CONLUCRAREA ÎN APLICAȚII A TEHNOLOGIILOR CONVENȚIONALE ȘI A TEHNICILOR DE INTELIGENȚĂ ARTIFICIALĂ

drnd. ing. Ion Leon Bațachia

Institutul de Cercetări în Informatică

Rezumat: În acest articol se face o prezentare a arhitecturilor de integrare a tehnicilor de inteligență artificială (IA) și a tehnologiilor convenționale (gen sisteme suport pentru decizie, de exemplu) în sisteme globale performante. Se argumentează că cea mai avantajoasă, dar și cea mai dificilă din punctul de vedere al implementării, este arhitectura de tip "embedded". De asemenea se aduc argumente asupra faptului că într-un sistem de tip "embedded" este inacceptabilă ideea ca subsistemul de IA să aibă controlul. Sistemul gazdă global este cel care trebuie să aibă controlul. Spre deosebire de sistemele de IA "stand-alone", în care arhitectura sistemului global este dominată de cerințele soluției de IA, într-un sistem de tip "embedded", componenta de IA face parte integrantă dintr-un sistem mai larg, concurând prin funcții și servicii specifice la îndeplinirea misiunii și a serviciilor globale pentru care sistemul gazdă a fost proiectat. Mai mult, într-o implementare corectă, componenta IA trebuie să fie făcută invizibilă la nivelul interfeței utilizator a sistemului global. Pentru acest tip de integrare, se expun cerințele funcționale generale, problemele care trebuie avute în vedere și depășite și se evidențiază faptul că există deja pe piață instrumente software comerciale, care pot fi utilizate cu succes la rezolvarea multora din problemele ridicate.

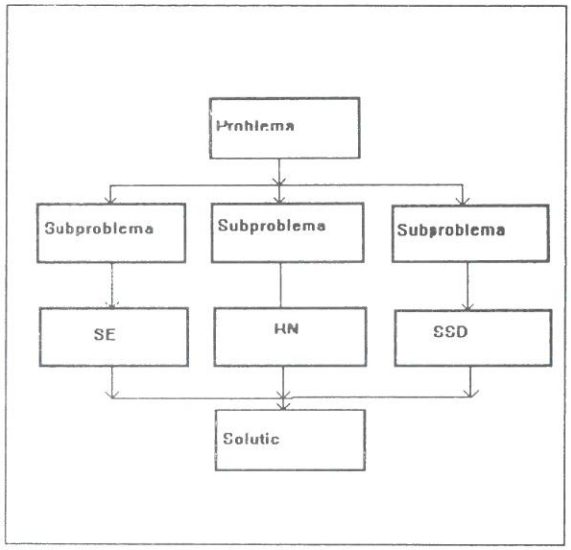
Cuvinte cheie: inteligență artificială, tehnologii convenționale, SSD, sisteme inteligente, arhitecturi de integrare, sisteme slab cuplate, sisteme strâns cuplate, sisteme "embedded", cerințe funcționale.

1. Arhitecturi generale de integrare

Un posibil mod de a integra natural tehnologii convenționale și tehnici din domeniul IA în aplicații este de a împărți o problemă complexă, care trebuie rezolvată în subprobleme sau sarcini și apoi folosirea sistemelor expert (SE), rețelelor neurale (RN), și posibil a altor tehnologii inteligente sau convenționale (de exemplu, sisteme suport pentru decizie - SSD) pentru rezolvarea problemei complexe, fiecare tehnologie rezolvând subproblema care îi este mai potrivită. Figura de mai jos ilustrează o modalitate de conlucrare, într-un sistem global, a tehnologiei SSD împreună cu o componentă rețea neurală și un subsistem expert.

Există mai multe moduri de integrare a tehnologiilor convenționale și a tehnicilor de IA în realizarea de sisteme globale cu performanțe superioare. În general, există trei căi generale de integrare, iar în literatura de specialitate [4], [5], [7] se regăsesc, în funcție de specificul problemei care trebuie rezolvată de sistemul global și de paradigmele de IA folosite în rezolvarea de subprobleme, mai multe variante ale acestor căi clasice de integrare.

Cele trei tipuri majore de arhitecturi sunt: integrarea cu cuplaj slab (*loosely coupled*), integrarea cu cuplaj strâns (*tightly coupled*) și sistemele *embedded* (sau *full integration*).



1.1. Sistemele cu cuplaj slab

În acest mod de integrare, două sau mai multe componente comunică prin intermediul fișierelor de date pentru o interfațare rapidă. Într-un astfel de caz, subsistemele individuale (RN, SE, SSD, etc) sunt capabile de funcționare independentă. Este un tip de integrare foarte comun și implică faptul că sistemele sunt distribuite. Cuplarea slabă oferă anumite avantaje, printre care primul și cel mai important este faptul că este fezabilă cu software-ul și metodologiile curente. Astfel shell-uri sistem expert cum ar fi VP expert, First Class, Level 5 etc. pot fără greutate să se interfațeze cu un instrument de dezvoltare rețele neurale, cum ar fi BrainMaker, NeuroShell etc. De asemenea, shell-urile sistem expert pot să se interfațeze cu anumite generatoare SSD (de exemplu Lotus 1-2-3) sau/și cu sisteme de gestiune a bazelor de date. Un alt avantaj este că fiecare componentă poate fi dezvoltată complet separat, reducând astfel timpul de dezvoltare. Un dezavantaj evident este acela al performanțelor mai reduse ale sistemului global datorită vitezei mai mici de comunicație.

1.2. Sistemele cu cuplaj strâns

Această arhitectură este similară cu cea în care cuplajul între subsisteme este slab (amândouă arhitecturile sunt distribuite), dar aici comunicația între modulele componente nu se face prin fișiere

de date, ci prin intermediul parametrilor sau transferului de date.

Avantajul major este creşterea vitezei de comunicaţie, care este esenţială în cazul aplicaţiilor în timp real. Dezavantajul major este nevoia de a face ca procesul de construcţie a sistemului global, să fie orientat pe aplicaţie, componentele sale neputând să mai fie proiectate şi dezvoltate în mod independent. Aceasta presupune o creştere a complexităţii proceselor de proiectare, dezvoltare şi validare, şi o mărire a duratei şi a costului dezvoltării.

1.3. Sistemele *embedded*

În sistemele *embedded*, o tehnologie este integrată complet într-o alta. Desigur că particularizând discuţia la cazul tehnologiei SSD şi tehnologiei IA, este evident că este de preferat ca tehnologia care este înglobată să fie tehnologia IA şi nu invers, sistemul global de tip SSD fiind cel care trebuie să aibă controlul şi nu invers, din motive care vor mai fi expuse în acest articol (a se vedea şi articolul *SSD şi SI integrate pentru conducerea producţiei*, semnat de Alexandru Dan Donciulescu, din acest număr al revistei). Mecanismele de control sunt integrate total sau operează în mod cooperativ. Cerinţele funcţionale şi problemele care le ridică o astfel de arhitectură vor fi tratate mai pe larg în secţiunea următoare a articolului.

Avantajele majore ale unor astfel de sisteme sunt adaptabilitatea crescută

, generalizarea, flexibilitatea şi eficienţa calculului. Un alt avantaj este redundanţa minimă în achiziţia de date şi în folosirea resurselor.

O dificultate majoră în cazul acestei arhitecturi o reprezintă integrarea manipulării simbolice a componentei sistem expert, de exemplu, cu forma numerică a modelelor matematice sau cu alte forme de reprezentare şi de manipulare de date specifice precum şi integrarea mecanismelor de control.

2. Cerinţe funcţionale generale. Probleme care trebuie depăşite

Se va lua în considerare varianta *embedded* care, aşa cum s-a precizat în secţiunea precedentă, prezintă avantajele cele mai importante, dar şi cele mai deosebite dificultăţi în implementare. Această tendinţă de integrare a componentelor de IA în sisteme existente deja sau în sisteme care urmează a fi realizate, în scopul îmbunătăţirii performanţelor sistemului gazdă global, reprezintă una din provocările fundamentale ale dezvoltării tehnologiei IA. Deşi ideea pare simplă în concept, există multe aspecte care trebuie luate în seamă şi cercetate. Problemele generale care trebuie avute în vedere şi depăşite nu sunt specifice doar conlucrării tehnologiei SSD şi subsistemelor de IA în cadrul unui sistem global complex, ele sunt comune tuturor sistemelor care înglobează IA.

Una din problemele cele mai ample este noţiunea de IA însăşi. Furnizarea de inteligenţă de natură umană sistemelor bazate pe computer este o căutare lungă şi dificilă. Cercetarea în ceea ce priveşte modelele de reprezentare şi mecanismele de raţionament este continuuă, o diversitate de tehnici fiind elaborate până în prezent. Tehnici de bază cum ar fi *forward* şi *backward chaining*, demonstrarea de teoreme, reţele neurale, algoritmi genetici, diferitele tehnici de *machine learning* etc. acoperă o zonă largă de necesităţi computaţionale adesea contradictorii, afectând arhitectura sistemului bazat pe computer. Fiecare din acestea pot avea felia lor în tipologia de probleme pe care le pot rezolva, dar multe dintre probleme necesită integrarea de tehnici multiple pentru o soluţie completă.

Au fost făcute progrese importante în ultimii ani în ceea ce priveşte producerea unor sisteme inteligente. Dar aceste sisteme nu există în izolare. Pentru a fi cu adevărat folositoare, ele trebuie integrate cu alte sisteme care le furnizează informaţie şi carora le prezintă soluţiile.

Provocarea curentă pentru dezvoltarea sistemelor de IA este integrarea sau înglobarea (*embedding*) sistemelor inteligente în sistemele bazate pe calculator, şi astfel în viaţa noastră de fiecare zi.

Problema de înglobare ia două forme: cum să se implementeze sistemul bazat pe calculator, în aşa fel ca el să accepte înglobarea de IA, şi cum să fie schimbate sistemele de IA aşa fel ca să fie posibilă înglobarea lor.

2.1. Scurt istoric

În urmă cu circa zece ani, începeau să apară ca rezultat al muncii din universităţi şi laboratoare de cercetare, sisteme prototip şi instrumente comerciale de IA. Aceste instrumente (iniţial shell-uri expert şi sisteme bazate pe cunoştinţe şi, mai târziu, medii de dezvoltare a reţelelor neurale) puteau fi folosite să rezolve o varietate de probleme ce păreau să eludeze tehnicile programării convenţionale. Multe dintre aceste instrumente erau implementate în Lisp sau Prolog, şi nu se executau în condiţii bune pe o maşină de scop general. Pentru a depăşi anumite limite de performanţă au apărut maşinile cu hardware specializat (maşina Lisp, de exemplu).

Deoarece tehnologia era proaspăt ieşită din laboratoare şi încă era centrată pe rezolvarea *problemei de inteligenţă*, s-a dat relativ puţină atenţie problemelor de înglobare a acestor sisteme în mediile din lumea reală.

Sistemul inteligent a fost văzut ca un element software izolat, adesea cu procesorul său dedicat, proiectat special pentru maximizarea performanţei în problema de IA. Achiziţia de date de la alte sisteme era dificilă şi adesea consumatoare de timp. Date fiind resursele necesare pentru rezolvarea problemelor şi resursele disponibile în acea perioadă, achiziţia de date de-a lungul unei reţele locale sau direct de la utilizator era lucrul cel mai bun la care ne puteam aştepta.

De atunci, necesitatea înglobării de IA a fost recunoscută. Sistemele Lisp au fost înlocuite pe scară largă de implementările C/C++. Integrarea de tehnici cum ar fi apelurile de software extern, interfeţe cu baze de date, interfeţe utilizator sofisticate, filtrarea datelor şi subsisteme de procesare a cunoştinţelor au devenit între timp comune. Performanţa a fost mereu crescută prin îmbunătăţirea continuă a mecanismelor de inferenţă. Au devenit, de asemenea, comune tehnicile de convertire a reprezentărilor declarative ale cunoştinţelor în cod executabil eficient.

Dar aceste soluţii tratează numai implementarea eficientă a sistemului pe hardware-ul calculatorului şi furnizează facilităţi mai bune pentru interfaţa cu alte sisteme. Cei care dezvoltă aplicaţii IA trebuie în continuare să se confrunte cu impactul integrării asupra proiectării sistemului inteligent şi a căilor prin care interacţionează cu mediul înconjurător.

2.2. Ce înseamnă înglobarea de IA ?

Sistemele de IA în mare pot fi clasificate în două clase, plecând de la arhitectura acestora: *stand-alone* şi *embedded*.

Un sistem de IA *stand-alone*, fie există independent, fie reprezintă componenta principală a unui sistem care se bazează pe alte sisteme pentru achiziţia datelor. În ultimul caz, arhitectura sistemului gazdă este dominată de cerinţele soluţiei IA. Multe din primele sisteme de IA au fost numite *stand-alone* datorită orientării pe aplicaţia IA şi pe capabilităţile instrumentelor disponibile.

Cu toate acestea, puţine probleme din lumea reală sunt de natură izolată. Cele mai interesante probleme necesită schimb de date şi integrarea cu alte sisteme şi baze de date. De fapt multe din primele sisteme de IA stand-alone au avut, de asemenea, această proprietate, dar solicitau date de la utilizator sau foloseau alte tehnici brute de intrare pentru a colecta date aparţinând de drept altor sisteme.

Prin contrast, în arhitectura *embedded*, într-un sistem cu IA înglobată, componenta de IA este o parte integrantă a unui sistem mai larg.

Acest sistem global furnizează un spectru larg de funcţii care concură la îndeplinirea misiunii sale (în cazul unui sistem global de asistarea deciziei, misiunea principală este sprijinirea decidentului uman în luarea deciziilor la un nivel calitativ superior) şi care definesc arhitectura sa. În cadrul sistemului global, funcţiile care îi revin componentei/componentelor de IA pot fi asigurate direct sau indirect prin intermediul serviciilor.

În fiecare caz, folosirea tehnologiei de înglobare a componentei de IA trebuie să fie invizibilă pentru utilizator şi pentru sistemul global. A face componenta de IA invizibilă este una din cele mai importante cerinţe funcţionale care trebuie rezolvată.

Alte probleme care, implicit, trebuie abordate şi rezolvate, sunt legate de modul în care componenta de IA interacţionează cu sistemul global în culegerea şi furnizarea de date, modul în care controlul execuţiei este partajat între componenta de IA şi sistemul care o înglobează şi modul de control al resurselor.

2.3. Controlul execuţiei şi al resurselor

Într-un sistem de tip *embedded*, fie sistemul IA este cel care are controlul, fie sistemul global care-l conţine are controlul.

În multe din primele sisteme de IA *stand-alone*, sistemul de IA era cel care avea în totalitate controlul. Era invocat în mod obişnuit de către utilizator şi rula independent de mediul înconjurător până când sarcina pentru care era invocat era rezolvată.

Pentru cele mai multe aplicaţii cu sistem de IA înglobat, este inacceptabilă soluţia ca sistemul IA să aibă controlul complet. Cel care trebuie să aibă controlul este sistemul global, care trebuie să decidă când şi dacă, în anumite cazuri, componenta IA trebuie activată şi cum trebuie să-şi execute sarcina. Acest aspect al controlului este critic în multe aplicaţii de timp real, care impun ca munca să fie realizată între limite stricte de timp.

Un aspect al controlului mai subtil se referă la controlul resurselor sistemului de către componenta IA. Într-un mediu cu subsisteme înglobate, subsistemul de IA trebuie să partajeze capacitatea procesorului, memoria, spaţiul de memorare pe disc, bazele de date şi alte resurse cu celelalte subsisteme componente. Subsistemele înglobate trebuie construite astfel încât să partajeze aceste resurse şi să utilizeze mecanismele sistemului global pentru alocarea şi controlul lor.

Activitățile de calcul intensiv, cum ar fi de *pattern matching* în cazul sistemelor expert, sau de învățare, în cazul rețelelor neurale, trebuie tratate cu atenție pentru a fi compatibile cu sistemul global. În mod similar, folosirea memoriei pentru structurile de date mari ale sistemelor de IA trebuie controlate. Din punctul de vedere al performanței, nu sunt acceptate discontinuități de genul *garbage collection* din sistemele din trecut, bazate pe LISP.

În dezvoltarea sistemelor de IA există tendința de a rezolva multe din aceste probleme prin selecția potrivită de instrumente. Shell-rile, sistemele comerciale curente bazate pe cunoștințe, furnizează multe facilități inclusiv interfețe la baze de date standard, apeluri de proceduri externe, controlul asupra paradigmei de inferență folosit și facilități de filtrare pentru date de timp real, care simplifică problemele de interfață. Implementările de sisteme de IA și instrumente folosind C sau C++ ca limbaj de bază, algoritmi de înaltă performanță și tehnici care compilează reprezentări în cod executabil sprijină rezolvarea problemelor de control, făcând sistemul rezultat compatibil cu mediul în care este înglobat și reducând necesarul de resurse pentru acesta. Dar instrumentele și vehiculele de implementare nu rezolvă toate problemele de înglobare. Proiectarea și structura soluției de IA însăși trebuie, în ultimă instanță, să trateze problema de înglobare.

2.4. Interacțiunea și achiziția de date

Un sistem IA înglobat poate achiziționa date de la mai multe surse incluzând:

- intrarea utilizator
- accesul la baze de date partajate
- invocarea directă a altor sisteme (de exemplu, prin apeluri de proceduri)
- invocarea indirectă a altor sisteme prin transmitere de mesaje sau alte mecanisme de comunicație.

Gradul în care această interacțiune este corespunzătoare sistemului global este cheia pentru o înglobare de succes.

Multe din primele sisteme de IA au fost însoțite de probleme de utilizare și de instruire datorită faptului că utilizatorii interacționau cu acestea direct, și nu prin intermediul unei interfețe standard a sistemului global. De asemenea, lipsa partajării datelor a condus la duplicarea intrării utilizator.

O dată cu evoluția tehnologiei bazelor de date, a devenit fezabil pentru sistemele IA să partajeze datele cu alte sisteme prin intermediul sistemelor de gestiune a bazelor de date.

Această capabilitate a adus cu sine necesitatea de a menține integritatea datelor între toți utilizatorii de date. Dar cum majoritatea sistemelor de baze de date asigură integritatea datelor, responsabilitatea sistemului IA este de a folosi în mod corect respectiva bază de date.

Totodată, achiziția directă și indirectă de date de la alte sisteme asigură o integrare strânsă cu sistemul global și permite distribuția corectă a paternității datelor.

Pentru ca această capabilitate să fie practicabilă, sistemul IA trebuie să aibe facilitățile potrivite pentru a identifica și a procesa tipurile de date de la aceste sisteme străine.

Achiziția indirectă de date prin intermediul interfețelor bazate pe mesaje necesită ca sistemul IA să fie confruntat cu date furnizate asincron. Pentru aceasta, sistemele IA trebuie proiectate ca procesoare tranzacționale care pot accepta date în secvențe nespecificate, reacționează la date atunci când devin disponibile și mențin starea curentă.

De asemenea, datele asincrone pot introduce și o doză de nesiguranță, sugerând necesitatea unor tehnici de tratare a incertitudinii și raționament bazat pe modele.

O problemă generală pentru sistemele IA înglobate, care partajează și primesc date din afară, este reprezentarea. Rolul reprezentării în rezolvarea problemelor și în eficiență este atât de important în IA, încât majoritatea sistemelor de IA folosesc reprezentări speciale ale datelor. Sistemele externe cu care interacționează nu au aceste cerințe. Acest aspect implică o problemă de translatare, datele achiziționate de la sistemele externe trebuind să fie translatate într-o reprezentare internă specifică, în vederea rezolvării respectivei probleme de IA, și apoi un proces invers, de translatare la prezentarea soluției.

Așa cum am mai precizat în acest articol, din fericire multe din problemele ridicate de integrarea elementelor de IA, pot fi rezolvate ușor, prin folosirea unor instrumente software existente pe piață. În secțiunea următoare, se prezintă succint câteva din produsele software cu acest rol.

3. Exemple de produse comerciale pentru dezvoltarea, testarea și integrarea de tehnologii inteligente

NeuralDesk

NeuralDesk este un software format din trei pachete integrate, realizat de Neural Computer

Sciences (NCS) din Marea Britanie, care permite utilizatorilor să creeze şi să ruleze propria reţea neurală. Rulează sub Microsoft Windows, cel puţin versiunea 3.0.

Folosind *NeuDesk*, datele sunt introduse prin intermediul programului însuşi, şi nu prin intermediul unui program extern cum ar fi Excel sau Superbase, deşi acestea pot fi legate la acest pachet software. De îndată ce datele sunt introduse, NeuDesk proiectează automat o reţea potrivită cu informaţiile furnizate. Împreună cu NeuDesk este şi *NeuRun*, care permite reţelelor neurale să fie legate cu alte programe Windows. În final, pachetul *NeuModel* permite customizarea şi editarea reţelelor neurale, create anterior cu NeuModel sau de NeuDesk.

NeuralDesk este proiectat pentru programele Windows, iar componentele sale folosesc la maximum avantajele capabilităţilor oferite de Windows. Folosind facilitatea numită Dynamic Data Exchange (DDE), care permite aplicaţiilor să comunice unele cu altele, în sensul că pot partaja date şi una din ele poate avea controlul asupra altora, reţele neurale antrenate pot fi înglobate în alte aplicaţii Windows.

THINKS

THINKS este produsul lui Logical Designs Consulting Inc. din Statele Unite şi reprezintă un software pentru Windows 3.1 pentru învăţarea reţelelor neurale. THINKS integrează algoritmi de reţele neurale cu o interfaţă Windows user-friendly.

Algoritmii nu sunt folosiţi de proprietar şi de un utilizator experimentat sau nu poate testa uşor configuraţii noi de reţele. Oferind funcţii de esenţă privind tehnologia reţelelor neurale în forme diferite, inclusiv biblioteci run-time sub DOS pentru diferite compilatoare, un DLL Windows şi cod sursă portabil în orice mediu, THINKS permite utilizatorului să dezvolte în mod eficient orice aplicaţie bazată pe reţele neurale.

MEDAL

MEDAL (Matrix and Expert System Developpment Aid Language) este un program interactiv, care sprijină dezvoltarea de sisteme cuplate în inginerie şi ştiinţă. Sintaxa limbajului său este similară cu a limbajului MatLab: el reţine toate principalele caracteristici ale MatLab, inclusiv sintaxa MatLab şi fişierele M. În plus, MEDAL include un shell sistem expert integrat pentru dezvoltarea sistemelor bazate pe cunoştinţe şi logică fuzzy, care poate, de asemenea, executa calcul numeric sofisticat.

Astfel, predicatele adiţionale ale sistemul expert extind sintaxa limbajului de comandă MatLab. De asemenea, MEDAL suportă un set bogat de structuri de date pentru reprezentarea obiectelor în mediul de programare. Cunoştinţele pot fi reprezentate folosind fapte, reguli şi frame-uri.

MEDAL este gratuit pentru scopuri de cercetare şi academice şi este disponibil via Internet.

O versiune profesională a MEDAL este, de asemenea, furnizată de Waterloo MEDAL Software â Technologies Inc.

EXSYS Professional 4.0.

Ultima versiune a shell-ului pentru dezvoltarea sistemelor expert, EXSYS Professional lucrează în configuraţii Windows, Macintosh şi Windows NT. EXSYS permite realizarea de sisteme expert probabilistice, bazate pe cunoştinţe folosind reguli de producţie IF-THEN-ELSE, integrarea invizibilă şi interfaţarea uşoară a componentelor sisteme expert cu alte aplicaţii, şi baze de date.

Editorul de reguli permite generarea rapidă, folosind ecrane de help pentru selecţia elementelor din liste de opţiuni diferite, pentru asignarea şi combinarea valorilor de probabilitate. Regulile sunt editate la fel de uşor ca în lucrul cu un procesor de texte. Problemele complexe pot fi împărţite în sisteme expert mai mici, care comunică prin intermediul fişierelor de date comune.

O dată cu versiunea 4.0 vine şi EXSYS RuleBook for Windows, un pachet de dezvoltare sistem expert stil diagrame arborescente. Sistemele expert sunt construite folosind diagrame arborescente care descriu un întreg aspect al problemei. Nodurile în arbore reprezintă întrebări puse de utilizatorul final; când un nou nod este adăugat, alte ramuri sunt construite automat pentru toate valorile posibile de intrare. Sistemele expert sunt construite prin crearea de arbori multipli, reprezentând aspecte independente ale problemei.

Furnizorul pachetului integral EXSYS este EXSYS Inc. din Statele Unite.

De remarcat că versiunea 5.0, probabil apărută deja, fiind anunţată de mai multă vreme, include şi logică fuzzy.

Bibliografie

1. **BAŢACHIA, L., DONCIULESCU, D.**: CTT-SSD. Recomandări tehnice privind sistemele suport pentru decizie, Raport de cercetare, ICI, iulie, 1995.
2. **HIGHLAND, F.** (Guest Editor): Embedded AI. În: Intelligent Systems & Their Applications, June, 1994, Vol. 2, No. 3, pp. 18-21.

3. **SCHUTZER, D.**: Business Expert Systems: The Competitive Edge. În: Expert Systems with Applications, 1990, Vol. I, pp. 17-21.

4. **MEDSKER L., TURBAN, E.**: Integrating Expert Systems and Neural Computing for Decision Support. În: Expert Systems With Applications, 1994, Vol. 7, No. 4, pp. 553-562.

5. **SONDAK, N.E., SONDAK, V.K.**: Embedding Artificial Neural Networks and Expert Systems. În: Proc. California State University Anual Meeting on AI, June, 1992, Long Beach, CA., pp. 150-157.

6. **ODETTE, L.L.** : Intelligent Embedded Systems, Reading, MA: Addison-Wesley, 1991.

7. **DONCIULESCU, D., BAȚACHIA, L., BUȚA, O.**: Sisteme de simulare bazate pe modele mixte pentru conducerea operativă a producției în IMM, Raport de cercetare, ICI, noiembrie, 1995.

8. **BISWAS, G., OLIFF, M., SEN, A.**: An Expert Decision Support System for Production Control. În: Decision Support Systems, 1988, pp 235-247.