CAPITOLUL II

PROBLEMATICA AGENŢILOR SOFTWARE

Conceptul de ***agent software inteligent*** a captat imediat imaginaţia populară. Oamenilor le place ideea de a delega sarcini complexe unor agenţi software. Agenţii pot face rezervări de bilete de avion, pot comanda cărţi dintr-un magazin on-line, să afle ultimul cântec al muzicianului favorit, să monitorizeze vînzările de produse, etc.

Agenţii software pot naviga pe Internet pentru a localiza informaţii pentru noi. Agenţii software mai sofisticaţi pot negocia pentru a cumpăra materiale pentru o fabrică, să planifice producţia fabricii, să negocieze diverse orare cu agentul software al unui client, sau să automatizeze procesul de plată.

Oricum, pentru a dezvolta agenţi inteligenţi sunt necesare cunoştinţe specifice, iar procesul de dezvoltare poate fi dificil, consumator de timp şi predispus spre erori. Sunt necesare mecanisme noi care să permită dezvoltatorilor software să construiască mai uşor aceşti agenţi software sofisticaţi.

Agenţii software inteligenţi reprezintă o nouă clasă de software care acţionează în favoarea utilizatorului pentru a găsi şi a filtra informaţii, pentru a negocia servicii, pentru a automatiza sarcini complexe, sau pentru a colabora cu alţi agenţi software pentru a rezolva probleme complexe. Agenţii software reprezintă o puternică *abstractizare* în scopul vizualizării şi structurării problemelor complexe. Procedurile, funcţiile, metodele şi obiectele sunt abstractizări software pe care dezvoltatorii de programe le utilizează zilnic. Agenţii software, totuşi, sunt o paradigmă fundamentală nouă, nefamiliară multor dezvoltatori de soft.

Ideea centrală în jurul conceptului de agent software este *delegarea*. Deţinătorul sau utilizatorul unui agent software deleagă o sarcină agentului, iar acesta va executa în mod *autonom* sarcina în favoarea utilizatorului. Agentul trebuie să fie capabil să *comunice* cu utilizatorul pentru a primi instrucţiuni din partea acestuia şi să furnizeze utilizatorului rezultatele activităţilor sale. În cele din urmă, un agent trebuie să fie capabil să *monitorizeze* starea propriului său mediu (de execuţie) şi să ia deciziile necesare pentru a-şi îndeplini sarcinile care i-au fost delegate.

Sunt două abordări de bază pentru a construi sisteme bazate pe agenţi:

- dezvoltatorul poate folosi un singur agent independent (de sine stătător);
- poate implementa un sistem multiagent.

Un agent de sine-stătător comunică doar cu utilizatorul şi furmizează întreaga funcţionalitate necesară pentru implementarea unui program bazat pe agenţi. ***Sistemele multiagent*** sunt sisteme computaţionale în care mai mulţi agenţi cooperează în scopul realizării unor sarcini care ar fi mult prea greu sau poate imposibil de realizat pentru un singur agent. De foarte multe ori, aceste sisteme multiagent sunt numite în literatură ***agenţii***. Agenţii din cadrul unei agenţii comunică, cooperează şi negociază între ei pentru a găsi soluţia unei probleme particulare.

2.1 De ce sunt Agenţii Software Inteligenţi importanţi?

În fiecare zi, dezvoltatorii software au ca sarcini construirea unor aplicaţii software din ce în ce mai complexe şi mai voluminoase. Din ce în ce mai multe companii şi corporaţii trebuie să-şi integreze sistemele informaţionale cu acelea ale furnizorilor şi cumpărătorilor lor. Noile sisteme trebuie să unifice diverse organizaţii şi platforme de aplicaţii într-un sistem global de gestiune a informaţiei, care utilizează World Wide Web şi tehnologiile distribuite bazate pe obiecte.

Sistemele din noua generaţie trebuie să furnizeze o legătură prin intermediul Internetului şi a diverse Intraneturi. Printre utilizatorii acestor sisteme vor fi: muncitori şi funcţionari din fabrici, furnizori, muncitori mobili, grupuri de muncitori, vânzători precum şi muncitori la distanţă. Platformele aplicaţiilor vor fi, cu siguranţă, foarte variate, începând de la calculatoare personale până la sisteme complexe multiprocesor. Tipul aplicaţiilor care trebuie să comunice şi să opereze unele cu altele, va fi şi acesta foarte variat ca şi complexitate a aplicaţiilor, de la programe simple până la aplicaţii pentru gestiunea întreprinderilor (aplicaţii *enterprise*).

Dezvoltarea de aplicaţii pentru aceste domenii necesită metode şi tehnici deosebit de puternice şi robuste pentru conceptualizarea şi implementarea sistemelor software. Agenţii software inteligenţi furnizează, în acest scop, o paradigmă foarte puternică pentru rezolvarea problemelor, foarte potrivită pentru dezvoltarea *aplicaţiilor enterprise* complexe.

2.2 De ce sunt Agenţii Software Inteligenţi greu de construit?

Agenţii şi tehnologia bazată pe agenţi a fost şi este în continuare un domeniu foarte activ de cercetare în domeniul Inteligenţei Artificiale. Construirea agenţilor software inteligenţi este o sarcină dificilă, care necesită mult timp şi care necesită înţelegerea unor tehnologii avansate cum ar fi reprezentarea cunoaşterii, inferenţa, metode şi protocoale de comunicare în reţele, etc. Aplicaţiile mai complexe de cele mai multe ori necesită şi expertiză în tehnologiile învăţării automate şi a planificării automate

Un dezvoltator care foloseşte un agent inteligent într-o aplicaţie nouă trebuie să decidă asupra arhitecturii generale a agentului, asupra mecanismului de raţionament (inferenţă) a agentului, asupra cunoştinţelor interne şi a reprezentării datelor, asupra protocoalelor de comunicare între agenţi precum şi a formatului mesajelor. Mai mult, dacă agentul învaţă din mediu sau de la utilizatorul acestuia, atunci sunt necesare şi tehnologii de învăţare automată. Agenţii complecşi pot necesita şi abilităţi de planificare, ceea ce înseamnă că dezvoltatorul tebuie să aleagă un algoritm de planificare şi o implementare corespunzătoare. Dacă aplicaţia necesită comunicare între mai mulţi agenţi, atunci dezvoltatorul trebuie să stabilească un protocol robust de comunicare între agenţi. Ceea ce va însemna că dezvoltatorul trebuie să aibă cunoştinţe şi expertiză în domeniul tehnologiilor de comunicaţii utilizate pentru comunicaţii între agenţi.

2.3 Caracteristici ale Agenţilor Software Inteligenţi

Înainte de a defini caracteristicile unui agent inteligent, să vedem pentru început caracteristicile unui agent software. Un agent software poate fi văzut ca o construcţie software autonomă, capabilă de a executa acţiuni fără intervenţia utilizatorului.

Două sunt cerinţele pe care ar trebui să le îndeplinească un produs software pentru a fi agent software:

- un agent trebuie să aibă abilitatea de a comunica cu un alt agent software sau uman;
- un agent trebuie să aibă abilitatea de a-şi percepe şi monitoriza mediul.

Abilitatea de comunicare implică faptul că agentul are abilitatea de a coopera cu alţi agenţi (în cele din urmă, cooperarea este necesară în scopul primirii şi recunoaşterii unei comunicaţii). Cooperarea este de maximă importanţă şi este prima raţiune în foavoarea utilizării mai multor agenţi într-o arhitectură software.

Chiar dacă mulţi cercetători privesc în mod diferit noţiunea de agent, vom defini agentul software ca fiind o componentă software care:

- se execută în mod autonom;
- comunică cu alţi agenţi sau cu utilizatorul;
- monitorizează starea mediului în care se execută.

Având definite caracteristicile generale ale unui agent software, se poate defini întrebarea legată de caracteristicile care îl fac pe un agent software să devină inteligent, altfel spus să încercăm să definim ce înseamnă *software inteligent*.

Făcând o sinteză a părerilor cercetătorilor, putem considera că pentru ca un software să fie considerat inteligent, trebuie să posede următoarele capabilităţi şi atribute:

- Să aibă capacitatea de a exploata cunoştinţele semnificative din domeniul pe care îl adresează.
- Să fie tolerant la erori şi date geşite.
- Să fie capabil să folosească simboluri şi abstractizări.
- Să fie capabil de comportament adaptiv, orientat spre scop.
- Să fie capabil să înveţe din mediul în care este situat.
- Să fie capabil de a efectua operaţii în timp real.
- Să fie capabil să comunice foloisnd limbajul natural.

Un argument deosebit de puternic este că un agent software inteligent nu trebuie să aibă, în mod necesar, toate capacităţile şi atributele descrise anterior. Spre exemplu, multe aplicaţii nu trebuie să dea răspuns în timp real, doar în timp rezonabil. Alte aplicaţii necesită doar interacţiune între agenţi, ca urmare nu necesită abilitatea de a comunica în limbaj natural.

Pe de altă parte, mulţi cercetători au arătat că pot fi construiţi agenţi inteligenţi extrem de capabili, fără a avea capacitatea de a învăţa.

Cercetătorii IBM definesc agenţii inteligenţi ca fiind: “entităţi software care efectuează numite operaţii în favoarea unui utilizator cu o anumită autonomie, şi care utilizează fie cunoştinţele fie reprezentările scopurilor şi dorinţelor utilizatorului”.

Atribute ale unui Agent Inteligent

Agent	Se execută autonom
	Comunică cu alţi agenţi sau cu utilizatorul
	Monitorizează starea mediului în care se execută
Agent Inteligent	Capabil să folosească simboluri şi abstractizări
	Capabil să exploateze cunoştinţe semnificative din domeniu
	Capabil de comportament adaptiv şi orientat spre scop
Agent	Capabil să înveţe din mediu

"într-adevăr" Inteligent	Tolerant la erori şi date greşite
	Capabil de operaţii în timp real
	Capabil să comunice în limbaj natural

Tabelul 2.1 Atributele agenţilor software

Ca urmare, agenţii "într-adevăr" inteligenţi vor avea capabilităţile unui agent software (autonomie, comunicabilitate, percepţie) precum şi capabilităţile software-ului inteligent (abilitatea de a exploata cunoaşterea şi toleranţa la erori, raţionament simbolic, învăţare şi raţionament în timp real, comunicare şi raţionament în limbaj natural).

Sunt foarte multe aplicaţii în care nu este necesar ca agenţii să aibă capabilitatea de a învăţa, ca urmare, învăţarea nu numai că este nenecesară, poate este chiar de nedorit. În astfel de aplicaţii, cerinţele pentru agenţi vor fi să aibă un comportament raţional în mediul pentru care au fost proiectaţi (de exemplu, agenţii bazaţi pe logică).

Unii cercetători sunt de părere că reţele mari de agenţi foarte simpli care comunică şi cooperează între ei, fiecare efectuând doar acţiuni simple, ar putea fi cerinţa de bază a calculului inteligent.

Trebuie totuşi remarcat faptul că astfel de componente software, posedând doar unele dintre capabilităţile amintite anterior pot fi cu greu considerate agenţi inteligenţi. Spre exemplu, un sistem expert este capabil să exploateze cunoaşterea, să folosească simboluri şi abstractizări, fiind capabil şi de comportament orientat spre scop. Totuşi, sistemele expert, în general nu se pot executa autonom, nu pot învăţa, comunica şi coopera cu alţi agenţi. Astfel încât, în cele mai multe cazuri, nu considerăm sistemele expert agenţi inteligenţi.

2.4 Ce nu este un Agent Software Inteligent

Tehnologia agenţilor inteligenţi a fost victima hiperbolizării şi exagerării. Patti Maes observa că mulţi dintre agenţii comerciali disponibili în prezent cu greu îşi justifică numele de agent, cu atât mai puţin adjectivul inteligent.

Este foarte important să avem grijă în clasificarea agenţilor software de gradul de inteligenţă. Termenul inteligenţă are foarte puternice contaţii istorice şi emoţionale. Se consideră că este mult mai bine să apreciem agentul software inteligent în termenii competenţei sale. Un agent foarte competent va manifesta toate atributele descrise în Tabelul 2.1. (până la un anumit grad). Agenţii cu o competenţă mai mică vor manifesta mai puţine dintre aceste atribute.

2.5 O clasificare a agenţilor

Sunt, probabil, atâtea clasificări ale agenţilor inteligenţi câţi cercetători sunt în domeniu. Nwana furnizează o topologie definind patru tipuri de agenţi bazaţi pe capacitatea lor de a coopera, învăţa şi acţiona autonom; numeşte aceşti agenţi *agenţi inteligenţi (smart agents), agenţi colaborativi, agenţi colaborativi care învaţă* şi *agenţi de interfaţă*. Figura 2.1 prezintă modul în care aceste trei tipuri de agenţi folosesc capabilităţile descrise anterior.

2.5.1 Agenţi Colaborativi

Agenţii colaborativi evidenţiază autonomia şi cooperarea pentru efectuarea sarcinilor prin comunicare şi posibil negociere cu alţi agenţi pentru a ajunge la înţelegeri comune.

Acest agenţi sunt folosiţi pentru a rezolva probleme distribuite, acolo unde un agent central nu este practic (spre exemplu în controlul traficului aeraian). O problemă esenţială pentru aceşti agenţi este un limbaj bine definit pentru comunicare între agenţi, cum ar fi limbajul KQML, care va fi descris în capitolul IV.

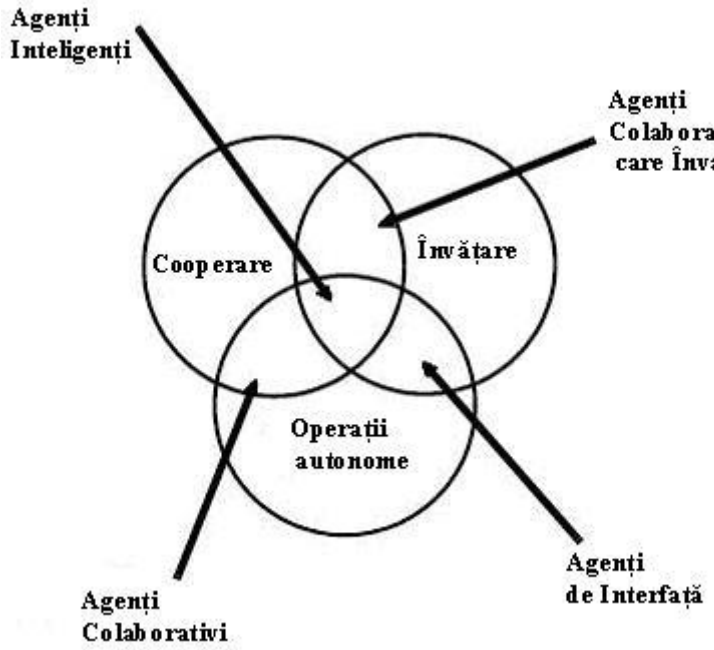

Figura 2.1 Topologia agenţilor

2.5.2 Agenţi de Interfaţă

Agenţii de interfaţă sunt autonomi şi folosesc învăţarea pentru a efectua sarcini pentru utilizatorii lor. Modelul de inspiraţie pentru această clasă de agenţi este un asistent personal care colaborează cu utilizatorul. Această clasă de agenţi este folosită pentru a implelementa asistenţi, ghizi, filtre; ei efectuează acţiuni în scopul utilizatorului sau cumpără şi vând în folosul utilizatorului.

2.5.3 Agenţi Mobili

Agenţii mobili sunt procese computaţionale capabile să se mişte în cadrul unei reţele (posibil reţele de dimensiuni foarte mari cum ar fi Internet-ul sau World Wide Web), interacţionând cu gazde străine, culegând informaţii pentru utilizator şi apoi să se întoarcă la utilizator după ce au efectuat sarcinile cu care au fost delegaţi. Agenţii mobili sunt implementări de programe la distanţă (*remote programs*) - programe dezvoltate pe o maşină şi trimise pe o altă maşină pentru a fi executate. Foarte multe dintre aspectele care trebuie abordate în programarea la distanţă trebuie deasemenea abordate şi în probleme legate de agenţi mobili. Aceste aspecte includ:

- denumirea programelor - asignarea de nume programelor agent pentru a le putea distinge;
- autentificarea programelor - autentificarea implementatorului programului agent;
- migrarea programelor - mutarea unui program de pe o maşină pe alta;
- securitatea programelor - asigurarea faptului că un program nu produce pagube pe maşina pe care se execută.

Literatura de specialitate a tratat, în multe cazuri, agenţii mobili ca fiind singura întruchipare a agenţilor inteligenţi. Dar, aşa cum arată Nwana "mobilitatea nu este o condiţie necesară şi nici suficientă pentru agenţi."

2.5.4 Agenţi de Informaţii (Information Agents)

Una dintre cele mai populare utilizări ale agenţilor inteligenţi este pentru găsirea, analizarea şi regăsirea unor cantităţi mari de informaţii. Agenţii pentru obţinerea de informaţii sunt instrumente pentru a uşura gestionarea cantităţilor uriaşe de informaţie disponibile în cadrul unor reţele, cum ar fi WWW şi Internet. Agenţii pentru obţinerea de informaţii accesează o reţea căutând informaţii particulare, filtrându-le şi apoi reîntorcându-se la utilizator. Aceşti agenţi folosesc, de obicei, protocolul HTTP pentru a accesa informaţiile, deşi ar putea să se folosească şi de avantajele unor limbaje de comunicare între agenţi, cum ar fi KQML.

2.5.5 Sisteme agent eterogene

Sistemele agent eterogene se referă la o colecţie de doi sau mai mulţi agenţi având diferite arhitecturi. Datorită varietăţii foarte mari a domeniilor de aplicare, este puţin probabil că aceeaşi arhitectură de agenţi va fi folosită exclusiv în toate domeniile; pentru fiecare domeniu, va fi aleasă cea mai potrivită arhitectură. Agenţii în aceste sisteme eterogene vor comunica, coopera şi interopera. O cerinţă majoră pentru interoperare între agenţi este existenţa unor limbaje de comunicare între agenţi, care vor permite diferitelor tipuri de agenţi software să comunice.

După cum am arătat deja, în abordarea din această carte, considerăm un agent inteligent ca fiind orice program software care poate opera autonom (fără intervenţia utilizatorului), poate învăţa obiceiurile şi preferinţele utilizatorului, poate primi instrucţiuni de la utilizator şi/sau alţi agenţi, să comunice cu aceştia, precum şi să efectueze sarcinile pe care utilizatorul sau alţi agenţi i le-au dat. Agenţii inteligenţi sunt specializaţi pentru a furniza un nivel înalt de competenţă şi capabilitate în domenii specifice. Un agent inteligent trebuie să fie capabil să dezvolte planuri şi scopuri de nivel înalt şi să încerce să satisfacă aceste scopuri.

2.6 Caracteristici ale Agenţilor Inteligenţi

Termenul **agent** este deseori folosit în literatura de specialitate din domeniul bazelor de date, al sistemelor de operare, al reţelelor, pentru a indica un proxy pentru un proces de calcul sau o locaţie (site) - o tranzacţie, un proces, un router de reţea, care poate fi folosit pentru a interacţiona cu entitatea corespunzătoare.

Din acest punct de vedere un agent este o **interfaţă formală** cu un sistem oarecare.

Exceptând taxonomia prezentată în secţiunea 2.5, în literatura de specialitate întâlnim clasificări ale agenţilor inteligenţi după mai multe criterii.

1) **Clasificare după sarcina pe care o au agenţii**. Conform acestui criteriu se deosebesc următoarele tipuri de agenţi:
   - agenţi de interfaţă;
   - asistenţi personali pentru obţinere de informaţii;
   - agenţi pentru *data mining* (extragerea informaţiilor semnificative din date);
   - agenţi pentru filtrarea mesajelor;
   - agenţi pentru planificarea activităţilor;
   - agenţi pentru stabilirea (planificarea) întâlnirilor;
   - agenţi pentru gestionarea informaţiilor.
2) **Clasificare după modul de comportament**. Conform acestui criteriu se deosebesc următoarele tipuri de agenţi:
   - agenţi reactivi, deliberativi, bazaţi pe scop, bazaţi pe utilităţi (raţionali), creativi;
   - agenţi competitivi, cooperativi, antagonici;
   - agenţi mobili / staţionari;
   - agenţi controlaţi / autonomi;
   - agenţi preprogramaţi / adaptivi / care învaţă;
   - agenţi cu auto-replicare / care evoluează.
3) **Clasificare după paradigma Inteligenţei Artificiale**. Conform acestui criteriu se deosebesc următoarele tipuri de agenţi:
   - agenţi procedurali / declarativi;
   - agenţi simbolici / hibrizi;
   - agenţi determinişti logici (monotoni, nemonotoni) / probabilistici;
   - agenţi cu reprezentare explicită / implicită;
   - agenţi ce realizează inferenţe logice, probabilistici, analogici, inductivi.

În ceea ce priveşte posibile modele pentru sisteme de agenţi (agenţii), în literatura de specialitate se propun următoarele:

- **sisteme raţionale: logice**: cum ar fi sistemele de inferenţă bazate pe baze de cunoştinţe, sisteme pentru menţinerea raţiunii şi adevărului;
- **sisteme raţionale: economice**: agentul păstrează preferinţe asupra stărilor mediului şi selectează acţiuni care îi maximizează preferinţele;
- **sisteme sociale**: cooperare, competiţie, coordonare;
- **sisteme interactive**: interacţiunile presupun comunicare între agenţi, prin intermediul unor limbaje paratajate (sintaxă, semantică);
- **sisteme adaptive**: agenţii învaţă prin interacţiunea cu mediul în care sunt situaţi (care poate include şi alţi agenţi);
- **sisteme care evoluează**: populaţiile de agenţi auto-replicativi se adaptează la mediul în care sunt situaţi prin **evoluţie**.

Proiectarea sistemelor bazate pe agenţi a fost principalul obiect de studiu în Inteligenţa Artificială de-a lungul ultimelor decenii.

Russel şi Norvig propun următoarele exemple de modele de agenţi:

- agenţi reflex;
- agenţi pentru rezolvare de probleme;
- agenţi deliberativi;
- agenţi raţionali;
- agenţi bazaţi pe baze de cunoştinţe;
- agenţi de planificare;
- agenţi reactivi;

- agenţi proactivi;
- agenţi adaptivi;
- agenţi care învaţă.

Posibilele abordări pentru modelarea agenţilor sunt: programarea directă, învăţarea automată şi tehnicile evolutive.

În ceea ce priveşte limbajele folosite în realizarea sistemelor bazate pe agenţi, amintim:

- limbaje de implementare a agenţilor (Java, Lisp, Agent0, etc);
- limbaje de comunicare şi coordonare între agenţi (KQML);
- limbaje pentru modelarea şi descrierea legilor şi comportamentelor mediului (incluzând şi cele ale altor agenţi);
- limbaje pentru reprezentarea şi raţionamentul cu şi despre cunoştinţe (incluzând fapte despre mediu, intenţii şi păreri ale altor agenţi, etc);
- limbaje de specificare a agenţilor.

2.7 Sisteme MultiAgent

Un Sistem MultiAgent (SMA) este format din mai mulţi agenţi care interacţionează unul cu celălalt. În cazul general, agenţii acţionează în folosul unor utilizatori cu diferite scopuri şi motivaţii. Pentru a interacţiona cu succes, este necesară abilitatea de a **coopera**, a se **coordona** şi a **negocia** între ei, aşa cum marea majoritate a oamenilor o fac.

Conform definiţiei date de Feber, termenul de "**sistem multiagent**" este aplicat unui sistem care cuprinde următoarele elemente :

- Un mediu E, care este, un spaţiu de dimensiuni mari, în general.
- Un set de obiecte, O. Fiecare obiect are asociat o poziţie în mediul E. Aceste obiecte pot fi create, percepute, distruse şi modificate de agenţi.
- Un ansamblu de agenţi, A, care sunt obiecte specifice.
- Un ansamblu de relaţii, R, care fac legătura între obiecte (şi astfel între agenţi).
- Un ansamblu de operaţii, Op, cu ajutorul cărora agenţii din A pot percepe, transforma, modifica, manipula obiectele din O.
- Operatori care specifică condiţiile în care pot fi aplicate operaţiile şi rezultatul realizării acestora.

Un exemplu de un sistem cu mai mulţi agenţi este prezentat în Figura 2.2.

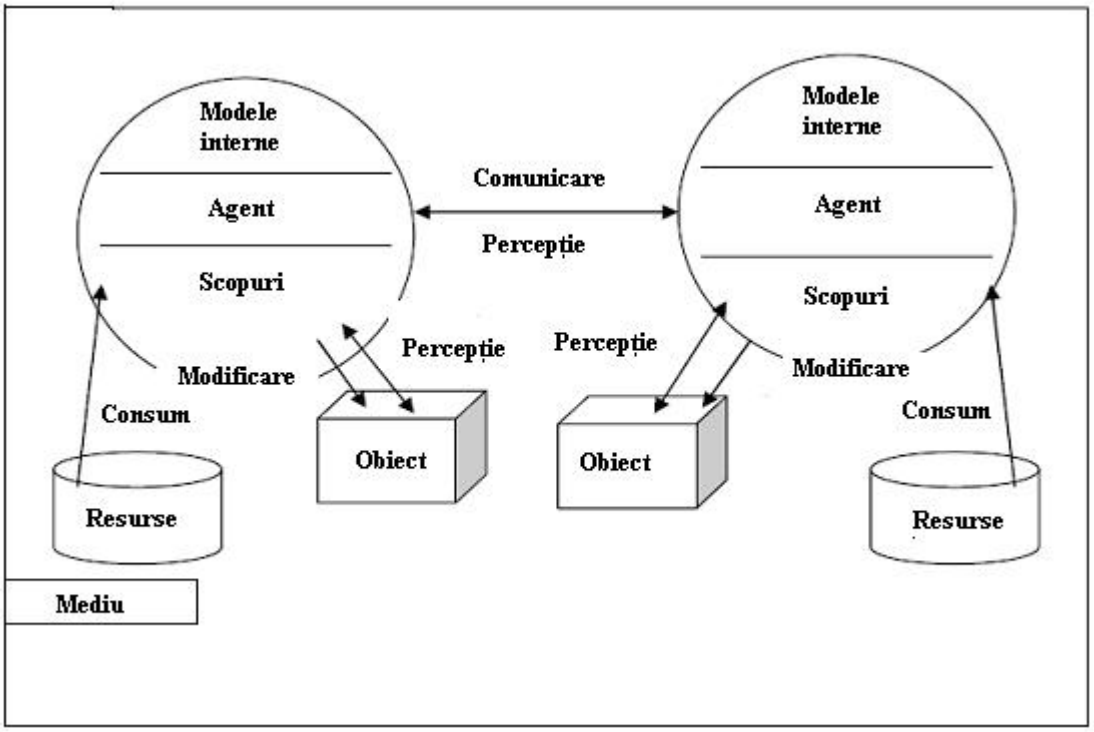

Figura 2.2. Exemplu de sistem de agenţi distribuiţi

Pe de o parte, sistemele multiagent permit rezolvarea distribuită şi colaborativă a problemelor prin intermediul unor colecţii de agenţi care se autoorganizează în mod dinamic. Pe de altă parte permit o abordare modulară şi extensibilă la proiectare sistemelor informaţionale complexe.

În consecinţă, următoarele aspecte sunt esenţiale în cadrul unui sistem cu mai mulţi agenţi:

- **comunicarea** între agenţi: comunicarea cunoştinţelor, intenţiilor, opiniilor agenţilor;
- **colaborarea** între agenţi prin negociere între agenţi raţionali cu interese proprii (*self-interested agents*);
- **coordonare** şi **control** în sistemele multiagent.

Cooperarea între agenţi în cadrul unui sistem multiagent se poate realiza prin: grupare, comunicare, specializare, partajare de sarcini şi resurse, coordonarea acţiunilor, rezolvarea conflictelor prin arbitrare şi negociere.

Organizaţiile de sisteme multiagent sunt definite printr-un ansamblu de clase de agenţi caracterizaţi de rolurile care le-au fost asignate şi de o mulţime de relaţii abstracte (subordonare, conflict, etc) între roluri.

Pentru organizaţiile de sisteme multiagent sunt posibile următoarele abordări: orizontale, ierarhice, democraţii, anarhii creative, societăţi guvernate de convenţii (protocoale de negociere), colonii de furnici, sisteme imune, sisteme evolutive.

Cele două probleme cheie legate de sistemele multiagent sunt:

- **proiectarea agenţilor**. Cum să construim agenţi care să fie capabili de acţiuni independente şi autonome în scopul executării cu succes a sarcinilor pe care au fost delegaţi să le efectueze?
- **proiectarea societăţii**. Cum să construim agenţi care să fie capabili să interacţioneze (coopereze, coordoneze, negocieze) cu alţi agenţi în scopul executării cu succes a sarcinilor pe care au fost delegaţi să le efectueze,

mai ales în cazurile în care ceilalţi agenţi nu au neapărat aceleaşi interesuri/scopuri?

Cele două probleme enunţate anterior pot fi privite ca fiind perspective *micro* şi *macro* ale sistemelor multiagent.

Domeniul sistemelor multiagent poate fi privit din cel puţin două puncte de vedere:

- agenţii ca o paradigmă a *ingineriei soft*;
- agenţii ca mecanisme pentru înţelegerea societăţilor umane (sistemele multiagent furnizează mecanisme noi şi inedite pentru simularea societăţilor, mecanisme care ar putea clarifica diferitele tipuri de procese sociale).

Iată câteva dintre cele mai importante caracteristici ale sistemelor multiagent:

- Agenţii au posibilităţi limitate şi deţin informaţii incomplete.
- Controlul sistemului este distribuit.
- Informaţia este descentralizată
- Calculul este asincron.

Ne-am putea întreba de ce avem nevoie să construim un sistem distribuit complex în loc să folosim tehnicile de rezolvare a problemelor din IA standard (tradiţională). Avantajele utilizării sistemelor distribuite au fost punctatete de Moulin şi Chaib-Draa:

- rezolvarea problemelor este mai rapidă – căutarea soluţiilor este efectuată în paralel;
- se reduce comunicarea – doar soluţiile parţiale se schimbă între agenţi, în locul trimiterii soluţiilor neprelucrate pentru centralizare;
- creşte flexibilitatea;
- creşte exactitatea – agenţii se pot substitui între ei în cazul eşuării unui agent.

Observăm din cele prezentate până acum că cele mai importante componente ale unui sistem multiagent sunt: agenţii, mediul pe care îl populează şi interacţiunile dintre aceştia. Prin modificarea acestor componente putem crea diferite tipuri de sisteme multiagent.

Din punctul de vedere al arhitecturii agenţii folosiţi în SMA pot fi de două tipuri: agenţi reactivi şi agenţi cognitivi. Agenţii cognitivi cunosc foarte bine mediul în care se găsesc şi folosesc aceste cunoştinţe pentru realizarea scopului, în timp ce agenţii reactivi nu-şi cunosc deloc mediul.

2.8 Agenţi Inteligenţi şi Obiecte

Obiectele, după cum se ştie, sunt definite ca fiind nişte *entităţi* computaţionale care *încapsulează* o serie de stări, sunt capabile să efectueze acţiuni, sau *metode* asupra acestor stări şi să comunice prin transmiterea de mesaje.

În timp ce între *agenţi inteligenţi* şi *obiecte* există similarităţi evidente, sunt şi o serie de diferenţe semnificative.

1. *Gradul de autonomie* al obiectelor şi al agenţilor. După cum se ştie, principala caracteristică a programării obiectuale este încapsularea – ideea că obiectele au control asupra stării lor interne. Datorită posibilităţii de a avea atribute (atât variabile de instanţă cât şi metode) **private** şi **publice**, un obiect poate fi considerat ca exprimându-şi autonomia asupra propriei stări: are control asupra acesteia. Dar un obiect nu-şi poate exprima controlul asupra *comportamentului* său (spre exemplu, dacă o metodă **m** este disponibilă celorlalte obiecte, ea poate fi

apelată de acestea ori de câte ori vor – odată ce un obiect îşi declară o metodă **publică**, nu mai poate avea controlul dacă acea metodă a fost sau nu executată).

Nu la fel este situaţia într-un sistem *multiagent*. Nu se poate admite faptul că agentul *i* va executa o acţiune (metodă) *a* doar pentru faptul că un alt agent *j* vrea acest lucru – *a* poate că nu este varianta optimă pentru *i* .

Astfel, dacă *j* cere agentului *i* să efectueze acţiunea *a, i* poate satisface sau nu cererea primită. Controlul deciziei despre executarea sau nu a unei acţiuni, este diferită în sistemele "cu obiecte" sau "cu agenţi". În cazul agenţilor, decizia este a agentului care recepţionează cererea, pe când în cazul obiectelor, decizia este a obiectului care lansează cererea. Această diferenţă între obiecte şi agenţi este sugestiv redată de expresia:

"Obiectele execută un lucru în mod liber, pe când agenţii execută un lucru pentru bani."

2. *Gradul de flexibilitate* al comportamentului. Modelul obiectual standard, nu prea are multe de spus despre cum să se construiască sisteme care să integreze un astfel de tip de comportament. Gradul de flexibitate se referă, în esenţă, la trei lucruri:
   - *reactivitate*: agenţii inteligenţi au abilitatea de a-şi percepe mediul şi să răspundă oportun la schimbările ce apar, în scopul atingerii obiectivelor pentru care au fost proiectaţi;
   - *sunt activi*: agenţii inteligenţi sunt capabili să dea dovadă de comportament specific, luând iniţiative în vederea atingerii obiectivelor – aşa numitul "*comportament orientat spre scop*";
   - *abilitate socială*: agenţii inteligenţi sunt capabili să interacţioneze cu alţi agenţi (posibili umani – poate fi cazul Sistemelor Expert), în vederea atingerii obiectivelor.
3. Se consideră că fiecare agent are propriul său *fir* (thread) de execuţie (control) – într-un model obiectual standard există un singur fir de control în sistem.

Recent a apărut noţiunea de *concurenţă* în programarea obiectuală, cu aşa numitele *obiecte active*, care în esenţă sunt agenţi, dar care nu dau dovadă, în mod necesar, de comportament autonom *flexibil*.

"Un obiect activ este unul care incorporează propriul său fir de control. Obiectele active sunt în general autonome, ceea ce înseamnă că pot da dovadă de un anumit comportament fără a fi acţionate de un alt obiect. Obiectele pasive, în schimb, pot explicit să-şi schimbe starea doar la o acţiune a unui alt obiect".

În concluzie, există trei deosebiri fundamentale între punctul de vedere tradiţional, asupra unui obiect, şi punctul de vedere asupra unui agent:

- agenţii incorporează mai puternic decât obiectele noţiunea de *autonomie*, iar în particular, pot decide pentru ei înşişi dacă să execute sau nu o acţiune solicitată de un alt agent;
- agenţii sunt capabili de comportament *flexibil* (în cei trei termeni în care a fost definită flexibilitatea anterior), ceea ce lipseşte modelului obiectual;
- un sistem multiagent e în mod inerent *multi-"thread"*, în care se consideră că fiecare agent are cel puţin un fir de control.

2.9 Agenţi Inteligenţi şi Sisteme Expert

2.9.1 Sisteme Expert (SE)

Sistemele Expert (SE) sunt produse ale Inteligenţei Artificiale, ramură a ştiinţei calculatoarelor ce urmăreşte dezvoltarea de programe şi sisteme inteligente.

Momentul de naştere a sistemelor expert a fost în 1960, când NASA a hotărât trimiterea unui vehicul pe Marte, unul dintre scopuri fiind cercetarea structurii chimice a solului acestei planete. Programul realizat în acest scop, numit DENDRAL, a fost considerat ca fiind primul sistem expert, tocmai datorită înglobării experienţei chimiştilor, utilizând atât deducţii empirice cât şi raţionamente ştiinţifice.

Astfel, se dezvoltă sistemele expert, ca fiind programe ce urmăresc cunoştinţele, raţionează pentru obţinerea rezultatelor în activităţi dificile, ce sunt în general efectuate doar de experţii umani.

Cercetătorii în Inteligenţă Artificială au dezvoltat, din momentul apariţiei acesteia, o serie de tehnici care separă cunoştinţele cu care se operează într-un program, de procedurile care manipulează aceste cunoştinţe.

Astfel, a apărut o nouă orientare în programare, *programarea simbolică*, care, spre deosebire de programarea convenţională este orientată nu atât spre procesarea numerică, cât spre *procesarea simbolică*.

Având ca şi caracteristică uşurinţa de a explica raţionamentul efectuat, bazându-se pe cunoştinţe simbolice structurate într-o *bază de cunoştinţe*, folosind *euristici* în loc de algoritmi, şi punând în evidenţă inteligenţa experţilor umani, *sistemele expert* devin sisteme de Inteligenţă Artificială comparabile, sau poate superioare ca performanţă cu experţii umani.

Ca şi concluzie, *sistemele expert* reprezintă o altă direcţie de studiu în domeniul sistemelor inteligente, putând fi considerate ca fiind dezvoltări în noile câmpuri ale achiziţiei de cunoştinţe ca forme specializate de învăţare, în care cunoştinţele sunt achiziţionate direct de la expert.

2.9.1.2 Problematica SE

SE fac parte dintr-o gamă de instrumente indinspensabile pentru realizarea de instrumente automate sau interactive capabile să realizeze sarcini complexe. Sistemele Expert pot fi considerate programe soft care operează intr-un sistem sofisticat la fel cum o fac şi experţii umani. Cu alte cuvinte, acestea sunt aplicaţii software care imită judecata experţilor umani dintr-un domeniu bine conturat, realizând o expertiză într-un anumit domeniu.

Sistemele expert sunt de fapt sisteme de programare bazate pe tehnicile Inteligenţei Artificiale, care înmagazinează cunoştinţele experţilor umani dintr-un domeniu bine definit şi apoi le folosesc, pentru rezolvarea problemelor din acest domeniu.

Sistemele expert sunt folosite în foarte multe domenii: medicină, economie, finanţe, industrie. O clasificare a sistemelor expert ar putea fi următoarea:

- Sisteme expert pentru diagnostic.
- Sisteme expert pentru reparaţii.
- Sisteme expert pentru instruire.
- Sisteme expert de inerpretare.
- Sisteme expert pentru prognoză.
- Sisteme expert pentru proiectare si planificare.
- Sisteme expert de monitorizare şi control.

Profesorul Edward Feigenbaum de la Universitatea din Stanford, unul din cei mai reprezentativi cercetători în domeniul sistemelor expert, definea un sistem expert astfel: "*un program inteligent care utilizează cunoştinţe şi proceduri de inferenţă*

pentru a rezolva probleme care sunt suficient de complexe şi dificile pentru a necesita expertiză umană semnificativă pentru a fi soluţionate".

Cunoştinţele utilizate (aşa numita *bază de cunoştinţe*) precum şi procedurile de inferenţă (ce constituie *motorul de inferenţă*) utilizate într-un sistem expert într-un anumit domeniu, pot fi considerate ca fiind un model de expertiză a celor mai buni practicieni din domeniu.

2.9.1.3 Componentele de bază ale unui SE

Cele trei componente de bază ale unui sistem expert sunt: baza de cunoştinţe, mecanismul (motorul) de inferenţă, baza de fapte.

Baza de cunoştinţe este reprezentată ca o structură de date ce conţine ansamblul cunoştinţelor specializate introduse de către expertul uman. Este o componentă cognitivă, cunoaşterea fiind memorată într-un spaţiu special organizat, spaţiu în care se descriu situaţii evidente, fapte reale sau ipotetice, precum şi euristici. În cazul memorării cunoaşterii sub formă de reguli de producţie, baza de cunoştinţe conţine două componente: baza de fapte şi baza de reguli, iar motorul de inferenţă se mai numeşte interpretor de reguli.

Mecanismul (motorul) de inferenţă poate fi denumit şi interpretor deoarece, el preia cunoştinţele din baza de date şi le utilizează pentru construirea unui raţionament, pentru a forma inferenţe şi a trage concluzii. Acesta urmăreşte o serie de obiective majore cum ar fi: alegerea strategiei de control în funcţie de problema curentă, executarea acţiunilor care sunt prevazute în planul de rezolvare.

Baza de fapte este reprezentată de o memorie auxiliară ce conţine toate datele utilizatorului (fapte iniţiale ce descriu enunţul problemei de rezolvat) şi rezultatele intermediare produse în cursul procedurii de deducţie.

Dintre metodele cele mai importante de reprezentare a cunoaşterii, amintim două: reprezentarea cunoaşterii folosind reguli de producţie şi metodele bazate pe logică.

Metoda reprezentării cunoaşterii folosind **reguli de producţie** este cea mai folosită metodă de reprezentare a cunoaşterii Sistemelor Expert, fiind foarte eficace pentru reprezentarea recomandărilor, directivelor sau strategiilor.

Utilizarea regulilor la codificarea cunoştinţelor este o problemă de o deosebită importanţă, deoarece sistemele de raţionare bazate pe reguli au jucat un rol foarte important în evoluţia sistemelor cu inteligenţă artificială.

În locul reprezentării cunoştinţei într-un mod declarativ (ca o serie de lucruri care sunt adevărate), regulile pot specifica ce ar trebui să facă sau ce se poate include în diferite situaţii. Un sistem bazat pe reguli, conţine o serie de reguli DACĂ-ATUNCI (IF-THEN), o serie de fapte şi un interpretor ce controlează aplicarea regulilor.

Există două tipuri de sisteme bazate pe reguli: *sisteme cu înlănţuire înainte*, şi *sisteme cu înlănţuire înapoi*. În primul model, se începe cu o serie de fapte iniţiale, şi se continuă să se folosească regulile care derivează noi concluzii din acele fapte. În modelul înlănţuirii înapoi, se porneşte cu o serie de scopuri (goals) care se încearcă a fi demonstrate, şi se caută reguli care ar permite să deducă acele scopuru. Primul model, este numit şi *data-driven* iar al doilea *goal-driven*.

Reprezentarea exactă a cunoaşterii dintr-un domeniu particular necesită un număr mare de reguli, pentru a putea fi surprinse toate detaliile. Numărul regulilor este direct proporţional cu complexitatea cunoaşterii. Pe lângă autonomia ridicată pe

care o conferă acest mod de reprezentare, regulile pot fi privite ca piese de cunoaştere independente, ele reprezintă şi o modalitate eficientă de procesare a cunoaşterii.

Metoda bazată pe logică este una dintre cele mai importante metode de reprezentare a cunoaşterii, care ne permite să reprezentăm fapte complexe despre lume, şi să derivăm noi fapte într-un mod care ne garantează că, dacă faptele iniţiale sunt adevărate, atunci si concluziile sunt la fel. Logica predicatelor de ordinul întâi (cea mai folosită), are o sintaxă bine definită, reguli şi semantici de inferenţă.

Propoziţiile cu valoare de adevăr, numite şi *aserţiuni*, se referă la obiecte, concepte, evenimente din domeniul problemei, sunt reprezentate cu tehnici specifice predicatelor de ordinul I, calculul acestora, permiţând descrierea atributelor obiectelor, conceptelor, evenimentelor.

Limbajele de programare logică (spre exemplu Prolog) ar fi foarte potrivite pentru pentru implementarea componentelor de bază ale unui sistem expert. După cum se ştie, în structura de control folosită de interpretorul Prolog se aplică raţionamentul înapoi pentru a demonstra o concluzie, ca urmare motorul de inferenţă al sistemului expert nu trebuie implementat, el fiind incorporat în interpretorul Prolog.

Ca urmare, dacă ar fi să folosim limbajul Prolog pentru a descrie un mini-sistem expert, baza de cunostinţe ar fi formată din reguli Prolog, iar baza de fapte ar fi formată din fapte Prolog.

Calculul predicatelor este preferat a fi folosit, datorită structurii sale şi puterii de inferenţiere. Logica are o importanţă deosebită în Sistemele Expert în care motoarele de inferenţă raţionează de la premise către concluzii. Un termen semnificativ pentru programarea logică şi Sistemele Expert este acela de sisteme cu raţionament automat (Automated Reasoning Systems).

2.9.1.4 De ce un Sistem Expert?

Răspunsul la această întrebare este unul simplu: introducerea unui sistem expert va ameliora calitatea deciziilor luate, datorită faptului că acesta deschide largi posibilităţi raţionamentelor, prin exploatarea unui număr mult mai mare de variante decât cele posibile în mintea omului, aceasta datorându-se şi faptului că ele au o capacitate mult mai mare de memorare decât un specialist uman.

După cum am văzut în secţiunea anterioară, sistemele expert sunt sisteme bazate pe cunoştinţe şi care raţionează euristic pentru obţinerea rezultatelor într-o activitate dificilă, care în mod uzual este efectuată de experţi umani.

Baza de cunoştinţe a unui sistem expert e constituită din *fapte* şi *euristici*. *Faptele* constituie "corpul" de informaţii ce este accesibil, disponibil (în mod public) şi în general acceptat de majoritatea experţilor din domeniu. *Euristicile* sunt în mare parte reguli private (caracteristici sistemului expert pe care-l definesc) şi reprezintă reguli de raţionament *plauzibil*, reguli ce asigură un comportament "bun" al sistemului, caracterizând nivelul de *decizie* al acestuia.

Se poate spune că sistemele expert sunt programe ce exploatează intensiv *cunoaşterea* (din diverse domenii de activitate), conţinând o multitudine de cunoştinţe specifice. *Regulile euristice* specifice sistemului indică spre aspectele "cheie" ale problemei particulare şi spre manipularea descrierilor simbolice, necesare pentru a raţiona despre cunoştinţele ce îi sunt furnizate.

Cele mai semnificative diferenţe între un *sistem expert* şi un *program convenţional* sunt următoarele:

- un sistem bazat pe cunoştinţe (ce exploatează cunoaşterea) are un foarte mare grad de *interactivitate*;

- inginerii de cunoştinţe şi experţii întreţin sistemele expert, în timp ce programele convenţionale sunt întreţinute de programatori;
- baza de cunoştinţe a unui sistem expert poate fi consultată şi este uşor de modificat;
- structura programelor convenţionale se bazează pe algoritmi, în timp ce structura sistemelor bazate pe cunoştinţe se bazează pe euristici.

Pe baza celor menţionate mai sus, rezultă de fapt diferenţele între *programarea convenţională* şi *programarea simbolică*.

Programarea convenţională	**Programarea simbolică**
▪ algoritmi	▪ euristici
▪ bază de date adresabilă numeric	▪ bază de cunoştinţe simbolice
▪ orientare spre procesare numerică	▪ orientare spre procesare simbolică
▪ procesare secvenţială	▪ grad foarte ridicat de interactivitate a procesării
▪ justificări în timpul execuţiei imposibile	▪ justificări în timpul execuţiei posibile (simple)

În fapt, cele trei idei fundamentale ale Inteligenţei Artificiale sunt:
- noi modalităţi de reprezentare a cunoaşterii;
- căutare euristică;
- separarea cunoştinţelor de mecanismul de inferenţă şi control.

Utilizând euristici, programele de Inteligenţă Artificială sugerează acţiuni în situaţii în care programele covenţionale nu o pot face. Tehnicile euristice de programare dezvoltă astfel varietatea de sarcini pe care calculatoarele o pot efectua, permiţând programatorilor să dezvolte programe mult mai vaste şi mai complexe, capabile să analizeze probleme destul de neclare (incerte) şi să sugereze posibile soluţii.

Importanţa utilizării Sistemelor Expert rezidă din o serie de carateristici generale ale acestora:
- Sistemele Expert pot rezolva probleme dificile tot aşa de bine sau chiar mai bine decât experţii umani.
- Raţionează euristic, utilizând ceea ce specialiştii consideră a fi metode empirice eficiente.
- Interacţionează cu omul în diferite moduri, inclusiv prin folosirea limbajului natural.
- Manipulează descrierile simbolice ale cunoştinţelor şi raţionează asupra acestor descrieri simbolice.
- Funcţionează cu date eronate şi cu reguli de raţionament imprecise.
- Iau în consideraţie în mod simultan ipoteze conflictuale.
- Explică de ce pun o anumită întrebare.
- Îşi argumentează concluziile.

Figura 2.3 ilustrează legăturile existente între domeniul *Inteligenţei Artificiale* şi *Ingineria cunoaşterii* (domeniu din care fac parte *Sistemele Expert*).

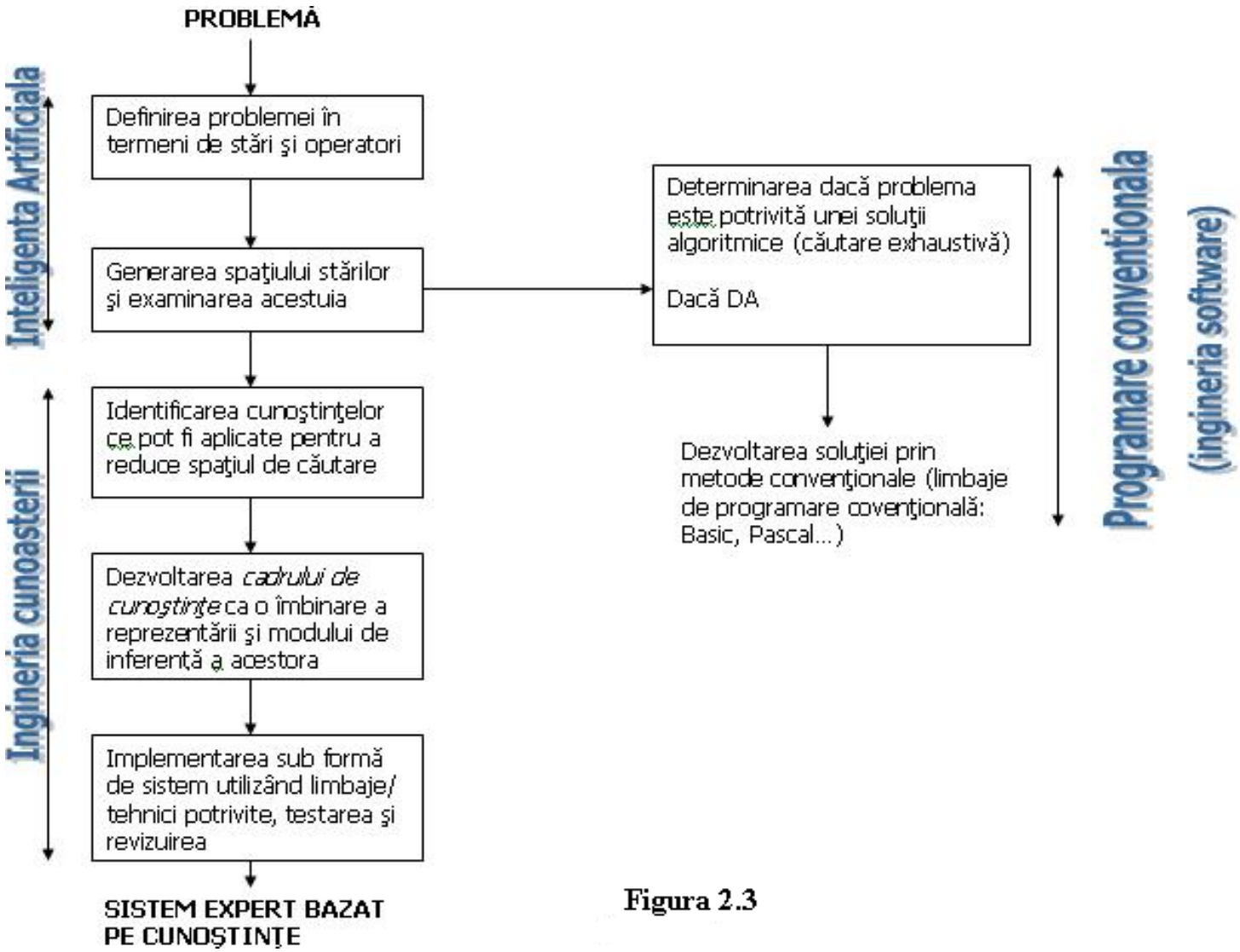
PROBLEMĂ
Definirea problemei în termeni de stări şi operatori
Generarea spaţiului stărilor şi examinarea acestuia
Identificarea cunoştinţelor ce pot fi aplicate pentru a reduce spaţiul de căutare
Dezvoltarea cadrului de cunoştinţe ca o îmbinare a reprezentării şi modului de inferenţă a acestora
Implementarea sub formă de sistem utilizând limbaje/ tehnici potrivite, testarea şi revizuirea
SISTEM EXPERT BAZAT PE CUNOŞTINŢE
Inteligenta Artificiala
Ingineria cunoasterii
Determinarea dacă problema este potrivită unei soluţii algoritmice (căutare exhaustivă)
Dacă DA
Dezvoltarea soluţiei prin metode convenţionale (limbaje de programare covenţională: Basic, Pascal...)
Programare conventionala
(ingineria software)

Figura 2.3

2.9.2 Sistemul Expert – Agent Inteligent?

O întrebare posibilă, şi dealtfel foarte justificată, legată de problematica agenţilor inteligenţi ar putea fi următoarea: ***Nu sunt de fapt agenţii inteligenţi Sisteme Expert, dar sub un alt nume?***

Răspunsul la această întrebare este: ***Categoric, nu.*** Justificarea o vom da în cele ce urmează. După cum am aflat deja, Sistemele Expert efectuează o expertiză într-un anumit domeniu (abstract) de conversaţie. Spre exemplu, MYCIN este un sistem expert care "ştie" despre bolile de sânge la oameni, conţinând foarte multe cunoştinţe despre bolile de sânge, sub formă de reguli. Un doctor poate obţine sfaturi experte despre bolile de sânge furnizându-i sistemului fapte, răspunzând la întrebări şi interogând sistemul.

Principalele ***diferenţe*** între agenţi şi sisteme expert sunt:

- Sistemele Expert nu sunt, în general, autonome – ele necesită intervenţia oamenilor (expertului).
- Agenţii sunt situaţi într-un mediu, pe când sistemele expert (spre exemplu MYCIN) nu sunt "conştiente" de mediul asupra căruia fac expertiza – singura informaţie pe care o pot obţine despre mediu este punând întrebări utilizatorului.
- Agenţii acţionează asupra mediului în care sunt situaţi, pe când sistemele expert (MYCIN în cazul nostru) nu acţionează asupra pacienţilor (doar răspund la întrebări legate de pacienţi).

Cu toate diferenţele enumerate mai sus, anumite sisteme expert în timp real (în special cele legate de controlul proceselor) sunt agenţi.

2.10. Agenţi mobili

O infrastructură de **agenţi mobili** oferă o abordare atractivă pentru transformarea reţelelor publice în platforme computaţionale. Într-o astfel de arhitectură, agenţii sunt **mobili**: ei sunt capabili să se deplaseze dintr-un loc într-altul, caz în care programul şi starea agentului sunt codificate şi transmise prin reţea într-un alt loc, unde execuţia reîncepe. Agenţii mobili sunt de fapt agenţi care circulă prin reţea şi realizează diferite sarcini pe maşini care au capacitatea de a "găzdui" agenţi (aici intervine şi partea de securitate a maşinii gazdă).

În continuare vom enumera câteva caracteristici ale agenţilor mobili:

- agenţii mobili au un program, o stare internă persistentă şi alte atribute (planul de deplasare în reţea, istoricul deplasării, informaţii legate de privilegii de acces);
- agenţii mobili se deplasează în reţea de la un calculator gazdă la altul, în scopul îndeplinirii sarcinilor cu care au fost delegaţi;
- agenţii mobili se bazează pe paradigma programării la distanţă (*remote programming*), spre deosebire de paradigma tradiţională a apelului de proceduri la distanţă, paradigmă a programării distribuite. Diferenţa esenţială între apelurile de proceduri la distanţă (a) şi agenţi mobili (b) este ilustrată în Figura 2.4.
- agenţii mobili facilitează comunicarea între aplicaţii care implică date, surse de cunoştinţe şi servicii distribuite;
- agenţii mobili pot combina cunoştinţe şi date de la client şi de la server, acolo unde sunt localizate datele şi resursele de calcul;

- agenţii mobili pot facilita interacţiuni în timp real cu un server.

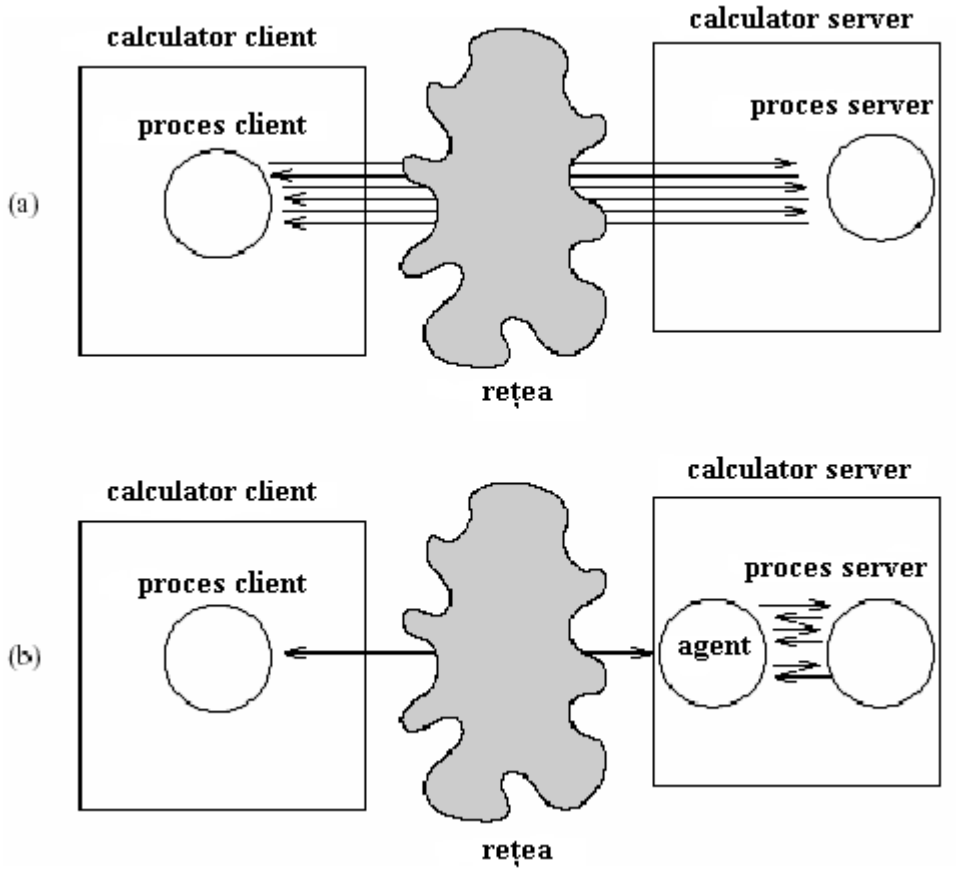

Figura 2.4

Platformele de agenţi mobili furnizează:

- un mediu de execuţie independent de calculatorul gazdă pentru programele agenţilor mobili;
- limbaje de comunicare standard care vor fi folosite de agenţi şi de server pentru a dialoga.

Facilitatea agenţilor mobili (*Mobile Agent Facility*) este o încercare de a standardiza anumite aspecte ale platformelor de agenţi mobili pentru a uşura interoperabilitatea între diferite platforme.

Câteva exemple de platforme de agenţi mobili sunt: **Voyager** (scris în Java, pentru Data Mining, descoperire de cunoştinţe, bioinformatică), **JavaMob** (scris în Java, foloseşte CORBA pentru gestionarea obiectelor distribuite şi e folosit pentru Data Mining şi descoperire de cunoştinţe).

O întrebare legitimă legată de problematica agenţilor mobili ar fi: "*De ce agenţi mobili?*". Răspunsurile la această întrebare ar fi: pe de o parte folosirea eficientă a resurselor reţelei, pe de altă parte sunt utili în reţele cu lăţime de bandă mică (PDA-uri de mână, cum ar fi NEWTON).

Fără a intra prea mult în amănunte legate de problematica agenţilor mobili, ar mai trebui menţionat faptul că sunt multe aspecte care trebuie luate în considerare atunci când se doreşte dezvoltarea unor mecanisme software care să suporte agenţi mobili:

- probleme de securitate între agenţi şi calculatoarele gazdă;
- eterogenitatea calculatoarelor gazdă;
- legăturile dinamice.

Pentru construirea sistemelor bazate pe agenţi mobili au fost construite diverse medii de dezvoltare (spre exemplu TELESCRIPT). TELESCRIPT este un mediu bazat pe limbaj pentru construirea sistemelor bazate pe agenţi mobili. Tehnologia TELESCRIPT este de fapt numele dat unei familii de concepte şi tehnici care au fost dezvoltate de către *General Magic* pentru a-şi gestiona produsele.

Sunt două concepte de bază în terminologia TELESCRIPT: **locuri** şi **agenţi**. **Locurile** sunt locaţii virtuale ocupate de agenţi. Un loc poate corespunde unei singure maşini, sau unui grup de maşini. **Agenţii** sunt mobili - ei sunt capabili să se desplaseze dintr-un loc într-altul.

O altă posibilitate de a dezvolta agenţi mobili este folosirea unor medii care să permită acest lucru, spre exemplu mediul **JADE** (Java Agent DEvelopment Framework), care furnizează clase predefinite pentru construirea de agenţi mobili.

2.11. Domenii de utilizare ale Agenţilor Inteligenţi

Agenţii sunt indicaţi a fi utilizaţi în domenii în care sunt necesare *acţiuni autonome*. **Agenţii Inteligenţi** sunt indicaţi a fi utilizaţi în domenii în care sunt necesare *acţiuni autonome* şi *flexibile*. Tehnologia bazată pe agenţi, ne oferă, totuşi, o modalitate de a construi a sisteme a căror realizare este **grea** pentru ingineria soft obişnuită.

Sistemele MultiAgent sunt indicate a fi utilizate în domenii în care:

- controlul, datele şi expertiza sunt distribuite;
- controlul centralizat este imposibil sau nepractic;
- nodurile procesate au puncte de vedere sau obiective competitive/conflictuale.

Principalele domenii de aplicare ale agenţilor sunt: sisteme distribuite/concurente, reţele, interfeţe om-calculator. Câteva aplicaţii în care sunt folosiţi agenţii sunt: agenţi de interfaţă (spre exemplu: căutarea pe Internet pentru a afle răspunsul la o anumită interogare), agenţi în Internet, agenţi în comerţul electronic (*e-commerce*).

Din punct de vedere al implementării, agenţii sunt în general implementaţi ca:

- procese UNIX;
- procese în C;
- thread-uri în JAVA.

În cazul implementării agenţilor folosind tehnici OO (orientate obiect), simularea interacţiunii dintre un agent şi un mediu conţine, în general, următoarele clase de bază:

- *Agent* - reprezintă agentul al cărui comportament se doreşte a fi implementat. Agentul interacţionează cu mediul, primeşte percepţii de la acesta şi selectează acţiuni. Agenţii pot sau nu învăţa şi pot sau nu să dezvolte un model al mediului;
- *Mediu* - mediul cu care agentul va interacţiona;
- *Acţiune* - reprezintă acţiunile pe care le poate efectua agentul;

- *Percepţie* - reprezintă perecepţiile pe care le primeşte agentul de la mediul său. În cazul cel mai simplu, percepţiile vor fi stări ale mediului;
- *Simulare* - gestionează interacţiunea între agent şi mediu. O instanţă a clasei *Simulare* este asociată în momentul creerii cu o instanţă a unui *Mediu* şi una sau mai multe instanţe ale unui *Agent*.

Comportamentul unui *Agent* va fi dat de o operaţie (metodă) **actiune(p:Perceptie):Actiune** care selectează acţiunea agentului pe baza percepţiei curente, iar *Mediul* va avea o operaţie **urmator(p:Perceptie;a:Actiune):Perceptie** care mapează percepţia curentă şi acţiunea selectată la o nouă percepţie, în cazul în care mediul este determinist. Dacă mediul este nedetreminist operaţia **urmator** va returna o mulţime de percepţii.

Domeniile în care se folosesc agenţii inteligenţi sunt numeroase, cum ar fi:

- ***industrie***: controlul proceselor industriale,controlul traficului aerian,etc.
- ***economie***: gestiunea afacerilor, comerţ electronic,etc.
- ***medical***: pentru monitorizarea pacienţilor,sănatate.
- ***cercetare guvenamentală*** (Guvernul US): domeniile transmiterii de mesaje, automatizarea procesului de muncă şi regăsirea informaţiei, urmărirea şi controlul mecanisme de luptă sofisticate (ca de exemplu cele care presupun lansări de bombe).
- ***administrarea reţelelor***: monitorizează şi măsoară in mod continuu activitatea reţelei; realizează sarcini de genul plăţi, servicii client, înregistrări.
- ***jocuri, filtrarea poştei electronice, căutare de informaţii***.

Agenţii inteligenţi pot imbunătăţi ***interacţiunea om-calculator*** prin:

- ascunderea complexităţii task-urilor dificile;
- realizarea de task-uri (acţiuni) laborioase;
- conducerea unor tranzacţii în numele utilizatorului;
- pregătire şi învăţare;
- oferirea de ajutor (asistenţă) unor anumiţi utilizatori în vederea colaborării acestora;
- monitorizarea de evenimente şi proceduri diverse.

În ceea ce priveşte construirea sistemelor bazate pe agenţi putem face următoarele observaţii:

- nu există nişte standarde stabilite pentru agenţi;
- dezvoltatorii de sisteme bazate pe agenţi de multe ori cred că nu au altă posibilitate decât de a proiecta şi a construi de la început toate componentele specifice sistemelor bazate pe agenţi, ignorând faptul că există deja nişte standarde *de facto* cum ar fi: CORBA, HTML, KQML, FIPA.

2.12. Exemple de agenţi inteligenţi

"Norns go to war" , un proiect de colaborare între firma CyberLife si DERA (Defense Evaluation and Research Agency a Guvernului Britanic) destinat creerii de agenţi care să se comporte ca adversari în misiunile de pilotaj în simulatoarele de zbor. Departamentul militar doreşte crearea unor piloţi de elită de luptă (virtuali), adică agenţi/piloţi software care să menţină în aer avioanele indiferent de atac şi să poată urmări ţintele perioade lungi de timp. De asemenea, Ministerul Apărării al UK

împreună cu CyberLife au iniţiat un contract pentru construirea unui aparat de zbor militar simulat controlat în întregime de un agent software.(el va fi capabil, printre altele să ia decizii raţionale în funcţie de context pentru a-şi finaliza misiunea cu succes).

"Do-I-Care Agent ?" (DICA), dezvoltat la University of California at Irvine, tratează problema re-descoperirii resurselor pe Web. O dată ce s-a găsit un site interesant pe Web, cum se poate şti când a fost adăugat material nou şi interesant pe acel site? Soluţia DICA pentru această problemă constă în verificarea/vizitarea periodică a site-ului şi informarea utilizatorului numai în cazul când a fost adaugat ceva nou şi interesant. Pentru aceasta utilizatorul trebuie să furnizeze feedback sistemului DICA pentru a-l "învata" pe acesta ce înseamnă "interesant" din punctul de vedere al utilizatorului.

"ReferralWeb", crează automat reţele sociale reprezentând relaţii între prieteni, colegi, şi parteneri în afaceri, din informaţiile publice disponibile pe Web. Două persoane sunt relaţionate (există un link intre ele) dacă sistemul a adunat informaţie că cele două persoane sunt dintr-un anumit punct de vedere relaţionabile. Reţeaua poate fi astfel folosită de către experţi pentru a ghida căutarea de informaţii. De exemplu, cineva doreşte să utilizeze sistemul pentru a găsi experţi în criptografie care se găsesc la 3 paşi de el. Sistemul va găsi experţii şi îi va prezenta solicitantului în ce fel sunt relaţionaţi, permiţându-i acestuia să găsească informaţie credibilă de la experţi de încredere, care sunt dispuşi să ajute, deoarece sunt prieteni ai prietenilor acelei persoane.