădăcină al unui document *XFiles* este *Properties*. Acest element poate conţine, în orice ordine, următoarele sub-elemente:

- *Type* – elementul reflectă tipul unui fişier: normal, director, *pipe*, legătură spre alt fişier (*link*), dispozitiv de tip bloc ori caracter sau *socket*[7], în cazul unui sistem UNIX/Linux; atributul *mime* specifică tipul *MIME* (*Multipurpose Internet Mail Extensions*) care poate fi asociat unei resurse (de exemplu, `text/html`, `image/png` sau `video/mpeg`) [Borenstein & Freed, 1992];

- *Location* – acest element desemnează adresa IP a gazdei care stochează fişierul considerat; atributul *dns* se poate utiliza pentru a specifica numele simbolic al gazdei, adoptându-se sistemul *DNS* (*Domain Name System*) [Tanenbaum, 2003]; atributul *port* va indica un alt port de conectare (implicit fiind considerat portul standard la care este asociat serviciul respectiv – *e.g.* pentru protocolul HTTP portul standard este 80 [Fielding *et al.*, 1997]);

- *Auth* – elementul specifică metoda de autentificare folosită pentru accesarea unei resurse date (de exemplu, se poate recurge la utilizarea metodelor *Basic* sau *Digest* [Franks *et al.*, 1997], puse la dispoziţie de serverele Web);

- *Owner* – acest element conţine informaţii privitoare la posesorul (proprietarul) unui fişier: numele de cont (*login*), parola, grupul de utilizatori din care face parte, numele real; este posibil să existe posesori multipli pentru acelaşi fişier considerat;

[6] Conform teoriei limbajelor formale [Jucan, 1999, Jucan & Andrei, 2002].
[7] Amănunte în lucrările [Buraga & Ciobanu, 2001] şi [Comer & Stevens, 1993].

- *Size* – acest element specifică dimensiunea actuală a unui fişier; atributul *max* poate fi folosit pentru a stabili o dimensiune maximă permisă;

- *Permissions* – elementul include o serie de permisiuni (drepturi de acces) asociate unui fişier; de exemplu, putem avea permisiuni de citire (*read*), scriere (*write*) şi execuţie (*execute*) în cazul unui sistem UNIX şi drepturi ca *Full Control*, *Read* sau *Take Ownership* în cazul unui sistem Windows [Silberschatz *et al.*, 2001, Tanenbaum, 2001];

- *Timestamp* – acest element dă posibilitatea consultării timpilor de acces (*access*), de modificare (*modification*) sau de schimbare (*change*) a stării unui fişier dat;

- *Version* – elementul se poate utiliza în cadrul unui mediu de control al versiunilor – *e.g.*, *CVS* (*Concurrent Versions Systems*) [CVS] –, pentru realizarea unui istoric al versiunilor conţinutului unui fişier;

- *Parse* – acest element specifică o aplicaţie care poate fi utilizată pentru procesarea conţinutului unui fişier (*e.g.*, un editor de texte, un compilator, un vizualizator etc.); poate fi folosit şi atributul *params* pentru a pasa opţiuni, parametri sau argumente suplimentare programului de prelucrare respectiv.

3.2.5.2 Exemplu

În continuare, va urma un exemplu folosind construcţii RDF referitoare la resursele unui sistem de fişiere distribuit [Buraga, 2002b]:

Exemplul 21 Pentru a specifica o proprietate de posesie şi o metodă de autentificare bazată pe parolă în vederea accesului la un set de fişiere stocate pe mai multe calculatoare – locale şi aflate la distanţă –, vom putea defini următoarele aserţiuni RDF (omitem declararea spaţiilor de nume pentru limbajele RDF şi *XFiles*):

```
<?xml version="1.0" ?>
<rdf:RDF>
  <!-- o colectie de resurse -->
  <rdf:Bag rdf:ID="myfiles">
    <!-- fisier disponibil local prin schema URI "file" -->
    <rdf:li rdf:resource="file:///tmp/article.tex" />
    <!-- fisier disponibil la distanta via protocolul FTP
         (se foloseste schema URI "ftp" -->
    <rdf:li rdf:resource="ftp://ftp.tuiasi.ro/pub/gaen" />
    <!-- eventual si alte resurse -->
  </rdf:Bag>

  <!-- asertiuni RDF referitoare la colectia de resurse -->
  <rdf:Description rdf:about="#myfiles">
    <!-- metadatele si alte informatii sunt specificate
         utilizand constructii XFiles -->
    <xf:Properties>
      <xf:Auth>Basic</xf:Auth>
      <xf:Owner>
        <!-- proprietarul este identificat si
             prin intermediul sitului Web -->
        <rdf:Description
          rdf:about="http://www.infoiasi.ro/~busaco">
          <xf:Login xf:uid="714">busaco</xf:Login>
          <xf:Password>NU74b33cs</xf:Password>
        </rdf:Description>
      </xf:Owner>
      <xf:Permissions>
        <xf:Permission>User-Read</xf:Permission>
        <xf:Permission>User-Write</xf:Permission>
        <xf:Permission>Group-Read</xf:Permission>
      </xf:Permissions>
      <xf:Timestamp xf:type="modification">
        2003-09-01T14:00
      </xf:Timestamp>
    </xf:Properties>
  </rdf:Description>
</rdf:RDF>
```

Au fost exprimate următoarele:
“Pentru colecţia de fişiere (resurse) dată, posesorul acestor fişiere este utilizatorul cu numele de cont busaco. Fişierele vor putea fi accesate prin intermediul unei parole şi doar posesorul lor le va putea modifica (permisiunea *User-Write*) şi consulta (permisiunea *User-Read*). Membrii din grupul la care aparţine proprietarul fişierelor vor avea acces doar la conţinutul acestora (permisiunea *Group-Read*)”.

Spaţiul de nume *xf* corespunde tuturor elementelor şi atributelor limbajului *XFiles* (a se vedea anexa B).

3.2.6 Modelarea resurselor stocate de un server Web

În cadrul acestei secţiuni, vom descrie modalitatea de utilizare a limbajului *XFiles* – prezentat în cadrul secţiunii 3.2.5.1 –, în cazul resurselor stocate pe un sit sau grup de situri Web.

Pentru aceasta plecăm de la premisa că oricare resursă Web va fi memorată pe un dispozitiv de stocare oferit de serverul Web via un sistem de fişiere. Cum resursele Web pot fi stocate distribuit, considerăm drept suport de memorare un sistem de fişiere distribuit.

Formal, putem defini următoarele:

Definiţia 13 *Un* server Web $\mathcal{W}$ *poate fi reprezentat abstract printr-o funcţie* $\mathcal{W} : \mathsf{U} \rightarrow \mathsf{R}$ *care asociază unei resurse solicitate desemnată de un identificator uniform de resursă* $u \in \mathsf{U}$ *o pereche* $\langle c, r \rangle \in \mathsf{R}$*, unde* c *reprezintă un cod de răspuns (stare) HTTP* [8] *(*e.g.*, “200 Ok” sau “404 Not found”), iar* r *este conţinutul acelei resurse –* r *aparţine mulţimii tipurilor MIME:*

$$\mathcal{W}(u) = \langle c, r \rangle \tag{3.8}$$

[8] A se consulta lucrările [Fielding *et al.*, 1997] sau [Buraga *et al.*, 2002a].

Remarca 5 Funcţia $\mathcal{W}$ este totală [Dyreson, 2001] din moment ce un server va întoarce întotdeauna o resursă-răspuns, chiar şi în cazul în care resursa solicitată nu este disponibilă (într-o astfel de situaţie serverul va genera un conţinut HTML indicând indisponibilitatea resursei şi codul de stare corespunzător).

În vederea modelării versiunilor posibile ale unei resurse disponibile pe un server Web, putem lua în consideraţie faptul că, pentru fiecare modificare (editare) a unei resurse, este creată o nouă versiune a acesteia. Din acest punct de vedere, dinamica actualizării conţinutului resurselor poate fi considerată ca fiind o *tranzacţie*[9], oferindu-se astfel suport pentru replicarea datelor (conform celor prezentate în secţiunea 3.2.4.4). În ceea ce priveşte numirea, modelul prezentat se bazează pe facilităţile oferite de identificatorii uniformi de resurse (*e.g.*, scheme de adresare, independenţă de platformă, uniformitate etc.).

3.2.6.1 Exemplu

Exemplul 22 Vom prezenta în continuare un exemplu de aserţiuni RDF care includ construcţii *XFiles* pentru specificarea de alternative pentru execuţia de la distanţă a unui script (în acest caz particular, un analizor XML). Se va recurge la construcţia `<Alt>` oferită de RDF (a se vedea secţiunea 2.2.6 a capitolului 2).

```
<?xml version="1.0" ?>
<rdf:RDF>
  <!-- o descriere RDF a unui fisier XML local -->
  <rdf:Description
    rdf:about="file://localhost/SEAweb/teza.xml">
    <xf:Properties>
      <!-- diverse metadate asociate -->
      <xf:Type xf:mime="text/xml">
           ordinary
```

[9]Conform cercetărilor întreprinse în domeniul bazelor de date – a se vedea, de exemplu, [Roddick & Snodgrass, 1995] şi [Tsotras, 1996].

```
</xf:Type>
<xf:Owner uid="714">
     busaco
</xf:Owner>
<!-- aplicatia cu care se va prelucra
     continutul documentului XML -->
<xf:Parser xf:params="-q">
  <rdf:Alt>
    <!-- vor exista doua posibilitati (utilizarea
         a doua programe de analiza XML,
         disponibile pe platforme diferite)
    <rdf:li>
      <!-- descrierea unei aplicatii
           disponibile local -->
      <rdf:Description
        rdf:about="file:///sbin/expat">
        <xf:Type xf:mime="application/executable">
          ordinary
        </xf:Type>
        <xf:Location xf:dns="localhost">
          127.0.0.1
        </xf:Location>
      </rdf:Description>
    </rdf:li>
    <rdf:li>
      <!-- descrierea unui program
           disponibil la distanta -->
      <rdf:Description
        rdf:about="https://winhost.ro/xmled.exe">
        <xf:Type xmime="application/octet-stream">
          ordinary
        </xf:Type>
        <xf:Location xf:dns="it2.infoiasi.ro">
          193.231.30.228
        </xf:Location>
      </rdf:Description>
    </rdf:li>
  </rdf:Alt>
```

```
      </xf:Parser>
    </xf:Properties>
  </rdf:Description>
</rdf:RDF>
```

În acest exemplu, putem observa utilizarea declaraţiilor RDF pentru specificarea a două aplicaţii (elemente ale colecţiei de resurse de tip *Alt*), una disponibilă – via serverul Web – pe maşina locală, iar cealaltă accesată în manieră securizată (HTTPS) de pe un sistem Windows. Unul dintre aceste programe va putea fi folosit pentru procesarea documentului XML dat. Se remarcă, de asemenea, utilizarea elementului `<Type>` al limbajului *XFiles* pentru a exprima tipul fişierului ce urmează a fi prelucrat, atât din punctul de vedere al sistemului de fişiere nativ (în acest caz, fişier normal – *ordinary*), cât şi din punctul de vedere al aplicaţiilor Web (se foloseşte atributul `mime` pentru definirea tipului MIME corespunzător).

3.2.7 Modelarea relaţiilor temporale dintre resurse

Conform [Hobbs & Pustejovsky, 2002], printre necesităţile importante ale Web-ului semantic se numără specificarea informaţiilor temporale.

În cadrul acestei secţiuni, ne vom concentra asupra modelării relaţiilor temporale stabilite între documentele Web. Pentru a exprima relaţiile temporale care se pot stabili între diferite resurse Web sau (fragmente de) situri Web, vom recurge la modelul formal ITL prezentat în secţiunea 3.2.3. Lucrarea propune o descriere de nivel înalt bazată pe construcţii RDF a acestor relaţii, scopul principal fiind cercetarea dinamicii modificărilor survenite în conţinutul şi structura siturilor Web, în general, şi ale resurselor multimedia, în special.

Expresiile temporale vor fi specificate într-un limbaj bazat pe XML, limbaj purtând numele *TRSL* (*Temporal Relation Specification Language*) [Buraga & Ciobanu, 2002]. Acest limbaj va exprima relaţiile $Before$, $Meets$, $Overlaps$, $Start$, $During$ şi $Finishes$ care pot fi stabilite între două resurse Web r_1 şi r_2. Aceste resurse vor fi identificate prin adresele lor (identificatori uniformi de resurse – URI) uri_1 şi uri_2, respectiv.

Nu impunem nici o restricţie în ceea ce priveşte specificul resurselor considerate, acestea putând fi:

- fişiere media de sine-stătătoare (*e.g.*, imagini în formatele raster PNG sau JPEG ori în format vectorial SVG, fişiere audio MP3 sau AU, documente video MPEG ori AVI etc.)[10];
- colecţii de resurse multimedia (de exemplu, prezentări multimedia sincronizate SMIL [Ayars *et al.*, 2001] sau animaţii Flash);
- elemente programabile, precum:
  - scripturi CGI concepute în limbaje precum bash, C/C++ şi Perl [Buraga *et al.*, 2002a];
  - programe rulând în cadrul unor servere de aplicaţii ca PHP, JSP sau ASP [Buraga, 2001a];
  - *servlet*-uri Java [Tanasă *et al.*, 2003];
  - servicii Web [Vasudevan, 2001];
  - agenţi software disponibili pe Web [Alboaie & Buraga, 2002, Luck, McBurney & Preist, 2003];
- fragmente sau situri Web complete.

Colecţiile de resurse vor putea fi exprimate prin structuri `<Bag>`, `<Alt>` sau `<Seq>` puse la dispoziţie de RDF şi desemnate de atributul `about` (a se vedea secţiunea 2.2 a capitolului 2 şi exemplele de mai jos).

Pentru a exprima un model temporal cât mai general, limbajul TRSL va permite specificarea atât a intervalelor de timp (desemnate de puncte de început şi sfârşit), cât şi a momentelor.

Vom defini *t* ca spaţiu de nume pentru toate construcţiile sintactice ale limbajului TRSL. Pentru fiecare relaţie temporală introdusă de algebra ITL şi prezentată în cadrul secţiunii 3.2.3.3, TRSL oferă un

[10] A se consulta şi [Buraga, 2002a] sau [Brut & Buraga, 2004] pentru o listă a tipurilor şi formatelor de resurse multimedia care pot intra în componenţa unui sit Web.

element corespunzător acelei relaţii (de exemplu, elementul <Meets> va corespunde relaţiei binare $Meets$). Începutul şi sfârşitul perioadelor de timp vor fi exprimate prin intermediul a două atribute denumite begin şi, respectiv, end. Aceste atribute sunt inspirate de specificaţia limbajului *SMIL* (*Synchronized Multimedia Integration Language*) [Ayars *et al.*, 2001]. Valorile atributelor begin şi end pot fi furnizate în formatul specificat de XML Schema [Fallside, 2001].

De asemenea, TRSL defineşte un atribut dur pentru a se putea specifica o perioadă de timp *a-priori* cunoscută sau predictibilă. Aceasta va permite, de exemplu, agenţilor Web să iniţieze diverse acţiuni care vor putea fi executate pe o durată determinată de atributul dur). Valorile care le poate lua atributului dur sunt identice cu cele specificate în [Fallside, 2001] (convenţiile utilizate de schemele XML).

Sursa şi destinaţia (adică operanzii) relaţiilor temporale vor fi exprimate prin intermediul unor aserţiuni RDF.

3.2.7.1 Sintaxa şi semantica limbajului TRSL

Pentru a defini *sintaxa* limbajului TRSL, vom folosi o schemă XML, ca şi în cazul limbajului *XFiles* prezentat în secţiunea 3.2.5.1. Această schemă poate fi parcursă în anexa C.

Elementele <Meets>, <Before>, <Overlaps>, ... corespund operatorilor temporali $Meets$, $Before$, $Overlaps$, ... prezentaţi grafic în figura 3.2. Fiecare astfel de element va putea include un atribut care specifică o durată de timp (*e.g.*, 1S pentru 1 secundă sau 3H30M pentru 3 ore şi 30 de minute).

Elementul <link> defineşte relaţiile (spaţiale sau temporale) ce pot fi stabilite între două resurse Web r_1 şi r_2. Tipul relaţiei va fi dat prin intermediul atributului type. Dacă relaţia modelată este spaţială, atunci se vor putea folosi extensiile limbajului *XFiles* pentru a specifica construcţii RDF privitoare la proprietăţile, locaţiile şi distanţa acestor resurse. Dacă relaţia este considerată a fi temporală, atunci elementul <link> poate include unul dintre operatorii temporali ITL. După cum am văzut, aceşti operatori sunt specificaţi prin elemente sintactice TRSL.

În conjuncţie cu atributul dur, elementul <link> poate include

atributele `begin` şi `end` pentru specificarea intervalelor. Suportul oferit pentru specificarea acţiunilor poate fi deosebit de important la implementarea unui mecanism de răspuns la evenimente sau la realizarea unui şir de acţiuni în manieră secvenţială, concurentă sau chiar paralelă. Aceasta poate deschide perspective promiţătoare în modelarea, de exemplu, a construcţiilor specifice TLA+ [Lamport, 2003] în vederea definirii comportamentului entităţilor statice ori dinamice ale unui sistem distribuit complex (roboţi, agenţi, servicii Web etc.).

Pentru a oferi o mai mare flexibilitate, s-a definit şi un element `<action>` care poate specifica o colecţie RDF de acţiuni care se intenţionează să fie executate la un moment dat.

O acţiune particulară ar fi aceea de a menţine consistenţa relaţiei temporale avută în vedere. De exemplu, ştiind că două resurse r_1 şi r_2 sunt în relaţia $After$, pentru a menţine această relaţie şi în viitor un agent ar putea realiza o oglindire a conţinutului resursei r_1 la locaţia corespunzătoare resursei r_2, astfel încât fiecare modificare a resursei r_1 să determine şi actualizarea resursei r_2. Desigur, asupra acţiunilor specificate de elementul `<action>` pot fi impuse restricţii suplimentare, prin extinderea limbajului cu alte construcţii utile.

Semantica elementelor TRSL este definită în modul următor: fiecărui element TRSL, exceptând `<link>`, îi corespunde un operator temporal prezentat în secţiunea 3.2.3.

Specificarea formală a elementelor `<Before>` şi `<During>` poate fi exprimată astfel:

$$i \prec j \equiv < \texttt{Before dur} = \texttt{"t"} / > \tag{3.9}$$

$$i \sqsubset j \equiv < \texttt{During dur} = \texttt{"t"} / > \tag{3.10}$$

unde intervalul $t = j.\texttt{start} - i.\texttt{end}$, iar i şi j sunt momente de timp (a se vedea modelul temporal caracterizat în secţiunea 3.2.3.4). Uzual, un astfel de moment desemnează cel mai recent timp de actualizare a conţinutului resursei căreia îi corespunde – exprimat prin construcţia *XFiles* `<Timestamp type="modification">`.

Similar pot fi exprimate semantic şi celelalte elemente TRSL.

3.2.7.2 Exemple

În continuare, vom da două exemple de construcţii RDF modelând diverse relaţii spaţio-temporale cu ajutorul limbajelor *XFiles* şi TRSL. Primul exemplu poate fi regăsit şi în [Buraga & Ciobanu, 2002].

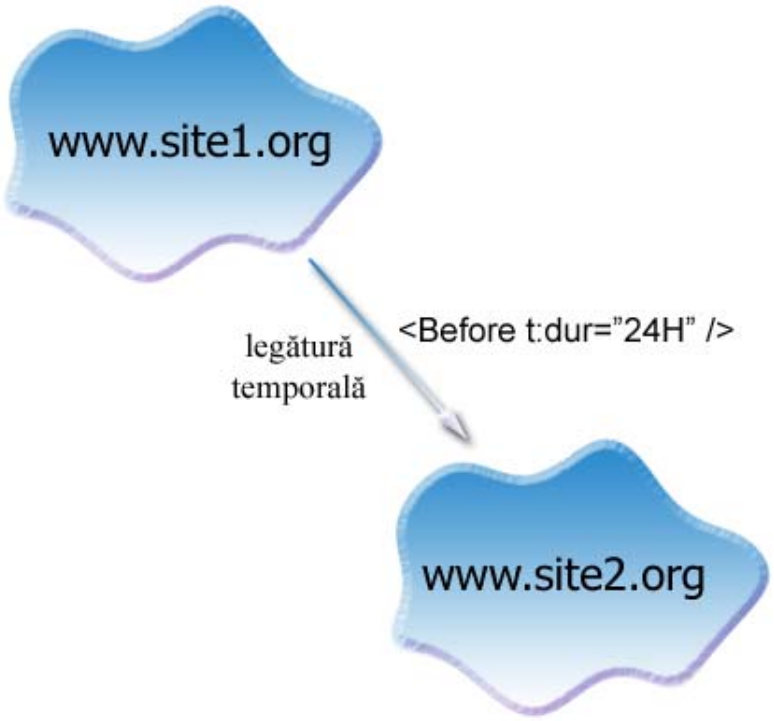

Figura 3.3: Reprezentarea grafică a legăturii temporale dintre două situri Web

Exemplul 23 Considerăm două situri Web s_1 şi s_2, identificate prin adresele lor $url_1 \equiv$ `www.site1.org` şi $url_2 \equiv$ `www.site2.org`, respectiv.

Conţinutul primului sit (văzut ca o colecţie de resurse stocate pe serverul Web identificat prin url_1) este transferat dinamic (oglindit) pe cel de-al doilea sit de către un robot Web la o întârziere cu o zi faţă de actualizarea primului sit. Această operaţie de copiere a conţinutului hipertext începe zilnic la ora 8:00 am.

Scenariul temporal descris poate fi modelat prin intermediul a două aserţiuni RDF recurgându-se la utilizarea construcţiilor TRSL pentru a specifica relaţia $Before$ dintre cele două situri (a se vedea şi figura 3.3). Astfel, $s_1 \prec s_2$ poate fi exprimată în XML în modul următor (omitem declaraţiile spaţiilor de nume RDF şi TRSL):

```
<rdf:RDF>
  <!-- primul operand -->
  <rdf:Description rdf:about="http://www.site1.org">
     <!-- legatura temporala dintre s1 si s2 -->
     <t:link t:type="temporal" t:start="08:00:00.000">
         <!-- al doilea operand -->
         <rdf:Description rdf:about="http://www.site2.org">
            <!-- operatorul temporal -->
            <t:Before t:dur="24H" />
         </rdf:Description>
     </t:link>
  </rdf:Description>
</rdf:RDF>
```

Exemplul 24 Vom presupune că un agent Web descoperă diferite relaţii temporale stabilite între resurse (documente) Web şi generează următorul document RDF cu scopul de a interschimba, cu alţi agenţi, informaţii privitoare la resursele găsite (spaţiul de nume *f* corespunde elementelor şi atributelor limbajului *XFiles* definit în secţiunea 3.2.5.1, iar spaţiul de nume *dc* defineşte construcţiile DCMI referitoare la metadate [Purl]). O parte dintre legăturile stabilite între resursele Web amintite este ilustrată de figura 3.4.

```
<rdf:RDF>
   <!-- o colectie de resurse modificate recent -->
   <rdf:Bag id="RecentlyChanged">
      <rdf:li resource="index.html" />
      <rdf:li resource="figure.gif" />
      <rdf:li resource="styles.xsl" />
   </rdf:Bag>
```

```
<!-- specificarea relatiilor spatio-temporale -->
<rdf:Description rdf:about="#RecentlyChange">
   <!-- o asertiune privind resursele din colectie -->
   <!-- informatii spatiale -->
   <f:Location f:dns="www.site.org">
       193.231.30.1
   </f:Location>

   <!-- informatii despre metadatele asociate -->
   <!-- ...exprimate in XFiles -->
   <f:Owner>
      <!-- posesor -->
      <rdf:Description
         rdf:about="http://www.infoiasi.ro/~busaco/">
         <f:Login f:uid="74">busaco</f:Login>
         <!-- eventual si alte date... -->
      </rdf:Description>
   </f:Owner>
   <!-- ...exprimate in DCMI -->
   <dc:Description>
      O colectie de resurse modificate recent
   </dc:Description>
   <dc:Publisher>
      Site.Org
   </dc:Publisher>

   <!-- informatii temporale -->
   <t:link t:type="temporal" t:action="Update"
           t:end="2003-09-01T14:00">
      <t:During>
        <t:Action t:start="0">
           <!-- specificarea actiunii ce
                va fi indeplinite:
                invocarea imediata a unui agent -->
           <rdf:Description
              rdf:about=
              "http://www.omega.ro/agent.a?ac=Query">
```

```
                    <f:Location f:dns="www.omaga.ro">
                       192.103.33.74
                    </f:Location>
                 </rdf:Description>
              </t:Action>
           </t:During>
        </t:link>
     </rdf:Description>
</rdf:RDF>
```

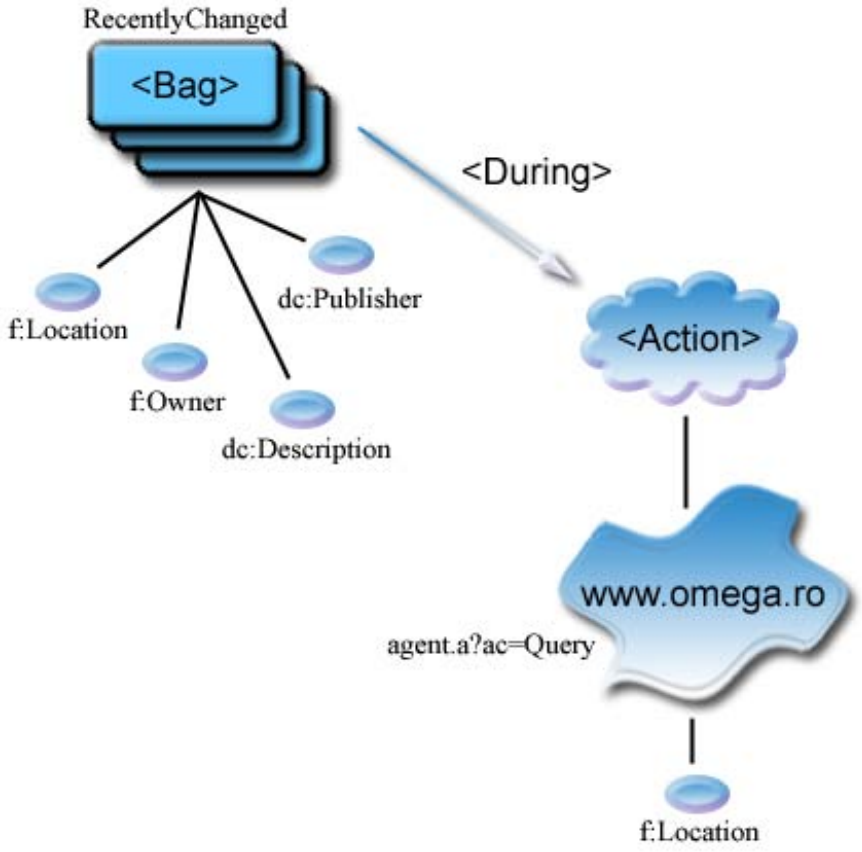

Figura 3.4: Reprezentarea grafică a legăturilor stabilite între resursele Web

Colecţia de resurse identificată de $R \equiv$ `RecentlyChanged` este stocată de maşina `www.site.org` şi are asociate diverse metadate, pre-

cum date despre posesorul resurselor, o descriere în limbaj natural şi organizaţia responsabilă cu publicarea lor pe Web.

Pentru această colecţie este planificată o operaţie de actualizare (specificată de atributul `action="Update"`), care trebuie îndeplinită cel târziu pe data de 1 Septembrie 2003, la ora 14. Notăm această acţiune cu a_1.

Pe cât posibil concomitent cu a_1, se iniţiază automat o alta, notată a_2 şi specificată printr-o aserţiune RDF în cadrul elementului `<Action>`. Observăm că în cazul acţiunii a_2 sunt specificate mai multe informaţii: invocarea unui agent localizat pe maşina `www.omega.ro`, cu trimiterea unui parametru `ac=Query` folosind convenţiile sintactice pentru identificatori uniformi de resurse. [11]

Din punct de vedere formal, s-a exprimat faptul că pentru toate resursele colecţiei date se stabileşte o relaţie implicită $Before$ între timpul ultimei modificări a resurselor şi acţiunea de actualizare a_1 de forma $\forall r \in R, r \prec a_1$.

Între cele două acţiuni a_1 şi a_2 a fost stabilită relaţia temporală $During$, astfel încât avem: $a_1 \sqsubseteq a_2$ conform notaţiei introduse în secţiunea 3.2.3 a capitolului de faţă (deoarece cele două acţiuni vor fi invocate pe maşini diferite, ele vor putea fi executate fizic în paralel).

Remarca 6 După cum se poate remarca din ultimul exemplu, modelul propus poate specifica relaţii temporale care se pot stabili nu doar între resurse desemnate de documente Web, ci şi între entităţi dinamice (procese, fire de execuţie, servicii Web etc.).

3.2.7.3 Suportul pentru Web-ul semantic

Prin trăsăturile pe care le pune la dispoziţie, ca principal avantaj al limbajului TRSL putem considera abilitatea de a specifica, via construcţii bazate pe XML, relaţii temporale între resurse. Spre deosebire de alt limbaj de marcare – TimeML [Pustejovsky *et al.*, 2002] –, care se

[11] O altă abordare ar fi fost aceea de a recurge la definirea acţiunii ca fiind o proprietate specificată în OWL.

concentrează asupra realizării de adnotări privitoare la timp în cadrul documentelor, TRSL ia în consideraţie informaţiile temporale dintre resursele Web şi nu cele stabilite între elementele interne ale unui document.

Pentru descrierea conţinutului temporal al paginilor Web sau pentru specificarea proprietăţilor temporale ale serviciilor Web, s-a propus DAML+Time [Hobbs & Pustejovsky, 2002, DAML+TIME], o ontologie bazată pe modelul ITL, privitoare la conceptele de timp. O serie dintre conceptele exprimate sunt deja modelate în DAML+OIL şi OWL.

Limbajul TRSL poate fi folosit în contextul DAML+Time, deoarece oferă suport atât pentru exprimarea momentelor şi intervalelor, cât şi a relaţiilor dintre intervalele de timp. Relaţia $Before$, prezentă în ontologia DAML+Time, poate fi modelată prin elementul `<Before>`, la fel şi celelalte relaţii. Unităţile de măsură a timpului specificate de DAML+Time în cazul TRSL sunt cele din specificaţia XML Schema.

3.3 Căutarea

3.3.1 Argument

În continuare ne vom concentra asupra modalităţilor de căutare a resurselor multimedia disponibile pe Web. Pentru început, vom prezenta succint structura motoarelor de căutare existente. Una dintre componentele arhitecturale de bază este cea referitoare la mecanismul de formulare a cererilor, lucrarea propunând un limbaj bazat pe XML pentru exprimarea unor interogări care să implice specificarea relaţiilor temporale dintre resurse şi care să ia în calcul structura internă a documentelor adnotate căutate.

Prin intermediul unui limbaj de marcare, prezentat în cadrul secţiunii 3.3.3, vom putea identifica poziţiile şi numărul de apariţii ale unor elemente XML [Buraga & Rusu, 2000, Buraga & Brut, 2001]. În acest sens, lucrarea aducând contribuţii la identificarea în general a documentelor disponibile pe Web şi în special a prezentărilor hipermedia sincronizate, exprimate prin SMIL. Astfel, materialul se înscrie pe linia cercetărilor deschise de [Botafogo *et al.*, 1991].

3.3.2 Motoare de căutare

3.3.2.1 Preliminarii

Spaţiul WWW are ca *raison d'être* oferirea unui mecanism facil de a stoca, într-o manieră bazată pe hipertext, şi de a pune, ulterior, la dispoziţie informaţii, în mod intuitiv, nesecvenţial. Studiile efectuate estimează că între 85% şi 90% dintre utilizatori se bazează pe *motoarele de căutare* pentru a localiza resursele dorite [Buraga, 2001a].

Motoarele de căutare pot oferi servicii de căutare pe bază de indecşi (i.e., Altavista [Altavista] sau Google [Google]) sau pe baza unor ierarhii de termeni – aşa-numitele *servicii director*, în fapt cataloage bazate pe ontologii (*e.g.*, Yahoo! [Yahoo]). În ultimii ani, aceste servicii au devenit hibride.

3.3.2.2 Regăsirea informaţiilor de către utilizatori

În funcţie de intenţiile pe care le au, utilizatorii adoptă diverse stiluri de parcurgere a structurilor hipertext prezente pe Web, dintre care se pot enumera [Buraga, 2001a]:

- *scanarea* (*scanning*) – utilizatorii acoperă, superficial, o arie largă de informaţii, aparţinând de obicei unui anumit subiect sau unui grup de subiecte conexe;
- *răsfoirea* (*browsing*, *surfing*) – utilizatorii vizitează locaţiile care pe captează interesul, fără a avea stabilit un model mental al informaţiilor dorite;
- *căutarea* (*searching*) – utilizatorii sunt motivaţi să găsească o categorie particulară de informaţii-ţintă. De cele mai multe ori este adoptată căutarea bazată pe cuvinte-cheie (*e.g.* “Petri nets”) sau pe construcţii formulate în limbaj natural (de exemplu, “Unde sunt disponibile documentaţii despre design Web?”)[12];

[12] A se consulta şi lucrările [Marchiori, 1997], [Moldovan & Mihalcea, 1999] şi [Mendelzon *et al*, 1997].

- *explorarea* (*exploring*) – utilizatorii investighează legăturile referitoare la o anumită resursă informativă şi pe cele conexe;
- *rătăcirea* (*wandering*) – utilizatorii realizează o navigare complet nestructurată.

Una dintre problemele survenite este cea a ignorării structurii interne a documentelor căutate. Conform [Louka, 1994, Buraga, 2002a], documentele nu sunt stocate în mod structurat pe serverele Web, iar structura hipertext pe care, eventual, o formează poate fi recunoscută de cele mai multe ori doar examinând URI-urile asociate. Mai mult, legăturile dintre documente sunt unidirecţionale şi nu au asociate semantici (foarte puţine documente utilizează mecanismul de exprimare a legăturilor multidirecţionale oferit de *XLink* [DeRose *et al.*, 2001]).

Este dificil de alcătuit o hartă locală care să ilustreze semantic toate sursele şi destinaţiile legăturilor dintre paginile Web ale unui sit[13].

Un alt impediment este cel legat de menţinerea legăturilor. Legăturile fiind stocate în interiorul documentelor, posibilitatea de a adăuga direct adnotări[14] sau de a modifica legăturile dintr-o pagină este deţinută doar de proprietarul acesteia. Menţinerea integrităţii legăturilor pentru situri Web care conţin un număr foarte mare de documente este dificilă, deoarece structura hipermedia a acestuia poate fi deosebit de complexă [Balasubramanian, 1994].

3.3.2.3 Anatomia unui motor de căutare

În continuare, vom descrie pe scurt structura internă a unui motor de căutare (pentru mai multe detalii a se vedea [Brin & Page, 1998], [Buraga, 2001a], [Butler, 2000], [Marchiori, 1997] sau [Wallace, 2000]).

În general, un motor de căutare este constituit din trei componente de bază (a se vedea şi figura 3.5):

[13] Asupra descoperirii de structuri particulare de grafuri pentru fragmente de Web, a se vedea [Chakrabarti *et al.*, 1999], [Butler, 2000] şi [Lowe, 2000], fără a se pune însă problema asocierii de descrieri, procesabile de către maşină, a relaţiilor dintre nodurile hipertext.

[14] A se vedea eforturile Consorţiului Web în această privinţă, concretizate în proiectul *Annotea* [Annotea] bazat pe RDF, XPointer şi alte tehnologii XML.

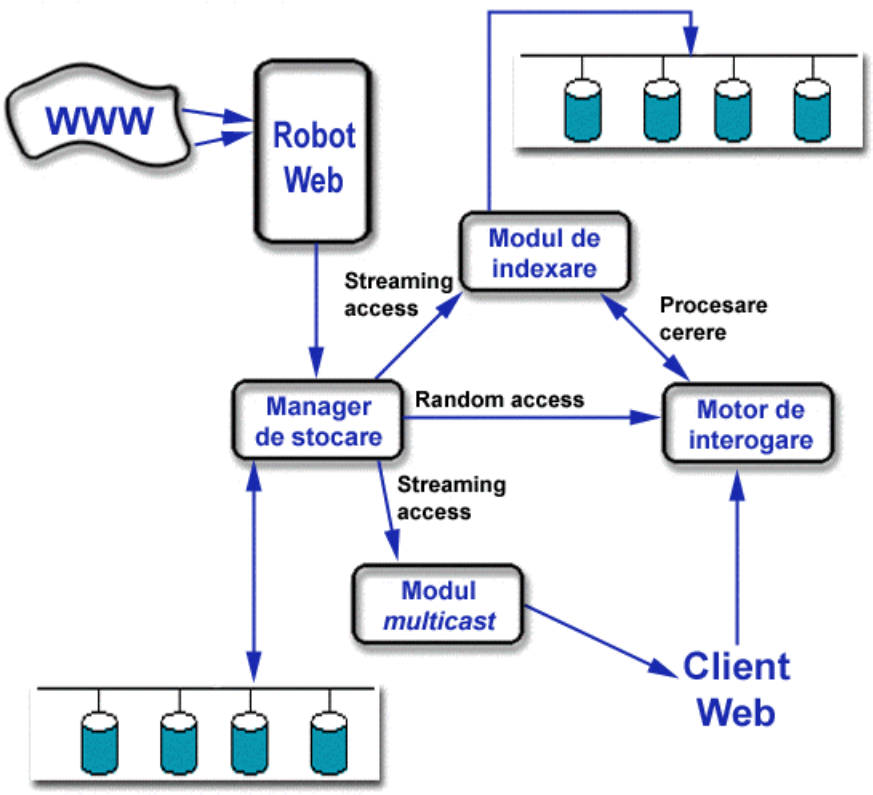

Figura 3.5: Arhitectura internă a unui motor de căutare

1. o aplicaţie, denumită *robot Web* (*spider*, *crawler*), având misiunea de a parcurge spaţiul WWW şi de a vizita anumite pagini (resurse), extrăgând informaţii privitoare la ele. Aceste informaţii vor fi stocate pe serverul/serverele motorului de căutare, într-o bază de date distribuită (numită şi index);

2. un depozit de memorare a informaţiilor despre paginile parcurse de robot, depozit numit *index* (*catalog*). Acesta conţine de cele mai multe ori câte o copie a fiecărei pagini şi a URI-ului corespunzător acesteia, organizarea informaţiilor în cadrul indexului efectuându-se conform unor criterii specifice;

3. un *mecanism de evaluare* (*ranking*) a importanţei paginilor din

index în conformitate cu cererea formulată de utilizator, cerere introdusă prin intermediul unei interfeţe Web (partea vizibilă a motorului de căutare). În ordinea importanţei, adresele paginilor (plus alte metadate) sunt returnate – sub forma unui document Web – utilizatorului care a formulat cererea. Utilizatorul este cel care decide care pagină (sau grup de pagini) întruneşte preferinţele sale.

Putem defini un *robot Web* ca fiind un program care traversează în mod automat structura hipertext a spaţiului WWW, cu scopul extragerii unor informaţii. În esenţă, activitatea unui robot constă în a transmite o cerere HTTP [Fielding *et al.*, 1997] către un server Web – pornind de la un identificator uniform de resurse – şi de a extrage informaţiile dintr-un document HTML şi din toate documentele desemnate de legăturile acestuia. Activitatea de căutare poate viza nu numai documentele hipertext, ci şi alte tipuri de resurse (i.e. multimedia, arhive, aplicaţii etc.) după cum vom vedea mai jos.

Vizitarea tuturor documentelor prezente pe Web nu poate fi realizată practic din cel puţin două motive [Brin & Page, 1998]: în primul rând, indexul motorului de căutare are o capacitate limitată şi deci motorul nu este capabil să indexeze şi să analizeze toate paginile (spaţiul WWW se dezvoltă într-un ritm alert), iar în al doilea rând Web-ul are un caracter dinamic – modificându-se extrem de rapid, atât structural, cât şi din punctul de vedere al conţinutului – şi robotul nu va avea posibilitatea să parcurgă o serie de pagini.

Nu toate paginile vor avea aceeaşi importanţă pentru robotul Web, ţinându-se cont de mai mulţi factori. Se pot lua în calcul:

- compatibilitatea cu posibilele cereri ale utilizatorului;
- numărul legăturilor spre resurse care includ, la rândul lor, legături spre pagina analizată de robot – gradul de “citare” a documentului;
- relevanţa conţinutului paginii;
- numărul de legături conţinute de pagină şi metrica locaţiei (de exemplu, o pagină din domeniul `.com` se consideră a fi mai importantă decât una a domeniului `.za`).

Relevanţa unui document Web depinde, aşadar, de contextul apariţiei acestuia, relaţiile stabilite între resursele căutate având o importanţă majoră [Sheth, Arpinar, Kashyap, 2002], mai ales în contextul Web-ului semantic.

Cea de-a doua componentă a motorului de căutare este *indexul* (*catalogul*), alcătuit dintr-o serie de baze de date disponibile pe un *server de stocare* [Brin & Page, 1998], constituindu-se astfel un depozit distribuit de date în cadrul unui *cluster* sau Grid [Buyya, 2002]. Acest depozit va memora documentele indexate returnate de robotul asociat motorului de căutare.

Modulul de stocare realizează diverse activităţi, cele mai importante fiind:

- inserarea datelor noi privitoare la paginile parcurse de roboţi,
- actualizarea conţinutului vechilor documente stocate,
- programarea diferitelor cereri de accesare a informaţiilor referitoare la o serie de documente.

În vederea stocării, fiecare pagină va avea asociat un identificator stabilit pe baza URI-ului absolut al resursei respective. URI-urile vor fi *normalizate* prin intermediul următorului algoritm, prezentat în lucrarea [Brin & Page, 1998]:

Algoritmul 1 (Normalizarea unui URI)

1. *se elimină prefixul desemnând schema de acces (*e.g. `http://`*), dacă aceasta este prezentă;*
2. *se elimină numărul portului implicit (*`:80`*), dacă există, dar se păstrează orice alt număr de port (*e.g. `8000`*);*
3. *adresa simbolică a serverului este convertită în litere mici;*
4. *caracterele "`/`" de la finalul URI-ului sunt eliminate.*

Exemplul 25 Presupunând că avem la intrare identificatorul uniform de resurse `http://www.InfoIasi.Ro:80/`, în urma algoritmului de normalizare vom obţine URI-ul normalizat `www.infoiasi.ro`.

Modulul de indexare şi metadate este responsabil cu extragerea metadatelor[15] din paginile colectate şi cu indexarea atât a metadatelor, cât şi a conţinutului hipertext al documentelor Web [Brin & Page, 1998, Butler, 2000, Lowe, 2000, Wallace, 2000]. O altă tehnică abordată este cea a *indexării semantice*, pentru aceasta recurgându-se la folosirea ontologiilor.

Extragerea metadatelor variază în funcţie de motorul de căutare. Cea mai populară tehnică este cea a indexării documentelor pe baza *cuvintelor-cheie* furnizate fie explicit de creatorul acestor documente (via construcţii `<meta>` sau aserţiuni RDF/RSS [RSS], în conjuncţie cu calificatori DCMI [Purl]), fie în urma unei catalogări automate realizate de robot. Unele metode apelează la sumarizări automate, la utilizarea de sinonime ale cuvintelor-cheie sau la folosirea de euristici. În activitatea de catalogare a informaţiilor, de multe ori intervine ierarhizarea datelor în funcţie de subiectul tratat, această clasificare conducând la apariţia *serviciilor director*, bazate pe generarea de ontologii (a se vedea [Beckett, 2002] şi [Carr *et al.*, 2001]).

Indecşii pot include indecşi text obişnuiţi, dar şi indecşi ai metadatelor extrase.

Depozitul de documente indexate suportă trei moduri de acces:

- *acces direct* (*random access*) – se realizează pe baza identificatorului unic (cheie primară) asociat fiecărei pagini indexate;
- *acces bazat pe interogări* (*query-based access*) – în urma unei interogări, vor fi furnizate toate documentele satisfăcând o anumită cerere. Cererea poate să se refere la diverse atribute ale metadatelor (*e.g.*, autor, locaţie, titlu) sau la conţinutul (textual) al paginilor;

[15] A se revedea şi cele discutate în secţiunea 3.2 a capitolului de faţă.

- *acces flux de date* (*streaming access*) – este folosit atunci când din depozitul de date se extrage un grup de pagini (resurse) pentru a fi trimis ca flux de date spre o anumită aplicaţie (de exemplu, atunci când se reindexează o serie de documente).

În cadrul depozitului de date, este de dorit să se memoreze cea mai recentă versiune a resurselor găsite de roboţii Web. Trebuie avute în vedere aspecte importante precum *consistenţa indecşilor* şi *eliminarea paginilor vechi/inexistente pe Web*. Roboţii Web pot avea proprii lor parametri pentru realizarea unei actualizări regulate, ţinând cont de diverse raţiuni, cum ar fi cele legate de congestia reţelelor sau de relaţiile temporale stabilite între resurse.

Pentru asigurarea scalabilităţii, depozitul de date poate fi unul distribuit, constituindu-se în acest scop o colecţie de *noduri de stocare* independente, controlul realizându-se prin intermediul unui server de management al nodurilor [Brin & Page, 1998]. Pentru sporirea eficienţei, nodurile pot conţine pagini grupate pe diverse criterii (cuvinte-cheie, tematică, localizare etc.).

A treia componentă a unui motor de căutare este reprezentată de *mecanismul de regăsire şi de evaluare* a paginilor Web furnizate utilizatorului în urma cererii acestuia, formulată prin intermediul unei interfeţe puse la dispoziţie de motor. Interfaţa de căutare (denumită şi *motor de interogare*) oferă diferite posibilităţi de formulare a cererilor prin intermediul diverşilor operatori logici, în limbaj natural, explorând ierarhii de domenii catalogate (directoare Web), stabilind localizarea resurselor etc.[16]

În cazul cererilor formulate în limbaj natural (în speţă, în limba engleză), problemele care trebuie rezolvate sunt cele legate de eliminarea caracterului ambiguu al termenilor, eliminarea cuvintelor nerelevante sau expandarea interogării (de exemplu, pot fi automat formulate noi cereri prin utilizarea reţelei WordNet) [Moldovan & Mihalcea, 1999].

Procesul de evaluare a interogării specificate de utilizator poate fi derulat conform etapelor sintetizate de algoritmul de mai jos:

[16] A se vedea [Buraga, 2001a] şi [Buraga, 2002a].

Algoritmul 2 (Evaluarea unei interogări)

1. *analizarea cererii formulate de utilizator;*
2. *căutarea termenilor rămaşi după analiza cererii în indecşii corespunzători ;*
3. *scanarea tuturor documentelor stocate de motor care întrunesc întregul set de condiţii de căutare;*
4. *evaluarea gradului de relevanţă a fiecărei pagini în funcţie de cererea formulată;*
5. *eliminarea duplicatelor şi sortarea în ordinea relevanţei;*
6. *afişarea adreselor (identificatorilor uniformi de resurse) ale celor mai relevante N documente, eventual furnizându-se şi alte informaţii precum lungimea documentului, formatul, limba, contextul apariţiei etc.*

Sistemul de evaluare depinde de motorul ales. În cazul motorului de căutare Google [Google], sunt menţinute mai multe metadate referitoare la documentele indexate [Brin & Page, 1998], faţă de alte motoare, ceea ce conduce la o precizie mai bună.

3.3.2.4 Meta-căutătoare şi portaluri

Pentru a formula interogări şi a primi rezultate de la o multitudine de motoare de căutare, utilizatorii au posibilitatea de a apela la serviciile oferite de un *meta-căutător*. Funcţia principală a acestuia este cea de a compila listele de pagini obţinute de la motoarele obişnuite (interogate în manieră paralelă) şi de a prezenta utilizatorului cele mai relevante documente găsite. Deseori, un meta-căutător – precum Kartoo [Kartoo] – are implementat propriul sistem de evaluare a relevanţei paginilor, în funcţie de anumite criterii proprii sau formulate de utilizator.

De asemenea, un meta-căutător poate oferi o vedere ierarhizată a resurselor găsite, similare structurii de directoare Web (ierarhii de termeni) pe care acestea le formează. O parte dintre serviciile de căutare specializată (*e.g.* căutarea de conţinut multimedia) se regăsesc şi în cadrul motoarelor clasice.

Pe Web, pot fi căutate informaţii specifice unui domeniu de interes: afaceri, comunităţi umane, divertisment etc. Aplicaţiile specializate în acest sens poartă numele de *portaluri Web*, punând în plus la dispoziţie şi alte servicii, precum accesarea unui cont de poştă electronică, starea meteo, înscrierea la liste/grupuri de discuţie şi altele. Un portal poate fi privit, aşadar, drept integrator de conţinut, fără a oferi însă o colecţie nestructurată de legături spre alte situri.

Pentru mai multe detalii, se pot consulta lucrările [Buraga, 2001a], [Buraga, 2002a] sau [Wallace, 2000].

3.3.3 Exprimarea interogărilor prin WQFL

3.3.3.1 Preliminarii

Dacă în cadrul secţiunilor anterioare ne-am concentrat mai ales asupra modelării, într-un dialect XML, a relaţiilor care se stabilesc între resursele multimedia disponibile pe Web, în continuare ne vom referi la posibilitatea de a formula interogări flexibile pentru căutarea acestor resurse. În acest sens, vom defini un limbaj bazat pe XML, scopul principal fiind acela de a putea fi folosit la exprimarea unor interogări complexe şi flexibile.

În faza de proiectare a limbajului *WQFL* (*Web Query Formulating Language*) [Buraga & Rusu, 2000, Buraga & Brut, 2001] s-au observat următoarele:

- interogările complexe recurg la folosirea conectorilor logici (precum operatorii *and*, *or* sau *not* puşi la dispoziţie de componenta de interfaţă a fiecărui motor de căutare actual – a se vedea secţiunea 3.3.2.3);
- utilizatorii preferă să obţină documente Web prezentând diverse structuri şi tipuri de conţinut (*e.g.*, fără tabele, având mai puţin de trei imagini plasate în partea inferioară a paginii găsite etc.);

- activitatea de căutare poate fi mult îmbunătăţită dacă se adoptă diverse tehnici specifice domeniului inteligenţei artificiale.

Exemplul 26 În activitatea de căutare a unor informaţii, utilizatorii ar putea formula o interogare având următoarea formă:

```
"temporal Petri nets" + with <7 paragraphs on top
   + with <3 images on bottom + without links
   + without tables + without multimedia
```

Această interogare compusă specifică faptul că utilizatorul doreşte să se realizeze o căutare bazată pe cuvinte-cheie, documentele returnate trebuind să conţină maxim 7 paragrafe la început, maxim 3 imagini la final, iar conţinutul nu trebuie să includă alte legături spre alte documente, nici tabele şi nici alte informaţii multimedia (sunet, animaţii, video). Operatorul *and* a fost notat prin simbolul "+", iar ghilimelele încadrează expresia textuală căutată.

Interogarea ar putea include şi informaţii privitoare la timp, ca de exemplu să specifice faptul ca toate resursele găsite să fie într-o relaţie $After$ cu o alta:

```
"temporal Petri nets" + with <7 paragraphs on top
   + with <3 images on bottom + without links
   + without tables + without multimedia
   + modified after 5 hours from site: www.infoiasi.ro
```

În exemplul de mai sus, s-a specificat suplimentar ca documentele găsite să aibă conţinutul modificat cu 5 ore după actualizarea sitului `www.infoiasi.ro`.

Limbajul WQFL va putea fi folosit atât pentru a modela interogările provenite din partea utilizatorilor, cât şi pentru a desemna unele informaţii privind structura, conţinutul şi relaţiile documentelor găsite de motorul de căutare.

3.3.3.2 Activitatea de căutare

Activitatea de căutare propusă constă din următoarele faze principale (conform [Buraga & Rusu, 2000] şi [Buraga & Brut, 2002]):

1. Formularea unei interogări de către utilizator, prin intermediul unei interfeţe, şi generarea unui document WQFL care modelează în termenii XML acea interogare.

2. Cuvintele-cheie ale expresiei de interogare (*e.g.*, "temporal Petri nets") sunt trimise unui serviciu de căutare – un motor de căutare tradiţional, un serviciu Web special sau un sistem de agenţi de căutare (a se vedea capitolul 4) – pentru a realiza efectiv operaţiunea de căutare a resurselor dorite. Serviciul de căutare va returna primele N adrese ale documentelor Web semnificative. Aceste adrese vor fi folosite pentru a salva local conţinutul resurselor hipertext în vederea procesării ulterioare. Expresiile adiţionale ale interogării (*e.g.*, `with <3 images on bottom`) vor fi folosite la pasul următor.

3. Scopul acestei etape este de a codifica informaţiile privitoare la structura conţinutului resurselor Web găsite. Aceste informaţii vor fi stocate de documente WQFL. Fiecare document Web va fi procesat într-o asemenea manieră încât să fie reţinute anumite informaţii privitoare la locul şi numărul de apariţii ale unor elemente/atribute XML (în particular, XHTML) şi la relaţiile spaţio-temporale între documentele regăsite.

   Procesul de generare a documentelor WQFL poate fi văzut ca o funcţie $g : Pages \rightarrow \mathsf{WQFL}$, unde $Pages$ reprezintă mulţimea resurselor (paginilor) Web găsite şi WQFL este mulţimea documentelor WQFL create.

   Exemplul 27 În cazul XHTML, pentru a se permite căutare unor documente Web cu o anumită structură, se va putea considera o submulţime $\mathcal{S}$ a mulţimii $\mathcal{H}$ a tuturor elementelor limbajului XHTML [Pemberton *et al.*, 2002] permise.

   Conform [Gogan & Buraga, 2000, Buraga & Brut, 2001], în cadrul mulţimii $\mathcal{S}$, vor putea fi incluse elementele:

- <p> (paragraf);
- <img> (imagini statice în formate precum JPEG, GIF sau PNG);
- <object> (obiect multimedia – sunet, animaţie, video, lume virtuală 3D – sau obiect generic – *e.g.*, componentă ActiveX, applet Java, prezentare Flash etc.);
- <table> (date tabelare);
- <a> (ancoră);
- <script> (cod într-un limbaj de tip script interpretat de navigatorul client, precum JavaScript sau VBScript);
- <meta> (metadate asociate documentului).

După cum se poate remarca, se au în vedere mai ales constituenţi multimedia – statici sau dinamici – care pot apărea în cadrul unei pagini Web marcate în XHTML.

Remarca 7 În funcţie de limbajul folosit în marcarea datelor, mulţimea $\mathcal{S} \subset \mathcal{H}$ poate îngloba alte elemente/atribute specifice. O direţie care poate prezenta interes este aceea de a considera construcţii sintactice ale limbajului SMIL [Ayars *et al.*, 2001].

Pentru fiecare element $e \in \mathcal{S}$, se vor reţine – în diferite construcţii WQFL – apariţiile fiecăruia în cadrul documentului Web. Conform celor discutate în lucrările [Buraga & Rusu, 2000] şi [Buraga & Rusu, 2001], contextul apariţiei se va raporta la poziţiile de început şi de final ale unui document Web.

Aşadar, pentru fiecare din cele N documente găsite la faza anterioară, se va genera un document WQFL.

4. În cadrul acestei etape, fiecare document WQFL generat, $w \in$ WQFL, este comparat cu documentul WQFL corespunzător interogării formulate de utilizator. Cea mai bună potrivire găsită va

desemna documentul cu informaţia structurală cât mai apropiată de cerinţele iniţiale ale utilizatorului în ceea ce priveşte structura şi tipul de conţinut al resurselor dorite.[17]

Tot în cadrul acestui pas se va verifica dacă este satisfăcută relaţia temporală dintre fiecare resursă şi cea specificată în interogarea iniţială.

3.3.3.3 Limbajul WQFL

Pentru validarea documentelor WQFL, este specificată o schemă XML prezentată în anexa D.

Fiecare document WQFL va avea `<webquery>` ca element rădăcină. Acest element va include elementele:

- `<engine>` – desemnează motorul de căutare folosit, eventual parametrii opţionali ce trebuie transmişi pentru realizarea interogării, folosindu-se atributul `params`;
- `<query>` – specifică interogarea textuală aşa cum este ea formulată de utilizator (poate include şi operatorii uzuali acceptaţi de motoarele de căutare);
- `<structure>` – defineşte structura conţinutului documentelor ce se doresc a fi căutate: elementele XML, numărul şi contextul apariţiilor acestora, plus eventual alte informaţii folositoare; elementele vor fi specificate prin `<element>`, gradul de importanţă al apariţiei acestora putând fi definit de valoarea atributului `order` (cu cât valoarea numerică a atributului `order` este mai mare, cu atât importanţa apariţiei elementelor specificate este mai mică);
- `<content>` – desemnează tipul de conţinut dorit a fi returnat (*e.g.*, limba sau tipul MIME) şi poate include atât conţinutul pro-

[17] Una dintre tehnicile care pot fi folosite în acest sens este cea recurgând la reţele neuronale cu auto-organizare bazate pe învăţare competitivă (a se vedea [Gogan & Buraga, 2000]), direcţie continuată de [Gaura & Newman, 2003].

priu-zis al documentelor returnate de motor – folosindu-se elementul <page> –, cât şi alte informaţii privitoare la metadate şi relaţii cu alte documente (eventual, putându-se utiliza construcţii *XFiles* sau TRSL; a se vedea secţiunile 3.2.5.1 şi 3.2.7.1 ale capitolului de faţă) – prin intermediul elementului <meta>.

Exemplul 28 Considerăm următoarea interogare complexă, formulată de utilizator [Buraga & Brut, 2001]:

```
"temporal Petri nets"
   + with <7 paragraphs on top
   + without links
   + without table
```

În urma procesării, interogării date îi va corespunde documentul WQFL următor (construcţiile sintactice WQFL vor aparţine spaţiului de nume *wq* – a se vedea şi anexa D):

```
<!DOCTYPE webquery PUBLIC "-//WQFL 1.0//EN">
<?xml version="1.0"
      xmlns:wq="http://www.infoiasi.ro/~busaco/wqfl"?>
<wq:webquery>
   <!-- motoarele de cautare interogate in paralel -->
   <wq:engine wq:url="http://www.google.com/">
        Google
   </wq:engine>
   <wq:engine wq:url="http://www.altavista.com/">
        Altavista
   </qw:engine>

   <!-- interogarea textuala (cuvinte-cheie) -->
   <wq:query>temporal Petri nets</wq:query>

   <!-- structura continutului resurselor cautate -->
   <wq:structure>
     <!-- maxim 7 paragrafe la inceputul paginii -->
     <wq:element wq:name="p" wq:order="0">
       <wq:occur wq:top="7" />
```

```
    </wq:element>
    <!-- nu se permit elemente ancora (legaturi) -->
    <wq:element wq:name="a" wq:appear="no"
                wq:order="1">
      <wq:occur wq:top="0" wq:bottom="0" />
    </wq:element>
    <!-- nu se permit elemente 'table' (tabele) -->
    <wq:element wq:name="table" wq:appear="no"
                wq:order="1">
      <wq:occur wq:top="0" wq:bottom="0" />
    </wq:element>
  </wq:structure>
</wq:webquery>
```

După ce interogarea textuală este trimisă spre prelucrare motoarelor de căutare specificate, se va returna un număr de pagini găsite (considerate cele mai relevante conform cuvintelor-cheie date). Pentru fiecare document al cărui identificator uniform de resurse a fost returnat de motorul de căutare, se va genera un fişier WQFL care va modela structura conţinutului acelui document.

De exemplu, am putea avea următorul document WQFL (omitem declaraţiile spaţiilor de nume folosite, corespunzătoare construcţiilor sintactice RDF, *XFiles*, TRSL şi WQFL):

```
<!DOCTYPE webquery PUBLIC "-//WQFL 1.0//EN">
<?xml version="1.0"?>
<wq:webquery wq:timestamp="2003-09-01T14:33">
  <!-- motorul care a returnat URI-ul resursei -->
  <wq:engine wq:url="http://www.google.com/">
       Google
  </wq:engine>
  <!-- interogarea textuala (cuvinte-cheie) -->
  <wq:query>temporal Petri nets</wq:query>
  <!-- structura continutului resurselor cautate -->
  <wq:structure>
    <!-- s-au gasit 5 paragrafe la inceputul paginii -->
    <wq:element wq:name="p">
      <wq:occur wq:top="5" />
```

```
    </wq:element>
    <!-- nu s-au gasit elemente ancora (legaturi) -->
    <wq:element wq:name="a">
      <wq:occur wq:top="0" wq:middle="0" wq:bottom="0" />
    </wq:element>
    <!-- nu s-au gasit elemente 'table' (tabele) -->
    <wq:element wq:name="table">
      <wq:occur wq:top="0" wq:middle="0" wq:bottom="0" />
    </wq:element>
  </wq:structure>
  <wq:content>
     <!-- asertiuni RDF privitoare la resursa gasita -->
     <rdf:Description rdf:about=
        "http://www.infoiasi.ro/~jucan/index.html">
        <!-- metadate specificate in XFiles -->
        <xf:Type xf:mime="text/html">
           ordinary
        </xf:Type>
        <!-- alte constructii... -->
        <!-- relatia temporala cu www.infoiasi.ro -->
        <t:link t:type="temporal">
           <rdf:Description
              rdf:about="http://www.infoiasi.ro">
              <!-- operatorul temporal -->
              <t:After t:dur="48H" />
           </rdf:Description>
        </t:link>
     </rdf:Description>
   </wq:content>
</wq:webquery>
```

Resursa găsită – localizată la adresa desemnată de identificatorul uniform de resurse `http://www.infoiasi.ro/~jucan/index.html` – conţine 5 paragrafe, nu include ancore sau tabele, are tipul MIME `text/html` (document HTML) şi este în relaţia temporală $After$ (a se vedea secţiunea 3.2.3) cu situl `www.infoiasi.ro`. Conţinutul propriu-zis al documentului găsit a fost omis în exemplul de faţă.

3.3.3.4 Extinderea limbajului WQFL pentru a oferi suport pentru expresii regulate

Vom extinde WQFL pentru a permite folosirea de expresii regulate în stilul Perl în cadrul interogărilor formulate de utilizatori.

O *expresie regulată Perl* reprezintă un şablon (*pattern*) căruia, pe baza unor reguli precise, i se poate asocia (*matching*) un text.[18]

Pentru a se permite utilizarea expresiilor regulate Perl în cadrul interogărilor, vom adăuga elementului `<query>` atributul `regexp`. Acest atribut va indica dacă o anumită interogare conţine expresii regulate.

Înainte de a trimite motoarelor de căutare şirul de cuvinte-cheie, în unele cazuri interogarea va trebui pre-procesată local. După returnarea rezultatelor, se va putea efectua o post-procesare care va consta în analiza textuală a conţinutului fiecărei pagini Web găsite pentru verificarea potrivirii cu expresia regulată specificată de utilizator.

Exemplul 29 Pentru a găsi toate documentele conţinând termenul "Petri" urmat la oricare distanţă de una sau mai multe apariţii ale termenilor "nets" ori "network", vom putea utiliza următoarea expresie regulată (această expresie poate fi parţial pre-procesată imediat ce a fost specificată de utilizator prin intermediul interfeţei Web):

```
<webquery>
   ...
   <query regexp="yes">
      /(Petri.*|(net|(s|work)))+/i
   </query>
   ...
</webquery>
```

Meta-caracterul "`.`" substituie oricare alt caracter, "`|`" specifică o alternativă, "`*`" reprezintă operatorul Kleene (desemnând zero, una sau mai multe apariţii ale unei construcţii), operatorul "`+`" specifică măcar o apariţie a unui element, iar "`/i`" realizează o potrivire independentă

[18] A se consulta şi lucrările [Wall *et al.*, 2000], [Trăuşan-Matu, 2001] sau [Buraga *et al.*, 2002a].

de modul de scriere – cu minuscule sau majuscule – ale caracterelor literale.

3.3.3.5 WQFL ca limbaj de interogare XML

În prezent, există mai multe propuneri de tehnici pentru localizarea, indexarea, interogarea şi regăsirea conţinutului marcat în limbaje bazate pe XML[19].

Una dintre direcţiile timpurii de cercetare a fost aceea de a combina căutarea bazată de cuvinte-cheie cu suportul pentru interogare oferit de sistemele de gestiune a bazelor de date relaţionale. Putem menţiona limbajele *Web3QL* [Konopnicki & Shmueli, 1995] şi *WebSQL* [Mendelzon *et al*, 1997] – inspirate de standardul *SQL* (*Structured Query Language*) – ori *WebLog* [Lakshmanan, Sadri & Subramanian, 1996] – variantă orientată spre spaţiul WWW a limbajului DataLog.

Drept limbaje de interogare a conţinutului XML se pot enumera *XQuery* [DeRose, 1998], *XML-QL* [Deutsch *et al.*, 1998] sau *XML-GLrec* [Oliboni & Tanca, 2000] (extensie a *XML-GL* [Ceri *et al.*, 1999])[20]. Aceste limbaje oferă posibilitatea de a folosi o sintaxă XML pentru formularea de interogări care conduc la localizarea de informaţie în cadrul documentelor cu scopul extragerii (semi-)automate a datelor dorite.

Actualmente, pentru a specifica exclusiv printr-o sintaxă XML interogările asupra documentelor XML, a fost propus limbajul în curs de standardizare *XQueryX* [Malhotra *et al.*, 2003].

Prin caracteristicile sale, WQFL poate fi considerat un limbaj simplu de interogare a conţinutului marcat în XML, oferind suplimentar posibilitatea de a indica numărul de apariţii a fiecărui element XML căutat. De asemenea, în cadrul elementului `<structure>` pot fi incluse construcţii sintactice provenite din limbajele de interogare amintite mai sus, în vederea modelării unor interogări complexe.

[19] Premisele unui astfel de demers se pot parcurge în lucrările [Ceri *et al.*, 1999] sau [Florescu, Levy & Mendelzon, 1998].

[20] Pentru detalii, a se consulta [Buraga & Brut, 2001, Buraga & Brut, 2002].

3.4 Concluzii

Capitolul de faţă a prezentat principalele noastre contribuţii referitoare la modul de specificare prin intermediul meta-limbajului XML a metadatelor şi a relaţiilor spaţio-temporale ce pot fi asociate resurselor disponibile pe Web.

În cadrul subcapitolului 3.2 s-a descris maniera de modelare a resurselor dintre resursele Web prin intermediul a două limbaje – *XFiles* şi *TRSL* (*Temporal Relation Specification Language*). Primul limbaj, detaliat în secţiunea 3.2.5.1 şi propus în premieră în [Buraga, 2002b], poate fi considerat drept o contribuţie originală la nivelul de metadate al Web-ului semantic, asociind diverse metadate documentelor Web, văzute drept componente ale unui sistem de fişiere distribuit. Al doilea limbaj pune la dispoziţie suportul pentru descrierea semantică a relaţiilor temporale stabilite între (fragmente de) situri, din prisma timpului. Construcţiile sintactice TRSL, prezentate în secţiunea 3.2.7.1 şi [Buraga & Ciobanu, 2002], se bazează pe formalismul ITL. De asemenea, limbajul TRSL oferă suport pentru specificarea parţială a ontologiei DAML+Time, putând fi astfel considerat ca o contribuţie integrată în nivelul ontologic al Web-ului semantic.

A doua parte a capitolului s-a concentrat asupra definirii unui limbaj capabil să exprime interogări care să conţină informaţii referitoare la structura documentelor căutate şi, posibil, la relaţiile stabilite între aceste documente şi alte resurse. Acest limbaj, descris în secţiunea 3.3.3, oferă suport pentru expresii regulate Perl, putând include date suplimentare privitoare la resursele dorite a fi regăsite. Limbajul propus poate fi considerat, de asemenea, ca fiind un limbaj simplu de interogare a conţinutului XML. Variantele preliminare ale limbajului sunt disponibile în [Buraga & Rusu, 2000], [Buraga & Brut, 2001] şi [Buraga & Brut, 2002].

Cele trei limbaje detaliate mai sus vor fi folosite de diverse componente ale platformei de regăsire a resurselor multimedia, a cărei propunere de implementare va fi descrisă în cadrul următorului capitol.

Capitolul 4

Soluţii de implementare

Subiectul central al capitolului de faţă este platforma distribuită multi-limbaj *ITW* destinată descoperirii de resurse (multimedia) disponibile pe Web, folosind diverse componente precum agenţi şi servicii Web.

4.1 Preliminarii

CAPITOLUL de faţă prezintă o soluţie originală de implementare a unui sistem distribuit de regăsire a resurselor multimedia, sistem compus din entităţi programabile eterogene, precum agenţi software, servicii Web bazate pe XML şi alte programe (*e.g.*, scripturi CGI). De asemenea, se va discuta o posibilă integrare a sistemului în contextul Grid-ului, ca viitoare componentă a Web-ului semantic, şi se vor enumera două domenii de utilizare vizând aplicaţii de tip *e-learning* şi *e-enterprise*.

Capitolul este divizat în trei părţi, primele două făcând o trecere în revistă a subiectelor referitoare la agenţi şi servicii Web.

Ultima parte prezintă în detaliu sistemul *ITW* propus, aflat în stadiu de prototip. Agenţii *ITW* care compun sistemul vor comunica prin

intermediul unor mesaje XML, oferindu-se un model flexibil de serializare a datelor. Serviciile Web *ITW* vor rula pe două platforme software (Linux şi Windows), putând fi implementate în mai multe limbaje de programare – prototipul foloseşte limbajele Perl şi C#. De altfel, printre scopurile iniţiale ale proiectului s-au enumerat cele legate de asigurarea scalabilităţii, extensibilităţii şi independenţei de platformă. După cum se va putea desprinde din cele discutate în secţiunea 4.4, toate aceste scopuri pot fi considerate ca fiind îndeplinite.

O serie dintre rezultatele cercetărilor întreprinse s-a concretizat în lucrări precum [Buraga, 2001c], [Buraga, 2003b], [Buraga, 2003c], [Buraga, 2003g], [Buraga & Alboaie, 2004], [Buraga & Cioca, 2003], [Buraga & Găbureanu, 2003], [Alboaie & Buraga, 2003a] şi [Alboaie & Buraga, 2003b].

4.2 Agenţi software

În continuare, vom realiza o trecere în revistă a problematicii agenţilor software, axându-ne pe modelarea formală a agenţilor inteligenţi integraţi într-un sistem multi-agent. Ulterior, în cadrul secţiunii 4.4, vom prezenta o infrastructură de creare a agenţilor ca parte componentă a sistemului *ITW* dezvoltat.

4.2.1 Definire

În multitudinea de încercări de a oferi o definiţie cât mai completă a termenului de agent, se detaşează două abordări distincte, dar înrudite, una bazată pe noţiunea de agent privit ca o atribuire de identitate comportamentală unei componente software, cealaltă definind agentul ca sumă de descrieri ale unor atribute pe care acesta le posedă.

4.2.1.1 Agenţii ca entităţi comportamentale

Plecând de la premisa că agenţii trebuie să manifeste un comportament inteligent, aceştia trebuie să “cunoască” informaţii specifice privi-

toare la contextul situaţiei cu care sunt confruntaţi. Din prisma acestei premise, se poate da următoarea definiţie [Jennings *et al.*, 1998]:

Definiţia 14 Agenţii software *reprezintă sisteme informaţionale care se comportă asemenea unor alte entităţi într-o manieră autonomă, execută diverse acţiuni, prezentând un anumit nivel de reacţie, şi etalează atribute precum învăţare, cooperare şi mobilitate, putând asista utilizatorii în activităţile pe care aceştia le întreprind.*

Agenţii pot fi priviţi ca fiind sisteme intenţionale. Intenţiile agenţilor pot fi considerate drept o combinaţie între *decizii* şi *scopuri*, fundamentându-se o teorie a interacţiunilor dintre agenţi. O altă abordare formală a noţiunilor mentale asociate agenţilor este cea fundamentată pe logici multi-modale, în care primitivele considerate sunt *convingerile*, *dorinţele* şi *intenţiile* – a se vedea secţiunea 4.2.2.2 a capitolului de faţă.

4.2.1.2 Atributele agenţilor

O altă definiţie a noţiunii de agent este cea de mai jos [Bradshow, 1997]:

Definiţia 15 Agenţii *sunt entităţi software posedând funcţii comportamentale, rulând într-o manieră autonomă şi continuă în medii colective, compuse din alţi agenţi şi procese.*

Cerinţa ca agenţii să asigure autonomia şi continuitatea este dată de dorinţa programatorilor ca agenţii proiectaţi să fie capabili să acţioneze într-un mod flexibil şi inteligent, adaptându-se situaţiilor fără aportul utilizatorului uman. În mod ideal, un agent ar trebui să poată învăţa din propria lui experienţă şi să poată dezvolta tehnici de comunicare şi de cooperare cu alţi agenţi similari sau să poată manifesta mobilitate.

Conform [Wooldridge & Ciancarini, 2000], se poate da următoarea definiţie:

Definiţia 16 *Un* agent *reprezintă un sistem software (sau hardware) care posedă următoarele caracteristici (atribute):*

- autonomie – *comportament direcţionat spre un scop specific, independent de utilizator; agentul operează fără intervenţia directă a unui utilizator uman ori a altor entităţi (*e.g.*, procese, aplicaţii, sistem de operare etc.), având o stare internă şi un anumit grad de control asupra acţiunilor sale;*
- adaptabilitate – *posibilitatea unui agent de a învăţa şi de a se dezvolta, ţinând cont de experienţa acumulată şi versatilitate în rezolvarea unor situaţii inedite;*
- colaborare – *posibilitatea ca un agent să fie capabil să interacţioneze cu alţi agenţi (şi posibil cu utilizatori umani) prin intermediul unui limbaj de comunicare orientat-agent* (agent communication language) *în vederea îndeplinirii unui scop comun;*
- reactivitate – *capacitatea unui agent de a-şi percepe mediul de execuţie – care poate fi, după caz, un utilizator exploatând o interfaţă grafică* (graphical user interface)*, o colecţie de alţi agenţi compunând un sistem multi-agent, un sistem distribuit la nivel global (i.e. spaţiul WWW sau un Grid) etc. – şi de a reacţiona la schimbările din cadrul acestui mediu;*
- mobilitate – *abilitatea agentului de a migra de la sine de pe o platformă (gazdă –* host*) pe alta.*

4.2.1.3 Caracterizări ale agenţilor

Din perspectiva inteligenţei artificiale, agenţii pot fi caracterizaţi şi clasificaţi pe baza măsurării capacităţii de rezolvare a problemelor (de îndeplinire a scopurilor) propuse. Astfel, un agent *reactiv* este acea entitate care manifestă reacţii la modificări ale mediului sau ale mesajelor recepţionate de la alţi agenţi. Un agent *intenţional* este capabil să-şi elaboreze comportamentul în funcţie de “intenţii” şi de “convingeri”, să

creeze planuri de acţiuni şi să le execute. Un agent *social* urmăreşte diverse modele inspirate din societatea umană[1] şi interacţionează cu alţi agenţi, în cadrul aşa-numitei agenţii.

Conform [Bradshow, 1997], agenţii inteligenţi pot fi descrişi în termenii unui spaţiu tridimensional definit prin dimensiunile de *agenţie*, *inteligenţă* şi *mobilitate*:

- *agenţia*

  Reprezintă gradul de autonomie şi autoritate de care se bucură un agent şi poate fi măsurată calitativ de natura interacţiunilor dintre agenţi şi alte entităţi prezente în sistem.

  Un agent poate rula în mod asincron, sistemul trebuind să pună la dispoziţie mecanisme specifice de comunicare asincronă, precum servicii de rutare (dirijare) a mesajelor. În cazul comunicării sincrone, de cele mai multe ori se adoptă paradigma invocării de la distanţă a metodelor (*RMI – Remote Method Invocation*). Agenţii trebuie să expună o serie de *interfeţe* de apelare (invocare).

  Gradul de complexitate a agenţiei creşte dacă agentul îl reprezintă tot mai fidel pe utilizator (agentul devenind astfel un avatar al utilizatorului real). Un agent mai avansat poate interacţiona cu datele, cu aplicaţiile, cu serviciile oferite de platforma gazdă ori cu alţi agenţi. Din acest punct de vedere, agenţia poate fi considerată ca fiind areal de operare a agenţilor (mediu de execuţie – *agent framework*), gazdă utilizată pentru managementul activităţilor agenţilor [Martin, Cheyer, Moran, 1999].

  Mai multe agenţii pot conduce la realizarea unui mediu global orientat-agent bazat pe standarde deschise precum *FIPA* (*The Foundation of Intelligent Physical Agents*) [FIPA]. Ca exemple de soluţii de implementare menţionăm *Agentcities* [Agentcities] şi *FIPA-OS* (*FIPA Open Source*) [Poslad, Buckle, Hadingdam, 2000].

- *inteligenţa* reprezintă gradul de raţionare şi auto-învăţare, adică abilitatea de a accepta şi de a înţelege scopurile utilizatorilor şi de

[1] A se vedea, de exemplu, [Cabri, 2001] şi [Capretz & Osano, 2002].

a le îndeplini, indepedent de aceştia. Cerinţa minimă este aceea de a da posibilitatea agentului să opereze cu un set de preferinţe ale utilizatorului. Nivelurile superioare de inteligenţă includ modele de *raţionament* (vezi şi secţiunea 4.2.2.2), posibilităţi de *învăţare* şi *adaptare* la mediu etc.

- *mobilitatea* furnizează gradul de migrare a agenţilor în reţea. *Scripturile mobile* pot fi concepute pe o maşină şi trimise altei gazde spre interpretare şi execuţie. *Obiectele mobile* sunt transportate din gazdă în gazdă în timpul execuţiei, acumulând diverse informaţii privitoare la mediul în care operează şi pe care pot să-l modifice.

  Suportul pentru migrare[2] este bazat de cele mai multe ori pe un protocol de transport standard, ca HTTP [Fielding *et al.*, 1997], dar pot fi utilizate şi alte tehnologii, mai ales în medii fără fir, cu topologii dinamice.

  Vehicularea datelor despre un agent (cod, stare, metadate etc.) se realizează prin intermediul *serializării*. Actualmente, mecanismul de serializare nu este standardardizat, în cadrul lucrării de faţă propunându-se o metodă originală de standardizare bazată pe XML – a se vedea secţiunea 4.4.2.1 a capitolului de faţă. Acţiunea efectivă de realizare a mobilităţii are loc prin intermediul unui *serviciu de migrare* (*migration service*) [Tripathi *et al.*, 1999].

  Una dintre problemele importante care pot apărea în acest context o constituie *securitatea*. Printre principalele teme de interes se pot enumera autentificarea entităţilor participante la procesul de migrare şi criptarea transferului efectiv de date între platforme. De asemenea, trebuie avute în vedere protecţia nodurilor de reţea de acţiunea agenţilor mobili distructivi şi protecţia agenţilor de gazdele pe care aceştia rulează (conform [Fong, 1998], [Sander & Tschudin, 1998] şi [Patrick, 2002]).

  O altă problemă este cea a *descoperirii* gazdelor, sistemul implementând deseori un *serviciu de descoperire*. Pentru descoperirea

[2] A se consulta şi lucrările [Hohlfeld & Yee, 1998], [Milojicic, 1999], [Grigoraş, McInerny, Mulcahy, 2002] şi [Grigoraş, 2004].

serviciilor oferite de gazde sau de agenţi pot fi utilizate standarde precum *SSDP* (*Simple Service Discovery Protocol*) [SSDP] sau *SLP* (*Service Location Protocol*) [SLP].

4.2.1.4 Direcţii actuale de cercetare

Dintre provocările ridicate de domeniul calculului bazat pe agenţi se disting[3]:

1. creşterea calităţii aplicaţiilor orientate-agent la nivelul de standard industrial – va conduce atât la apariţia unor metodologii, instrumente şi medii de proiectare orientate-agent, cât şi la integrarea facilă a altor tehnologii actuale (*e.g.*, servicii Web sau Grid) [Moreau, 2002];

2. oferirea unor standarde efective în vederea dezvoltării de sisteme deschise – cercetările se vor focaliza asupra limbajelor de comunicare inter-agent (*e.g.*, FIPA), protocoalelor de interacţiune, arhitecturilor multi-agent, cu rezultate în domenii ca *e-commerce*;

3. oferirea unei infrastructuri semantice – se au în vedere integrarea în Web-ul semantic, dezvoltarea de ontologii, crearea de arhitecturi de tip *broker* etc.;

4. dezvoltarea capacităţilor de raţionare în cadrul mediilor deschise – va conduce la dezvoltarea arhitecturilor BDI, a algoritmilor de negociere sau a raţionamentului ontologic;

5. creşterea abilităţii agenţilor să înţeleagă intenţiile şi scopurile utilizatorilor umani, dar să şi se adapteze la schimbările mediului – astfel, un domeniu important de cercetare este cel direcţionat asupra agenţilor de interfaţă[4];

[3] Aceste direcţii de interes sunt dezirabile mai ales în cadrul spaţiului european, conform [Luck, McBurney & Preist, 2003].

[4] A se vedea şi [Buraga, 2003f].

6. asigurarea încrederii în agenţi – implicarea mai pronunţată a tehnologiilor de securitate, cu dezvoltarea unor modele şi infrastructuri bazate pe încredere şi reputaţie [Patrick, 2002].

După cum se va distinge din cele discutate în subcapitolul 4.4, soluţia de implementare propusă este aliniată direcţiilor de interes menţionate mai sus.

4.2.2 Sisteme multi-agent

4.2.2.1 Prezentare generală

Tehnologiile orientate-agent capătă un tot mai pronunţat rol în dezvoltarea de sisteme software pe scară largă, în domenii precum controlul traficului aerian, simulările interactive, procesele de afaceri sau managementul informaţiilor multimedia distribuite. Sistemele multi-agent se concentrează asupra rezolvării diferitelor probleme ce pot surveni, din prisma modelării comportamentului unei colecţii de agenţi autonomi care încearcă să soluţioneze aceste probleme.

Un *sistem multi-agent* (*multi-agent system*) poate fi definit ca o reţea slab conectată de entităţi computaţionale care lucrează împreună la rezolvarea unei probleme ce nu poate fi soluţionată în mod individual [Bradshow, 1997]. Asemenea entităţi – agenţii software – sunt autonome şi pot fi eterogene.

O caracteristică importantă a sistemelor multi-agent este aceea că fiecare agent posedă informaţii sau prezintă funcţionalităţi incomplete astfel încât nu poate fi capabil să soluţioneze problema, luată în ansamblu, în manieră individuală. Datele procesate de agenţi sunt descentralizate, iar calculul se desfăşoară asincron. De asemenea, nu există un control global al sistemului [Jennings *et al.*, 1998].

Dezvoltarea sistemelor multi-agent are în vedere, la nivel formal, teorii care privesc agenţii ca *sisteme intenţionale*. Convingerile (cunoştinţele) agenţilor ajută la formarea dorinţelor, intenţiilor, obligaţiilor sau opţiunilor agenţilor din mediu – este cazul agenţilor *BDI (Belief-Desire-Intention)* [Rao *et al.*, 1995], descrişi succint mai jos. Un alt model format este cel al semanticii lumilor posibile, inspirat de logica modală [Wooldridge & Jennings, 1995].

Din punct de vedere arhitectural, sistemele multi-agent pot adopta arhitecturi deliberative (bazate pe modelul BDI), arhitecturi reactive sau arhitecturi stratificate [Mangina, 2002].

4.2.2.2 Modele BDI

Agenţii inteligenţi reprezintă o abstractizare destinată conceptualizării şi modelării de sisteme complexe într-o manieră clară şi intuitivă. Una dintre formalizările cele mai potrivite o constituie logica *BDI* (*Belief-Desire-Intention*), punct de plecare pentru definirea unor limbaje de comunicare între agenţi – ca AgentTalk, AgentSpeak(L) sau 3APL – şi pentru dezvoltarea unor medii software – precum JACK, PRS ori dMARS [Ancona & Mascardi, 2003].

Modelul formal BDI are la bază *modelul lumilor posibile* şi e formulat în termenii logicii modale [Kripke, 1963].

Preliminarii

Conform [Rao *et al.*, 1995], sistemele multi-agent prezintă o serie de caracteristici importante:

1. În orice moment de timp, există mai multe modalităţi de evoluţie a mediului (mediul multi-agent este nedeterminist);

2. În orice moment de timp, există mai multe acţiuni (proceduri, metode) care pot fi executate (sistemul multi-agent este nedeterminist);

3. În orice moment de timp, pot exista mai multe obiective care trebuie îndeplinite;

4. Acţiunile care duc la îndeplinirea obiectivelor sunt dependente de starea mediului (contextul de lucru) şi sunt independente de starea internă a sistemului;

5. Măsura în care calculele şi acţiunile se desfăşoară este comparabilă cu măsura în care mediul evoluează.

Se observă că dată fiind condiţia 4., este esenţial ca sistemul să posede informaţii privitoare la starea mediului; denumim aceste informaţii *convingeri* (*beliefs*)[5]. Convingerile pot fi văzute aşadar ca fiind o componentă *informativă* a stării sistemului.

Este necesar ca sistemul să aibă informaţii despre obiectivele care urmează a fi îndeplinite ori sunt în curs de îndeplinire, sau – mai general – despre priorităţile sau costurile asociate obiectivelor curente (a se vedea condiţiile 3. şi 4.). Vom numi aceste informaţii *dorinţe* (*desires*), ele reprezentând starea *motivaţională* a sistemului.

Pentru modelarea sistemului se poate recurge la teoria deciziei. Calculul funcţiei de selecţie a următoarei stări în care va trece sistemul nu ia în consideraţie condiţia 5. şi anume că mediul se poate modifica esenţial şi nedeterminist chiar în timpul execuţiei funcţiei de selecţie sau în timpul execuţiei unei acţiuni. Astfel, va trebui inclusă o a treia componentă a stării sistemului, aceasta trebuind să reprezinte desfăşurarea curentă a acţiunii de îndeplinit, rezultatul întors de cel mai recent apel al funcţiei de selecţie. Această stare captează componenta *intenţională* a sistemului, modelând *intenţiile* (*intentions*) agenţilor.

Pentru a integra cele trei structuri de date – convingerile, dorinţele şi intenţiile –, se recurge la *planuri* (*plans*). Un set de planuri (numit şi bibliotecă de planuri) specifică acţiunile pe care le poate urma un agent pentru a-şi îndeplini intenţiile. Un plan constă din două părţi, prima fiind corpul (programul) care defineşte cursul acţiunilor agentului, iar a doua reprezentând un descriptor care defineşte circumstanţele în care un plan poate fi aplicat (*e.g.*, pre-condiţii) şi consecinţele care vor apărea după aplicarea planului (*e.g.*, post-condiţii). Planurile sunt statice, ele nu se schimbă pe parcursul ciclului de viaţă a unui agent.

Sistemele BDI se modelează prin diverse extensii ale logicii temporale ramificate CTL^* (*Computational Tree Logics*) [Emerson, 1990] [Emerson & Srinivasan, 1989]. Acest tip de logică va modela stările "mentale" ale unui agent. Extinzând logica prin definirea unor conectori şi operatori suplimentari, se vor putea modela convingerile şi sco-

[5]Această componentă poate fi implementată sub forma unei variabile (*e.g.*, membru de tip dată), baze de date, mulţime de expresii logice sau o altă structură de date adecvată.

purile agenţilor, reprezentându-se astfel la nivel formal acţiunile executate de agenţi [Rao *et al.*, 1995]. Lucrările [Rao & Georgeff, 1991c], [Rao & Georgeff, 1991a], [Cohen & Levesque, 1990], [Rao *et al.*, 1995] şi [Wooldridge & Jennings, 1995] oferă detalii privitoare la formalismele adoptate în cazul agenţilor BDI.

Ciclul de execuţie

Conform [Ancona & Mascardi, 2003], sistemele BDI includ o coadă de evenimente E memorând atât evenimentele externe (percepute din mediul de execuţie), cât şi cele interne (generate de agenţi în cursul îndeplinirii propriilor planuri).

Ciclul de execuţie al sistemului BDI este caracterizat de următorii paşi:

- observarea mediului (lumii) şi stării interne a agenţilor, cu actualizarea cozilor de evenimente E_a ale fiecărui agent a în parte;
- generarea de planuri noi ale căror descriptori se potrivesc unui eveniment din coada de evenimente (planuri "relevante") şi ale căror precondiţii sunt satisfăcute (planuri "aplicabile") – aceste planuri sunt incluse într-o mulţime de planuri P;
- selectarea unui plan p din mulţimea P în vederea execuţiei;
- memorarea planului p într-o stivă de intenţii asociată fiecărui agent, în cazul în care evenimentul reprezintă un (sub)scop;
- selectarea unei stive de intenţii şi alegerea planului p (elementul *top*), cu executarea următorului pas al planului: dacă acest pas reprezintă o acţiune, aceasta va fi îndeplinită, altfel dacă este un (sub)scop va fi stocat în coada de evenimente.

Detalii privitoare la modelarea sistemelor de agenţi BDI se găsesc în [Kinny, Georgeff, Rao, 1996] şi [Wooldridge & Ciancarini, 2000].

Exemplu de modelare a comportamentului unui agent

Pentru a modela – conform formalismului BDI – faptul că un agent intenţionează să realizeze o actualizare a conţinutului unei resurse r_1, ştiind că aceasta este în relaţia temporală $Before$ cu o resursă r_2, vom putea scrie (a se vedea şi exemplul 23 din capitolul 3):

$$\mathsf{INTEND}(\mathsf{EF}(\textit{Actualizare}(r_1))) \wedge \mathsf{BEL}(\mathsf{AF}(r_1.\texttt{start} \prec r_2.\mathsf{end})) \qquad (4.1)$$

unde relaţia $Before$ a fost notată $\prec$, conform celor prezentate în secţiunea 3.2.3.3 a capitolului anterior, iar operatorii INTEND şi BEL sunt operatori modali ai logicii BDI, desemnând faptul că agentul intenţionează şi, respectiv, crede (cunoaşte).

Modelul BDI apelează la o serie de operatori temporali[6]: X (următorul – *next*), U (până când – *until*), E (o posibilitate în viitor sau opţional – *eventually*), F (cândva în viitor – *future*). Operatorii G (toţi timpii din viitor sau oricând – *generarly*), B (înainte – *before*) sau A (toate drumurile din viitor sau inevitabil – *always*) pot fi definiţi pe baza celor enunţaţi anterior.

Am recurs la operatorul E deoarece acţiunea de actualizare s-ar putea să nu se poată realiza în viitor, din diverse raţiuni (*e.g.*, serverul Web pe care este stocată resursa ar putea să nu fie operaţional).

După cum se poate remarca, s-a realizat legătura dintre logica ITL şi modelul BDI, comportamentul agenţilor putând depinde de relaţiile temporale stabilite între resursele monitorizate. În acest mod, planurile agenţilor pot fi constituite din declaraţii RDF modelând relaţiile dintre resurse şi specificând acţiunile care trebuie întreprinse, cu concursul limbajelor *XFiles* şi TRSL detaliate în capitolul 3.

Extinderi

Modelul BDI originar a fost extins prin numeroase propuneri, dintre care se pot menţiona:

[6] A se consulta [Rao *et al.*, 1995].

- *BOID* (*Beliefs, Obligations, Intentions, Desires* – adaugă mediului o componentă corespunzătoare obligaţiilor, ciclul de viaţă luând în consideraţie efectele globale ale acţiunilor agenţilor înainte de a fi realizate efectiv [Broersen *et al*, 2001];

- *Coo-BDI* – extindere care ia în calcul caracterul cooperativ al agenţilor implicaţi [Ancona & Mascardi, 2003];

- *TOMAS* (*Transaction Oriented Multi-Agent System*) – model şi arhitectură destinate agenţilor mobili [Busetta *et al.*, 1998].

4.2.2.3 Comunicare inter-agent

Pentru ca agenţii să-şi îndeplinească scopurile, trebuie adoptat un *limbaj de comunicare între agenţi* (*ACL – Agent Communication Language*) standardizat, la nivel scăzut utilizându-se protocoale de comunicaţie ca TCP/IP, SMTP sau HTTP.

Printre abordările actuale se pot menţiona:

- Schimbul de cunoştinţe realizat între agenţii inteligenţi se poate realiza prin intermediul limbajului *KQML* (*Knowledge Query Manipulation Language*) [Bradshow, 1997], bazat pe schimb de mesaje şi pe un protocol de vehiculare a acestor mesaje. Limbajul KQML se bazează pe o abordare stratificată: nivelul inferior e reprezentat de protocolul de transport folosit, iar cel superior desemnează conţinutul specificat de aplicaţie.

- În vederea interpretării şi procesării conţinutului mesajelor vehiculate, s-a dezvoltat limbajul *KIF* (*Knowledge Interchange Format*) [Genesereth & Fikes, 1992]. Ca sintaxă pentru conţinutul mesajelor s-a adoptat calculul cu predicate de ordin întâi, semantica fiind una declarativă. Oferă metode pentru reprezentarea meta-cunoştinţelor şi poate juca rol de mediator între limbaje precum Prolog, Lisp sau XML.

- Pentru asigurarea inter-operabilităţii, s-a definit standardul *FIPA ACL* [FIPA] care include acţiuni de comunicare de bază, destinate a fi folosite în cadrul unor protocoale de interacţiune.

Una dintre direcţiile promiţătoare – aleasă şi de soluţia noastră de implementare – este cea de utilizare a meta-familiei XML ca limbaj, protocol şi mecanism de vehiculare a cunoştinţelor. Conform unor lucrări ca [Haustein, 2001] şi [Buraga & Alboaie, 2004], schimbul de date între agenţi se poate realiza, de asemenea, prin intermediul protocolului SOAP, cu implicarea unor limbaje de exprimare a metadatelor şi ontologiilor, precum DCMI, RDF sau OWL.

4.3 Servicii Web

4.3.1 Prezentare generală

În ultimii ani, lumea informaticii a evoluat destul de mult în privinţa dezvoltării şi exploatării de aplicaţii distribuite, disponibile pe calculatoare localizate oriunde pe mapamond, principala activitate fiind cea de *comunicare*. Interfeţele software utilizate pentru facilitarea comunicării între diverse situri sunt referite prin termenul generic de servicii Web.

O posibilă definiţie[7] a conceptului de *serviciu Web* este aceea că acesta este un sistem software proiectat pentru a suporta interacţiunea maşină-maşină la nivel de Web, având o interfaţă de programare descrisă într-un format procesabil de către calculator.

Un serviciu Web reprezintă o colecţie de funcţii împachetate, considerate ca fiind o singură entitate şi publicate în spaţiul WWW cu scopul de a fi folosite de alte programe; serviciile Web sunt cărămizile pentru crearea de sisteme distribuite deschise, permiţând organizaţiilor şi dezvoltatorilor independenţi să-şi facă disponibile pe Web bunurile digitale într-un mod rapid şi facil [Buraga, 2003a, cap. 7].

Serviciile Web formează un cadru general pentru facilitarea modului de comunicare pe reţea (şi, totodată, de interconectare a componentelor programabile ale Web-ului), având la bază ideea atomică de serviciu (oferirea unei funcţionalităţi specifice). O asemenea standar-

[7] Nu există o definiţie unanim acceptată. A se consulta şi [Vasudevan, 2001], [Gorman, 2001] sau [Sycare & Payne, 2003].

dizare este posibilă cu concursul altor standarde deja existente şi utilizate pe scară largă, dintre care cel mai important este meta-limbajul XML. Multe dintre caracteristicile serviciilor Web sunt similare celor ale agenţilor software.

Figura 4.1: Nivelurile de standardizare ale serviciilor Web

4.3.2 Standarde

4.3.2.1 Descrierea unui serviciu Web

În vederea exploatării funcţionalităţilor puse la dispoziţie de un serviciu Web, trebuie mai întâi specificată *descrierea* acestuia.

Descrierea unui serviciu Web se realizează în limbajul *WSDL* (*Web Service Description Language*) [Curbera *et al.*, 2002, W3C], care oferă un model şi o sintaxă bazată pe XML pentru aceasta. Standardul WDSL permite separarea descrierii abstracte a funcţionalităţii oferite de un serviciu de detaliile concrete ale descrierii acelui serviciu (de tipul "cum" şi "unde" este disponibilă acea funcţionalitate).

Limbajul WSDL descrie serviciile Web începând cu mesajele care sunt schimbate între entitatea care oferă şi cea care invocă un anumit serviciu. Mesajele sunt descrise în manieră abstractă şi apoi sunt asociate unui protocol de reţea, precizându-se de asemenea şi un format (o sintaxă). Un mesaj constă dintr-o colecţie de date de anumite tipuri, iar schimbul de mesaje este descris ca o operaţie.

Conform [Sycare & Payne, 2003], au fost propuse diverse alte limbaje de descriere a serviciilor Web, dintre care se disting:

- *BPEL4WS* (*Business Processing Execution Language for Web Services*)
- *WSCM* (*Web Services Component Model*)
- *WSMF* (*Web Services Modeling Framework*)
- *WSML* (*Web Services Meta Language*)
- *WSOL* (*Web Services Offerings Language*)
- *WSXL* (*Web Services Experience Language*)
- *WSUI* (*Web Services User Interface*)
- *XLANG* (*Web Services for Business Processes*)

4.3.2.2 Publicarea şi regăsirea unui serviciu Web

Apare necesitatea punerii la dispoziţie, în manieră publică, a descrierii unui anumit serviciu Web în vederea invocării sale ulterioare de o altă componentă programabilă. Este posibil ca o anumită funcţionalitate oferită de un grup de servicii Web să poată fi regăsită prin intermediul unei interogări formulate de partea care intenţionează să folosească acea funcţionalitate.

Pentru aceasta, s-a instituit un catalog al tuturor furnizorilor de servicii Web în vederea regăsirii informaţiilor dorite – *UDDI* (*Universal Description, Discovery, and Integration*) [Curbera *et al.*, 2002, UDDI] –, oferind o bază de date distribuită a serviciilor Web din prisma tipurilor de afaceri electronice pe care acestea le pot modela. Uzual, căutarea în

cadrul UDDI se realizează în funcţie de clasa din care face parte afacerea, după un nume de serviciu sau după locaţia geografică a furnizorului afacerii respective.

Fiecare afacere înregistrată via UDDI îşi prezintă toate serviciile şi atribuie fiecăruia dintre aceste servicii un tip. Un tip de serviciu este derivat dintr-un set de tipuri de servicii de bază, *a-priori* înregistrate cu UDDI. Tipurile de servicii sunt numite *tModele* în limbajul UDDI. Fiecare *tModel* are asociat un tuplu compus dintr-un nume, o descriere şi un identificator unic, numit *tModelKey*:

$$tm \equiv \langle n, d, k \rangle \tag{4.2}$$

unde n şi d sunt cuvinte peste un alfabet de simboluri Σ, iar $k \in \mathsf{Keys}$, mulţimea Keys fiind o mulţime de identificatori de tip *tModelKey*.

Având la dispoziţie o mulţime de tipuri de servicii bine cunoscute, UDDI dă posibilitatea de a afla modalitatea prin care se pot iniţia afaceri electronice cu o companie. Acesta este principalul avantaj pe care îl are UDDI în comparaţie cu alte cataloage de afaceri bazate pe Web.

Alternativele la UDDI sunt *USML* (*UDDI Search Markup Language*) şi *WS-Inspection* (a se vedea [Sycare & Payne, 2003]).

4.3.2.3 Invocarea serviciilor Web

Pentru invocarea serviciilor Web apare necesitatea utilizării unui protocol de comunicare între entitatea care intenţionează să apeleze un serviciu Web şi serverul pe care va rula acel serviciu. În acest sens, trebuie găsit un format pentru transmisia în format XML a parametrilor de intrare şi întoarcerea răspunsului primit de la serviciul Web invocat. Se foloseşte un mecanism similar celui utilizat de paradigma *RPC* (*Remote Procedure Call*)[8].

Principalele protocoale de comunicare utilizate în prezent sunt [9]:

- *XML-RPC* [XML-RPC] – oferă o specificaţie şi un set de implementări care permit software-ului care rulează pe sisteme de ope-

[8] A se consulta [Comer & Stevens, 1993] şi [Buraga & Ciobanu, 2001, cap. 11].

[9] Un studiu comparativ între protocoalele XML-RPC şi SOAP se găseşte în [Rhodes, 2003].

rare dispersate şi în medii eterogene să realizeze apeluri de proceduri prin Internet. Foloseşte protocolul HTTP pentru transportul efectiv al mesajelor şi limbajul XML pentru codificarea acestora. Este proiectat pentru a fi cât mai simplu, în acelaşi timp însă permiţând ca structuri complexe de date să fie transmise, procesate şi returnate.

- *SOAP* (*Simple Object Access Protocol*) [Gorman, 2001, W3C] – propune facilităţi sofisticate şi oferă o paletă largă de posibilităţi de reprezentare (serializare şi deserializare) a datelor vehiculate[10].

  Un mesaj SOAP este compus din trei părţi: un *plic* (*envelope*) care defineşte cadrul de lucru pentru a descrie ceea ce conţine mesajul şi modul de procesare a acestui conţinut, un set de *reguli de codificare* (*encoding rules*) pentru a exprima instanţe ale tipurilor de date definite în aplicaţie şi o *convenţie de reprezentare* (*representation*) a apelurile de proceduri la distanţă şi răspunsurile furnizate în urma invocării acestor proceduri (metode).

  Actualmente, SOAP este standard industrial şi recomandare oficială a Consorţiului Web.

Remarca 8 Peste protocolul de comunicare poate exista un alt protocol specific, de exemplu *ebXML* [ebXML] folosit pentru conversaţii de afaceri şi reprezentând o metodă standardizată pentru modelarea de afaceri B2B (*business-to-business*) prin intermediul serviciilor Web[11].

4.3.3 Servicii Web semantice

4.3.3.1 Problematici actuale

Atât serviciile Web, cât şi sistemele multi-agent pot avea diverse beneficii prin ataşarea de semantici, din prisma interoperabilităţii, a com-

[10] A se vedea şi [Alboaie & Buraga, 2003a] pentru o propunere de serializare bazată pe SOAP a mesajelor interschimbate de agenţi.

[11] A se consulta şi [Wright *et al.*, 2003] privind diverse utilizări ale protocolului *ebXML* la constituirea unui mediu de tip *e-travel* bazat pe agenţi.

punerii, descoperirii şi invocării automate a serviciilor sau a accesării cunoştinţelor stocate de Internet.

Definirea de planuri pentru îndeplinirea de către agenţi a diverselor scopuri se poate baza pe invocarea unor servicii Web compuse. Interpretarea la nivel semantic a mesajelor vehiculate (cereri şi răspunsuri) între diverse entităţi şi servicii trebuie să ia în calcul limbaje bazate pe metadate şi ontologii [Sycare & Payne, 2003].

Printre problemele actuale care trebuie rezolvate se enumeră cele legate de neconcordanţe semantice la:

- nivel al conţinutului – de exemplu, un serviciu Web care furnizează localizarea unei resurse multimedia returnează adresa numerică a serverului, deşi solicitantul doreşte URI-ul acelei resurse;
- nivel de atribut – se solicită tipul MIME `video/mpeg`, dar serviciul Web returnează tipul generic `Video`;
- nivel al unităţilor de măsură – serviciul solicită ca dată de intrare dimensiunea unui fişier audio în octeţi, dar i se trimite un număr reprezentând dimensiunea în MB;
- nivel de mesaj – procesul solicitant poate furniza rezoluţia unei imagini ca pereche *lăţime* $\times$ *înălţime*, însă serviciul Web doreşte numărul total de pixeli alocaţi.

4.3.3.2 Caracterizare

Pentru descrierea semantică şi invocarea corespunzătoare a serviciilor Web, eforturile sunt concentrate asupra adoptării unei ontologii de nivel înalt, capabilă să descrie proprietăţi şi funcţionalităţi ale serviciilor şi agenţilor Web, într-o formă procesabilă de calculator.

O soluţie este *DAML-S* (*DARPA Agent Markup Language for Services*) [McIlraith, Son, Zeng, 2001, DAML-S], având la bază modelul DAML+OIL. Obiectivul principal este punerea la dispoziţie a unui mecanism de interoperabilitate semantică via standarde ale Web-ului semantic. Principalul avantaj va fi automatizarea descoperirii, selectării, invocării, compunerii şi monitorizării execuţiei serviciilor Web.

Ontologiile DAML-S sunt folosite la specificarea unor modele de servicii, avându-se în vedere profilurile serviciilor, modelele de procesare şi logica serviciilor. Un serviciu Web poate fi văzut ca proces atomic, simplu sau compus. Fiecărui proces i se asociază un profil şi are ataşate proprietăţi ca pre-condiţie, intrare, ieşire şi efect. Un proces compus posedă proprietăţi compuse, plus o proprietate care indică dacă e invocabil sau nu. Activităţile care trebuie îndeplinite pot fi astfel considerate drept servicii Web compuse, modul de compunere fiind specificat prin DAML-S[12].

Ca instrumente software folosite în cadrul DAML-S se pot menţiona aplicaţiile *MindSwap's Web Service Composer*, *KSL DAML-S Editor* şi *SotonCCWS* (detalii în [Sycare & Payne, 2003]).

O serie de alternative la DAML-S sunt *IRS* (*Internet Reasoning Service*) [IRS], *SWOBIS* [Sollazzo *et al.*, 2001] şi *WSML* (*Web Service Modeling Framework*) [Fensen, Bussler, 2002].

4.3.3.3 Exemplificare

În cadrul secţiunii de faţă, vom prezenta modul de specificare a unui serviciu Web semantic de căutare a resurselor multimedia. Acest serviciu este o componentă a platformei *ITW* descrisă în subcapitolul 4.4.

Ontologia DAML-S consideră serviciile Web ca fiind procese. Serviciul de căutare, denumit *ITWSearch*, va fi derivat din clasa *Process* a ontologiei utilizate:

```
<rdfs:Class rdf:ID="ITWSearch">
   <rdfs:subClassOf rdf:resource="&process;" />
</rdfs:Class>
```

În locul URI-urilor desemnând construcţiile DAML-S am folosit entităţi pentru a simplifica exemplele furnizate.

Abstractizând, putem considera pentru serviciul *ITWSearch* doar intrarea, ieşirea şi, eventual, o serie de parametri de control. Implementarea efectivă a serviciului nu are importanţă în acest moment.

[12] Detalii referitoare la compunerea serviciilor Web, privite drept componente *middleware*, sunt disponibile în [Benatallah *et al.*, 2003].

Ca date de intrare, transmise serviciului la invocarea sa, vom avea o interogare – compusă din elemente WQFL –, ieşirea fiind o listă de documente WQFL conţinând informaţii privitoare la resursele multimedia găsite. Intrările şi ieşirile sunt modelate ca sub-proprietăţi ale proprietăţilor generice *input* şi *output* [DAML-S].

În cazul nostru, vom avea (pentru fiecare proprietate va trebui să specificăm domeniul şi intervalul – a se vedea cele discutate în secţiunea 2.2.5.3 a capitolului 2):

```
<!-- intrarea -->
<rdf:Property rdf:ID="query">
   <rdfs:subPropertyOf rdf:resource="&input;" />
   <rdfs:domain
      rdf:resource="#ITWSearch" />
   <rdfs:range
      rdf:resource="http://www.infoiasi.ro/wqfl.xsd" />
</rdf:Property>

<!-- iesirea -->
<rdf:Property rdf:ID="foundResources">
   <rdfs:subPropertyOf rdf:resource="&output;" />
   <rdfs:domain
      rdf:resource="#ITWSearch" />
   <rdfs:range
      rdf:resource="http://www.infoiasi.ro/wqfl.xsd" />
</rdf:Property>
```

Vom putea defini şi diverşi parametri de control ai serviciului Web *ITWSearch*. Mai jos este specificat parametrul *regexpSwitch* care indică sistemului să proceseze sau nu expresiile regulate Perl care pot apărea în cadrul interogărilor. Acest parametru va fi exprimat de o proprietate DAML-S:

```
<!-- parametrul de control -->
<rdf:Property rdf:ID="regexpSwitch">
   <rdfs:subPropertyOf rdf:resource="&parameter;" />
   <rdfs:range
      rdf:resource="#regexp" />
</rdf:Property>
```

4.4 *ITW* – o platformă distribuită destinată descoperirii resurselor multimedia

4.4.1 Prezentare generală

Sistemul *ITW* reprezintă o soluţie software pentru descoperirea resurselor multimedia, concretizată într-o platformă distribuită compusă din agenţi software, servicii Web (semantice) şi alte entităţi programabile, precum scripturile *CGI* (*Common Gateway Interface*).

Suportul pentru implementare se bazează pe utilizarea modelelor formale prezentate în secţiunile anterioare în vederea dezvoltării de agenţi BDI şi de servicii Web semantice. Sistemul *ITW* foloseşte descrieri RDF şi XML pentru căutarea resurselor multimedia plecând de la interogări modelate prin WQFL (a se vedea sub-capitolul 3.3) şi se bazează pe relaţiile spaţio-temporale adnotate în limbajul TRSL definit în secţiunea 3.2.7.1 şi stabilite între diferite (fragmente de) situri Web.

4.4.2 Arhitectura sistemului

Aspectul-cheie avut în vedere în proiectarea sistemului *ITW* este cel de a oferi spre exploatare o arhitectură flexibilă, multi-platformă pentru descoperirea resurselor (multimedia) pe baza *informaţiilor semantice* asociate resurselor. În cadrul procesului de descoperirea a resurselor, una dintre cele mai importante probleme este aceea de a folosi relaţiile temporale care se pot stabili între acestea. Pentru aceasta, se pot dezvolta o serie de agenţi software care, folosind modelul RDF prezentat în secţiunea 3.2.7 a precedentului capitol, vor fi capabili să genereze şi apoi să utilizeze construcţii TRSL. Aceşti agenţi vor face parte dintr-un sistem de multi-agenţi BDI (a se vedea şi secţiunea 4.2.2.2).

Arhitectura generală a sistemului propus constă în două tipuri majore de entităţi programabile [Buraga & Găbureanu, 2003]:

- *agenţii* ITW

  Rolul acestora este cel de a descoperi resurse multimedia distribuite stocate pe diverse situri Web. Vor folosi reprezentări spaţio-temporale TRSL şi vor genera specificaţii *XFiles* (definite în

secţiunea 3.2.5.1; a se consulta şi anexa B) pentru a memora localizarea fizică a resurselor.

Agenţii *ITW* vor rula într-un mediu orientat-agent (a se vedea secţiunea 4.4.2.1).

- *servicii Web* ITW

  Scopul acestor servicii Web este cel de a oferi posibililor clienţi informaţiile dorite privitoare la resursele căutate şi accesul la aceste resurse. Serviciile Web vor putea fi invocate pe diverse aplicaţii, precum agenţi-utilizator (i.e. navigatoare Web sau alţi clienţi speciali), alţi agenţi Web ori servicii Web.

  Serviciile Web *ITW* pot avea asociate descrieri semantice exprimate prin declaraţii DAML-S.

Din punctul de vedere al conceperii şi exploatării serviciilor Web, *ITW* se doreşte a fi o infrastructură eterogenă interoperabilă bazată pe servicii Web. Prin intermediul unei interfeţe de tip portal Web, utilizatorul va putea formula interogări complexe implicând informaţii temporale şi de structură referitoare la resursele dorite a fi descoperite. Pentru a îndeplini acest deziderat se va folosi limbajul WQFL specificat în cadrul secţiunii 3.3.3 a capitolului precedent.

Informaţiile şi metadatele marcate de aserţiuni RDF vor fi stocate pe serverele independente, în vederea asigurării toleranţei la defecte şi a scalabilităţii sistemului. Chiar dacă unul dintre servere nu va putea fi operaţional, sistemul *ITW* îşi va putea continua activitatea, graţie structurii sale flexibile compuse din agenţi şi servicii Web. Sistemul permite, de asemenea, interogarea în paralel a unor servicii Web – punând la dispoziţie funcţionalităţi noi – care pot fi ataşate arhitecturii software fără restartarea sistemului. Aceste servicii Web pot fi considerate *extensii* (*plug-in*-uri) *ITW*.

Aplicaţia *ITW* are o structură modulară, prezentată în figura 4.2.

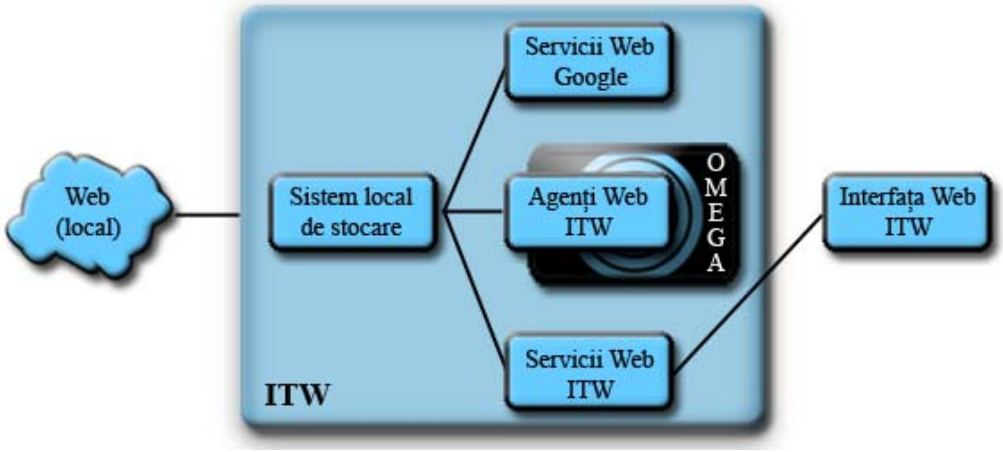

Figura 4.2: Arhitectura internă a sistemului *ITW*

4.4.2.1 Agenţi *ITW*

Ca agenţi *ITW* vom considera entităţile software făcând parte din sistemul *Omega* [Alboaie & Buraga, 2002]. Acest sistem reprezintă o infrastructură orientată-obiect oferind un spaţiu de adresare ierarhică pentru obiectele Web şi punând la dispoziţie o serie de funcţionalităţi pentru accesul la distanţă a resurselor Web distribuite.

Principalul scop în dezvoltarea sistemului *Omega* a fost acela de a oferi o structură obiectuală distribuită, prin intermediul unor reprezentări independente de platformă ale tipurilor de date, instrucţiunilor, funcţiilor şi obiectelor ce compun un limbaj de programare orientat-agent. De asemenea, *Omega* pune la dispoziţie un mediu interpretat de dezvoltare de multi-agenţi, oferind mecanisme pentru controlul execuţiei şi pentru serializarea informaţiilor vehiculate între agenţi.

Pentru numirea şi regăsirea obiectelor (agenţilor), sistemul implementează un *serviciu de nume* (*name service*) [Alboaie & Buraga, 2002].

Clasa de bază pentru fiecare clasă *Omega* este clasa `IObject`. Fiecare obiect care necesită un spaţiu de stocare va utiliza `IObject`. În acest mod se pune la dispoziţie o memorie distribuită comună care poate fi utilizată de toţi agenţii *ITW*. Fiind un mediu obiectual, *Omega* expune un număr de tipuri de obiecte predefinite, instanţe ale unor

clase fundamentale precum *String*, *Number*, *List* şi *Control*.

Clasa *Control* este folosită pentru controlul execuţiei agenţilor (i.e. suportul pentru fire de execuţie virtuale, interpretarea instrucţiunilor unui limbaj de tip *script* etc.).

În cadrul mediului, tipurile de date sunt reprezentate de diferite clase precum `IString`, `INumber`, `IOmegaStack`, `IOmegaList` sau `IOmegaQueue`, derivate din clasa generică `IObject`. Tipurile de date sunt divizate în două mari categorii [Alboaie & Buraga, 2002]:

- *tipuri de date simple (scalare)* – nu au alte elemente în componenţă (ca exemple se pot da `INumber` sau `IString`);
- *tipuri de date compuse* – reprezintă un ansamblu de două sau mai multe tipuri simple (*e.g.*, `IName`, `IOmegaStack`, `IOmegaList` sau `IAThread`).

Sistemul *Omega* pune la dispoziţie suportul pentru crearea de agenţi BDI, planurile agenţilor putând fi interpretate de mediu, conform celor descrise în secţiunea 4.2.2.2.

Partea activă a fiecărui obiect (agent) constă în scurte secvenţe de cod scris într-un limbaj de tip *script*. Folosindu-se drept abstractizare `IObject`, a fost implementat un suport pentru fire de execuţie reprezentate de obiecte din clasa `IAThread`. Mini-nucleul *Omega* – prezentat în [Alboaie & Buraga, 2002] – oferă un model de date (un sistem de tipuri de bază, plus o manieră de construcţie a noilor obiecte), un spaţiu de adresare (fiecare obiect posedă propria sa adresă consistentă la nivelul Internet) şi diverse tehnici de implementare a construcţiilor unui limbaj de programare de nivel înalt (*e.g.*, instrucţiuni precum `if`, `while`, `for` sau `goto`) [Alboaie & Buraga, 2003a].

Ideea de la care s-a plecat a fost aceea că, în momentul rulării, maniera de instanţiere şi de execuţie a agenţilor să se reducă la crearea de obiecte de tip `IObject` în cadrul spaţiului comun de obiecte distribuite (se oferă astfel suport pentru interschimbul de informaţii).

Starea obiectelor *Omega* poate fi exprimată prin intermediul unor aserţiuni RDF [Buraga & Alboaie, 2004]. Fiecărui obiect făcând parte

din sistemul de multi-agenţi considerat i se pot asocia diferite metadate. Aceste meta-descrieri pot fi utilizate, printre altele, la controlul versiunilor şi accesului la diverse informaţii. Din raţiuni de securitate, metadatele vor putea stoca lista permisiunilor de acces asociate fiecărui obiect. Această abordare este inspirată de modelul descris în cadrul secţiunii 3.2.5.1 a capitolului precedent.

Sistemul va păstra aceste descrieri drept construcţii RDF care pot fi transportate către alte obiecte în timpul activităţii de actualizare a informaţiilor (i.e. replicarea obiectelor). Dinamica agenţilor implicaţi şi a legăturilor dintre ei poate fi, de asemenea, exprimată prin intermediul descrierilor RDF, în conjuncţie cu relaţiile temporale marcate în TRSL (a se vedea secţiunea 3.2.7.1 a capitolului 3).

Sistemul pune la dispoziţie şi un *mecanism de serializare*. Toate clasele derivate din `IObject` trebuie să implementeze metode de serializare şi de deserializare a datelor vehiculate între agenţi. Mecanismul de serializare adoptă modelul RPC [Buraga & Ciobanu, 2001, cap. 11].

Pentru asigurarea independenţei de platformă, s-a recurs la o serializare bazată pe XML [Alboaie & Buraga, 2003a], utilizându-se ca tipuri primare – ale datelor interschimbate de agenţi în cadrul proceselor de serializare şi deserializare – pe cele specificate de XML Schema.

Exemplul 30 În cazul unui obiect instanţiat din clasa `IString` vom putea avea (ca valoare efectivă am furnizat un tip MIME corespunzător conţinutului unei resurse video):

```
<IString>
   <name xsi:type="xsd:string">
      video/mpeg
   </name>
</IString>
```

Stilul de codificare se bazează pe cel specificat de XML Schema. Toate tipurile de date utilizate în cadrul sistemului vor fi serializate conform tipurilor XML Schema sau pe baza celor derivate din tipurile de date *Omega*.

Maniera de serializare bazată pe tipuri XML Schema nu este impusă, pentru o mai mare flexibilitate putându-se utiliza şi alte mecanisme.

O altă modalitate de serializare este aceea folosind protocolul *SOAP* (*Simple Object Access Protocol*) [Gorman, 2001, W3C].

În mod similar cu alte modele distribuite obiectuale [Bakken, 2001] – precum CORBA IIOP sau DCOM –, SOAP poate invoca metode, servicii, componente şi obiecte pe servere aflate la distanţă, utilizându-se ca protocol de transport pe scară largă HTTP. Spre deosebire de alte protocoale, mesajele vehiculate nu sunt binare, ci sunt marcate în XML. De asemenea, protocolul SOAP poate oferi suport pentru activităţile de interschimb şi rutare a mesajelor în cadrul sistemului de multi-agenţi considerat, integrându-se în standarde actuale ca *FIPA* (*The Foundation of Intelligent Physical Agents*) [FIPA].

În ceea ce priveşte serializarea, SOAP se bazează pe un model de date orientat-obiect simplu, pentru validarea construcţiilor sintactice utilizându-se scheme XML. Serializarea bazată pe SOAP se integrează în cadrul modelului oferit pe *UML* (*Unified Modeling Language*) [OMG]. Meta-modelul UML poate fi utilizat la serializarea modelelor UML prin intermediul sintaxei de serializare SOAP, conform [Haustein, 2001].

Un exemplu de serializare folosind mecanismul SOAP este detaliat în lucrarea [Alboaie & Buraga, 2003a], iar [Haustein, 2001] discută maniera de serializare bazată pe SOAP în conjuncţie cu RDF şi alte limbaje înrudite.

4.4.2.2 Servicii Web *ITW*

Partea bazată pe servicii Web a aplicaţiei constă din următoarele componente distribuite [Buraga, 2003g, Buraga & Găbureanu, 2003]:

- o interfaţă Web bazată pe limbajul XUL [Oeschger, 2003], oferind o interfaţă flexibilă de interogare; această componentă este similară celei de la un motor de căutare (a se vedea secţiunea 3.3.2.3);
- un număr de m servicii Web locale menite a pune la dispoziţie informaţii privitoare la resursele stocate pe un Web local (i.e. intranetul sau situl Web public ale unei organizaţii);

- un număr de n servicii Web externe, dezvoltate de alte organizaţii, invocate în vederea interogării altor motoare de căutare (de exemplu, serviciul Web public de căutare oferit de Google – a se vedea şi secţiunea 4.4.2.4 de mai jos);

Localizarea fizică (i.e. adresa Web) şi modul de execuţie (*e.g.* platforma hardware, sistemul de operare, serverul Web, limbajul de programare etc.) ale celor $(m + n)$ servicii Web sunt transparente pentru utilizatorul final. Aceste servicii Web pot fi considerate, în acest sens, drept o singură entitate (foarte similară unui Grid de tip computaţional [Foster & Kesselan, 1999, Buyya, 2002]) a sistemului *ITW*.

Dacă adresele IP ale serviciilor Web locale sunt fixate şi publice, atunci aceste servicii pot fi invocate în manieră independentă de alte entităţi externe. Desigur, informaţiile interschimbate între serviciile Web *ITW* şi clienţii care le invocă sunt încapsulate de mesaje SOAP.

Sistemul *ITW* oferă – via construcţii WSDL – descrieri pentru serviciile Web locale şi utilizează descrierile WSDL ale serviciilor Web externe în vederea invocării. De asemenea, fiecărui tip de serviciu – de exemplu, *ITWSearch* – îi sunt asociate descrieri semantice prin intermediul DAML-S, conform celor discutate în secţiunea 4.3.3.3.

După cum s-a menţionat, informaţiile şi metadatele privitoare la resursele regăsite de agenţii *ITW* sunt stocate ca aserţiuni RDF, incluzând marcaje *XFiles* şi TRSL. Metadatele asociate şi localizarea resurselor sunt memorate de documente *XFiles*, iar informaţiile temporale privitoare la resursele Web se stochează ca fişiere TRSL, conform celor descrise în capitolul 3.

Aceste documente vor fi automat generate de către agenţii *ITW*. O soluţie preliminară folosea un robot Web rulat la momente regulate de timp pentru actualizarea informaţiilor referitoare la resursele hipermedia ale unui sit Web disponibil local.

Exemplul 31 Metadatele asociate unei resurse multimedia – în cazul de faţă, un film stocat în formatul MPEG – sunt reprezentate în modul următor (omitem declaraţiile spaţiilor de nume XML):

```
<rdf:RDF>
  <rdf:Description
```

```
rdf:about="http://www.infoiasi.ro/Video.mpg">
<!-- informatii temporale -->
<temporal:link temporal:begin="2003-05-09T17:15:00"
               temporal:end="2004-05-09T10:00:00"
               temporal:type="temporal">
   <temporal:Before temporal:dur="7D">
     <rdf:Description
        rdf:about="http://mirror.org/video/1.mpeg">
        <!-- metadate privitoare la copie -->
        <dc:Description>
            Copie a resursei Video.mpg
            localizata initial la
            http://www.location.org/Video.mpg.
        </dc:Description>
        <dc:Format>
            <rdf:Bag>
               <rdf:li>video/mpeg</rdf:li>
            </rdf:Bag>
        </dc:Format>
        <!-- relatia cu serverul actual de stocare -->
        <dc:Relation rdf:parseType="Resource">
            <dcq:ResourceType
               rdf:resource="http://purl.org/metadata/
                 dublin_core_qualifiers#IsPartOf" />
               <rdf:value
                  rdf:resource="http://mirror.org" />
            </dcq:ResourceType>
        </dc:Relation>
     </rdf:Description>
   </temporal:Before>
</temporal:link>

<!-- metadate privitoare la resursa originala -->
<meta:Properties>
   <meta:Location meta:ip="193.231.30.225"
                  meta:port="80">
      www.infoiasi.ro
   </meta:Location>
```

```
        <meta:Auth meta:type="basic">
           UsersGroup
        </meta:Auth>
        <meta:Owner>
           <meta:Login>web</meta:Login>
           <meta:Password meta:method="md5">
              ...
           </meta:Password>
        </meta:Owner>
     </meta:Properties>
  </rdf:Description>
</rdf:RDF>
```

Aici, în locul spaţiilor de nume *xf* şi *t* corespunzătoare construcţiilor sintactice ale limbajelor *XFiles* şi, respectiv, TRSL, au fost folosite *meta* şi *temporal* (a se consulta şi anexele B şi C).

Documentul de mai sus defineşte relaţia $Before$ între două resurse. Resursa video identificată de `http://www.infoiasi.ro/Video.mpg` este considerată ca fiind resursa originală, iar copia ei este localizată pe alt sit, la adresa `http://mirror.org/video/1.mpeg`. Fiecare actualizare a resursei originale implică o actualizare a copiei după o perioadă de 7 zile. Resursa originală este disponibilă pe Web în perioada de timp specificată de atributele `begin` şi `end`. Metadatele asociate stochează informaţii referitoare la adresa maşinii pe care este memorată resursa, proprietarul acesteia şi mecanismul de autentificare folosit pentru accesarea documentului video. Agenţii Web vor genera, de asemenea, şi alte informaţii folositoare, ca de exemplu elemente DCMI [Purl] sau RSS [RSS] (în exemplul dat, se poate observa utilizarea unor construcţii DCMI, aparţinând spaţiilor de nume *dc* şi *dcq*, pentru ataşarea de metadate copiei resursei multimedia).

4.4.2.3 Interfaţa *ITW*

Interfaţa sistemului *ITW* foloseşte o extensie a limbajului WQFL prezentat în secţiunea 3.3.3 a capitolului 3 pentru a oferi utilizatorilor finali o modalitate de formulare a unor interogări complexe. Această extensie este *WQGL* (*Web Query Graphical Language*). Limbajul, descris

pe larg în [Buraga & Brut, 2001] şi [Buraga & Brut, 2002], poate fi utilizat pentru generarea de interfeţe grafice Web dinamice, componentele acestora fiind construite în *XUL* (*Extensible User-interface Language*) [Oeschger, 2003]. Documentele XUL sunt suportate în prezent de navigatorul Mozilla pentru a genera o interfaţă Web similară celei utilizate de sistemele de ferestre actuale (*e.g.*, XWindow System sau Microsoft Windows) sau de mediile de dezvoltare (precum Borland Delphi, Glade, KDevelop ori Microsoft Visual Studio .NET).

Documentele WQGL vor extinde construcţiile WQFL via un nou element `<interface>` care va putea include elementele şi atributele XUL necesare generării interfeţei dorite. De asemenea, în locul marcatorilor XUL pot fi direct folosite *tag*-uri *WML* (*Wireless Markup Language*) [Buraga, 2003a] pentru a putea exploata interfaţa într-un mediu fără fir.

Exemplul 32 Pentru a specifica un control grafic de tip *tab*, documentul WQGL corespunzător interogării prezentate în cadrul secţiunii 3.3.3 este (spaţiul de nume *xul* desemnează construcţiile XUL utilizate):

```
<?xml version="1.0" ?>
<webquery>
  <engine url="http://www.google.com">
     Google
  </engine>
  <query>temporal Petri nets</query>

  <!-- interfata utilizata -->
  <interface target="Mozilla"
             language="XUL" type="text/xul">
     <xul:window xul:title="WQGL" css:style="width: 400px"
        xmlns:xul="http://www.mozilla.org/keymaster/
                   gatekeeper/there.is.only.xul">
       <xul:tabcontrol xul:align="vertical">
          <xul:tabbox value="Structure" />
          <xul:tabbox value="Textual query" />
          <xul:tabbox value="Options" />
```

```
      </xul:tabcontrol>
    </xul:window>
  </interface>

  <!-- structura continutului -->
  <structure>
    <element name="p">
      <occur top="7" />
    </element>
    <element name="a" appear="no">
      <occur top="0" middle="0" bottom="0" />
    </element>
    <element name="table" appear="no">
      <occur top="0" middle="0" bottom="0" />
    </element>
  </structure>
</webquery>
```

Remarca 9 Limbajul XUL nu oferă suport pentru modelarea răspunsului la evenimente (*e.g.*, apăsarea unui buton, selectarea unei opţiuni, reafişarea unui control grafic) sau exprimarea funcţiilor pe care interfaţa trebuie să le îndeplinească. Un alt limbaj bazat pe XML suplineşte aceste lipsuri: *XUP* (*Extensible User-interface Protocol*) [W3C], dezvoltat cu scopul de a comunica evenimente de interfaţă într-o formă independentă de platformă şi de dispozitiv. Pentru mai multe detalii privitoare la folosirea limbajului XUP în cadrul unei interfeţe Web se poate consulta [Buraga, 2003b].

4.4.2.4 Implementarea curentă

Sistemul *ITW*, aflat în stadiu de prototip, include următoarele componente (ilustrate în figura 4.2):

1. un script CGI conceput în limbajul Perl, cu rolul de a genera interfaţa WQGL şi de a funcţiona drept client generic pentru serviciile

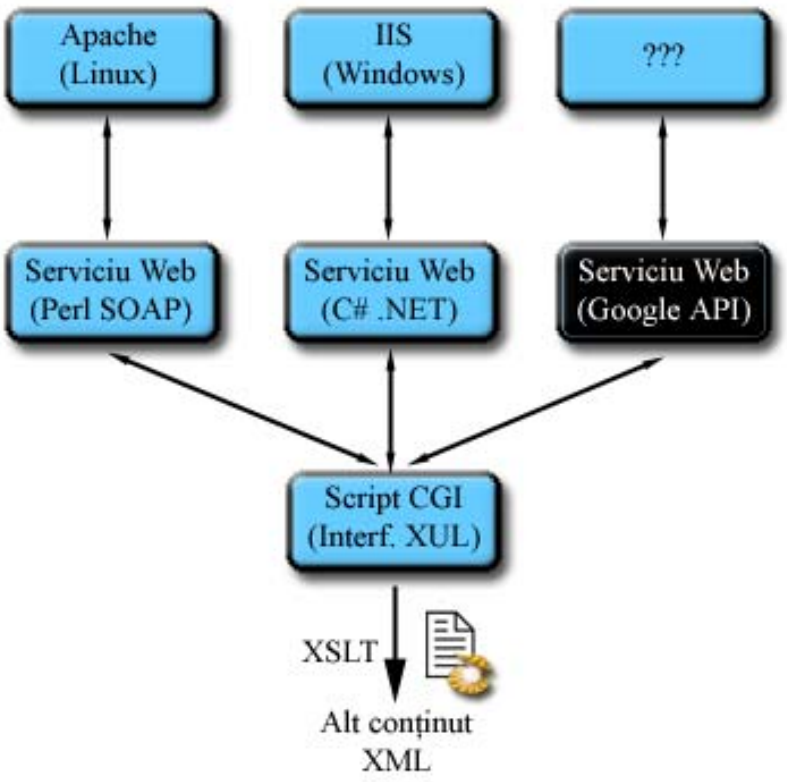

Figura 4.3: Serviciile Web locale şi externe folosite de *ITW*

Web implicate în procesul de descoperire a resurselor. Folosind diferite transformări XSL, aceste documente WQGL pot fi automat convertite în alte documente bazate pe XML, precum limbajele de marcare XHTML sau WML în vederea exploatării interfeţei în medii clasice sau fără fir (*wireless*)[13]. Transformările XSL pot fi efectuate de orice program *wrapper*, scris în oricare dintre limbajele de programare existente.

2. două servicii Web *ITW* locale, disponibile pe sistemele de operare Linux şi Windows: un serviciu Web implementat în Perl, utilizând

[13] Pentru amănunte, se pot consulta lucrările [Buraga, 2003a], [Buraga, 2003b] şi [Buraga & Brut, 2001].

modulul *SOAP::Lite* [CPAN] şi rulând pe serverul Apache şi un serviciu Web implementat în limbajul C# pe platforma .NET, folosind serverul IIS ca server Web.

3. un serviciu Web extern, furnizat de motorul de căutare Google, în vederea îmbunătăţirii procesului de descoperire a resurselor multimedia localizate pe Web.

4. un sistem de multi-agenţi compus din agenţi software, dezvoltat conform celor precizate în secţiunea 4.4.2.1.

5. un sistem de stocare a resurselor găsite, folosind un sistem *open-source* de management al bazelor de date (*e.g.*, MySQL sau PostgreSQL).

Figura 4.3 ilustrează structura serviciilor Web utilizate de sistemul *ITW* propus.

Elementele implementate au fost testate pe platformele Windows XP – folosind mediul .NET Framework 1.1 – şi Linux (distribuţiile RedHat/Fedora şi Mandrake) – utilizând Perl 5.8 şi Apache 2.0. Pentru procesarea documentelor XML, s-a utilizat modulul *XML::Parser* pentru Perl, recurgându-se la procesorul SAX *Expat*.[14] Accesarea motorului Google s-a realizat prin intermediul unei chei speciale unice oferite de proiectanţii motorului via descrierea WSDL publică a serviciului [Google].

4.4.2.5 Utilizări

Ca posibile utilizări ale sistemului *ITW* se pot enumera cele legate de activităţi de tip *e-learning* şi *e-enterprise*.

Sistemul *ITW* poate fi exploatat în cadrul unui sistem de învăţare *on-line*, circumscris domeniului *e-learning*.

Conform celor expuse în [Buraga, 2003c, Buraga, 2003e], paradigma orientată-agent poate fi utilizată cu succes în dezvoltarea unui sistem

[14] A se vedea detalii în [Buraga, 2001a] şi [Buraga *et al.*, 2002a].

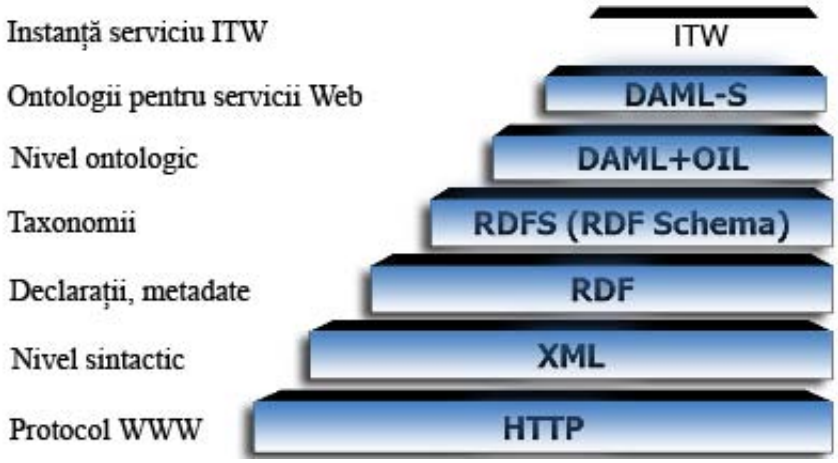

Figura 4.4: Structura pe niveluri a componentelor *ITW* implementând servicii Web semantice

complex de învăţare bazată pe Web (*Web-based learning*). Acest aspect devine cu mult mai atractiv în cazul adoptării unor standarde deschise, precum familia de limbaje XML, şi a unor tehnologii flexibile bazate pe interfeţe şi servicii Web. Demersurile de integrare de obiecte multimedia într-un astfel de sistem au fost cercetate în lucrările [Pecheanu, Ştefănescu, Buraga, Istrate, 2001], [Buraga & Cioca, 2003] şi [Ştefănescu, Pecheanu, Buraga, 2001].

De asemenea, sistemul *ITW* poate fi considerată o platformă de regăsire a informaţiilor multimedia în cadrul intranetului unei organizaţii, formând astfel un portal de întreprindere. Acest portal poate conduce la instituirea de întreprinderi virtuale, compuse din resurse umane, date şi logistică distribuite la nivel planetar. O serie de cercetări pot fi regăsite în [Cioca & Buraga, 2003a], [Cioca & Buraga, 2003b] şi [Cioca & Buraga, 2003c].

O altă direcţie pe care o avem în vedere este cea de integrare a sistemului propus în sisteme disponibile pe arie largă de tip Grid, urmând liniile descrise în [Alboaie & Buraga, 2003b].

4.5 Concluzii

Capitolul de faţă a prezentat maniera de dezvoltare a unui sistem Web distribuit utilizat la regăsirea de resurse multimedia.

Conform rezultatelor cercetărilor desfăşurate şi concretizate în lucrări precum [Buraga, 2003b], [Buraga, 2003c], [Buraga, 2003g], [Buraga & Alboaie, 2004], [Alboaie & Buraga, 2003a] şi [Buraga & Găbureanu, 2003], am descris o manieră universală, independentă de platformă, de integrare via meta-limbajul XML a mai multor componente eterogene, ca agenţi şi servicii Web, în vederea creării unei infrastructuri software situată la nivelurile de metadate şi scheme în cadrul Web-ului semantic.

Prin caracteristicile sale (adoptarea de agenţi inteligenţi, folosirea de logici temporale pentru modelarea relaţiilor stabilite între resurse, utilizarea de tehnologii Web precum XML şi RDF, recurgerea la servicii Web semantice etc.), sistemul *ITW* – detaliat în cadrul subcapitolului 4.4 – poate fi considerat ca reprezentând o contribuţie originală aliniată problematicilor Web-ului semantic.

Capitolul 5

La final

Capitolul de faţă realizează o trecere în revistă a problematicilor dezbătute pe parcursul cărţii şi punctează o serie dintre posibilele direcţii viitoare ale cercetărilor efectuate.

5.1 Privire de ansamblu

DATĂ fiind amploarea cu care se dezvoltă continuu spaţiul World-Wide Web, una dintre principalele problemele care trebuie rezolvate este cea a *regăsirii* informaţiilor atât într-o manieră cât mai facilă pentru utilizatorii finali, cât – mai ales – într-un mod standardizat de către maşină.

Tematica lucrării de faţă se înscrie în sfera manipulării resurselor multimedia disponibile în Internet, pentru aceasta prezentându-se un model original de adnotare în limbaje bazate pe XML a relaţiilor spaţio-temporale care se pot stabili între componentele sau fragmentele de situri Web, conform cercetărilor întreprinse în lucrări ca [Buraga, 2000a], [Buraga, 2002b], [Buraga, 2002c] şi [Buraga & Ciobanu, 2002].

Relaţiile spaţiale sunt specificate pornind de la maniera de stocare a documentelor într-un sistem de fişiere distribuit, prin intermediul unui

limbaj bazat pe XML, extins însă la nivelul unui Web (local). Pentru exprimarea relaţiilor temporale, la nivel formal s-a plecat de la logica ITL, specificarea relaţiilor dintre resurse realizându-se prin aserţiuni RDF, baza pentru Web-ul semantic [Berners-Lee *et al.*, 2001]. Această specificare ia în consideraţie şi posibilele metadate ce pot fi asociate resurselor Web, metadate marcate într-un limbaj bazat pe XML.

Regăsirea informaţiilor hipermedia poate fi facilitată de modul în care sunt formulate de către utilizator cererile de interogare, focalizându-se asupra specificării structurii şi tipului de conţinut al resurselor dorite a fi găsite, având în vedere că aceastea pot fi marcate în diverse limbaje bazate pe XML. Rezultatele principale sunt concretizate în lucrări precum [Buraga & Rusu, 2000], [Buraga & Brut, 2001] şi [Buraga & Brut, 2002].

De asemenea, materialul prezintă şi o soluţie de implementare, concretizată în prototipul unui sistem multi-platformă, distribuit şi eterogen – *ITW* [Buraga, 2003g, Buraga & Găbureanu, 2003] – compus în principal din agenţi software şi servicii Web bazate pe XML. În proiectarea sistemului de agenţi s-a avut în vedere formalizarea prin logici temporale ramificate a comportamentului agenţilor prin prisma paradigmei BDI. Structura conceptuală a sistemului este prezentată în [Alboaie & Buraga, 2002] şi [Buraga & Alboaie, 2004]. Comunicarea între agenţi se realizează folosind mecanisme specifice protocolului SOAP, în acest sens permiţându-se o integrare strânsă cu serviciile Web. O parte dintre cercetări s-au ocupat de adoptarea unei maniere de serializare originale, descrisă pe larg în [Alboaie & Buraga, 2003a]. Serviciile Web dezvoltate au asociate descrieri semantice exprimate de declaraţii DAML-S, în vederea regăsirii acestor servicii de alte aplicaţii aliniate problematicilor Web-ului semantic.

Interfaţa sistemului a fost realizată conform cerinţelor actuale ale interacţiunii om-calculator, detaliate în lucrările [Buraga, 2003b] şi [Buraga, 2003f].

Prin caracteristicile sale, sistemul poate fi văzut ca o componentă a Web-ului semantic, putând de asemenea fi încorporat în cadrul unui Grid, urmărind ideile din [Alboaie & Buraga, 2003b].

Aplicabilitatea sistemului se întrevede în domenii ca *e-learning* şi *e-enterprise*. Cercetările întreprinse s-au concretizat în [Buraga, 2003c],

[Buraga, 2003e], [Buraga & Cioca, 2003], [Cioca & Buraga, 2003a] şi [Cioca & Buraga, 2003b].

Părţi din materialul de faţă s-au bazat şi pe lucrările cu caracter monografic [Buraga, 2001a], [Buraga, 2002a], [Buraga, 2003a], [Buraga & Ciobanu, 2001] şi [Buraga *et al.*, 2002a].

5.2 Direcţii viitoare de cercetare

Ca viitoare direcţii de cercetare intenţionăm:

- studierea altor formalisme pentru exprimarea relaţiilor dintre resurse, folosindu-se diverse tipuri de logici temporale;
- includerea construcţiilor bazate pe RDF în cadrul altor limbaje utilizate de Web-ului semantic, în vederea generării automate de ontologii pe baza tipurilor şi structurii conţinutului resurselor multimedia, conform direcţiilor expuse de [Davies *et al.*, 2003];
- studierea în profunzime la nivel formal a modului de specificare a sistemului *ITW*, folosind diverse limbaje de specificare;
- realizarea unei interoperabilităţi mai strânse între agenţii şi serviciile Web semantice via OWL [Dean & Schreiber, 2004];
- implicarea problematicilor lucrării de faţă în cadrul direcţiilor de cercetare întreprinse în domeniilor serviciilor semantice pentru Grid, conform [Sycare & Payne, 2003, SGrid].

Anexa A

Folosirea metadatelor în contextul *e-commerce*

În cadrul acestei anexe, vom descrie modalitatea de generare a unor aserţiuni RDF privitoare la produsele unui magazin electronic. Pentru fiecare produs (identificat în mod unic printr-un număr – cheie primară în cadrul unei tabele dintr-o bază de date), se va genera o declaraţie RDF care asociază diverse metadate privitoare la numele, descrierea, preţul, categoria de produse din care face parte acel produs. De asemenea, se va construi o listă de tip *Bag* conţinând produsele înrudite cu un anumit produs.

Generarea documentului RDF se va realiza prin folosirea modelului *DOM* (*Document Object Model*) [Buraga, 2001a] în PHP.

Un exemplu de document RDF este cel prezentat în continuare:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:m="urn:www.infoiasi.ro:eshop">
  <rdf:Description rdf:about="urn:www.infoiasi.ro:34">
    <m:Descriere>Boxe active 500 W SUBWOOFER</m:Descriere>
    <m:Pret>67.9</m:Pret>
    <m:Categorie>Boxe</m:Categorie>
```

```
    <m:Inrudite>
      <rdf:Bag/>
    </m:Inrudite>
  </rdf:Description>
  <rdf:Description rdf:about="urn:www.infoiasi.ro:488">
    <m:Descriere>HDD 60 GB IBM 7200</m:Descriere>
    <m:Pret>183.1</m:Pret>
    <m:Categorie>Hard Disk</m:Categorie>
    <m:Inrudite>
      <rdf:Bag>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:489"/>
      </rdf:Bag>
    </m:Inrudite>
  </rdf:Description>
  <rdf:Description rdf:about="urn:www.infoiasi.ro:489">
    <m:Descriere>HDD 75 GB IBM 7200</m:Descriere>
    <m:Pret>252.5</m:Pret>
    <m:Categorie>Hard Disk</m:Categorie>
    <m:Inrudite>
      <rdf:Bag>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:488"/>
      </rdf:Bag>
    </m:Inrudite>
  </rdf:Description>
  <rdf:Description rdf:about="urn:www.infoiasi.ro:86">
    <m:Descriere>CPU PENTIUM III 1000EB</m:Descriere>
    <m:Pret>200.4</m:Pret>
    <m:Categorie>Procesoare</m:Categorie>
    <m:Inrudite>
      <rdf:Bag>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:81"/>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:82"/>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:83"/>
```

```
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:84"/>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:7320"/>
      </rdf:Bag>
    </m:Inrudite>
  </rdf:Description>
  <rdf:Description rdf:about="urn:www.infoiasi.ro:81">
    <m:Descriere>CPU PENTIUM III CELERON 700</m:Descriere>
    <m:Pret>49.8</m:Pret>
    <m:Categorie>Procesoare</m:Categorie>
    <m:Inrudite>
      <rdf:Bag>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:86"/>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:82"/>
        <rdf:li
           rdf:resource="urn:www.infoiasi.ro:83"/>
      </rdf:Bag>
    </m:Inrudite>
  </rdf:Description>
</rdf:RDF>
```

Spaţiul de nume *m* desemnează construcţiile sintactice ale limbajului XML folosit pentru a exprima metadatele asociate fiecărui produs în parte.

Programul PHP care generează un astfel de document RDF este prezentat mai jos[1]:

```
<?php
// definitii ale claselor utilizate
// pentru acces la bazele de date
include("def.php");
```

[1]Principalul autor al programelor PHP exemplificate în cadrul acestei anexe este Florin Bandas, viitor absolvent al Facultăţii de Informatică a Universităţii "Al. I. Cuza" din Iaşi.

```
$db = new DataBase_Shop;
// conectarea la baza de date
$db->connect();
// selectam toate produsele
$sql = "select * from products";
$db->query($sql);

// cream documentul RDF
// folosim DOM
$newdoc = domxml_new_doc("1.0");
$node = $newdoc->create_element("rdf:RDF");
// specificam spatiile de nume utilizate
$node->set_attribute("xmlns:rdf",
  "http://www.w3.org/1999/02/22-rdf-syntax-ns#");
$node->set_attribute("xmlns:m",
  "urn:www.infoiasi.ro:eshop");
$newnode = $newdoc->append_child($node);

// parcurgem inregistrarile returnate
while( $db->next_record() )
{
        $prod_id = $db->Record["id"];
        $short_d = $db->Record["descr"];
        $price = $db->Record["pret"];
$category = $db->Record["categorie"];

        // pentru fiecare produs,
        // cream o asertiune RDF
$description = $newdoc->create_element("rdf:Description");
$description->set_attribute("rdf:about",
           "urn:www.infoiasi.ro:" . $prod_id);
$description = $newnode->append_child($description);

$descr = $newdoc->create_element("m:Descriere");
$descr = $description->append_child($descr);
$text = $newdoc->create_text_node($short_d);
$text = $descr->append_child($text);
```

```
        $pret = $newdoc->create_element("m:Pret");
        $pret = $description->append_child($pret);
        $text = $newdoc->create_text_node($price);
        $text = $pret->append_child($text);

        $categ = $newdoc->create_element("m:Categorie");
        $categ = $description->append_child($categ);
        $text = $newdoc->create_text_node($category);
        $text = $categ->append_child($text);

        $related = $newdoc->create_element("m:Inrudite");
        $related = $description->append_child($related);

$bag = $newdoc->create_element("rdf:Bag");
        $bag = $related->append_child($bag);

        // preluam lista produselor inrudite
        // si o includem intr-un <rdf:Bag>
$db1 = new DataBase_Shop;
$db1->connect();
$sql1 = "select * from products
           where categorie=\"" . $category . "\"
           and id <> " . $prod_id;
        $db1->query($sql1);

while($db1->next_record())
{
$related_id = $db1->Record["id"];
        $li = $newdoc->create_element("rdf:li");
$li->set_attribute("rdf:resource",
           "urn:www.infoiasi.ro:" . $related_id);
        $li = $bag->append_child($li);
}
}

// salvam fisierul XML generat
$newdoc->dump_file("rdf.xml", false, true);
?>
```

Folosind aserţiunile RDF privitoare la produsele magazinului electronic, putem acum implementa un motor de căutare care va furniza atât detalii referitoare la produsul găsit, cât şi informaţii despre produsele înrudite. Ulterior, relaţia de înrudire poate fi construită pornind de la diverse aspecte semantice regăsite între anumite produse (de exemplu, se poate folosi o ontologie care să exprime conceptele utilizate în domeniul *e-business*).

Codul-sursă al programului PHP care reprezintă motorul de căutare este:

```
<html>
<head><title>Cautare</title></head>
<body>
<?php
// daca nu au fost furnizate date,
// atunci se genereaza formularul XHTML
if ($postback <> 1)
{
?>
<form action="search.php" method="POST">
<input type="hidden" name="postback" value="1" />
Produsul cautat: <input type="text" name="query" />
<input type="submit" value="Cauta" />
</form>
<br />
Cautarea se realizeaza in descrierea produsului.
<br />
Cautati, de exemplu "HDD", "SDRAM", "boxe", "cpu" etc.
<?php
}
else
{
// folosim documentul RDF generat anterior
// procesarea se va realiza prin DOM
$reference = domxml_open_file("rdf.xml");
$para = $reference->get_elements_by_tagname("*");
$count = 0;
foreach($para as $par) {
```

```
// cautam in continutul elementului <m:Descriere>
if ($par->tagname == "Descriere") {
  $textul = $par->get_content();
  if (strpos(strtolower($textul), strtolower($query))
    !== false) {
    //cautam in nodul parinte ca sa aflam ID-ul
    $parent = $par->parent_node();
    $childs = $parent->child_nodes();

    foreach($childs as $child) {
       if ($child->tagname != "" &&
           $child->tagname != "Inrudite")
          echo "<b>" . $child->tagname . ":</b> " .
              $child->get_content() . "<br />";
       // aflam produsele inrudite
       if ($child->tagname == "Inrudite") {
         $rel = $child->child_nodes();
         foreach($rel as $bag) {
           if ($bag->tagname == "Bag")
             if ($bag->has_child_nodes()) {
               $related = $bag->child_nodes();
               echo "<b>Va mai recomandam:</b><br />";
               foreach($related as $li) {
               $para1 =
                 $reference->get_elements_by_tagname("*");
               foreach($para1 as $par1) {
                 if ($par1->tagname == "Description")
                   if (($li->tagname == "li") &&
                     $par1->get_attribute("about") ==
                     $li->get_attribute("resource")) {
                     $par1_child = $par1->child_nodes();
                     foreach($par1_child as $child) {
                       if ($child->tagname == "Descriere")
                         echo $child->get_content() .
                           "<br />";
                     }
                   }
                }
```

```
                }
            }
        }
    }
  }

  // construim legatura spre produs,
  // pentru a accesa magazinul virtual
  $a = $parent->get_attribute("about");
  // extragem doar ID-ul
  $id_ul = substr($a, 20);
  $url = "http://students.infoiasi.ro/~wiz/e-shop/
     scripts/product_info_1.php?products_id=" . $id_ul;
  echo "pentru mai multe detalii: <a href=" .
        $url . " target=\"_blank\">click aici</a>";

  $count++; // nr. de rezultate
  echo "<hr />";
  }
}
}

// afisam cite produse au fost gasite
if ($count > 0)
  echo "\n<h5>Au fost gasite $count inregistrari</h5>";
else
  echo "\n<h5>Ne pare rau,
        nu a fost gasit nici un produs</h5>";
}
?>
</body>
</html>
```

Detalii privitoare la serverul de aplicaţii şi limbajul PHP pot fi consultate în [Buraga, 2001a] şi [Buraga, 2003a].

Anexa B

Schema pentru limbajul *XFiles*

Prezentăm mai jos schema XML pentru limbajul *XFiles* detaliat în cadrul secţiunii 3.2.5.1 a capitolului 3.[1]

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

  <xs:annotation>
     <xs:documentation xml:lang="ro">
        Schema XML pentru limbajul XFiles.
        Copyright (c)2001-2004
        Sabin-Corneliu Buraga - busaco@infoiasi.ro
     </xs:documentation>
  </xs:annotation>

  <!-- tipul fisierului/resursei -->
  <xs:element name="Type">
     <xs:complexType>
        <xs:simpleContent>
           <xs:restriction base="xs:string">
              <xs:attribute name="mime"
```

[1]O variantă preliminară a schemei a fost prezentată iniţial în [Buraga, 2000a].

```
                              type="xs:string"
                              use="optional" />
            </xs:restriction>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>

   <!-- locatia fisierului/resursei -->
   <xs:element name="Location">
      <xs:complexType>
         <xs:simpleContent>
            <xs:restriction base="xs:string">
               <xs:attribute name="dns"
                              type="xs:string"
                              use="optional" />
               <xs:attribute name="port"
                              type="xs:unsignedInt"
                              use="optional" />
            </xs:restriction>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>

   <!-- metoda de autentificare -->
   <xs:element name="Auth" type="xs:string" />

   <!-- numele de cont (login)
        al posesorului resursei -->
   <xs:element name="Login">
      <xs:complexType>
         <xs:simpleContent>
            <xs:restriction base="xs:string">
               <xs:attribute name="uid"
                              type="xs:unsignedInt"
                              use="optional" />
            </xs:restriction>
         </xs:simpleContent>
      </xs:complexType>
```

```
</xs:element>

<!-- grupul de utilizatori
     din care face parte posesorul resursei -->
<xs:element name="Group">
   <xs:complexType>
      <xs:simpleContent>
         <xs:restriction base="xs:string">
            <xs:attribute name="gid"
                          type="xs:unsignedInt"
                          use="required" />
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<!-- parola folosita la autentificare -->
<xs:element name="Password">
   <xs:complexType>
      <xs:simpleContent>
         <xs:restriction base="xs:token">
            <!-- metoda de criptare folosita -->
            <xs:attribute name="method"
                          type="xs:string"
                          use="optional" />
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<!-- numele real al posesorului fisierului/resursei -->
<xs:element name="Name" type="xs:string" />

<!-- informatii privitoare la posesor -->
<xs:element name="Owner">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="Login"
```

```
                    minOccurs="1"
                    maxOccurs="unbounded" />
         <xs:element ref="Group"
                    minOccurs="1"
                    maxOccurs="unbounded" />
         <xs:element ref="Password"
                    minOccurs="0"
                    maxOccurs="1" />
         <xs:element ref="Name"
                    minOccurs="0"
                    maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>
</xs:element>

<!-- dimensiunea fisierului (resursei) -->
<xs:element name="Size">
   <xs:complexType>
      <xs:simpleContent>
         <xs:restriction base="xs:positiveInteger">
            <xs:attribute name="max"
                          type="xs:positiveInteger"
                          use="optional" />
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<!-- tip al permisiunii (al dreptului de acces) -->
<xs:element name="Permission">
   <xs:complexType>
      <xs:simpleContent>
         <xs:restriction base="xs:string">
            <!-- enumerare a valorilor pentru
                 permisiunile care se pastreaza -->
            <xs:attribute name="preserved"
                          use="optional">
               <xs:simpleType>
```

```
                  <xs:restriction base="xs:NMTOKEN">
                     <xs:enumeration value="on" />
                     <xs:enumeration value="off" />
                  </xs:restriction>
               </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inherited"
                          use="optional">
            <!-- enumerare a valorilor pentru
                 permisiunile care se mostenesc -->
               <xs:simpleType>
                  <xs:restriction base="xs:NMTOKEN">
                     <xs:enumeration value="on" />
                     <xs:enumeration value="off" />
                  </xs:restriction>
               </xs:simpleType>
            </xs:attribute>
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<!-- permisiuni asociate fisierului/resursei -->
<xs:element name="Permissions">
   <xs:complexType>
      <xs:all>
         <xs:element ref="Permission"
                     minOccurs="0"
                     maxOccurs="unbounded" />
      </xs:all>
   </xs:complexType>
</xs:element>

<!-- data si timpul asociate fisierului -->
<xs:element name="Timestamp">
   <xs:simpleContent>
      <xs:restriction base="xs:time">
         <xs:attribute name="type" use="required">
```

```
            <!-- enumerare a valorilor
                 timpilor de acces, de modificare
                 si de schimbare de stare -->
            <xs:simpleType>
               <xs:restriction base="xs:NMTOKEN">
                  <xs:enumeration
                        value="access" />
                  <xs:enumeration
                        value="modification" />
                  <xs:enumeration
                        value="change" />
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:restriction>
   </xs:simpleContent>
</xs:element>

<!-- versiunea fisierului (utilizata in medii
     de control al versiunilor: CVS) -->
<xs:element name="Version">
   <!-- poate include orice constructie RDF -->
   <xs:complexType>
      <xs:any
         namespace=
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         minOccurs="0" maxOccurs="unbounded" />
      <xs:anyAttribute
         namespace=
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
   </xs:complexType>
</xs:element>

<!-- aplicatie asociata (e.g. procesor, vizualizator) -->
<xs:element name="Parser">
   <xs:simpleContent>
      <xs:restriction base="xs:string">
         <xs:attribute name="params" type="xs:string"
```

```
                        use="optional" />
      </xs:restriction>
   </xs:simpleContent>
</xs:element>

<!-- proprietati ale fisierului/resursei (metadate)
     "Properties" reprezinta elementul radacina
     al documentului -->
<xs:element name="Properties">
   <xs:complexType>
      <xs:all>
         <xs:element ref="Type"
                     minOccurs="0"
                     maxOccurs="1" />
         <xs:element ref="Location"
                     minOccurs="0"
                     maxOccurs="1" />
         <xs:element ref="Auth"
                     minOccurs="0"
                     maxOccurs="unbounded" />
         <xs:element ref="Owner"
                     minOccurs="0"
                     maxOccurs="unbounded" />
         <xs:element ref="Size"
                     minOccurs="0"
                     maxOccurs="1" />
         <xs:element ref="Permissions"
                     minOccurs="0"
                     maxOccurs="1" />
         <xs:element ref="Timestamp"
                     minOccurs="0"
                     maxOccurs="unbounded" />
         <xs:element ref="Version"
                     minOccurs="0"
                     maxOccurs="unbounded" />
         <xs:element ref="Parser"
                     minOccurs="0"
                     maxOccurs="unbounded" />
```

```
        </xs:all>
      </xs:complexType>
  </xs:element>
</xs:schema>
```

Anexa C

Schema pentru limbajul TRSL

Această anexă descrie schema XML pentru *TRSL* (*Temporal Relation Specification Language*), limbaj definit în cadrul secţiunii 3.2.7.1 a capitolului 3.[1]

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">
  <xs:annotation>
     <xs:documentation xml:lang="ro">
        Schema XML pentru limbajul TRSL.
        Copyright (c)2002-2003
        Sabin-Corneliu Buraga - busaco@infoiasi.ro
     </xs:documentation>
  </xs:annotation>

  <!-- grup de atribute "begin", "end" si "dur" -->
  <xs:attributeGroup name="TempAttr">
     <xs:attribute name="begin" type="xs:time"
                   use="optional" />
     <xs:attribute name="end" type="xs:time"
```

[1] O variantă preliminară a schemei XML a fost specificată iniţial în lucrarea [Buraga & Ciobanu, 2002].

```
                  use="optional" />
   <xs:attribute name="dur" type="xs:timeDuration"
                  use="optional" />
</xs:attributeGroup>

<!-- tip complex modeland relatiile temporale -->
<xs:complexType name="TempRel">
   <!-- fiecare element de acest tip poate include
        orice alte constructii (e.g., "Action") -->
   <xs:all>
      <xs:element name="Before">
         <any minOccurs="0" maxOccurs="unbounded" />
         <xs:attributeGroup ref="TempAttr" />
      </xs:element>
      <xs:element name="Starts">
         <any minOccurs="0" maxOccurs="unbounded" />
         <xs:attributeGroup ref="TempAttr" />
      </xs:element>
      <xs:element name="Overlaps">
         <any minOccurs="0" maxOccurs="unbounded" />
         <xs:attributeGroup ref="TempAttr" />
      </xs:element>
      <xs:element name="During">
         <any minOccurs="0" maxOccurs="unbounded" />
         <xs:attributeGroup ref="TempAttr" />
      </xs:element>
      <xs:element name="Finishes">
         <any minOccurs="0" maxOccurs="unbounded" />
         <xs:attributeGroup ref="TempAttr" />
      </xs:element>
      <xs:element name="Meets">
         <any minOccurs="0" maxOccurs="unbounded" />
         <xs:attributeGroup ref="TempAttr" />
      </xs:element>
   </xs:all>
</xs:complexType>

<!-- declararea elementului "link" -->
```

```
<xs:element name="link">
   <xs:complexType>
      <xs:simpleContent>
         <xs:restriction base="xs:string">
            <xs:attributeGroup ref="TempAttr" />
            <xs:attribute name="action"
                          type="xs:string"
                          use="optional" />
            <xs:attribute name="type" use="required">
               <!-- enumerare a tipurilor de legaturi -->
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="spatial" />
                     <xs:enumeration value="temporal" />
                     <xs:enumeration value="any" />
                     <!-- eventual si altele, definite
                          printr-o schema derivata -->
                  </xs:restriction>
               </xs:simpleType>
            </xs:attribute>
         </xs:restriction>
      </xs:simpleContent>
      <!-- poate include si operatori temporali -->
      <xs:element ref="TempRel"
                  minOccurs="0" maxOccurs="unbounded" />
   </xs:complexType>
</xs:element>

<!-- declararea elementului "action" -->
<xs:element name="action">
   <xs:complexType>
      <xs:simpleContent>
         <xs:restriction base="xs:string">
            <xs:attributeGroup ref="TempAttr" />
         </xs:restriction>
      </xs:simpleContent>
      <!-- permitem inserarea de constructii RDF -->
      <xs:any namespace=
```

```
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                minOccurs="0" maxOccurs="unbounded" />
          <xs:anyAttribute namespace=
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          />
       </xs:complexType>
    </xs:element>

    <!-- definirea elementului radacina al documentului -->
    <xs:element name="Properties">
       <xs:complexType>
          <xs:all>
             <xs:element ref="TempRel"
                         minOccurs="0"
                         maxOccurs="unbounded" />
             <xs:element ref="link"
                         minOccurs="0"
                         maxOccurs="unbounded" />
             <xs:element ref="action"
                         minOccurs="0"
                         maxOccurs="unbounded" />
          </xs:all>
       </xs:complexType>
    </xs:element>
</xs:schema>
```

Anexa D

Schema pentru limbajul WQFL

Anexa de faţă pune la dispoziţie schema XML utilizată pentru specificarea formală a gramaticii limbajului *WQFL* (*Web Query Formulating Language*). Limbajul WQFL [1] a fost definit în cadrul secţiunii 3.3.3 a capitolului 3.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">
  <xs:annotation>
     <xs:documentation xml:lang="ro">
        Schema XML pentru limbajul WQFL.
        Copyright (c)2000-2004
        Sabin-Corneliu Buraga - busaco@infoiasi.ro
     </xs:documentation>
  </xs:annotation>

  <!-- specificarea motorului de cautare -->
  <xs:element name="engine">
     <xs:complexType>
        <xs:simpleContent>
```

[1] O serie de versiuni preliminare ale WQFL (în varianta DTD) au fost specificate în [Buraga & Rusu, 2000], [Buraga & Brut, 2001] şi [Buraga & Brut, 2002].

```
         <xs:restriction base="xs:string">
            <xs:attribute name="url"
                          type="xs:uriReference" />
            <xs:attribute name="params"
                          type="xs:string"
                          use="optional" />
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<!-- specificarea interogarii textuale -->
<xs:element name="query">
   <xs:complexType>
      <xs:simpleContent>
         <xs:restriction base="xs:string">
            <xs:attribute name="regexp">
               <!-- se permit doar valorile
                    "yes" si "no" -->
               <xs:simpleType>
                  <xs:restriction base="string">
                      <xs:enumeration value="yes">
                      </xs:enumeration>
                      <xs:enumeration value="no">
                      </xs:enumeration>
                  </xs:restriction>
               </xs:simpleType>
            </xs:attribute>
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<!-- specificarea locului si numarului de aparitii -->
<xs:element name="occur">
   <xs:simpleContent>
      <xs:restriction base="xs:string">
         <xs:attribute name="top"
```

```
                        type="xs:positiveInteger"
                        use="optional" />
            <xs:attribute name="bottom"
                        type="xs:positiveInteger"
                        use="optional" />
        </xs:restriction>
    </xs:simpleContent>
</xs:element>

<!-- specificarea unui element al structurii cautate -->
<xs:element name="element">
    <xs:complexType>
        <xs:sequence>
            <xs:restriction base="xs:string">
                <xs:element ref="occurs"
                            minOccurs="1"
                            maxOccurs="unbounded" />
                <xs:attribute name="name" type="xs:string" />
                <xs:attribute name="appear">
                    <xs:simpleType>
                        <!-- permitem doar "yes"/"no" -->
                        <xs:restriction base="string">
                            <xs:enumeration value="yes">
                            </xs:enumeration>
                            <xs:enumeration value="no">
                            </xs:enumeration>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="order"
                              type="xs:positiveInteger"
                              use="optional" />
            </xs:restriction>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<!-- specificarea structurii documentului -->
```

```
<xs:element name="structure">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="element"
                     minOccurs="0"
                     maxOccurs="unbounded" />
      </xs:sequence>
   </xs:complexType>
</xs:element>

<!-- specificarea de meta-informatii -->
<xs:element name="meta">
   <xs:simpleContent>
      <xs:restriction base="xs:string" />
   </xs:simpleContent>
</xs:element>

<!-- specificarea continutului propriu-zis
     al resursei gasite -->
<xs:element name="page">
   <xs:complexType>
      <xs:any minOccurs="0" maxOccurs="unbounded" />
      <xs:anyAttribute />
   </xs:complexType>
</xs:element>

<!-- specificarea continutului -->
<xs:element name="content">
   <xs:complexType>
      <xs:all>
         <!-- permitem inserarea de asertiuni RDF -->
         <xs:any namespace=
           "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           minOccurs="0" maxOccurs="unbounded" />
         <xs:anyAttribute namespace=
           "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         />
         <xs:element ref="page"
```

```
                        minOccurs="0"
                        maxOccurs="1" />
            <xs:element ref="meta"
                        minOccurs="0"
                        maxOccurs="unbounded" />
         </xs:all>
      </xs:complexType>
   </xs:element>

   <!-- specificarea elementului radacina -->
   <xs:element name="webquery">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="engine"
                        maxOccurs="1"
                        maxOccurs="unbounded" />
            <xs:element ref="query"
                        maxOccurs="1"
                        maxOccurs="1" />
            <xs:element ref="structure"
                        maxOccurs="0"
                        maxOccurs="1" />
            <xs:element ref="content"
                        maxOccurs="0"
                        maxOccurs="1" />
         </xs:sequence>
         <xs:attribute name="timestamp"
                       type="xs:timePeriod"
                       use="optional" />
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Anexa E

Acronime

Următoarele acronime au fost folosite pentru termenii utilizaţi în lucrarea de faţă:

ACL – Agent Communication Language
API – Application Programmer Interface
ASCII – American Standard Code for Information Interchange
ASN.1 – Abstract Standard Notation One
BASH – Bourne Again SHell
BDI – Belief-Desire-Intention
BNF – Bachus-Naur Form
CERN – Centre Européen pour la Recherche Nucléaire
CGI – Common Gateway Interface
CORBA – Common Object Request Broker Architecture
CSS – Cascading Style Sheet
CTL – Computational Tree Logics
CVS – Concurrent Versions System
DAML – DARPA Agent Markup Language
DAML-S – DAML for Services
DB – Data Base
DCMI – Dublin Core Metadata Initiative
DCOM – Distributed Component Object Model
DHTML – Dynamic HTML
DNS – Domain Name System
DOM – Document Object Model
DTD – Document Type Definition
EDI – Electronic Document Integration
EOR – Extensible Open RDF toolkit
EPS – Encapsulated PostScript
FIPA – Foundation of Intelligent Physical Agents
FS – File System

FTP – File Transfer Protocol
GIF – Graphics Interchange Format
GNU – Gnu's Not Unix
HPFS – High Performance File System
HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol
IEEE – Institute of Electrical and Electronic Engineers
IETF – Internet Engineering Task Force
IP – Internet Protocol
ISOC – Internet SOCiety
ITL – Interval Temporal Logic
JPEG – Joint Picture Experts Group
JSP – Java Server Pages
KIF – Knowledge Interchange Format
KQML – Knowledge Query Manipulation Language
LAN – Local Area Network
MIME – Multipurpose Internet Mail Extensions
MPEG – Moving Pictures Experts Group
MAS – Multi-Agent System
NCSA – National Center for Supercomputing Applications
NFS – Network File System
NIS – Network Information Service
NS – NameSpace
OS – Operating System
OIL – Ontology Inference Layer
OWL – Web Ontology Language
P2P – Peer-to-Peer
PDF – Portable Document Format
PHP – PHP: Hypertext Processor
PNG – Portable Network Graphics
PPP – Point-to-Point Protocol
PS – PostScript
RAP – RDF API for PHP
RDF – Resource Description Framework
RDFS – RDF Schema
PTL – Propositional Time Logic
RFC – Request For Comments
RPC – Remote Procedure Call
RSS – RDF Site Summary
RTF – Rich Text Format
SAX – Simple API for XML
SGML – Standard Generalized Markup Language
SLP – Service Location Protocol
SMIL – Synchronized Multimedia Integration Language
SOAP – Simple Object Access Protocol
SQL – Structured Query Language
SSDP – Simple Service Discovery Protocol
SVG – Scalable Vector Graphics
SW – Semantic Web
SWS – Semantic Web Services
TCP – Transmission Control Protocol
TDL – Temporal Description Logic
TEI – Text Encoding Initiative
TRSL – Temporal Relation Specification Language
TLA – Temporal Logic of Actions
UDDI – Universal Description, Discovery, and Integration
UDP – User Datagram Protocol
UML – Unified Modeling Language
URI – Uniform Resource Identifier
URL – Uniform Resource Locator

URN – Uniform Resource Name
VRML – Virtual Reality Modeling Language
W3C – World-Wide Web Consortium
WAP – Wireless Application Protocol
WML – Wireless Markup Language
WS – Web Service
WSDL – Web Services Description Language
WQFL – Web Query Formulating Language
WQGL – Web Query Graphical Language
WWW – World-Wide Web
X3D – Extensible Three Dimensions
XHTML – Extensible HTML
XOL – XML Ontology Language
XML – Extensible Markup Language
XSD – XML Schema Definition
XSL – Extensible Stylesheet Language
XSLT – XSL Transformation
XUL – Extensible User-interface Language
XUP – Extensible User-interface Protocol

Bibliografie

[Acostăchioaie & Buraga, 2004] D. Acostăchioaie, S. C. Buraga, *Utilizarea sistemului Linux*, Polirom, Iaşi, 2004: `http://www.infoiasi.ro/~linux/`

[Adler *et al.*, 2001] S. Adler *et al.*, *Extensible Stylesheet Language (XSL) Version 1.0*, W3C Recommendation, Boston, 2001: `http://www.w3.org/TR/xsl/`

[Alboaie & Buraga, 2002] S. Alboaie, S. C. Buraga, L. Alboaie, "An XML-based Object-Oriented Infrastructure for Developing Software Agents", Scientific Annals of the "A. I. Cuza" University of Iaşi, Computer Science Series, tome XII, 2002, pag. 109–134

[Alboaie & Buraga, 2003a] S. Alboaie, S. C. Buraga, L. Alboaie, "An XML-based Serialization of Information Exchanged by Software Agents", *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics – SCI 2003*, Orlando, USA, 2003

[Alboaie & Buraga, 2003b] L. Alboaie, S. C. Buraga, S. Alboaie, "*tuBiG* – A Layered Infrastructure to Provide Support for Grid Functionalities", în M. Paprzycki (ed.), *Proceedings of the 2nd International Symposium on Parallel and Distributed Computing*, IEEE Computer Society Press, 2003 (în curs de apariţie)

[Allen, 1991] J. Allen, "Time and Time Again: The Many Ways to Represent Time", *International Journal of Intelligent Systems*, 6 (4), 1991

[Allen, 1983] J. Allen, "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, 26 (11), 1983

[Allen & Hayes, 1989] J. Allen, P. Hayes, "Moments and Points in an Interval-based Temporal Logic", *Computational Intelligence*, 5 (4), 1989

[Allen & Ferguson, 1997] J. Allen, G. Ferguson, "Actions and Events in Interval Temporal Logic", în O. Stock (ed.), *Spatial and Temporal Reasoning*, Kluwer Academic Publishers, 1997

[Alur & Henzinger, 1992] R. Alur, T. A. Henzinger, "Logics and Model of Real Time: a Survey", în J. W. de Bakker *et al.* (eds.), *Real-Time: Theory in Practice*, Lecture Notes in Computer Science – LNCS 600, Sprinver-Verlag, Berlin, 1992

[Alur & Henzinger, 1994] R. Alur, T. A. Henzinger, "A Really Temporal Logic", *The Journal of the ACM*, 41, 1994

[Ancona & Mascardi, 2003] D. Ancona, V. Mascardi, "Coo-BDI: Extending the BDI Model with Cooperativity", Technical Report, DISI – Universita di Geneva, 2003:
`ftp://ftp.disi.unige.it/pub/person/AnconaD/BCDI.ps.gz`

[Anutaryia *et al.*, 2001] C. Anutaryia *et al.*, *RDF Declarative Descriptions with Sets and Negation*, Technical Report, Computer Science and Information Management Program, Asian Institute of Technology, Thailand, 2001

[Artale & Franconi, 1999] A. Artale, E. Franconi, "Temporal Description Logics", în L. Vile *et al.* (eds.), *Handbook of Time and Temporal Reasoning in Artificial Intelligence*, MIT Press, 1999

[Ayars *et al.*, 2001] J. Ayars *et al.* (eds.), *Synchronized Multimedia Integration Language (SMIL 2.0)*, W3C Recommendation, Boston, 2001:
`http://www.w3.org/TR/smil20`

[Baader *et al.*, 2002] F. Baader *et al.* (eds.), *The Description Logic Handbook*, Cambridge University Press, 2002

[Baclawski *et al.*, 2002] K. Baclawski *et al.*, "Consistency Checking of Semantic Web Ontologies", în I. Horrocks, J. Hendler (eds.), *International Semantic Web Conference – ISWC 2002*, Lecture Notes in Computer Science – LNCS 2342, Springer-Verlag, 2002

[Bailey *et al.*, 2002] J. Bailey et al., "An Event-Condition-Action Language for XML", *Proceedings of the World-Wide Web 2002 Conference*, Honolulu, ACM Press, 2002

[Bakken, 2001] D. Bakken, "Middleware", în *Encyclopedia of Distributed Computing*, Kluwer Academic Press, 2001

[Balasubramanian, 1994] V. Balasubramanian, *State of the Art Review on Hypermedia Issues and Applications,* PhD Thesis, Rutgers University, New Jersey, 1994

[Balazinska *et al.*, 2002] M. Balazinska *et al.*, "INS/Twine: a Scalable Peer-to-Peer Architecture for Intentional Resource Discovery", *Pervasive 2002 – International Conference on Pervasive Computing*, Zurich, Switzerland, August 2002, Springer-Verlag, 2002

[Baumgartner & Furbach, 2002] P. Baumgartner, U. Furbach, "Model-Based Deduction for Knowledge Representation", *Semantic Web Workshop*, Hawaii, 2002

[Beckett, 2001] D. Beckett, "The Design and Implementation of the Redland RDF Application Framework", *Proceedings of the 10th World-Wide Web Conference*, Hong Kong, ACM Press, 2001

[Beckett, 2002] D. Beckett, *RDN-WSE – Web Crawling High-Quality Metadata using RDF and Dublin Core*, Tehnical Report, Institute for Learning and Research Technology, University of Bristol, 2002

[Beckett, 2004] D. Beckett (ed.), *RDF/XML Syntax Specification (Revised)*, W3C Recommendation, Boston, 2004:
`http://www.w3.org/TR/rdf-syntax-grammar`

[Berners-Lee, 1989] T. Berners-Lee, *Information Management: A Proposal*, CERN, 1989:
`http://www.w3.org/History/1989/proposal.html`

[Berners-Lee, 1999] T. Berners-Lee, *Weaving the Web*, Orion, Cambridge, 1999

[Berners-Lee, 2000] T. Berners-Lee, "Layers of the Semantic Web", *XML 2000 Invited Talk*, 2000:
`http://www.w3.org/2000/talks/`
`1206-xml2k-tbl/slide1-0.html`

[Berners-Lee, 2002] T. Berners-Lee, *The World-Wide Web – Past, Present and Future*, Japan Prize Speech, 2002

[Berners-Lee *et al.*, 2001] T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web", *Scientific American*, 5, 2001

[Berners-Lee *et al.*, 1998] T. Berners-Lee *et al.* (eds.), *Uniform Resource Identifiers (URI): General Syntax*, RFC 2396, IETF, 1998

[Benatallah *et al.*, 2003] B. Benatallah *et al.*, "The Self-Serv Environment for Web Services Composition", *IEEE Internet Computing*, January-February 2003

[Boley, 2000] H. Boley, "Markup Languages for Functional-Logic Programming", *Proc. 9th WFLP 2000*, Benicassim, Spain, 2000

[Borenstein & Freed, 1992] N. Borenstein, N. Freed, *MIME (Multipurpose Internet Mail Extensions)*, RFC 1341, IETF, 1992: `http://www.ietf.org/rfc/rfc1341.txt`

[Bos *et al.*, 1998] B. Bos *et al.*, *Cascading Style Sheets (CSS), level 2*, W3C Recommendation, Boston, 1998: `http://www.w3.org/TR/REC-CSS2/`

[Botafogo *et al.*, 1991] R. Botafogo *et al.*, "Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics", *ACM Transactions on Information Systems*, 1991

[Bouquet *et al.*, 2001] P. Bouquet *et al.*, *Theories and Uses of Context in Knowledge Representation and Reasoning*, Technical Report 0110-28, Centro per la ricerca scientifica e technologica, Trento, Italy, 2001

[Bozsak *et al.*, 2002] E. Bozsak *et al.*, *KAON – Towards a large scale Semantic Web*, Technical Report, Institute AIFB, University of Karlsruhe, 2002

[Bradshow, 1997] J. Bradshow (ed.), *Software Agents*, MIT Press, Cambridge, MA, 1997

[Bray *et al.*, 2004] T. Bray *et al.* (eds.), *Extensible Markup Language (XML) – version 1.0 (Third Edition)*, W3C Recommendation, Boston, 2004: `http://www.w3.org/TR/REC-xml`

[Bray *et al.*, 1999] T. Bray, D. Hollander, A. Layman (eds.), *Namespaces in XML*, W3C Recommendation, Boston, 1999: `http://www.w3.org/TR/REC-xml-names`

[Brickley & Guha, 2000] D. Brickley, R. V. Guha (eds.), *Resource Description Framework (RDF) Schema Specification*, W3C Candidate Recommendation, Boston, 2000:
http://www.w3.org/TR/rdf-schema

[Brin & Page, 1998] S. Brin, L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", *WWW7 Conference Proceedings*, Elsevier Science, 1998

[Broekstra *et al.*, 2001] D. Broekstra *et al.*, "Enabling Knowledge Representation on the Web by extending the RDF Schema", *Proceedings of 10th International WWW Conference*, ACM Press, 2001

[Broekstra *et al.*, 2002] D. Broekstra *et al.*, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema", în I. Horrocks, J. Hendler (eds.), *International Semantic Web Conference – ISWC 2002*, Lecture Notes in Computer Science – LNCS 2342, Springer-Verlag, 2002

[Broersen *et al*, 2001] J. Broersen, "The BOID Architecture: Conflicts between Beliefs, Obligations, Intentions and Desires", *Proceedings of the 5th International Conference on Autonomous Agents*, ACM Press, 2001

[Brut & Buraga, 2004] M. Brut, S. C. Buraga, *Prezentări multimedia pe Web*, Polirom, Iaşi, 2004:
http://www.infoiasi.ro/~multimedia/

[Butler, 2000] D. Butler, "Souped-up Search Engines", *Nature*, vol. 405, no. 5, 2000

[Buraga, 1998] S. C. Buraga, "GAEN – An Advanced Concurrent Teleconferencing System", în D. Grigoraş (ed.), *Proceedings of the 6th International Symposium on Automatic Control and Computer Science – SACCS'98*, vol. II, Matrix Rom, Bucureşti, 1998, pag. 171–175

[Buraga, 2000a] S. C. Buraga, "A RDF Description of Distributed File Systems". *Scientific Annals of the "Al. I. Cuza" University of Iaşi*, Computer Science Series, tome IX, 2000, pag. 27–44

[Buraga, 2000b] S. C. Buraga, "Reprezentarea sistemelor Lindenmayer ca documente XML", *Conferinţa de Informatică Teoretică şi Tehnologii Informatice – CITTI 2000*, Leda & Muntenia, Constanţa, 2000, pag. 192–198

[Buraga, 2001a] S. C. Buraga, *Tehnologii Web*, Matrix Rom, Bucureşti, 2001:
http://www.infoiasi.ro/~busaco/books/web.html

[Buraga, 2001b] S. C. Buraga, "A RDF Proposal for Modelling Relationships between a Teleconferencing System's Resources", în C. Lazăr (ed.), *The 7th International Symposium on Automatic Control and Computer Science (SACCS 2001) – Abstracts Book*, Editura "Gh. Asachi", Iaşi, 2001

[Buraga, 2001c] S. C. Buraga, "An Extensible Framework for Building Interactive Courses on Web", în *The Proceedings of the 5th International Conference on Economic Informatics*, INFOREC Printing House, Bucureşti, 2001, pag. 1–6

[Buraga, 2001d] S. C. Buraga, "An XML-based Representation of Relationships Between Resources of an Internet Teleconferencing System", în V. Preju (ed.), *Abstracts of the International Conference "Information Technologies"*, Chişinău, 2001, pag. 48

[Buraga, 2002a] S. C. Buraga, *Proiectarea siturilor Web*, Polirom, Iaşi, 2002:
http://www.infoiasi.ro/~design/

[Buraga, 2002b] S. C. Buraga, "A Model for Accessing Resources of the Distributed File Systems", în D. Grigoraş *et. al* (eds.), *Advanced Environments, Tools, and Applications for Cluster Computing. NATO Advanced Research Workshop, IWCC 2001 – revised papers*, Lecture Notes in Computer Science – LNCS 2326, Springer-Verlag, Berlin, 2002, pag. 224–230

[Buraga, 2002c] S. C. Buraga, "Modeling Relations between Web Resources", *Transactions on Automatic Control and Computer Science*, vol. 47 (61), No. 2, Politehnica Press, Timişoara, 2002, pag. 17–22

[Buraga, 2003a] S. C. Buraga (coord.), *Aplicaţii Web la cheie*, Polirom, Iaşi, 2003:
http://www.infoiasi.ro/~phpapps/

[Buraga, 2003b] S. C. Buraga, "An XML-based Approach in Designing and Building of Web User-Interfaces", în I. Ivan, I. Rocşa (eds.), *The Proceedings of the 6th International Conference on Economic Informatics*, INFOREC Printing House, Bucureşti, 2003, pag. 838–843

[Buraga, 2003c] S. C. Buraga, "Development Agent-Oriented E-learning Systems", în I. Dumitrache, C. Buiu (eds.), *Proceedings of the 14th International Conference on Control Systems and Computer Science – CSCS 14*, Politehnica Press, Bucureşti, 2003, pag. 19–24

[Buraga, 2003d] S. C. Buraga, "An XML-based Semantic Description of Distributed File Systems", în *Networking in Education and Research. RoEduNet International Conference – second edition*, Samia Press, Iaşi, 2003, pag. 41–49

[Buraga, 2003e] S. C. Buraga, "An XML-Based Agent-Oriented E-Learning System", *The Annals of "Dunărea de Jos" University of Galaţi – Electrotechnics, Electronics, Automatic Control and Informatics Section*, Fascicle V, 2003

[Buraga, 2003f] S. C. Buraga, "Suportul pentru implementare", în C. Pribeanu (coord.), *Introducere în interacţiunea om-calculator*, Matrix Rom, Bucureşti, 2003, pag. 155–178

[Buraga, 2003g] S. C. Buraga, "*ITW* – An Architecture based on Distributed Web Components for Multimedia Resource Discovery", *Scientific Annals of the "Al. I. Cuza" University of Iaşi*, Computer Science Series, tome XIII, 2003, pag. 40–56

[Buraga & Alboaie, 2004] S. C. Buraga, S. Alboaie, L. Alboaie, "An XML/RDF-based Proposal to Exchange Information within a Multi-Agent System", D. Grigoraş *et al.* (eds.), *Proceedings of NATO Advanced Research Workshop on Concurrent Information Processing and Computing*, IOS Press, 2004 (în curs de apariţie)

[Buraga & Brut, 2001] S. C. Buraga, M. Brut, "A Proposal for a Web Structural Search Language Based on XML Technologies", *Scientific Annals of the "Al. I. Cuza" University of Iaşi*, Computer Science Series, tome X, 2001, pag. 79–98

[Buraga & Brut, 2002] S. C. Buraga, M. Brut, "Different XML-based Search Techniques on Web", *Transactions on Automatic Control and Computer Science*, vol. 47 (61), No. 2, Politehnica Press, Timişoara, 2002, pag. 12–16

[Buraga & Brut, 2003] S. C. Buraga, M. Brut, "Using Multimedia Presentations on Web", în I. Dumitrache, C. Buiu (eds.), *Proceedings of the 14th International Conference on Control Systems*

and Computer Science – CSCS 14, Politehnica Press, Bucureşti, 2003, pag. 25–28

[Buraga & Ciobanu, 2001] S. C. Buraga, G. Ciobanu, *Atelier de programare în reţele de calculatoare*, Polirom, Iaşi, 2001: `http://www.infoiasi.ro/~lrc/`

[Buraga & Ciobanu, 2002] S. C. Buraga, G. Ciobanu, "A RDF-based Model for Expressing Spatio-Temporal Relations Between Web Sites", *The Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE 2002)*, IEEE Computer Society Press, 2002

[Buraga & Cioca, 2003] S. C. Buraga, M. Cioca, "Open Methodologies for Developing Web-based E-learning Systems", *Proceedings of the International Conference on Manufacturing Science and Education – Chalanges of the European Integration*, Editura Universităţii "L. Blaga", Sibiu, 2003, pag. 103–106

[Buraga & Găbureanu, 2003] S. C. Buraga, P. Găbureanu, "A Distributed Platform based on Web Services for Multimedia Resource Discovery", în M. Paprzycki (ed.), *Proceedings of the 2nd International Symposium on Parallel and Distributed Computing*, IEEE Computer Society Press, 2003 (în curs de apariţie)

[Buraga & Rusu, 2000] S. C. Buraga, T. Rusu, "An XML-based Query Language Used in Structural Search Activity on Web", *Transactions on Automatic Control and Computer Science*, vol. 45 (59), No. 3, Politehnica Press, Timişoara, 2000, pag. 107–112

[Buraga & Rusu, 2001] S. C. Buraga, T. Rusu, "Search Semi-Structural Data on Web", în C. Lazăr (ed.), *The 7th International Symposium on Automatic Control and Computer Science (SACCS 2001) – Abstracts Book*, Editura "Gh. Asachi", Iaşi, 2001

[Buraga & Todoroi, 2000] S. C. Buraga, D. Todoroi, "Adaptabilitatea informaţională şi operaţională", *Sesiunea jubiliară de comunicări ştiinţifice "Creştere economică, dezvoltare, progress"*, vol. XXX, Cluj-Napoca, 2000

[Buraga *et al.*, 2001] S. C. Buraga, D. Ştefănescu, E. Pecheanu, "Modeling Relations between Resources of an Internet Teleconferencing System", în S. Caraman *et al.* (eds.), *The 11th International*

Symposium on Modeling, Simulation, and Systems' Identification – SIMSIS 2001, Editura Fundaţiei Universitare "Dunărea de Jos", Galaţi, 2001, pag. 68–73

[Buraga *et al.*, 2002a] S. C. Buraga, V. Tarhon-Onu, Ş. Tanasă, *Programare Web în bash şi Perl*, Polirom, Iaşi, 2002: `http://www.infoiasi.ro/~cgi/`

[Buraga *et al.*, 2002b] S. C. Buraga, D. Ştefănescu, E. Pecheanu, "VRML Representations of Lindenmayer Systems", *Proceedings of the 6th International Conference on Development and Application Systems – DAS 2002*, Editura "Muşatinii", Suceava, 2002, pag. 301–306

[Busetta *et al.*, 1998] P. Busetta *et al.*, "An Architecture for Mobile BDI Agents", *Proceedings of Symposium on Applied Computing – SAC'98*, ACM Press, 1998

[Buyya, 2002] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, PhD Thesis, Monash University, Melbourne, Australia, 2002

[Cabri, 2001] G. Cabri, "Role-based Infrastructures for Agents", *Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems – FTDCS 2001*, Bologna, 2001

[Capretz & Osano, 2002] M. Capretz, M. Osano, "Collaborative Psychological Agents", *Technical Report*, Department of Electrical and Computer Engineering, University of Western Ontario, Canada, 2002

[Carr *et al.*, 2001] L. Carr *et al.*, "Conceptual Linking: Ontology-based Open Hypermedia", *Proceedings of the 10th International Conference on Web Technologies – WWW10*, ACM Press, 2001

[Călin, Alexandru & Buraga, 1998a] M. Călin, I. Alexandru, S. C. Buraga, "Application of Fuzzy Database Management in Steel Selection", *EUROMAT'98 European Conference Proceedings*, Lisbon, Portugal, 1998

[Călin, Alexandru & Buraga, 1998b] M. Călin, I. Alexandru, S. C. Buraga, "Software Tool with Fuzzy Capabilities for Assisting Decision in Steel Selection Process", în C. Lazăr (ed.), *Proceedings of the 6th International Symposium on Automatic*

Control and Computer Science – SACCS'98, vol. I, Matrix Rom, Bucureşti, 1998, pag. 243–248

[Ceri *et al.*, 1999] S. Ceri *et al.*, "XML-GL: A Graphical Language for Querying and Restructuring XML Documents", *Proceedings of the Eight International World-Wide Web Conference – WWW8*, ACM Press, 1999

[Ceri *et al.*, 2000] S. Ceri *et al.*, "Web Modeling Language (WebML): a Modeling Language for Designing Web Sites", *Computer Networks*, 33, 2000

[Chakrabarti *et al.*, 1999] S. Chakrabarti *et al.*, "Hypersearching the Web", *Scientific American*, no. 6, 1999

[Chapin *et al.*, 1999] S. Chapin, J. Karpovich, A. Grimshaw, "The Legion Resource Management System", *Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing at IPDPD'99*, San Juan, Puerto Rico, 1999

[Chellas, 1980] B. Chellas, *Modal Logic: An Introduction*, Cambridge University Press, Cambridge, UK, 1980

[Chomicki, 1994] J. Chomicki, "Temporal Query Languages: a Survey", *Proceedings of the First International Conference on Temporal Logic*, Lecture Notes in Artificial Intelligence – LNAI 827, Springer-Verlag, Berlin, 1994

[Cioca & Buraga, 2003a] M. Cioca, S. C. Buraga, "New Tools for Human Resource Management in e-Business: Combining UML Language, Reference Architectures and Web Programming", *IEEE International Conference on Industrial Informatics – INDIN'03 Proceedings*, Banff, Alberta, Canada, 2003

[Cioca & Buraga, 2003b] M. Cioca, S. C. Buraga, "Instruments and Web Technologies for Implementing Architectures and Integration Informatics Systems in Virtual Enterprises", *Proceedings of the 3rd International Conference on Research and Development in Mechanical Industry – RaDMI 2003*, Herceg Novi, Montenegro Adriatic, 2003, pag. 1503–1506

[Cioca & Buraga, 2003c] M. Cioca, S. C. Buraga, "Integration Methodologies of Enterprises in "e-Europe" Utilizing Reference Architectures, Modelling Languages and Web Technologies", în C. Gyenge

(ed.), *Proceedings of the 6th International MTeM Conference*, Cluj, Romania, 2003, pag. 113–116

[Cohen & Levesque, 1990] P. Cohen, H. Levesque, "Intention Is Choice with Commitment", *Artificial Intelligence*, no. 42 (3), 1990

[Comer & Stevens, 1993] D. Comer, D. Stevens, *Internetworking with TCP/IP – vol. III: Client/Server Programming and Applications*, Prentice Hall, New Jersey, 1993

[Conen & Klapsing, 2000] W. Conen, R. Klapsing, "A Logical Interpretation of RDF", *Linköping Electronic Articles in Computer and Information Science*, 5, 2000:
`http://www.ida.liu.se/ext/epa/cis/2000/013/tcover.html`

[Corcho & Gomez-Perez, 2000] O. Corcho, A. Gomez-Perez, "A Roadmap to Ontology Specification Languages", *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management*, 2000

[Costello & Jacobs, 2003] R. Costello, D. Jacobs, *OWL Tutorial*, XML Technologies, 2003:
`http://www.xfront.com/owl/`

[Croitoru, 1992] C. Croitoru, *Tehnici de bază în optimizarea combinatorie*, Editura Universităţii "A. I. Cuza", Iaşi, 1992

[Curbera *et al.*, 2002] F. Curbera *et al.*, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI", *IEEE Internet Computing*, vol. 6, no. 2, March 2002

[Davies *et al.*, 2003] J. Davies, D. Fensel, F. van Harmelen (eds.), *Towards the Semantic Web*, John Wiley & Sons, England, 2003

[Dean & Schreiber, 2004] M. Dean, G .Schreiber (eds.), *OWL Web Ontology Language Reference*, W3C Recommendation, Boston, 2004:
`http://www.w3.org/TR/owl-ref/`

[DeRose, 1998] S. DeRose, "XQuery: An Unified Syntax for Linking and Querying General XML Documents", *Proceedings of QL'98 – The Query Languages Workshop*, Cambridge, Massachusetts, 1998

[DeRose *et al.*, 2001] S. DeRose *et al.* (eds.), *XML Linking Language (XLink) Version 1.0*, W3C Recommendation, Boston, 2001: http://www.w3.org/TR/xlink/

[De Roure *et al.*, 2003] D. De Roure, N. Jennings, N. Shadbolt, "The Evolution of the Grid", *International Journal of Concurrency and Computation: Practice and Experience*, 2003

[Deutsch *et al.*, 1998] A. Deutsch *et al.*, "XML-QL: A Query Language for XML", *Proceedings of QL'98 – The Query Languages Workshop*, Cambridge, Massachusetts, 1998: http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/

[Doan *et al.*, 2003] A. Doan *et al.* (eds.), *Semantic Integration – SI 2003 Workshop Proceedings*, Sanibel Island, Florida, 2003

[Dyreson, 2001] C. Dyreson, "Towards a Temporal World-Wide Web: A Transaction-time Server", *Proceedings of the 12th Australasian Conference on Database Technologies*, ACM Press, 2001

[Emerson, 1990] E. A. Emerson, "Temporal and Modal Logic", în J. von Leenwen (ed.), *Handbook of Theoretical Computer Science: Volume B, Formal Methods and Semantics*, Elsevier Science and MIT Press, 1990

[Emerson & Srinivasan, 1989] E. A. Emerson, J. Srinivasan, "Branching Time Temporal Logic", în J. W. de Bakker *et al.* (eds.), *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Springer-Verlag, Berlin, 1989

[Fallside, 2001] D. Fallside (ed.), *XML Schema*, W3C Recommendation, Boston, 2001: http://www.w3.org/TR/xmlschema-0/

[Fensen *et al.*, 2000] D. Fensen *et al.*, "OIL in a Nutshell", *Proceedings of the European Knowledge Acquisition Conference – EKAW 2000*, Lecture Notes in Artificial Intelligence – LNAI, Springer-Verlag, 2000

[Fensen, Bussler, 2002] D. Fensen, C. Bussler, *The Web Service Modeling Framework – WSMF*, White Paper and Internal Report Vrije Universiteit Amsterdam, 2002: http://www.cs.vu.nl/swws/download/wsmf.paper.pdf

[Ferraiolo *et al.*, 2003] J. Ferraiolo *et al.* (eds.), *Scalable Vector Graphics (SVG) 1.1 Specification*, W3C Recommendation, Boston, 2003:
http://www.w3.org/TR/SVG11/

[Fielding *et al.*, 1997] R. Fielding *et al.* (eds.), *HyperText Transfer Protocol – HTTP/1.1*, RFC 2068, IETF, 1997:
http://www.ietf.org/rfc/rfc2068.txt

[Fikes & McGuinness, 2001] R. Fikes, D. McGuiness, *An Axiomatic Semantics for RDF, RDF Schema, and DAML+OIL*, Technical Report KSL-01-01, Standord University, 2001:
http://www.ksl.stanford.edu/people/dlm/daml-semantics/abstract-axiomatic-semantics.html

[Florescu, Levy & Mendelzon, 1998] D. Florescu, A. Levy, A. Mendelzon, *Database Techniques for the WWW: a Survey*, SIGMOD, ACM, 1998

[Fong, 1998] P. Fong, *Viewer's Discretion: Host Security in Mobile Code Systems*, Technical Report SFU CMPT TR 1998-19, School of Computing Science, Simon Fraser University, 1998:
ftp://fas.sfa.ca/pub/cs/techreports/1998/CMPT1998-19.pdf

[Fontoura *et al.*, 2003] M. Fontoura *et al.*, "TSpaces Services Suite: Automating the Development and Management of Web Services", *Proceedings of the World-Wide Web 2003 Conference*, ACM Press, 2003

[Foster & Kesselan, 1999] I. Foster, C. Kesselman (eds.), *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kauffmann Publishers, 1999

[Franks *et al.*, 1997] J. Franks *et al.*, *HTTP Authentication: Basic and Digest Access Authentication*, RFC 2617, IETF, 1997:
http://www.ietf.org/rfc/rfc2617.txt

[Gaura & Newman, 2003] E. Gaura, R. Newman, "Using AI Techniques to Aid Hypermedia Design", *Proceedings of SIGDOC'03 Conference*, ACM Press, 2003

[Gelernter, 1989] D. Gelernter, "Multiple Tuple Spaces in Linda", în J. G. Goos (ed.), *Proceedings of PARLE'89*, Lecture Notes in Computer Science – LNCS 365, Springer-Verlag, Berlin, 1989

[Genesereth & Fikes, 1992] M. Genesereth, R. Fikes, *Knowledge Interchange Format. Version 3.0. Reference Manual*, Technical Report Logic-92-1, Stanford University, 1992

[Gogan & Buraga, 2000] O. Gogan, S. C. Buraga, "The Use of Neural Networks for Structural Search on Web", *Proceedings of the 10th Edition of the International Symposium on Systems Theory, Automation, Robotics, Computers, Informatics, Electronic and Instrumentation – SINTES 10*, ROM TPT, Craiova, 2000, pag. 53–56

[Goldfarb, 1990] C. Goldfarb, *The SGML Handbook*, Oxford Press, 1990

[Goossens *et al.*, 1994] M. Goossens *et al.*, *The LaTeXCompanion*, Addison-Wesley, Reading, Massachusetts, 1994

[Gorman, 2001] C. Gorman, *Programming Web Services with SOAP*, O'Reilly & Associates, 2001

[Governatori *et al.*, 2002] G. Governatori *et al.*, "On Fibring Semantics for BDI Logics", *Logics in Artificial Intelligence, European Conference – JELIA 2002*, Lecture Notes in Artificial Intelligence – LNAI 2424, Springer-Verlag, Berlin, 2002

[Grigoraş, 2004] D. Grigoraş, "Service-oriented Naming Scheme for Wireless Ad Hoc Networks", D. Grigoraş *et al.* (eds.), *Proceedings of NATO Advanced Research Workshop on Concurrent Information Processing and Computing*, IOS Press, 2004 (în curs de apariţie)

[Grigoraş, McInerny, Mulcahy, 2002] D. Grigoraş, R. McInerny, C. Mulcahy, "MAIS – The Mobile Agents Information System Support for Creating Dynamic Clusters", *Proceedings of ICA3PP 2002*, IEEE Computer Society Press, 2002

[Grosof *et al.*, 2003] B. Grosof *et al.*, "Description Logic Programs: Combining Logic Programs with Description Logic", *Proceedings of WWW 2003 International Conference*, ACM Press, 2003

[Gruber, 1993] T. R. Gruber, "A Translation Approach to Portable Ontologies", *Knowledge Acquisition*, no. 5 (2), 1993

[Guarino, 1998] N. Guarino, "Formal Ontology and Information Systems", *Proceedings of FOIS'98*, IOS Press, 1998

[Guarino & Giaretta, 1995] N. Guarino, P. Giaretta, "Ontologies and Knowledge bases: towards a Terminological Clarification", în N. Mars (ed.), *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, IOS Press, 1995

[Guha & Hayes, 2003] R. Guha, P. Hayes, *LBase: Semantics for Languages of the Semantic Web*, W3C Candidate Recommendation, Boston, 2003:
http://www.w3.org/TR/lbase

[Halpern & Moses, 1992] J. Halpern, Y. Moses, "A Guide to Completness and Complexity for Modal Logics of Knowledge and Belief", *Artificial Intelligence*, no. 54, 1992

[Hayes, 2004] P. Hayes (ed.), *RDF Semantics*, W3C Recommendation, Boston, 2004:
http://www.w3.org/TR/rdf-mt/

[Haustein, 2001] S. Haustein, "Semantic Web Languages: RDF vs. SOAP Serialization", *Proceedings of Semantic Web Workshop*, Hongkong, China, 2001

[Heery & Wagner, 2002] R. Heery, H. Wagner, "A Metadata Registry for the Semantic Web", D-Lib Magazine, vol. 8, no. 5, May 2002

[Hendler & McGuinness, 2001] J. Hendler, D. McGuinness, "The DARPA Agent Markup Language", *IEEE Intelligent Systems*, vol. 15, no. 6, Nov./Dec. 2001

[Hobbs & Pustejovsky, 2002] J. Hobbs, J. Pustejovsky, "Annotating and Resoning about Time and Events", *Journal of Artificial Intelligence*, AAAI Press, 2002

[Hohlfeld & Yee, 1998] M. Hohlfelf, B. Yee, *How to Migrate Agents*, Department of Computer Science and Engineering, University of California San Diego, 1998:
http://www-cse.ucsd.edu/users/bsy/pub/migrate.ps

[Horrocks & Patel-Schneider, 2003] I. Horrocks, P. Patel-Schneider, "Three Theses of Representation in the Semantic Web", *Proceedings of WWW 2003 International Conference*, ACM Press, 2003

[Horrocks *et al.*, 2000] I. Horrocks *et al.*, *The Ontology Inference Layer (OIL)*, Technical Report, Free University of Amsterdam, 2000:
http://www.ontoknowledge.org/oil/

[Horrocks *et al.*, 2001] I. Horrocks *et al.*, "Ontology Reasoning in the $\mathcal{SHIQ}$(D) Description Logic", *Proceedings of the IJCAI Conference*, 2001

[Hintikka, 1962] J. Hintikka, *Knowledge and Belief*, Cornell University Press, Ithaca, New York, 1962

[Huget, 2002] M. P. Huget, *Desiderata for Agent-Oriented Programming Languages*, Technical Report ULCS-02-010, Department of Computer Science, University of Liverpool, UK, 2002

[Hughes & Cresswell, 1968] G. E. Hughes, M. J. Cresswell, *An Introduction to Modal Logic*, Methnen & Co., London, UK, 1968

[Jacobs, 2003] I. Jacobs (ed.), *Architecture of World Wide Web*, W3C Working Draft, Boston, 2003:
`http://www.w3.org/TR/webarch/`

[Jeffery, 2000] K. G. Jeffery, "Knowledge, Information, and Data", *A briefing to the Office of Science and Technology*, UK, 2000:
`http://www.itd.clrc.ac.uk/ActivityPublications/239`

[Jennings *et al.*, 1998] N. Jennings, K. Sycara, M. Wooldridge, "A Roadmap of Agent Research and Development", *Autonomous Agents and Multi-Agent Systems*, no. 1, Kluwer Academic Publishers, Boston, 1998

[Jucan, 1999] T. Jucan, *Limbaje formale şi automate*, Matrix Rom, Bucureşti, 1999

[Jucan & Andrei, 2002] T. Jucan, Ş. Andrei, *Limbaje formale şi teoria automatelor. Teorie şi practică*, Editura Universităţii "A. I. Cuza", Iaşi, 2002

[Kahn & Cicalese, 2001] M. L.Kahn, C. D. T. Cicalese, "CoABS Grid Scalability Experiments", *Second International Workshop on Infrastructure for Scalable Multi-Agent Systems at Autonomous Agents*, Montreal, Canada, 2001

[Kamps & Marx, 2002] J. Kamps, M. Marx, "Notions of Indistinguishability for Semantic Web Languages", *Proceedings of International Semantic Web Conference – ISWC 2002*, Lecture Notes in Computer Science – LNCS 2342, Springer-Verlag, 2002

[Karp *et al.*, 1999] P. Karp *et al.*, *XOL: An XML-based Ontology Exchange Language*, version 0.3, 1999:
`ftp://smi.stanford.edu/pub/bio-ontology/ontologyexchange.doc`

[Kinny, Georgeff, Rao, 1996] D. Kinny, M. Georgeff, A. Rao, "A Methodology and Modelling Technique for Systems of BDI Agents", *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World – MAAMAW'96*, Lecture Notes in Artificial Intelligence – LNAI 1038, Springer-Verlag, 1996

[Klapsing & Neumann, 2000] R. Klapsing, G. Neumann, "Applying the Resource Description Framework to Web Engineering", *Proceedings of the First International Conference on Electronic Commerce and Web Technologies*, Springer-Verlag, 2000

[Knight & Ma, 1993] B. Knight, J. Ma, "An Extended Temporal System Based on Points and Intervals", *Information Systems*, 18 (2), 1993

[Knight & Ma, 1994] B. Knight, J. Ma, "Time Representation: A Taxonomy of Temporal Models", *Artificial Intelligence Review*, Vol. 7, 1994

[Kokkelink & Schwänzl, 2002] S. Kokkelink, R. Schwänzl, *Expressing Qualified Dublin Core in RDF/XML*, DCMI Proposed Recommendation, Dublin, 2002:
`http://dublincore.org/documents/dcq-rdf-xml/`

[Konopnicki & Shmueli, 1995] D. Konopnicki, O. Shmueli, "W3QL: A Query System for the World Wide Web", in *Proceedings of the 21th International Conference on Very Large Databases*, Zurich, 1995

[Kramer, 1996] M. Kramer, *Distributed File Systems*, IBM White Paper, Boston, 1996

[Kripke, 1963] S. Kripke, "A Semantical Analysis of Modal Logic: Normal Modal Propositional Calculi", *Z. Math. Logik Grundl. Math.*, no. 9, 1963

[Knuth, 1984] D. E. Knuth, *The TEXBook. Volume A of Computers and Typesetting*, Addison-Wesley, Reading, Massachusetts, 1984

[Lakshmanan, Sadri & Subramanian, 1996] L. Lakshmanan, F. Sadri, I. Subramanian, "A Declarative Language for Querying and Restructuring the Web", *Proceedings of RIDE-NDS*, IEEE Computer Society Press, 1996

[Lamport, 1980] L. Lamport, "'Sometime' is Sometimes 'Not Ever': a Tutorial on the Temporal Logic of Programs", *Proc. of the 7th Annual Symposium on Principles of Programming Languages*, ACM SIGACT–SIGPLAN, 1980

[Lamport, 1994] L. Lamport, *LaTeX: A Document Preparation System*, Second Edition, Addison-Wesley, Reading, Massachusetts, 1994

[Lamport, 2003] L. Lamport, *Specifying Systems. The* TLA+ *Language and Tools for Hardware and Software Engineers*, Addison-Wesley, 2003

[Lassila & Swick, 1999] O. Lassila, R. Swick (eds.), *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation, Boston, 1999:
`http://www.w3.org/TR/REC-rdf-syntax`

[Lee & Selgman, 2000] R. Lee, S. Selgman, *JNDI API Tutorial and Reference. Building Directory-Enabled Java Applications*, Addison-Wesley, 2000

[Louka, 1994] M. Louka, *A Review of Hypermedia Methodologies and Techniques*, Technical Report, University of Surrey, 1994

[Lowe, 2000] D. Lowe, "Improving Web Search Relevance: Using Navigational Structures to Provide a Search Context", *The Sixth Australian World-Wide Web Conference*, Cairns, 2000

[Lucanu, 1996] D. Lucanu, *Bazele proiectării programelor şi algoritmilor*, Editura Universităţii "A. I. Cuza", Iaşi, 1996

[Luck, McBurney & Preist, 2003] M. Luck, P. McBurney, C. Preist, *Agent Technology: Enabling Next Generation Computing. A Roadmap for Agent-Based Computing*, AgentLink.org, Southampton, 2003:
`http://www.agentlink.org/roadmap`

[Maedche *et al.*, 2003] A. Maedche *et al.*, "An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies", *Proceedings of WWW 2003 International Conference*, ACM Press, 2003

[Malhotra *et al.*, 2003] A. Malhotra *et al.* (eds.), *XML Syntax for XQuery 1.0 (XQueryX)*, W3C Working Draft, Boston, 2003: `http://www.w3.org/TR/xqueryx`

[Mangina, 2002] E. Mangina, *Review of Software Products for Multi-Agent Systems*, AgentLink.org, Southampton, 2002: `http://www.agentlink.org`

[Manola & Miller, 2004] F. Manola, E. Miller (eds.), *RDF Primer*, W3C Recommendation, Boston, 2004: `http://www.w3.org/TR/rdf-primer/`

[Marchiori, 1997] M. Marchiori, "The Quest for Correct Information on the Web: Hyper Search Engines", *WWW6 Conference Proceedings*, Elsevier Science, 1997

[Martin, Cheyer, Moran, 1999] D. Martin, A. Cheyer, D. Moran, "The Open Agent Architecture: A Framework for building Distributed Software Systems", *Applied Artificial Intelligence*, 13 (1-2), 1999

[McBride, 2002] B. McBride, "Jena: A Semantic Web Toolkit", *IEEE Internet Computing*, November-December 2002

[McIlraith, Son, Zeng, 2001] S. McIlraith, T. Son, H. Zeng, "Semantic Web Services", *IEEE Intelligent Systems*, 16 (2), 2001

[Mendelzon *et al*, 1997] A. Mendelzon, G. Mihăilă, T. Milo, "Querying the World-Wide Web", *Journal of Digital Libraries*, vol. 1 (1), 1997: `ftp://ftp.db.toronto.edu/pub/papers/jodl97.ps.Z`

[Metakides & Nerode, 1998] G. Metakides, A. Nerode, *Principii de logică şi programare logică*, Editura Tehnică, Bucureşti, 1998

[Miller *et al.*, 2002] L. Miller *et al.*, "Three Implementations of SquishQL, a Simple RDF Query Language", în I. Horrocks, J. Hendler (eds.), *International Semantic Web Conference – ISWC 2002*, Lecture Notes in Computer Science – LNCS 2342, Springer-Verlag, 2002

[Milojicic, 1999] D. Milojicic, "Mobile Agent Applications", *IEEE Concurrency*, no. 8-9, 1999

[Moldovan & Mihalcea, 1999] D. Moldovan, R. Mihalcea, "Improving the Search on the Internet by Using WordNet and Lexical Operators", *Siglex'99 Proceedings*, 1999

[Moreau, 2002] L. Moreau, "Agents for the Grid: A Comparison with Web Services (Part I: Transport Layer)", în *IEEE International Symposium on Cluster Computing and the Grid Proceedings*, Berlin, Germany, 2002

[Naik, 1998] D. C. Naik, *Internet Standards and Protocols*, Microsoft Press, Redmond, Washington, 1998

[Nielsen, 1990] J. Nielsen, *Hypertext and Hypermedia*, San Diego Academic Press, 1990

[O'Hara *et al.*, 2001] K. O'Hara *et al.*, *The AKT Manifesto*, 2001: `http://www.aktors.org/publications/Manifesto.doc`

[Oeschger, 2003] I. Oeschger, *XUL Programmer's Reference Manual*, Mozilla.Org, 2003: `http://www.mozilla.org/xpfe/Xulref.zip`

[Ohia, 2001] K. Ohia, "Necessities on a Descriptive Level for Reusing Metadata Descriptions", *Proceedings of the Internationational Conference on Dublin Core Metadata Applications*, 2001

[Oliboni & Tanca, 2000] B. Oliboni, L. Tanca, *Querying XML Specified WWW Sites: Links and Recursion in XML-GL*, Technical Report, University of Milano, 2000

[Owicki & Lamport, 1982] S. Owicki, L. Lamport, "Proving Liveness Properties of Concurrent Systems", *ACM Transactions on Programming Languages and Systems*, 4 (3), 1982

[Papadimitriou, 1994] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994

[Patel-Schneider & Simeon, 2002] P. Patel-Schneider, J. Simeon, "Building the Semantic Web on XML", în I. Horrocks, J. Hendler (eds.), *International Semantic Web Conference – ISWC 2002*, Lecture Notes in Computer Science – LNCS 2342, Springer-Verlag, 2002

[Patrick, 2002] A. Patrick, "Building Trustworthy Software Agents", *IEEE Internet Computing*, no. 11-12, 2002

[Pauly, 2001] M. Pauly, *Logic for Social Software*, PhD Thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam, 2001

[Pecheanu, Ştefănescu, Buraga, Istrate, 2001] E. Pecheanu, D. Ştefănescu, S. C. Buraga, A. Istrate, "Integrating Hypermedia Objects in an Intelligent Tutoring System", în S. Caraman *et al.* (eds.), *The 11th International Symposium on Modeling, Simulation, and Systems' Identification – SIMSIS 2001*, Editura Fundaţiei Universitare "Dunărea de Jos", Galaţi, 2001, pag. 170–177[1]

[Pemberton *et al.*, 2002] S. Pemberton *et al.*, *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)*, W3C Recommendation, Boston, 2002:
`http://www.w3.org/TR/xhtml1`

[Pnueli, 1977] A. Pnueli, "The Temporal Logic of Programs", *Proceedings of the 18th Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, 1977

[Poslad, Buckle, Hadingdam, 2000] S. Poslad, P. Buckle, R. Hadingdam, "The FIPA-OS Agent Platform: Open Source for Open Standards", *PAAM2002 Proceedings*, Manchester, UK, 2000

[Pustejovsky *et al.*, 2002] I. Pustejovsky *et al.*, *TimeML Specification 1.0*, 2002:
`http://time2002.org`

[Raggett *et al.*, 1999] D. Raggett *et al.* (eds.), *HyperText Markup Language (HTML) 4.01 Specification*, W3C Recommendation, Boston, 1999:
`http://www.w3.org/TR/html401`

[Rana & Moreau, 2000] O. F. Rana, L. Moreau, "Issues in Building Agent-based Computational Grids", *Third Workshop of the UK Special Interest Group on Multi-Agent Systems – UKMAS 2000*, Oxford, UK, 2000

[1]Lucrare publicată de asemenea şi în Analele Universităţii "Dunărea de Jos" din Galaţi, fascicolul III, 2001.

[Rana & Walker, 2000] O. F. Rana, D. W. Walker, "'The Agent Grid': Agent-Based Resource Integration in PSEs", *Proceedings of the 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, Laussanne, Switzerland, 2000

[Rao & Georgeff, 1991a] A. Rao, M. Georgeff, "Asymmetry Thesis and Side-Effect Problems in Linear Time and Branching Time Intention Logics", *Proceedings of the 12th International Joint Conference on Artificial Intelligence – IJCAI'91*, Sydney, Australia, 1991

[Rao & Georgeff, 1991b] A. Rao, M. Georgeff, "Deliberation and its Role in the Formation of Intentions", *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence – UAI'91*, Morgan Kaufmann Publishers, San Mateo, CA, 1991

[Rao & Georgeff, 1991c] A. Rao, M. Georgeff, "Modeling Rational Agents within a BDI-Architecture", în J. Allen *et al.* (eds.), *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, San Mateo, CA, 1991

[Rao *et al.*, 1995] A. Rao *et al.*, *Formal Methods and Decision Procedures for Multi-Agent Systems*, Technical Report 61, Australian Artificial Intelligence Institute, 1995

[Rhodes, 2003] K.Rhodes, *XML-RPC vs. SOAP*, 2003:
`http://weblog.masukomi.org/writings/xml-rpc_vs_soap.htm`

[Roddick & Snodgrass, 1995] J. Roddick, R. Snodgrass, "Transaction Time Support", în R. Snodgrass (ed.), *The TSQL2 Temporal Query Language*, Kluwer Academic Publishers, 1995

[Rotenberg, 1996] J. Rotenberg, "Metadata to Support Data Quality and Longevity", *First IEEE Metadata Conference*, Silver Spring, Maryland, 1996:
`http://www.computer.org/conferences/meta96/rothenberg_Paper/ieee.data-quality.html`

[Rusu, Buraga, Iojoiu, 2002] T. Rusu, S.Buraga, C. Iojoiu, "A Proposal for Exchanging Scientific Documents on the Web", *The 6th World Congress of Theoretically Oriented Chemists – WATOC 2002*, Lugano, Switzerland, 2002

[Sander & Tschudin, 1998] T. Sander, C. Tschudin, *Protecting Mobile Agents Against Malicious Hosts*, International Computer Science Institute, Berkeley, 1998:
`http://www.icsi.berkeley.edu/~tschudin/ps/ma-security.ps.gz`

[Scott-Cost *et al.*, 2002] R. Scott-Cost *et al.*, "ITtalks: A Case Study in the Semantic Web and DAML+OIL", *IEEE Intelligent Systems*, January-February 2002

[Seaborne, 2002] A. Seaborne, "An RDF NetAPI", în I. Horrocks, J. Hendler (eds.), *International Semantic Web Conference – ISWC 2002*, Lecture Notes in Computer Science – LNCS 2342, Springer-Verlag, 2002

[Shanmugasundaram *et al.*, 1999] J. Shanmugasundaram *et al.*, "Relational Databases for Querying XML Documents: Limitations and Opportunities", *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, 1999

[Silberschatz *et al.*, 2001] A. Silberschatz, J. Peterson, P. Galvin, *Operating Systems Concepts*, Sixth Edition, Addison-Wesley, Reading, Massachusetts, 2001

[Sintek & Decker, 2002] M. Sintek, S. Decker, "TRIPLE – A Query, Inference, and Transformation Language for the Semantic Web", în I. Horrocks, J. Hendler (eds.), *International Semantic Web Conference – ISWC 2002*, Lecture Notes in Computer Science – LNCS 2342, Springer-Verlag, 2002

[Sheth, Arpinar, Kashyap, 2002] A. Sheth, B. Arpinar, V. Kashyap, *Relationships at the Heart of Semantic Web: Modeling, Discovering, and Exploiting Complex Semantic Relationships*, Techical Report, LSDIS Lab., University of Georgia, 2002

[Sollazzo *et al.*, 2001] T. Sollazzo *et al.*, "Semantic Web Service Architecture: Evolving Web Service Standards toward the Semantic Web", *FLAIRS'02 Proceedings*, AAAI Press, 2001

[Staab & Mädche, 2000] S. Staab, A. Mădche, "An Extensible Approach for Modeling Ontologies in RDF(S)", *First Workskop on the Semantic Web*, Lisbon, Portugal, 2000

[Sticker, 2001] P. Sticker, "Metia – A Generalized Metadata Driven Framework for the Management and Distribution of Electronic Media", *Proceedings of the International Conference on Dublin Core and Metadata Applications – DC 2001*, Tokyo, 2001

[Strang *et al.*, 2003] T. Strang *et al.*, "A Context Ontology Language to Enable Contextual Interoperability", *The 4th IFIP International Conference on Distributed Applications and Interoperable Systems – DAIS'03*, Paris, 2003

[Sycare & Payne, 2003] K. Sycara, T. Payne, "Agent Mediated Semantic Web/Grid Services", *Second International Semantic Web Conference*, ACM Press, 2003

[Szalas, 1987] A. Szalas, "A Complete Axiomatic Characterization of First-Order Temporal Logic of Linear Time", *Theoretical Computer Science*, vol. 54, Elsevier, North-Holland, Amsterdam, 1987

[Ştefănescu, Pecheanu, Buraga, 2001] D. Ştefănescu, E. Pecheanu, S. C. Buraga, "Pedagogical Agents in Intelligent Tutoring Systems", în S. Caraman *et al.* (eds.), *The 11th International Symposium on Modeling, Simulation, and Systems' Identification – SIMSIS 2001*, Editura Fundaţiei Universitare "Dunărea de Jos", Galaţi, 2001, pag. 178–183

[Tanasă *et al.*, 2003] Ş. Tanasă, C. Olaru, Ş. Andrei, *Java de la 0 la expert*, Polirom, Iaşi, 2003

[Tanenbaum, 2001] A. Tanenbaum, *Modern Operating Systems*, Prentice Hall International, 2001

[Tanenbaum, 2003] A. Tanenbaum, *Reţele de calculatoare* (ediţia a patra), Byblos, Tg. Mureş, 2003

[Todoroi, 1992] D. N. Todoroi, *The Adaptable Programming*, ASEM Press, Chişinău, 1992

[Trăuşan-Matu, 2000] Ş. Trăuşan-Matu, *Interfaţarea evoluată om-calculator*, Matrix Rom, Bucureşti, 2000

[Trăuşan-Matu, 2001] Ş. Trăuşan-Matu *et al.*, *Prelucrarea documentelor folosind XML şi Perl*, Matrix Rom, Bucureşti, 2001

[Tripathi *et al.*, 1999] A. Tripathi *et al*, *Ajanta – A Mobile Agent Programming System*, Technical Report no. 16, Department of Computer Science, University of Minnesota, 1999:
http://www.cs.umn.edu/Ajanta/papers/TR98-016ps

[Tsotras, 1996] V. Tsotras, "Temporal Database Bibliography Update", *SIGMOD Record (ACM Special Interest Group on Management of Data*, 25 (1), 1996

[van der Hoek & Wooldrige, 2003] W. van der Hoek, M. Wooldrige, "Towards a Logic of Rational Agency", *Logic Journal of the IGPL*, vol. 11, no. 2, Oxford University Press, 2003

[Vasudevan, 2001] V. Vasudevan, *A Web Services Primer*, XML.com, 2001:
http://www.xml.com/pub/a/2001/04/04/webservices/

[Vieu, 1991] L. Vieu, *Sémantique des relations spatiales et inférences spatio-temporelles*, PhD thesis, Université Paul Sabatier, Toulouse, 1991

[Vilain *et al.*, 1990] M. Vilain, H. Kautz, P. van Beek, "Constraint Propagation Algorithms for Temporal Reasoning: a Revised Report", *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufman, San Mateo, CA, 1990

[Vogels & Re, 2003] W. Vogels, C. Re, "WS-Membership – Failure Management in a Web-Services World", *Proceedings of WWW 2003 International Conference*, ACM Press, 2003

[Yu, 1997] J. Yu, *A Simple Intuitive Hypermedia Synchronization Model and its Realization in the Browser/Java Environment*, Technical Report, DEC Systems Research Center, 1997

[Wall *et al.*, 2000] L. Wall *et al.*, *Programming Perl* (Third Edition), O'Reilly & Associates, Cambridge, 2000

[Wallace, 2000] M. Wallace, "Architecture Online: A Search Engine Review", *Searcher*, vol. 8, no. 2, 2000:
http://www.infotoday.com/searcher/
feb00/wallace.htm

[Wright *et al.*, 2003] J. Wright *et al.*, "Using the ebXML Registry Repository to Manage Information in an Internet Travel Support System", *Proceedings of the 6th International Conference*

on Business Information Systems – BIS 2003, Colorado Springs, 2003

[Wooldridge & Ciancarini, 2000] M. Wooldridge, P. Ciancarini, "Agent-Oriented Software Engineering: The State of the Art", *Proceedings of the First International Workshop on Agent-Oriented Software Engineering*, Lecture Notes in Computer Science – LNCS 1957, Springer-Verlag, Berlin, 2000

[Wooldridge & Jennings, 1995] M. Wooldridge, N. Jennings, "Intelligent Agents: Theory and Practice", *Knowledge Engineering Review*, 1995

[Wyckoff *et al.*, 1998] P. Wyckoff *et al.*, "TSpaces", *IBM Systems Journal*, 37 (3), 1998

[Agentcities] * * *, *Agentcities Network*:
`http://www.agentcities.net`

[Altavista] * * *, *Altavista*:
`http://www.altavista.com`

[Annotea] * * *, *Annotea*:
`http://www.w3.org/Annotea/`

[CVS] * * *, *Concurrent Versions System*:
`http://cvshome.org`

[CPAN] * * *, *Comprehensive Perl Archives Network*:
`http://www.perl.com/CPAN/`

[DAML+OIL] * * *, *DAML+OIL Specification*:
`http://www.daml.org/2000/10/daml-oil`

[DAML-S] * * *, *DAML-S Specification*:
`http://www.daml.org/services`

[DAML+TIME] * * *, *DAML+TIME Specification*:
`http://www.daml.org`

[ebXML] * * *, *ebXML*:
`http://www.ebxml.org`

[Edutella] * * *, *Edutella*:
`http://edutella.jxta.org`

[EOR] * * *, *Extensible Open RDF Toolkit*:
http://eor.dublincore.org

[Globus] * * *, *Globus Project*:
http://www.globus.org

[Google] * * *, *Google APIs*:
http://www.google.com/apis/

[IRS] * * *, *Internet Reasoning Service*:
http://kmi.open.ac.uk/projects/irs/

[ISOC] * * *, *Internet Society*:
http://www.isoc.org/

[Java] * * *, *Java Software*:
http://www.javasoft.com

[JINI] * * *, *JINI*:
http://www.jini.org

[JXTA] * * *, *JXTA*:
http://www.jxta.org

[Kartoo] * * *, *Kartoo*:
http://www.kartoo.com

[MSDN] * * *, *Microsoft Developers Network*, Microsoft Corporation, 2003:
http://msdn.microsoft.com

[.NET] * * *, *Microsoft .NET Framework*, Microsoft Corporation, 2003:
http://www.microsoft.com/net/basics/framework.asp

[NFS] * * *, *The NFS Distributed File Service*, Sun's NFS White Paper, 1995:
http://www.sun.com/software/white-papers/wp-nfs

[NIS] * * *, *Network Information Service (NIS)*:
http://www.suse.de/~kukuk/nis/

[Oasis] * * *, *Oasis*:
http://www.oasis-open.org

[OMG] * * *, *Object Management Group Activity*:
http://www.omg.org

[Prospero] * * *, *Prospero Web Site*:
http://www.cuhk.hk/guides/earn/prospero.html

[Purl] * * *, *Purl Organization*:
http://www.purl.org

[RAP] * * *, *RDF API for PHP – RAP Library*:
http://www.wiwiss.fu-berlin.de/suhl/bizer/rdfapi/

[RSS] * * *, *Rich Site Summary – RSS 1.0 Specification*:
http://web.resource.org/rss/1.0/

[SGrid] * * *, *Semantic Grid Project*:
http://www.semanticgrid.org

[SLP] * * *, *Service Location Protocol*:
http://playground.sun.com/srvloc/
slp_white_paper.html

[SSDP] * * *, *Simple Service Discovery Protocol*:
http://www.ietf.org/internet-drafts/
draft-cai-ssdp-v1-03.txt

[FIPA] * * *, *The Foundation of Intelligent Physical Agents*:
http://www.fipa.org

[VRML] * * *, *The Virtual Reality Modeling Language*, International Standard ISO/IEC 14772-1:1997, VRML Consortium, 1997:
http://www.vrml.org/Specifications/VRML97/

[XML-RPC] * * *, *XML-RPC Specification*:
http://www.xmlrpc.com/spec

[Yahoo] * * *, *Yahoo!*:
http://www.yahoo.com

[UDDI] * * *, *Universal Description, Discovery, and Integration*:
http://www.uddi.org

[WOI] * * *, *Web Object Integration*:
http://www.objs.com/
survey/web-object-integration.htm

[W3C] * * *, *World-Wide Web Consortium*:
http://www.w3.org

[WAP Forum] * * *, *WAP Forum*:
`http://www.wapforum.org`

[WS] * * *, *Web Services*:
`http://www.webservices.org`