CONSIDERAŢII PRIVIND DEZVOLTAREA ONTOLOGIEI UNUI SISTEM INTELIGENT

Mihaela M. Oprea

Universitatea Petrol-Gaze *din Ploieşti*

Rezumat: Dezvoltarea unui sistem inteligent eficient necesită dezvoltarea unei baze de cunoştinţe eficiente, care să utilizeze un vocabular de termeni bine definiţi, cu semnificaţia specifică domeniului de aplicaţie. Lucrarea prezintă o serie de consideraţii privind dezvoltarea ontologiei unor sisteme inteligente şi analizează ontologia dezvoltată în cadrul a două sisteme, DIAGNOZA_MEDIU şi COM_ELECTRON, din subdomenii ale inteligenţei artificiale, sistemele expert şi sistemele multiagent.

Cuvinte cheie: ontologie, sistem inteligent, bază de cunoştinţe, sistem expert, sistem multiagent.

1. Introducere

Dezvoltarea unui sistem inteligent implică dezvoltarea unei baze de cunoştinţe, care să cuprindă toate cunoştinţele generale şi specifice, necesare sistemului respectiv, pentru a rezolva probleme dintr-un domeniu bine definit [1]. Procesul de construire a bazei de cunoştinţe este denumit ingineria cunoaşterii. Inginerul de cunoştinţe este cel care investighează un anumit domeniu, determinând conceptele importante din acel domeniu şi care creează o reprezentare formală a obiectelor şi relaţiilor specifice acestuia. Reprezentarea conceptelor generale şi specifice este denumită şi inginerie ontologică. În ultimii ani, s-au dezvoltat diferite metodologii de construire a unei baze de cunoştinţe, însă, indiferent de metodologia aleasă, dezvoltarea ontologiei reprezintă unul din cei mai importanţi paşi în dezvoltarea unei baze de cunoştinţe coerente, complete şi neredundante.

Metodologia generală de dezvoltare a unei baze de cunoştinţe cuprinde cinci etape:

1. stabilirea domeniului de expertiză a sistemului inteligent;
2. definirea vocabularului de predicate, funcţii şi constante (conceptele domeniului, adică);
3. codificarea cunoaşterii generale despre domeniu;
4. codificarea unei descrieri a instanţei problemei specifice;
5. testarea bazei de cunoştinţe prin interogarea procedurii de inferenţă.

Paşii 1 şi 2 reprezintă aşa numita inginerie ontologică. Ontologia proiectată la pasul 2 constă, practic, dintr-o listă de concepte ale domeniului. Dezvoltarea ontologiei unui sistem este un proces iterativ. Conceptele din ontologie trebuie să reprezinte obiecte fizice sau logice precum şi relaţiile dintre acestea, în domeniul de aplicaţie. Astfel, în propoziţiile care descriu domeniul, substantivele reprezintă obiectele, iar verbele reprezintă relaţiile dintre obiecte. Pornind de la ontologia dezvoltată, se poate crea baza de cunoştinţe a sistemului inteligent.

La ora actuală, există o serie de medii de proiectare a ontologiei (Protégé, Ontolingua, CommonKADS, Chimaera), alegerea unui astfel de mediu depinzând de facilităţile cerute de aplicaţie. În plus, dezvoltarea web-ului semantic a condus la dezvoltarea unor instrumente care facilitează proiectarea ontologiei. De exemplu, limbajul OWL este cel mai popular limbaj pentru modelarea ontologiilor în domeniul web semantic.

Proiectarea ontologiei este necesară, cu precădere, în cazul sistemelor bazate pe cunoştinţe (incluzând sistemele expert) şi a sistemele multiagent. Exemplele prezentate în lucrare ilustrează dezvoltarea ontologiei în cazul unui sistem expert, DIAGNOZA_MEDIU, şi a unui sistem multiagent, COM_ELECTRON.

Utilizarea ontologiilor reprezintă o soluţie pentru problema partajării cunoştinţelor în sistemele inteligente. Ontologiile pot fi descrise ca fiind meta date care reprezintă în mod explicit semantica datelor într-un mod direct procesabil de către calculator. Ele realizează legătura între forma şi conţinutul informaţiei. Principalul avantaj al utilizării lor îl constituie realizarea interoperabilităţii semantice în special în sistemele distribuite, eterogene.

Lucrarea cuprinde cinci secţiuni. După secţiunea introductivă, în secţiunea 2 se defineşte ontologia unui sistem inteligent şi sunt prezentate consideraţii generale, referitoare la proiectarea şi implementarea ontologiei unui sistem inteligent. Secţiunea 3 prezintă succint ontologiile a două sisteme, DIAGNOZA_MEDIU, sistem expert destinat diagnozei poluării atmosferice în zonele urbane şi COM_ELECTRON, sistem multiagent, destinat comerţului electronic pentru produse electronice de tip second hand. Ultima secţiune prezintă o serie de concluzii.

2. Ontologia unui sistem inteligent

Ontologia reprezintă specificarea explicită a unei conceptualizări pentru un anumit domeniu. Astfel, ea este o descriere (similar specificaţiei formale a unui program) de concepte şi relaţii între acestea. O conceptualizare

este o viziune simplificată, abstractă, asupra lumii pe care dorim să o reprezentăm. Fiecare sistem inteligent este relaționat în mod explicit sau implicit, cu o anumită schemă de conceptualizare. La modul pragmatic, ontologia definește vocabularul de termeni utilizați în interogări și mesaje schimbate între agenții inteligenți dintr-un sistem multiagent. Deseori, ontologiile sunt echivalate cu ierarhii taxonomice de clase, dar ele conțin, pe lângă definiții, și axiome care restricționează interpretările posibile ale termenilor definiți.

Un concept reprezintă orice, notație sau idee. Conceptualizarea este o structură semantică intensională, care codifică regulile implicite, restricționând structura unei piese a realității. Teoria ontologică este o mulțime de formule care sunt întotdeauna adevărate în raport cu o anumită conceptualizare. Ontologia se referă la oricare din termenii anterior definiți. Una din schemele uzuale de dezvoltare a unei ontologii [2] include următoarele etape: identificarea scopului, construirea ontologiei (achiziția ontologiei, codificarea ontologiei, integrarea ontologiilor existente), evaluarea și documentarea. În plus, trebuie să includă o mulțime de tehnici, metode, principii și indicații pentru fiecare etapă, precum și indicarea relațiilor care există între etape (de exemplu, ordinea recomandată, interclasarea, intrări/ieșiri).

2.1. Identificarea scopului

Această etapă presupune stabilirea domeniului ontologiei (incluzând clasele de probleme ce pot fi rezolvate) și a potențialilor utilizatori ai acesteia. Anumite ontologii sunt construite în vederea reutilizării lor. Unii cercetători consideră ontologia ca o modalitate de structurare a bazei de cunoștințe, în timp ce alții consideră că ontologia este o parte a unei baze de cunoștințe. De asemenea, există și cercetători care privesc ontologia ca fiind un inter-limbaj specific aplicației.

2.2. Construirea ontologiei

2.2.1. Achiziția

În cadrul acestei etape sunt realizate următoarele operații: identificarea conceptelor cheie și a relațiilor dintre ele în domeniul de interes, adică în domeniul de expertiză a sistemului inteligent; producerea unor definiții text neambigue pentru aceste concepte și relații; identificarea termenilor care să refere aceste concepte și relații; acceptarea de către specialiști a rezultatelor operațiilor anterioare. În [3], Skuce argumentează necesitatea unei reprezentări intermediare a conceptualizării care este mai formală decât limbajele naturale cu structură slabă, dar mai puțin formală față de limbajele formale. El propune un format specific pentru o astfel de reprezentare intermediară, care include presupuneri, justificări și definiții precise ale cuvintelor din ontologie. În mod similar, a fost dezvoltată metodologia KADS [4] care recomandă dezvoltarea unui model al domeniului, înaintea codificării bazei de cunoștințe.

2.2.2. Codificarea

Reprezentarea explicită a conceptualizării achiziționate la pasul 2.1. într-un limbaj formal constituie codificarea ontologiei. Aceasta trebuie să fie conformă cu o metaontologie. Dintre limbajele de reprezentare care au fost utilizate remarcăm Prolog, Grafuri Conceptuale, Ontolingua, L-Lilog, KL-ONE.

2.3. Integrarea

În cadrul etapelor 2.1, și 2.2, se analizează posibilitatea utilizării altor ontologii existente. Aceasta este o problemă dificilă, care a fost abordată în Ontolingua și în lucrarea [3].

2.3.1. Evaluarea

Etapa de evaluare a unei ontologii presupune stabilirea unor criterii referitoare la specificația cerințelor, la întrebările competente din domeniu și la mediul real de lucru al ontologiei dezvoltate.

2. 4. Documentarea

Una din probleme partajării ontologiilor existente este aceea că nu există o documentație care să specifice clar caracteristicile ontologiei.

În [5] sunt prezentate detaliat toate principiile dezvoltării ontologiei unui sistem inteligent. Scopul principal al dezvoltării unei ontologii este acela de a partaja cunoștințele. În ultimii ani, ele au devenit extrem de importante pentru trecerea la Web Semantic, a doua generație de World-Wide Web. În prezent, ele există sub forma unor taxonomii ca în cazul Yahoo și Amazon, pentru specificarea categoriilor de produse comercializate și

a caracteristicilor lor. Limbajul RDF (Resource Description Framework) a fost dezvoltat pentru a codifica cunoaşterea din paginile web astfel încât să aibă înţeles pentru agenţii electronici de căutare a informaţiei. În acest sens, a fost dezvoltat limbajul DAML (DARPA Agent Markup Language) prin extinderea limbajului RDF. Au fost dezvoltate o serie de ontologii cu scop general pentru diferite domenii, UNPSC, UMLS, SNOMED etc.

Aşa cum am precizat, ontologia defineşte un vocabular comun pentru cercetători care trebuie să partajeze informaţia din domeniu. Ea include definiţii interpretabile de către calculator, pentru conceptele de bază din domeniu şi pentru relaţiile dintre ele. Utilitatea definirii unei ontologii constă în partajarea cunoştinţelor din domeniu între oameni şi sisteme inteligente artificiale (de exemplu, agenţi software), reutilizarea cunoaşterii din domeniu, realizarea unor presupuneri explicite, separarea cunoaşterii din domeniu, de cunoaşterea operaţională şi analiza cunoaşterii din domeniu.

La ora actuală există o serie de sisteme de proiectare a unei ontologii. Dintre acestea, amintim Protégé [6], Ontolingua [7], Chimaera [8], CommonKADS [9], care funcţionează ca medii de editare a ontologiei. Unele sisteme utilizează metodologia orientată pe obiecte, de definire a unor ierarhii de clase şi obiecte. Astfel, clasele se mai numesc şi concepte, proprietăţile şi atributele conceptelor formează slot-urile numite şi roluri sau proprietăţi şi restricţii asupra slot-urilor numite faţete sau restricţii asupra rolurilor. Ontologia împreună cu instanţele claselor constituie baza de cunoştinţe. Aranjarea conceptelor în clase şi subclase reprezintă prima etapă a dezvoltării ontologiei în viziunea orientată pe obiecte. Următoarea etapă constă în descrierea fiecărei clase şi subclase conform cu specificaţiile anterioare, urmând a defini instanţele claselor. Cele mai importante relaţii într-o astfel de ierarhie sunt relaţiile *is-a* şi *a-kind-of*, relaţii care permit realizarea de inferenţe de către motorul de inferenţă sau componenta rezolutivă a sistemului inteligent. În figura 1, este prezentată o ierarhie de clase, în cazul unui sistem expert, destinat protecţiei mediului atmosferic [10].

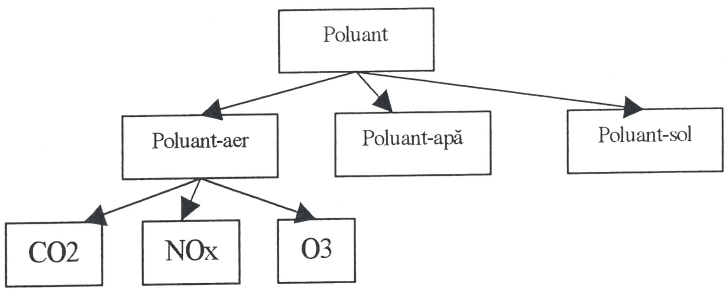

Figura 1. Ierarhie de clase

Abordarea orientată pe obiecte, a proiectării şi implementării ontologiei unui sistem inteligent, este cea mai naturală metodologie de abordare. Avantajul principal este cel al delimitării clare a conceptelor generale, de cele mai puţin generale, şi caracterizarea acestora prin atribute şi metode.

Sintetizând, dezvoltarea unei ontologii include: definirea claselor din ontologie, aranjarea claselor într-o ierarhie taxonomică (subclase-superclase), definirea slot-urilor şi descrierea valorilor permise pentru aceste slot-uri, completarea valorilor pentru slot-urile instanţelor. Crearea bazei de cunoştinţe se va realiza prin definirea instanţelor individuale ale acestor clase şi completarea valorilor specifice fiecărui slot şi a restricţiilor suplimentare pentru fiecare slot.

3. Aplicaţii

Principalele aplicaţii comerciale, care necesită proiectarea ontologiilor, sunt date de două subdomenii ale inteligenţei artificiale: sistemele expert şi sistemele multiagent, care utilizează agenţi software. În continuare, prezentăm succint ontologia proiectată pentru două aplicaţii, DIAGNOZA_MEDIU şi COM_ELECTRON.

3.1 Sisteme expert

Dezvoltarea unui sistem expert, care să fie eficient în domeniul lui de expertiză, implică dezvoltarea unei baze de cunoştinţe eficiente, care să fie completă, coerentă şi neredundantă. În plus, sistemele comerciale au nevoie de acceptul atât al unei clase largi de utilizatori, cât şi a majorităţii specialiştilor din domeniu. Astfel de deziderate implică proiectarea şi implementarea unei ontologii a domeniului de expertiză, care să permită o

înţelegere cât mai corectă a termenilor cu care lucrează sistemul şi a raţionamentelor pe care acesta le realizează. La ora actuală, majoritatea sistemelor expert au incluse ontologia specifică domeniului lor de expertiză.

În continuare, vom prezenta ontologia sistemului expert DIAGNOZA_MEDIU [10], sistem destinat diagnozei poluării atmosferice în zone urbane, având drept referinţă oraşul Ploieşti. Figura 2 prezintă schema bloc a acestui sistem. Componentele sistemului sunt următoarele: modulul de achiziţie a cunoştinţelor (MAC), baza de cunoştinţe (BC), motorul de inferenţă (MI), modulul explicativ (ME) şi interfaţa utilizator. În plus, sistemul utilizează date din bazele de date ale sistemului de monitorizare a mediului atmosferic şi are ataşate o serie de surse ale cunoaşterii domeniului de expertiză. Modulul MAC are rolul de a prelua cunoştinţele din diferite surse de cunoaştere (experţi umani, literatură de specialitate, standarde naţionale şi internaţionale de mediu etc.) şi de a le transpune sub o formă de reprezentare internă în cadrul sistemului astfel încât să fie tractabile pe un calculator. Baza de cunoştinţe reprezintă cunoştinţele din domeniu sub forma regulilor de producţie. Motorul de inferenţă este un motor de tip inductiv. Modulul explicativ oferă explicaţii ale regulilor din BC. Interfaţa utilizator are rolul de a prelua răspunsurile utilizatorului şi de a oferi răspunsurile sistemului într-o formă grafică prietenoasă.

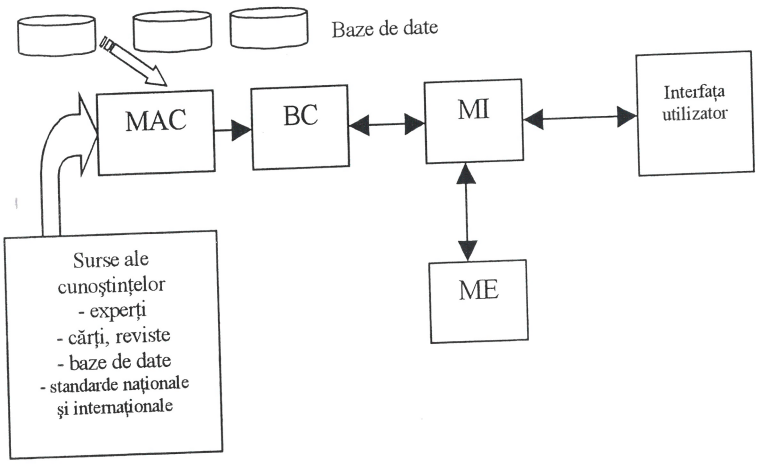

Figura 2. Schema bloc a sistemului DIAGNOZA_MEDIU

Sistemul prototip, de diagnoză a poluării atmosferice, a fost implementat în VP-Expert, un generator de sisteme expert, care foloseşte forma deductivă de reprezentare a cunoştinţelor. Ontologia domeniului de expertiză, `POLUARE_AER`, a fost proiectată şi implementată în Protégé-2000, un mediu de editare a ontologiilor scris în Java, bazat pe metodologia orientată pe obiecte, succint prezentată în secţiunea anterioară. Astfel, au fost identificaţi principalii termeni (concepte) din domeniu şi s-au creat ierarhiile de clase, împreună cu definiţiile acestor termeni. O secvenţă dintr-o ierarhie de clase a fost prezentată în figura 1. Din clasa de bază `Poluant` s-au derivat subclasele `Poluant-aer`, `Poluant-apă`, `Poluant-sol`. În figura 3, este prezentată o secvenţă din ontologia `POLUARE_AER`, într-o reprezentare prin slot-uri şi instanţe. De asemenea, pentru fiecare poluant atmosferic s-au stabilit sursele de poluare. În figura 4, sunt prezentate câteva definiţii ale unor termeni din ontologie.

Termen / Clasă	Instanţe / Slot-uri
Poluant	Poluant aer / apă / sol
Poluant aer	CO2, NOx, SO2, VOC, PM(2.5), PM(10), CO, Pb, N2O, CH4, NH3, SF6, HM, …
Sursă de poluare	Punct, arie, mobilă, biogenică
Industrie	Chimică, Petrochimică, Energetică, …
Transport	Vehicule, autobuze, trenuri, avioane
Sursă punct	Întreprinderi din ramura energetică, …
Sursă arie	Surse mai mici staţionare
Sursă mobilă	Maşini, autobuze, camioane, trenuri, avioane
Sursă biogenică	Arbori, vegetaţie, …
Standard de calitate a aerului	Naţional / internaţional – specific fiecărui poluant aer
Nivelul concentraţiilor	Specific fiecărui poluant – valoarea maximă admisibilă a concentraţiilor pentru un anumit interval de timp (CMA)

Considerăm clasa `Poluant-aer`, care are drept instanţe poluanţii atmosferici: NOx, CO2, SO2, CO, NH3, pulberi în suspensie etc. Principalele slot-uri ale clasei sunt nume poluant (`Nume`), nivelul concentraţiei

substanței poluante pentru o anumită fereastră de timp (`Nivel-Concentrație`), nivelul maxim admisibil al concentrației substanței poluante (`CMA`) pentru o anumită fereastră de timp, surse de poluare (`Surse-poluare`), măsuri de reducere/prevenire a poluării (`Măsuri-reducere-prevenire`). În figura 5, este prezentată reprezentarea clasei `Poluant-aer`.

Termen	Definiție
Sursă punct	O sursă staționară care poate fi identificată prin nume și locație.
Sursă arie	O sursă punct ale cărei emisii sunt foarte mici pentru a fi urmărite individual (de exemplu, clădiri) sau o sursă staționară difuză cum sunt cele din agricultură.
Sursă mobilă	Orice tip de vehicul sau echipament care utilizează benzină sau motor diesel, avion, tren, vapor.
Inventar de emisii	O listă a surselor punct de poluare care specifică locația, intervalul de timp și nivelul concentrațiilor monitorizate.

Figura 4. Secvență cu definiții ale unor termeni

INSTANCE:	**Pulberi-în-suspensie**
SLOT:	Nume
Fațeta:	PS
SLOT:	Nivel-concentrație
Fațeta-valoare:	0.1264 mg/m^3 (24h)
SLOT:	CMA
Fațeta-valoare:	0.15 mg/m^3 (24h)
SLOT:	Surse-poluare
Fațeta:	industrie (chimică - detergenți etc.), trafic vehicule
SLOT:	Măsuri-reducere-prevenire
Fațeta:	filtre performante, schimbare proces tehnologic, ...

Figura 6. Instanța Pulberi-în-suspensie

O instanță a clasei `Poluant-aer`, pulberi în suspensie, este reprezentată în figura 6.

3.2 Sisteme multiagent

Dezvoltarea agenților inteligenți a condus la dezvoltarea așa numitelor sisteme multiagent (SMA) care cuprind mai mulți agenți inteligenți (majoritatea agenți software) care comunică între ei și colaborează în vederea realizării unui scop comun sau a scopului lor individual. Unul din domeniile în care se utilizează agenții inteligenți este comerțul electronic, mediat de agenți. În această categorie, se încadrează sistemul COM_ELECTRON [11], care este destinat comercializării produselor electronice de tip second hand (calculatoare, imprimante, componente etc.). Sistemul a fost implementat în JADE [12], un mediu de dezvoltare a sistemelor multiagent conform cu standardul FIPA [13], scris în limbajul Java. Agenții software din cadrul sistemului COM_ELECTRON comunică între ei, astfel că ei trebuie să utilizeze termeni pe care să-i înțeleagă și mai mult, semnificația acestor termeni trebuind să fie aceeași. Cu alte cuvinte, agenții partajează o aceeași ontologie. În cazul în care ei intră în contact cu agenți care utilizează alte ontologii, pot apela la agenții mediatori (așa numiții agenți *matchmakers*), care realizează translatarea mesajelor dintr-o ontologie în alta. Figura 7 prezintă un exemplu de mesaj schimbat între doi agenți. Mesajul ACL (Agent Communication Language) stabilește cu ajutorul primitivelor de comunicare tipul de task cerut și furnizează parametrii respectivi, specificând ontologia utilizată, *OntologieProduseElectronice*, în acest caz. Exemplul prezentat este un mesaj schimbat între doi agenți, în cadrul unei licitații electronice, mediată de COM_ELECTRON. Figura 8 prezintă schema bloc a sistemului COM_ELECTRON, care utilizează o platformă multiagent, conformă cu standardul FIPA. Principalele componente ale sistemului sunt agenții de tip cumpărător (AC) și vânzător (AV), un agent care facilitează căutarea informațiilor, DF (Directory Facilitator), un canal de comunicație între agenți, ACC (Agent Communication Channel) și un agent broker, ARB (Agent Resource Broker).

```
(inform
: sender agent1
: receiver ServerProduseElectronice
: content
        (preț (licitație calculator_15) 470)
: in-reply-to round-3
: language sl
: ontology OntologieProduseElectronice
)
```

Figura 7. Exemplu de mesaj ACL

Agenții din sistemul COM_ELECTRON utilizează o aceeași ontologie conformă cu specificațiile date de standardul FIPA.

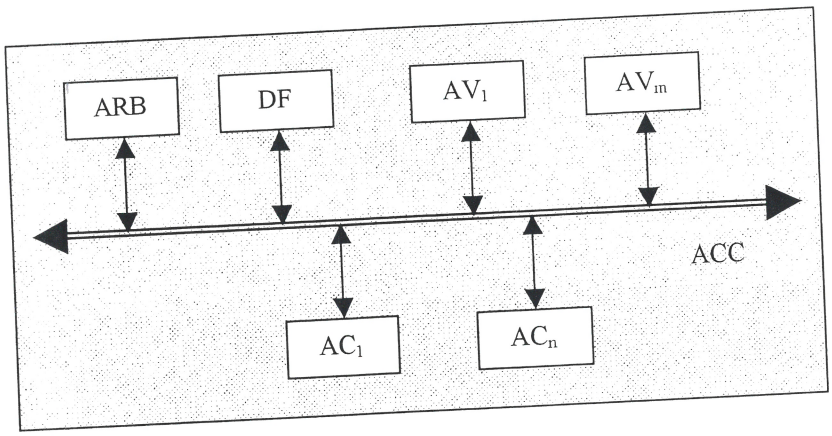

Figura 8. Schema bloc a sistemului COM_ELECTRON

JADE conține în pachetul `jade.content` metodele necesare dezvoltării unei ontologii. Mesajele ACL sunt reprezentate ca expresii cu conținut într-un anumit limbaj semantic (de exemplu, SL – Semantic Language) și sunt codificate într-un format corespunzător (de exemplu, șir de caractere). Fiecare agent JADE include un manager de conținut, care este accesibil prin metoda `getContentManager()` din clasa `Agent`. Clasa `Ontology` este inclusă în pachetul `jade.content.onto`. Verificările de ordin semantic, pentru o anumită expresie cu conținut, necesită clasificarea tuturor elementelor posibile din domeniul aplicației. Astfel, pot exista următoarele elemente: predicate, acțiuni agent, concepte, primitive, agregate, expresii referențiale de identificare și variabile. Un limbaj semantic trebuie să reprezinte și să distingă toate aceste elemente. O ontologie este o mulțime de scheme care definesc structura predicatelor, acțiunile agent și conceptele care sunt pertinente pentru domeniul aplicației.

Etapele principale ale dezvoltării ontologiei în JADE sunt următoarele: 1) definirea unei ontologii care să includă scheme pentru tipuri de predicate, acțiuni agent și concepte, 2) dezvoltarea claselor Java, corespunzătoare tuturor tipurilor de predicate, acțiuni agent și concepte din ontologie, 3) selectarea unui limbaj semantic, suportat de JADE, 4) înregistrarea ontologiei definite și a limbajului semantic, selectat pentru agent, 5) crearea și manipularea conținutului unei expresii drept obiecte Java, care sunt instanțe ale claselor dezvoltate la pasul 2) și realizarea de către JADE a traducerii acestor obiecte Java în/din șiruri de caractere sau secvențe de octeți care se potrivesc în slot-ul content al unui mesaj ACL.

O secvență din ontologia sistemului COM_ELECTRON este prezentată în figura 9. Numele ontologiei este `OntologieProduseElectronice`. Într-o primă etapă a dezvoltării sistemului COM_ELECTRON, am utilizat cinci concepte, **calculator**, **laptop**, **imprimantă**, **CD-Writer**, **scanner**, un predicat ***deține*** și două acțiuni agent, ***vinde*** și ***cumpără***, iar ca tipuri primitive, string, integer și float. Limbajul semantic utilizat este SL, furnizat de JADE.

```
package ontologieProduseElectronice;
import jade.content.onto.*;
import jade.content.schema.*;
public class OntologieProduseElectronice extends Ontology {
public static final String NUME_ONTOLOGIE="OntologieProduseElectronice";
// vocabularul de termeni
public static final String PRODUS="Produs";
public static final String NR_PROD="numarserial";
public static final String CALCULATOR="Calculator";
public static final String NUME_CALCULATOR="nume";
public static final String TIP_PROCESOR="tip-procesor";
public static final String CAPACIT_MEM_RAM="capacitate-memorie-RAM";
public static final String CAPACIT_HD="capacitate-hard-disc";
public static final String DETINE="Detine";
public static final String DETINE_PROPRIETAR="proprietar";
public static final String DETINE_PRODUS="produs";
public static final String VINDE="Detine";
public static final String VINDE_CUMPARATOR="cumparator";
public static final String VINDE_PRODUS="produs";
// restul termenilor ...
// instanta ontologiei
private static Ontology instanta=new OntologieProduseElectronice();
public static Ontology getInstance() { return instanta; }
// constructorul privat
private OntologieProduseElectronice() {
   super(NUME_ONTOLOGIE, BasicOntology.getInstance());
  try {
    add(new ConceptSchema(PRODUS), Produs.class);
    add(new ConceptSchema(CALCULATOR), Calculator.class);
    add(new ConceptSchema(LAPTOP), Laptop.class);
    add(new ConceptSchema(IMPRIMANTA), Imprimanta.class);
    add(new ConceptSchema(CD_WRITER), CD_Writer.class);
    add(new ConceptSchema(SCANNER), Scanner.class);
    add(new PredicateSchema(DETINE), Detine.class);
    add(new AgentActionSchema(VINDE), Vinde.class);
    add(new AgentActionSchema(CUMPARA), Cumpara.class);
    ConceptSchema cs = (ConceptSchema) getSchema(PRODUS);
    cs.add(NR_PROD, (PrimitiveSchema) getSchema(BasicOntology.INTEGER), ObjectSchema.OPTIONAL);
    cs = (ConceptSchema) getSchema(CALCULATOR);
    cs.addSuperSchema((ConceptSchema) getSchema(PRODUS));
    cs.add(NUME_CALCULATOR, (PrimitiveSchema) getSchema(BasicOntology.STRING));
    cs.add(TIP_PROCESOR, (PrimitiveSchema) getSchema(BasicOntology.STRING));
    cs.add(CAPACITATE_MEM_RAM, (PrimitiveSchema) getSchema(BasicOntology.INTEGER));
    cs.add(CAPACITATE_HD, (PrimitiveSchema) getSchema(BasicOntology.INTEGER));
    // restul descrierilor de structura ale conceptelor ...
}
catch (OntologyException oe) {
    oe.printStackTrace();
  }
```

Figura 9. Secvență din ontologia sistemului COM_ELECTRON

4. Concluzii

Sistemele inteligente necesită utilizarea de cunoştinţe dintr-un domeniu bine precizat. Pentru o utilizare eficientă a acestor sisteme, este necesară proiectarea şi implementarea unei ontologii care să permită în plus şi partajarea cunoaşterii cu alte sisteme inteligente şi o eventuală reutilizare a ei. La ora actuală, există o serie de limbaje de reprezentare a ontologiei, majoritatea dependente de aplicaţie. Principalele domenii în care se impune cu necesitate proiectarea ontologiilor sunt sistemele bazate pe cunoştinţe şi sistemele multiagent.

Lucrarea a prezentat două exemple de aplicaţii pentru care a fost proiectată ontologia specifică domeniului de expertiză. În cazul sistemului expert DIAGNOZA_MEDIU, ontologia `POLUARE_AER` a fost proiectată şi implementată în Protégé-2000, un mediu Java de editare a ontologiilor, iar în cazul SMA COM_ELECTRON ontologia `OntologieProduseElectronice` a fost scrisă în Java, utilizând facilităţile oferite de platforma multiagent JADE, prin pachetele `jade.content.onto` şi `jade.content.schema`.

Analiza ontologiei reprezintă un deziderat al dezvoltării oricărui sistem inteligent. De exemplu, sistemul Chimaera furnizează instrumente de diagnosticare pentru analiza ontologiilor. Analiza executată de Chimaera include o verificare a corectitudinii logice a ontologiei şi o diagnosticare a erorilor frecvente de proiectare a ontologiei. Calitatea unei ontologii se poate evalua doar prin utilizarea în aplicaţiile pentru care a fost proiectată.

Bibliografie

1. **RUSSEL, P., T. NORVIG**: Artificial Intelligence – A modern approach, Prentice Hall, 1995.

2. **USCHOLD, M., M. KING**: Towards a Methodology for Building Ontologies. În: Proc. of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, University of Edinburgh, 1995.

3. **SKUCE, D.**: Conventions for reaching agreement on shared ontologies. În: Proc. Of the 9th Knowledge Acquisition for Knowledge Based Systems Workshop, 1995.

4. **TANSLEY, D. S. W., C. C. HAYBALL**: Knowledge-based Systems Analysis and Design: a KAD Developers Handbook, Prentice Hall, 1993.

5. **GRUBER, T.**: Towards principles for the design of ontologies used for knowledge sharing. În: International Journal of Human-Computer Studies, 43(5/6), 1995, pp. 907-928.

6. **PROTÉGÉ**: http://protégé.stanford.edu, 2000.

7. **ONTOLINGUA**: http://www.ontolingua.org, 1997.

8. **CHIMAERA**: http://www.ksl.stanford.edu/software/chimaera/, 2000.

9. **COMMONKADS**: http://www.commonkads.org, 1995.

10. **OPREA, M.**: Knowledge Modelling in an Air Pollution Control Decision Support System. În: Proc. of the ECAI 2004 International Workshop Binding Environmental Sciences and Artificial Intelligence - BESAI-4, Valencia, Spain, 2004.

11. **OPREA, M**: COM_ELECTRON: Sistem de comerț electronic mediat de agenți inteligenți, raport de cercetare, Universitatea Petrol-Gaze din Ploiești, Catedra de Informatică, 2004.

12. **JADE**: http://www.jade.org, 2003.

13. **FIPA**: http://www.fipa.org, 2001.