Anca MIRON

SISTEME EXPERT ÎN ENERGETICĂ

Suport de curs

U.T.PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-802-3

Anca MIRON

SISTEME EXPERT ÎN ENERGETICĂ

Suport de curs

U.T.PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-802-3

Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
www.utcluj.ro/editura

Recenzia: Prof.dr.ing. Sorin G. Pavel
Conf.dr.ing. Radu A. Tîrnovan

Pregătire format electronic on-line: Gabriela Groza

ISBN 978-606-737-802-3

PREFAŢĂ

Prezentul material didactic este destinat să ofere un suport bibliogafic pentru informațiile predate și discutate în cadrul activității de curs din programa analitică a disciplinei *Sisteme expert în energetică,* pentru anul II, studii de master, specializarea de energetică.

Suportul pentru curs a fost conceput ca un material didactic complementar cursului, astfel încât noţiunile teoretice și practice acumulate să confere studenţilor un bagaj de cunoștințe menit să le permită abordarea cu succes a problemelor pe care vor trebui să le rezolve în calitate de specialişti energeticieni.

Elaborate în această concepţie, toate capitolele acestui material didactic sunt logic structurate, fiecare fiind corespondent unui curs și beneficiază de o tratare teoretică corespunzătoare. De asemenea, s-a urmărit ca fiecare capitol să asigure și exemple cuprinzătoare, menite să exemplifice principalele aspecte ale subiectelor studiate.

Cu toate eforturile depuse, autoarea este convinsă că această formă a prezentului material didactic poate fi îmbunătăţită prin continuarea preocupărilor în domeniu, prin sugestiile studenţilor care îl vor utiliza precum și ale altor specialişti.

Autoarea

CUPRINS

Introducere

Dezvoltarea ştiinţei calculatoarelor şi aprofundarea înţelegerii modului de funcţionare a creierului uman a dus în ultimele trei decenii la o explozie a aplicaţiilor tehnicilor de Inteligenţă Artificială în majoritatea domeniilor de activitate umană: inginerie, medicină, economie, tehnică, industria armamentului etc.

Inteligenţa Artificială, IA este domeniul ştiinţific care încearcă să explice și să simuleze comportamentul și gândirea umană cu ajutorul maşinilor, calculatoarelor și a proceselor computaţionale. „Din punctul de vedere al unui inginer, Inteligenţa Artificială urmăreşte generarea unor reprezentări şi proceduri concepute pentru rezolvarea automată a problemelor care până atunci au fost rezolvate de oameni. Scopul inteligenţei artificiale este înţelegerea ca o formă de manifestare a calculelor” [1].

Tehnicile de IA reprezintă modalităţile de implementare a principiilor Inteligenţei Artificiale pentru soluţionarea diferitelor probleme abordate. Conform spuselor multor specialişti, tehnicile de Inteligenţă Artificială sunt metodele de rezolvare ale secolului XXI [2].

Inteligenţa Artificială a interesat omul încă din antichitate, dar punctul de start a științei cu același nume este abia anul 1943 când apare primul model de neuron formal, propus de neuro-fiziologul W.S. McCulloch și matematicianul W. Pitts. Tot în acest an în S.U.A. şi Germania au apărut primele calculatoare electronice, care au făcut posibil progresul și dezvoltarea ulterioară a tehnicilor de Inteligenţă Artificială [2]. Șase ani mai târziu, în 1949 a apărut primul program comercial care putea fi asimilat de un calculator electronic; această nouă invenţie oferea noi perspective în domeniul calculatoarelor și implicit a Inteligenței Artificiale. Câţiva ani mai târziu, la sfârşitul anului 1956, a fost dezvoltat primul program de Inteligenţă Artificială, *The Logic Theorist*, de către A. Newell, C. Shaw și H.A. Simon. Tot în același an, John McCarthy, considerat părintele Inteligenței Artificiale, a organizat o conferinţă intitulată „The Dartmouth Summer Research Project on Artificial Intelligence”. În cadrul acestei manifestări ştiinţifice a fost utilizat pentru prima dată termenul de Inteligenţă

Artificială, și au fost conturate și discutate viitoarele direcţii de cercetare din domeniu.

Scopurile Inteligenţei Artificiale sunt de a realiza sisteme care:

- gândesc ca oamenii,
- acţionează ca oamenii,
- gândesc raţional,
- acţionează raţional.

Aceste scopuri deschid două direcţii fundamentale de cercetare: procese cognitive (gândire și raţionament) şi comportament.

Sistemele expert sunt tehnici de Inteligență Artificială care folosesc abordarea logico-simbolică pentru rezolvarea problemelor. Acestea sunt apropiate de modul de prezentare explicativ (declarativ), cu reguli clare, modul de operare este bazat pe logică, iar expertul uman este cel care sintetizează cunoștințele.

Sistemele expert sunt o categorie aparte a sistemelor bazate pe cunoștințe (knowledge-based systems). Alte astfel de sisteme sunt: sisteme de tutorat, sisteme bazate pe cazuri, sisteme manipulate prin hipertext și interfețe inteligențe complementate de baze de date [1].

In continuare, acest suport de curs are următoarea structură:

- Abrevieri – cuprinde lista tuturor abrevierilor folosite în corpul cărții.
- Definiția și istoricul sistemelor expert – prezinta elementele de baza ale Sistemelor Expert și fac o incursiune in etapele de dezvoltare ale sistemelor expert și a sistemelor bazate pe cunoștințe.
- Arhitecturi și tipuri de sisteme expert - prezintă tipuri de sisteme expert, precum și arhitectura de bază, respectiv arhitectura avansată a sistemelor expert.
- Cunoștințe și achiziția lor – se focalizează pe natura cunoștințelor și modul de achiziție a lor pentru a obține bagajul necesar construirii bazei de cunoștințe a sistemului expert.
- Baza de cunoștințe. Reprezentarea cunoștințelor – prezintă primul modul a unui sistem expert, precum și modul în care informațiile preluate de la experți pot fi introduse în sistemul expert pentru a acesta să poată lucra cu ele.
- Baza de cunoștințe. Baza de reguli și fapte – descrie cele două componente ale bazei de cunoștințe și anume, baza de reguli, respectiv

baza de fapte.

- Motorul de inferențe. Strategii de control – face o introducere în tematica motorului de inferențe și prezintă cea mai populară strategie de control, și anume strategia de control înainte.
- Motorul de inferențe. Metode de rezolvare a conflictelor – continuă cu prezentarea strategiei de control înapoi, și descrie metodele de rezolvare a conflictelor.
- Motorul de inferențe. Algoritmi de căutare – se focalizează pe algoritmii de căutare care sunt implementați în motorul de inferență pentru a crește eficiența acestuia.
- Modulul de interfață cu utilizatorul. Modelarea utilizatorilor – descrie modului de interfață cu utilizatorul, cea mai importantă componentă a sistemului expert, în ceea ce privește relația cu utilizatorul, dar și aspecte ce țin de modelarea utilizatorilor specifici ai unui sistem expert.
- Modulul explicativ. Tipuri de explicații și achiziția lor – prezintă modulul explicativ, care are rolul de a oferi utilizatorilor explicații cu privire la modul de rezolvare a problemei. De asemenea, cuprinde și aspecte legate de tipurile și achiziția diferitelor informații corelate cu explicațiile necesare.
- Module auxiliare – modulele auxiliare se referă la componentele suplimentare ale sistemelor expert, care au fost introduse pentru creșterea eficienței sistemelor expert. Printre acestea, se prezintă modulul de achiziție a cunoștințelor și modulul dinamic.
- Realizarea sistemelor expert – se referă la modul de construire a sistemelor expert, adică la resursele necesare pentru construirea unui sistem expert și posibilitățile de realizare.
- Instrumente pentru dezvoltarea sistemelor expert – se focalizează pe limbajele de programare dedicate constuirii sistemelor expert.
- Aplicații ale sistemelor expert în energetică – face o recapitulare a informațiilor prezentate și prezintă un exemplu de sistem expert folosit în energetică.
- Anexele – prezintă sisteme expert propuse în literatura de specialitate în domeniul electroenergetic. În aceste anexe se pune accent pe identificarea prinipalelor componente ale sistemelor expert și caracteristicilor acestora.

ABREVIERI

BC – Baza de cunoștințe
BD – Baza de date
BDE – Baze de date externe
ED – Expertul uman din domeniu
IA – Inteligență Artificială
IG – Inginerul de Cunoștințe
MAC – Modulul de Achizitie a Cunoștințelor
MD – Modulul dinamic
ME – Modulul explicativ
MI – Motorul de inferență
MIU – Modulul de interfață cu utilizatorul
SBC – Sistem bazat pe cunoștințe
SE – Sistem Expert
MP – Mediu de programare
UT - Utilizatori
ED – Expertul uman din domeniu

1

Definiția și istoria Sistemelor Expert

Acest capitol descrie în amănunt legătura dintre sistemele expert şi expertul uman, făcând o comparaţie între cei doi. De asemenea, aici sunt prezentate mai multe definiţii ale sistemelor expert din literatura de specialitate, făcându-se comentarii pe baza lor. La finalul capitolului se enumeră aplicații ale sistemelor expert în diferite domenii ale ingineriei, cu precădere în inginerie energetică.

1.1 Definiția sistemelor expert

Sistemele expert sunt o clasa a sistemelor bazate pe cunoștințe. Acestea din urma sunt aplicatii informatice care se bazeaza pe cunoștințe pentru a rezolva diferite probleme.

În literatura de specialitate sunt date multe definiţii ale sistemelor expert (SE), unele dintre ele fac referire la caracteristicile generale, altele sunt legate strict de conexiunea dintre sistemul expert și expertul uman. În continuare, sunt prezentate și comentate cinci definiţii ale sistemelor expert, prin acestea făcându-se o introducere la următorul curs unde se vor prezenta caracteristicile SE şi diferenţa dintre acestea și programele convenţionale de prelucrare a datelor.

O definiţie generală a SE este următoarea:

> Programe de Inteligență Artificială (IA) bazate pe cunoaştere, ce rezolvă probleme care în mod normal necesită experienţa unui specialist.

Urmărind modul de operare al unui specialist care posedă toate cunoștințele pentru rezolvarea unei probleme dintr-un domeniu specific, SE are o bază de cunoștințe pe care o foloseşte pentru găsirea soluţiei. Dacă specialistul face raţionamente şi trage concluzii pe baza cunoștințelor care le posedă, SE foloseşte motorul de inferenţă care realizează deducţii şi raţionamente de natură logică pentru a soluţiona problema.

Din punct de vedere funcţional SE sunt definite ca programe care se bazează pe o bancă de cunoștințe pentru a realiza unele sarcini, uneori chiar dificile, pe care de regulă le rezolvă omul care este expert în domeniul său.

O definiţie didactică a fost dată de Prof. Edward Feigenbaum (Universitatea Stanford – USA):

> „... un program inteligent care utilizează cunoştinţe, fapte și tehnici de raţionare pentru a rezolva probleme care în mod normal necesită cunoştinţele experţilor umani".

Conform acestei definiţie, sistemele expert sunt sisteme de prelucrare a datelor, care pe lângă elementele clasice conţin și o bază de cunoștințe (ce conţine baza de fapte şi reguli), precum și elementele necesare manipulării acestora (sub forma unui motor de inferenţă). Ele conţin și componente necesare achiziţiei de cunoștințe și implementării de noi cunoștințe, reprezentarea rezultatelor şi analizei a posteriori unui proces de obţinere a expertizei (componenta explicativă) [2].

Un SE este ansamblul unei serii de programe informatice care se sprijină pe informaţii actuale, o bază de date și de reguli de inferenţă pentru a rezolva probleme care necesită o mare competenţă umană. Baza de date și regulile de inferenţă sunt modelate prin competenţa celor mai buni practicieni de a rezolva problema [6]. Aceasta este una din definiţiile care menţionează faptul că sistemul expert conţine mai multe module de program interconectate, care au ca date de intrare cunoștințe actuale care sunt furnizate de specialişti umani, adică buni practicieni cu experienţă în domeniul problemei de soluţionat.

O altă definiţie cu caracter mai specific sistemelor expert este umătoarea:

> Un SE achiziţionează cunoștințele unui expert uman într-un domeniu restrâns, particular, într-o formă implementabilă pe un calculator. El le foloseşte pentru a furniza suportul deciziei la un nivel comparabil cu cel al expertului uman şi este capabil să-şi justifice raţionamentul [7].

Autoarea consideră că o definiţie corespunzătoare a sistemelor expert trebuie făcută luând în considerare toate cele menţionate anterior, şi anume:

Un SE este un program informatic de nivel înalt care se bazează pe cunoştinţele actuale ale unui expert uman din domeniul problemei de rezolvat, pe care le manipulează prin intermediul unor reguli şi deducţii logice specifice expertului, fiind capabil să interacţioneze cu utilizatorul prin intermediul unei interfeţe grafice prietenoase şi să-şi justifice raţionamentul ce a dus la găsirea soluţiei.

1.2 Istoricul sistemelor expert

Istoricul SE este strict legat de naşterea IA ca ştiinţă şi sintaxă în anul 1956, când a fost prezentat la conferinţa de la Dartmouth College primul program de demonstrare a logicii propoziţiilor. În următorii ani după acest eveniment au apărut primele programe de demonstrare a teoremelor bazate pe logica propoziţiilor. Programele din aceasta perioada au fost dezvoltate pentru rezolvarea unor probleme generale, precum General Program Solver. Insa, dupa putin timp cercetatorii au realizat ca aceasta abordare este gresita, deoarece programele puteau rezolva doar cazuri particulare. In consecinta, cercetarile s-au indreptat inspre aceasta noua directie: dezvoltarea de programe dedicate rezolvarii unor probleme dintr-un domeniu restans de activitate. Acest lucru a dus la aparitia programelor bazate pe cunoștințe și a sistemelor expert de generația întâi.

Sistemele expert de generatia intai au fost dezvoltate in etapa premergătoare apariţiei SE actuale, numite și sisteme expert de generatia a doua. In aceasta etapa, sistemele expert cuprindeau principiile de bază in ceea ce priveste modul de rezolvare a problemelor, a logicii din spatele gasirii solutiilor. Această etapă s-a realizat la puţini ani după conferinţa de la Dartmouth College din 1956, şi anume în anii 1960 – 1970, când s-au elaborat principiile majoritare în cercetarea arborescentă şi ideile de bază ale SE care se utilizează și în prezent. Avand in vedere ca aceste sisteme expert se focalizau doar pe logica de rezolvare a problemei, ele aveau doua dezavantaje: nu faceau distinctie intre cunoștințele cu care lucrau și explicatiile erau copiate din regulile introduse in program.

După această primă etapă, în anii 1970 preocupările specialiştilor din domeniu se mută de la algoritmii şi strategiile de căutare spre cercetarea naturii cunoștințelor. Mulţi cercetători din diferite domenii, precum informatica, psihologia, filozofia şi matematica au încercat să determine natura şi structura cunoștințelor. La început s-a pornit de la cunoștințe simple, clare, ca apoi interesul lor să se îndrepte spre cunoștințele cele mai complexe, precum

cunoștințe incerte, incomplete, polisemantice, imprecise şi vagi, adică spre cunoștințe din viaţa reală. Astfel, au luat nastere sistemele expert de generatia a doua, care nu mai aveau dezavantajele precedentelor variante.

Dezvoltarea sistemelor expert s-a facut cu ajutorul unor limbajele de programare specializate. Acestea au inceput sa fie dezvoltate la putin timp dupa conferinta de la Dartmouth. In cele ce urmeaza se face o scurat istorie a primelor limbaje dedicate dezvoltarii de sisteme expert.

In anul 1959 a apărut primul limbaj de acest fel, numit LISP, dezvoltat de J. McCharty, urmând ca în 1962 acesta să publice primul manual LISP [8].

LISP (LISt Programming) este un limbaj care lucrează doar cu două entităţi: atomi şi liste. Listele sunt structurate în arborescentă binară. Dezavantajul limbajului de programare este că nu face deosebire între proceduri şi date, dar permite adăugarea de reguli sau de cunoştinţe.

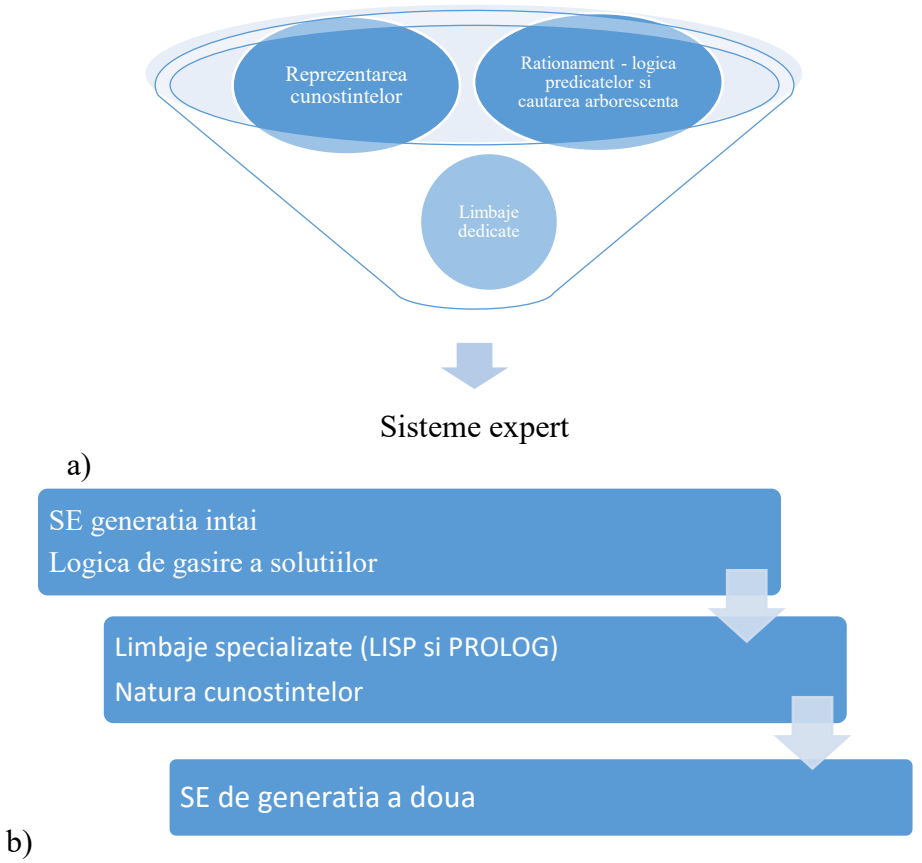


Fig. 1.1 Istoria sistemelor expert
a) Evolutia componentelor, b) etapele istorice

La începutul anilor 1970 a apărut ideea de a utiliza logica predicatelor în realizarea limbajelor de programare. Prin urmare, între anii 1970-1975 a fost dezvoltat un nou limbaj de programare, PROLOG, bazat pe logica predicatelor de ordinul întâi. Acesta a fost elaborat de Colmerauer și Roussel. PROLOG (PROgramming in LOGic) are propria strategie de demonstrare şi este un instrument puternic de elaborare a SE.

După anul 1975 a avut loc o dezvoltare rapidă a programelor bazate pe tehnicile de IA pentru diferite utilităţi: Knowledge Acquisition System - KAS (1979), Expert (1979), Knowledge Engineering Environment - KEE (1983), Vp-Expert, C Language Integrated Production System - CLIPS (1984) etc. [8].

De-a lungul timpului au fost scrise diverse limbaje dedicate dezvoltării sistemelor de IA (LISP, PROLOG, CLIPS). Totodată au fost dezvoltate motoare de inferenţă specializate în aplicaţii specifice unui anumit domeniu. In continuare se face o scurta istorie a celor mai importante aplicatii SE.

Prima aplicaţie de Sisteme Expert a fost dezvoltată de cercetătorii de la Stanford pentru NASA, care în 1960 a hotărât trimiterea pe Marte a unui vehicul care să cerceteze structura chimică a solului acestei planete. Programul elaborat a cuprins cunoștințele unor experţi umani în chimie care trebuiau să identifice structurile chimice pe baza spectrogramei de masă. Programul a fost numit DENDRAL și la final a înglobat cunoştinţele specialiştilor din domeniul spectrografiei de masă şi a analizei structurilor chimice. Fiind că acest program a cuprins atât raţionamente ştiinţifice, cât şi deducţii empirice a fost considerat primul SE. Pentru elaborarea DENDRAL a fost utilizat limbajul FORTRAN; în care cunoștințele proprii din domeniu sunt integrate cu mecanismul de inferenţă, astfel că ameliorarea SE este greu fezabilă [9].

Un alt SE care foloseşte experienţa din DENDRAL este MYCIN. Acesta operează ca un consultant pentru medici în investigarea pacienţilor pentru depistarea infecţiilor bacteriene ale sângelui, şi pentru prescrierea tratamentului adecvat. Programul este interactiv și oferă toate informaţiile legate de pacient şi rezultatele analizelor de laborator. MYCIN este capabil să îşi explice raţionamentul, iar baza de date putea fii schimbată în orice moment. Astfel că acesta la început, în 1977 a cuprins 350 de reguli, ulterior după câţiva ani în 1980, în urma experienţelor acumulate a ajuns la un număr de 450 de reguli. SE a fost scris în INTERLISP şi ocupa 130 K, dintre care 50 K ocupau sistemul de consultare, iar 20 K sistemul de achiziţie a cunoştinţelor.

HERSHAY-II este un SE de recunoaştere a parolei dezvoltat în 1979 de Erman. Acest program utiliza o memorie de lucru globală în care diferite tipuri și niveluri de informaţie erau integrate într-o structură uniformă. Este utilizată o tabelă pentru a face legătura între date, astfel că se putea avea o viziune globală asupra stării curente şi a strategiei de control.

Acestor aplicaţii ale sistemelor expert, le-au urmat multe altele, în diverse domenii precum: medicina (INTERNIST), calculatoare (XCON), chimie (PROSPECTOR), biologie moleculară (MOLGEN) etc., și firme de renume au devenit interesate de elaborarea unor SE pentru a-şi rezolva diverse probleme: General Electric, IBM, Schlumberger, HP etc.

1.3 Aplicarea SistemelorExpert în domeniul energetic

Sectorul energetic a adoptat mai lent noua tehnologie a Sistemelor Expert, în prezent asistăm la o explozie a aplicaţiilor SE în domeniul electroenergetic. Printre domeniile de aplicare ale SE în energetică amintesc: tratarea alarmelor, diagnosticarea defectelor, planificarea circulaţiei de puteri şi reglajul puterii reactive şi al tensiunii.

Complexitate crescută a sistemului electroenergetic a determinat de multe ori ca în procesul de conducere şi exploatare a acestuia să se apeleze la programe bine fundamentate matematic și bazate pe reguli euristice precum SE. Astfel, se observă tendinţa captării şi codificării cunoștințelor experţilor umani în cadrul programelor informatice și cuplarea acestora cu sisteme avansate pentru asistarea operatorilor umani. Motivaţia de a implementa Sistemele Expert în energetică rezultă în principal pentru sitautiile descrise în tabelul 1:

- Date inconsistente
- Complexitatea problemei
- Natura combinatorie
- Multitudinea datelor achizitionate

Tabelul 1.1 Motivaţia utilizării SE în energetică

Problema	**Observaţii**	**Rol SE**
Date inconsistente	Caracteristic diagnozei şi a prelucrării alarmelor	Extrage informaţia utilă dintr-o multitudine de date incomplete sau probabil conflictuale.

	Dispersarea geografică, modificarea în timp	Pentru rezolvare acestei probleme nu este disponibilă nici o tehnică convenţională
Complexitatea structurii sistemului electroenergetic	Diferite niveluri de tensiune, topologie complexă (buclată şi radială), diferite forme de relief	Modelează cunoștințele expertului uman pentru a identifica rapid regiunile slabe din sistemul electroenergetic
Natura combinatorie a soluţiilor	Restaurarea stării de funcţiune după o avarie în sistem	Strategie de căutare într-o problemă multidimesională – construieşte scenarii ale stărilor posibile și caută soluţia în spaţiul acestor stări
Multitudinea datelor achiziţionate	Reglarea unor procese complexe	Selectează datele și le sortează pentru a implica doar datele care sunt necesare rezolvării problemei apărute

Autoarea consideră că în conceperea şi introducerea SE în energetica trebuie avut în vedere că omul rămâne principalul element de decizie. Astfel, sistemele inteligente asistându-l în această activitate în situaţii de rutină sau excepţionale pentru a-l elibera de anumite sarcini şi pentru a-i da posibilitatea să se concentreze pe acele activităţi care implică în mod necesar elementul uman în alegerea şi executarea acţiunilor. În [8] se specifică că sistemele inteligente trebuie considerate, pentru o perioadă, ca auxiliare şi nu ca alternative pentru funcţiile actuale ale dispecerilor [8].

Prima aplicaţie a SE în electroenergetică a apărut dintr-o necesitate: asistarea operatorilor umani în centralele nuclearo-electrice. Acest fapt a apărut în anul 1979 în urma accidentului de la centrala nucleară Three Mile Island, S.U.A. Accidentul a început în sectorul auxiliar al centralei, urmată de un blocaj în sistemul primar, ceea de a determinat pierderea lichidului de răcire. Eşecurile

mecanice au fost agravate de eşecul iniţial al operatorilor umani şi incapacitatea lor de a recunoaşte gravitatea situaţiei. Astfel, accidentul a arătat că omul are anumite limite, iar volumul mare de mesaje sau informaţii de natură diversă (semnalizări acustice, luminoase, măsurători electrice de presiune, de temperatură, funcţionări de protecţii prin relee și altele) pe care trebuia să le prelucreze într-un interval de timp relativ scurt, fac dificilă munca și uneori chiar imposibilă luarea unor decizii corecte. În concluzie, primul SE în energetică a fost dezvoltat de EPRI (Electric Power Research Institute) care avea rolul de a asista operatorii umani şi de a găsi cauzele unor incidente apărute la centralele nuclearo-electrice. Acestui program inteligent i-au urmat multe altele, printre care amintesc: NPPC (Nuclear Power Plant Consultant) - consultant în asistarea operatorilor din centrale nucleare pentru stabilirea cauzelor unor avarii (1982, Georgia Institute of Technology), REACTOR - pentru diagnosticarea accidentelor într-un reactor nuclear (1982, EG şi Idaho) și altele [2].

În domeniul reţelelor electrice, în 1983 japonezii au prezentat primul SE dedicat reconstituirii unui sistem electric după o avarie. Apoi în 1986, americanii au lansat un SE pentru asistarea deciziilor în controlul tensiunii şi puterii reactive şi altul pentru diagnosticarea defectelor prin relee. În Europa, specialiştii francezi au realiza programul SEPT dedicat analizei incidentelor şi diagnosticarea funcţionării protecţiilor prin relee.

Domeniile de aplicare ale SE în electroenergetică sunt:

1. tratarea alarmelor - Estimarea stării ajută la filtrarea și validarea alarmelor generate de sistemul SCADA. Prin compararea valorilor măsurate și estimate, aceasta distinge evenimentele reale de alarmele false cauzate de erori ale senzorilor, erori de comunicare sau perturbații tranzitorii.
2. diagnosticarea defectelor - SE identifică modificări anormale ale parametrilor rețelei (cum ar fi căderi bruște de tensiune sau vârfuri de curent) care pot indica defecțiuni. Ajută la localizarea defecțiunii și la evaluarea severității acesteia, susținând decizii rapide de izolare și restaurare.
3. planificarea circulaţiei de puteri - SE oferă un model precis și în timp real al sistemului energetic, esențial pentru planificarea fluxurilor viitoare de energie în diferite scenarii de încărcare și generare. Acest lucru ajută la optimizarea dispecerizării generării și a programării transmisiei.

4. reglajul puterii reactive şi al tensiunii - Variabilele de stare precise (tensiuni și unghiuri) permit SE să susțină controlul puterii reactive și să mențină tensiunea în limite acceptabile. Identifică zonele cu tensiune slabă și sugerează acțiuni de control, cum ar fi comutarea condensatoarelor sau schimbarea prizelor.
5. configurarea reţelelor electrice - SE ajută la monitorizarea și verificarea configurației reale a sistemului energetic, inclusiv a stării întrerupătoarelor și comutatoarelor. Poate detecta inconsecvențe între topologiile așteptate și cele în timp real, îmbunătățind modelarea și funcționarea rețelei.
6. restaurarea sistemului - După o pană de curent sau o perturbație majoră, SE sprijină operatorii în restabilirea unei rețele stabile prin identificarea părților energizate ale sistemului și ghidarea secvențelor de restaurare pas cu pas.
7. evaluarea securităţii în funcţionare - SE introduce date în timp real în instrumentele de analiză a securității pentru a evalua contingențele N-1, riscurile de supraîncărcare, încălcările de tensiune sau marjele de stabilitate. Acest lucru ajută la prevenirea defecțiunilor în cascadă sau a întreruperilor de curent.
8. probleme de stabilitate tranzitorie - Estimarea stării în timp real îmbunătățește detectarea și analiza instabilităților dinamice, cum ar fi oscilațiile sau separările unghiulare. Poate fi integrată cu modele dinamice pentru a prognoza răspunsul sistemului după perturbații.
9. stabilirea tranşelor de deconectare - SE ajută la definirea strategiilor de deconectare bazate pe priorități (deconectare de sarcină) în timpul situațiilor de urgență. Prin estimarea condițiilor sistemului, aceasta susține definirea tranșelor care minimizează întreruperea serviciului, asigurând în același timp integritatea sistemului.
10. formarea şi instruirea operatorilor - Folosind simulări bazate pe SE, operatorii pot fi instruiți în scenarii realiste, bazate pe date. Aceasta include instruirea pentru condiții de defecțiune, restaurarea sistemului și gestionarea situațiilor de urgență bazată pe o modelare precisă a sistemului.
11. interfaţarea prietenoasă om – maşină - SE îmbunătățește calitatea datelor afișate în camera de control, prezentând operatorilor informații de sistem consistente, filtrate și validate. Îmbunătățește vizualizarea

variabilelor cheie și a stărilor sistemului, sprijinind o luare a deciziilor mai rapidă și mai bună.

12. planificarea întreţinerii reţelelor - Estimarea stării identifică legăturile slabe, liniile supraîncărcate sau echipamentele care se apropie de limitele operaționale. Acest lucru ajută la planificarea eficientă a întreținerii preventive și la reducerea riscului de întreruperi neașteptate.
13. automatizarea staţiilor electrice - SE îmbunătățește procesul decizional local în substații prin furnizarea de informații precise, de înaltă rezoluție. Permite controlul automat al întrerupătoarelor, comutatoarelor de prize și condensatoarelor pe baza condițiilor la nivelul întregului sistem.
14. prognoza de sarcină - Deși SE în sine nu este un instrument de prognoză, acesta oferă date istorice de înaltă calitate care îmbunătățesc precizia modelelor de predicție a sarcinii, în special în prognoza pe termen scurt, esențială pentru operațiunile în timp real.
15. gestionarea sarcinii - SE susține managementul cererii prin monitorizarea profilelor de sarcină în timp real, ajutând la implementarea strategiilor de răspuns la cerere sau la reducerea sarcinii țintită pentru a evita problemele legate de vârfurile de cerere sau supraîncărcarea rețelei.

1.4 Intrebări și exerciții

1. Sistemele SCADA sunt sisteme expert?

2. Sistemele expert sunt tehnici de IA care se bazează pe

Intuiție sau Logică ?

3. Un motiv pentru care se dorește utilizarea sistemelor expert în energetică este

Date inconsistente sau Logica circulației puterilor.

4. Primele sisteme expert în domeniul energeticii au apărut în

Rețele electrice sau Generare.

2

Arhitecturi și tipuri de sisteme expert

Acest capitol face o introducere în domeniul Sistemelor Expert prin descrierea caracteristicilor și a atributelor acestor programe de IA, respectiv prezentarea unei comparaţii detaliate între acestea şi programele convenţionale de prelucrare a datelor (cunoștințe).

2.1 Aspecte generale

Sistemele Expert sunt definite ca fiind programe informatice, inteligente bazate pe cunoaştere, ce rezolvă probleme care în mod normal necesită experienţa unui specialist din domeniu. Asemenea unui specialist care se folosește de memoria sa pe termen lung, experiență și raţionamente mai mult sau mai puţin obiective, un SE conţine o bază de cunoștințe asociată domeniului respectiv, respectiv utilizează raţionamente logice pentru a lua decizii şi a rezolva problemele apărute.

Principiile de bază ale SE au fost concepute în prima etapă de dezvoltare a limbajelor de programare dedicate, când s-a făcut distincţia dintre componenta care conţine cunoștințele şi componenta de manipulare a acestor informaţii. Acest mod de formulare a problemei poartă numele de abordare declarativă.

Metodele de rezolvare a problemelor în cadrul SE se bazează pe tehnici de raţionament (o înlănţuire de propoziţii în care plecând de la anumite cunoștințe se ajunge la o cunoştinţă nouă, sau o înlănţuire de reprezentări simbolice care trebuie să conducă la atingerea unui scop) calitativ care leagă cunoştinţele din baza de cunoştinţe.

Conceperea unui SE este un proces iterativ prin care în urma unor discuţii repetate cu experţii umani se dezvoltă baza de cunoștințe.

Diferenţa dintre Sistemele Expert şi programele convenţionale de gestiune a datelor începe de la limbajele şi mediile de programare folosite.

Limbajele algoritmice utilizate în dezvoltarea programelor convenţionale descriu într-un tot nediferenţiat elementele de logică necesare rezolvării

problemei, procedurile și proprietăţile caracteristice obiectelor problemei. Elementul de bază în programarea algoritmică este algoritmul – o procedură care garantează fie găsirea soluţiei corecte a unei probleme într-un timp dat, fie afişează răspunsul că nu există soluţie.

Precum a fost subliniat în subcapitolul precedent, în cadrul SE cunoștințele şi deducţiile logice sunt separate. Modul de implementare al programelor convenţionale le face greu de modificat şi actualizat. Acest dezavantaj nu se regăseşte la SE la care datorită separării între baza de cunoștințe şi motorul de inferenţe, modificarea oricăreia dintre ele nu o influenţează pe cealaltă, în acest sens SE sunt mai flexibile decât programele convenţionale.

O comparaţie amănunţită între SE şi programele convenţionale după mai multe criterii se prezentată în tabelul 2.1. O analiză a comparaţiei din tabelul 2.1 scoate în evidenţă următoarele avantaje ale SE:

- Căutarea soluţiei este accentuată pe găsirea unei soluţii, chiar dacă aceasta nu este varianta cea mai bună;
- Datele de intrare pot să aibă un caracter incomplet, incert. Acest tip de date sunt caracteristice situaţiilor reale;
- Utilizatorul primeşte explicaţii legate de raţionamentul de rezolvare utilizat, precum și paşii parcurşi;
- Modificarea SE este accesibilă.

Tabelul 2.1. Diferenţele dintre SE și programele convenţionale de gestiune a datelor

Nr.	Caracteristică	Program convenţional	Sistem Expert
1	Control de	Ordinea declaraţiei	Motor de inferenţă
2	Control & Date	Integrare implicită	Separare explicită
3	Control	Puternic	Slab
4	Soluţia dată de	Algoritm	Reguli şi inferenţă
5	Căutarea soluţiei	Mică sau deloc	Mare
6	Rezolvarea problemei	Algoritm	Reguli
7	Intrare	Presupus corectă	Incompletă, incorectă
8	Intrare neaşteptată	Dificil de lucrat cu	Foarte corespunzător

9	Ieşirea	Mereu corectă	Variază în funcţie de problemă
10	Explicarea	Nu există	De obicei
11	Aplicaţii	Numerice, fişiere şi text	Reprezentări simbolice
12	Execuţia	În general secvenţială	Reguli oportuniste
13	Design –ul programului	Structurat	Puţin sau deloc
14	Modificare	Dificilă	Rezonabilă
15	Expansiune	La fiecare pas	Incremental

Atributele SE sunt acele caracteristici care le fac să se distingă de celelalte programe de acelaşi tip. În paragraful precedent s-a prezentat o comparaţie între SE şi programele convenţionale; in continuare se prezintă principalele caracteristici ale SE care le definesc ca atare. Astfel, atributele unui SE sunt [1, 2]:

1. Lucrează la un nivel expert de competenţă – cunoștințele sunt dintr-un domeniu relativ restrâns şi sunt bine specificate;
2. Cunoştinţele utilizate se bazează exclusiv pe informaţiile date de un expert din domeniu studiat;
3. Cunoștințele sunt reprezentate sub formă declarativă, de cele mai multe ori sub formă de reguli;
4. Expertiza efectuată este în funcţie de cunoștințele special dobândite;
5. Foloseşte un mecanism de inferenţă pentru a realiza deducţiile – motor de inferenţă;
6. Baza de cunoștințe este net separată de mecanismul de manipulare a cunoștințelor. Baza de cunoștințe poate fi folosită ulterior în alte aplicaţii, sau modificată în funcţie de necesităţile problemei;
7. Comunicarea cu utilizatorul se face pe baza unor module specializate care realizează integrarea, comunicarea, explicarea și furnizarea de cunoștințe utilizatorului.

2.2 Structura sistemelor expert

Arhitectura SE cuprinde totalitatea componentelor necesare pentru buna funcţionare a SE şi rezolvarea problemelor într-un timp util, care să justifice aplicarea acestuia.

În literatura de specialitate există mai multe variante privind structura SE. Unele variante prezintă arhitecturi complexe, iar unele mai simple. Variantele simple conţin doar elementele strict necesare pentru funcţionarea SE. În comparaţie cu acestea, cele complexe sunt alcătuite atât din modulele principale care intră în componenţa structurilor simple (de bază) cât și din module auxiliare care fac SE mai accesibile utilizatorilor, și în acest sens mai comerciale.

În prezentul capitol se vor prezenta două arhitecturi de SE: arhitectura de bază a SE, respectiv o variantă complexă.

Arhitectura SE a fost construită la început după modelul expertului în domeniu: memoria pe termen lung, raţionamentele logice, explicarea raţionamentului. Aceste elemente au fost introduse în structura de bază al unui SE sub forma bazei de cunoștințe,a motorului de inferenţă şi a modulului explicativ. În ultimii ani, odată cu dezvoltarea ştiinţei calculatoarelor şi creşterea cerinţelor utilizatorilor, acestor elemente s-au adăugat altele care să facă SE mai uşor de înţeles și utilizat.

În arhitectura unui SE pot fi regăsite următoarele componente:

- Baza de cunoștințe, BC;
- Motorul de inferenţă, MI;
- Interfaţa Grafică cu Utilizatorul, IGU;
- Modulul Explicativ, ME;
- Modulul de Achiziţie a Cunoştinţelor, MAC;
- Interfaţa cu Inginerul de Cunoștințe, MIC;
- Modulul dinamic, MD.

Arhitectura de bază a unui SE este ilustrată în fig. 1, în care se observă componentele necesare care alcătuiesc un SE: baza de cunoștințe, motorul de inferenţă, interfaţa grafică cu utilizatorul și modulul explicativ.

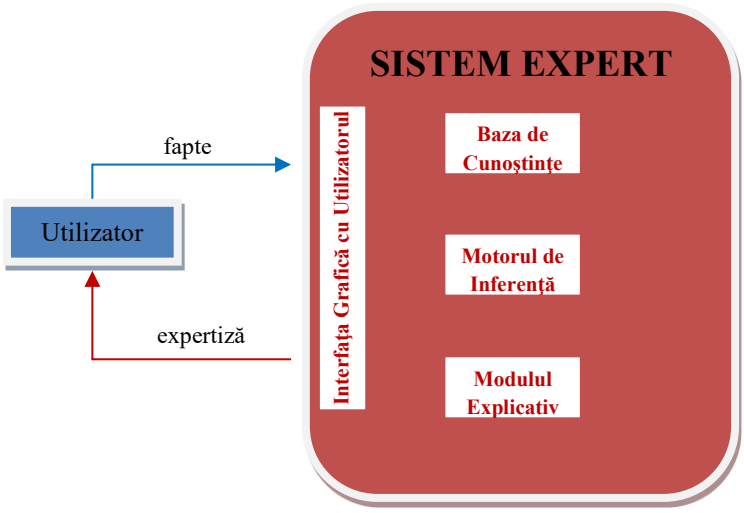


Fig. 2.2 Arhitectura de bază a SE

Arhitectura cea mai complexă a unui SE cuprinde pe lângă componentele de bază şi module auxiliare care fac programul eficient, comercial şi accesibil cât mai multor utilizatori.

Structura unui astfel de SE este descrisă în fig. 2, unde se pot observa şi legăturile dintre componentele SE şi personalul cu care acesta vine în contact: inginerul de cunoștințe, inginerul programator, expertul în domeniu și utilizatorul.

Baza de cunoștințe reprezintă componenta care conţine toate datele din domeniu de specialitate. Aceasta este alcătuită la rândul ei din reguli și fapte. Regulile se referă la operaţiile ce pot fi efectuate asupra obiectelor conţinute în baza de date. Regulile se modifică mai rar, ca urmare, de exemplu, a unor schimbări în regulamentele tehnice. În esenţă regulile constituie un ansamblu complet şi necontradictoriu de cunoștințe necesare rezolvării unei probleme. Faptele reprezintă elementele care arată starea problemei la un anumit pas în rezolvarea acesteia [1].

Baza de date este o componentă externă SE, dar este la fel de importantă ca orice componentă internă; ea are un caracter dinamic și cuprinde informaţiile relative legate de domeniu de aplicaţie studiat.

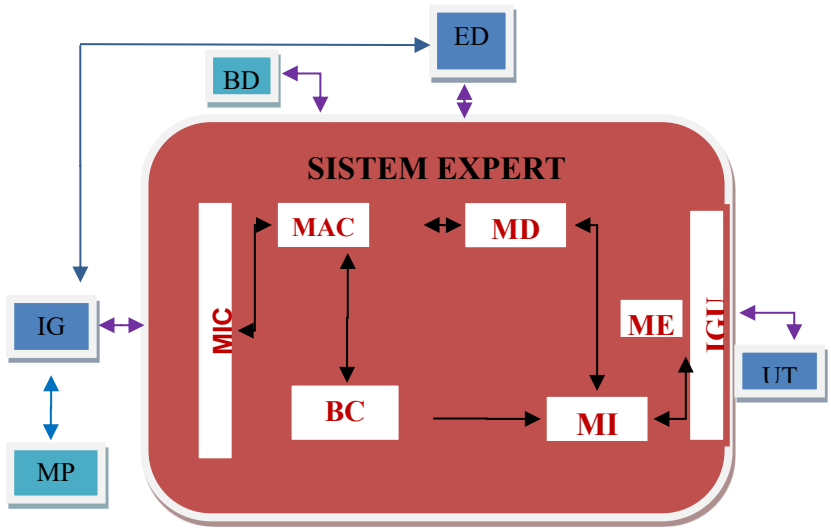


Fig. 2.3 Arhitectura complexă a Sistemelor Expert. IG-iginer cunoștințe, MP – mediu de programare, BD-bază de date, ED-expert din domeniu, U-utilizator, MIC-modulul de interfață cu inginerul de cunoștințe, MAC-modulul de achiziție de date, MD-modulul dinamic, BC-baza de cunoștințe, ME-modulul explicativ, MI-motorul de inferențe, IGU-interfața grafică cu utilizatorul

Motorul de inferenţă are rolul de a da sens regulilor de inferenţă, legându-le între ele, astfel încât să fie puse întrebările necesare pentru a obţine răspunsurile corecte. Fără o strategie adecvată de control, un SE riscă să necesite timpi de rezolvare foarte mari.

Modulul dinamic cuprinde memoria de lucru şi faptele dinamice. Nu este o componentă obligatoriu independentă, deoarece datele pot fi automat memorate de alte componente, şi astfel aceasta este cuprinsă în structura altor componente.

Modulul de achiziţie a cunoştinţelor realizează transferul cunoştinţelor de la anumite surse către SE. Expertiza din domeniu care trebuie să fie transferată unui SE în baza sa de cunoștințe este o colecţie de definiţii, relaţii, fapte specializate, euristici, proceduri, strategii şi ipoteze.

Interfaţa grafică cu utilizatorul este componenta care face dialogul cu utilizatorul. Aceasta cuprinde toate elementele grafice necesare pentru a ajuta utilizatorul în rezolvarea problemei şi înţelegerea raţionamentului de deducere a soluţiilor.

Interfaţa cu inginerul de cunoștințe este componenta care nu este accesibilă utilizatorului, ci doar celui care dezvoltă SE. Această componentă poate să

lipsească, dar în prezent există multe SE la care programatorii preferă să creeze o interfaţă specială care ulterior să îi ajute să modifice mai uşor SE (baza de cunoștințe și motorul de inferenţă).

2.3 Tipuri de Sisteme Expert

Sistemele expert se împart în mai multe clase în funcţie de utilizare, modul de reprezentare a cunoştinţelor și realizare a inferenţelor, respectiv de aplicaţia suport pe care a fost construit SE. Prezentul capitol descrie diferitele tipuri de SE care se regăsesc în aplicaţiile practice, respectiv face o descriere amănunţită a SE de clasificare, control şi diagnoză.

În funcţie de utilizarea lor, adică de natura scopului urmărit, SE se împart în SE de clasificare, de control, respectiv de diagnoză [3, 4]. Această clasificare este mai largă, în literatura de specialitate există o clasificare mai amănunţită (SE de diagnoză, reparaţie, instruire, monitorizare etc.). O analiză mai atentă a acestei clasificări relevă faptul că aceste tipuri de SE se restrâng în cele trei clase enunţate anterior. Nu se va insista pe descrierea acestora deoarece în compoziţia capitolului sunt prevăzute subcapitole dedicate acestora.

Din punctul de vedere al reprezentării cunoștințelor și a modului de parcurgere a Bazei de cunoștințe se disting Sisteme Expert bazate pe [5]:

- Reguli (Rule – based Expert Systems) – reprezentarea cunoștințelor se face cu ajutorul regulilor de producţie, iar soluţiile se obţine prin realizarea inferenţelor dintre acestea;
- Logică (Logic based Expert Systems) – baza de cunoștințe este alcătuită din formule logice. Un interes major în utilizarea acestei metode de reprezentare a cunoștințelor este în special pentru că oferă posibilitatea obţinerii unor rezultate noi folosind reguli de inferenţă de tipul modus ponens;
- Obiecte (Object oriented Expert Systems) – cunoștințele sunt reprezentate sub forma unor obiecte care au anumite caracteristici, iar raţionamentele se fac în funcţie de tipul problemei de rezolvat;
- Iniţiere (Induction based Expert Systems) – aceste SE sunt asemănătoare celor bazate pe reguli, însă unii autori fac distincţie între cele două, deoarece în cazul celor bazate pe reguli nu există o prioritate în reprezentarea cunoștințelor, fapt ce se întâlneşte în cazul celor bazate pe iniţiere;
- Sisteme Expert hibride – se caracterizează prin faptul că, în elaborarea bazei de cunoștințe și în realizarea raţionamentelor se folosesc şi alte

tehnici de IA, precum Logica Fuzzy, Reţele Neuronale Artificiale și Algoritmi Genetici.

O altă metodă de clasificare a Sistemelor Expert este în funcţie de tipul cunoștințelor selectate pentru a fi introduse în baza de cunoștințe, respectiv modalitatea de găsire a soluţiilor (case-based reasoning și model-based reasoning). Astfel, sunt SE bazate pe cazuri (Case-based Expert Systems , respectiv bazate pe modele (Model-based Expert Systems) [6].

Din punctul de vedere al modului în care se poate dezvolta un SE, se disting două tipuri: SE suport (cadru) şi SE individuale (personalizate). Prima categorie sunt acele SE dezvoltate pentru o anumită categorie de aplicaţii, programatorul trebuie doar să ataşeze o bază de cunoștințe specializată dedicată problemei de rezolvat. Al doilea tip de SE sunt cele care sunt construite de la început de către programator, care creează baza de cunoștințe, motorul de inferenţă şi celelalte module necesare.

Sistemele Expert de clasificare sunt primele tipuri de SE dezvoltate, şi în prezent reprezentând o mare majoritate comparativ cu celelalte tipuri. Aceste SE sunt folosite pentru identificarea şi clasificarea cauzelor posibile ale unei funcţionări defectuoase (cantitative, calitative, vizuale) în scopul de a determina semnificaţiile acestor tipuri de date [3] pentru a se lua măsurile necesare. De asemenea, se folosesc pentru instruirea personalului, prin care SE face o clasificare a metodelor aplicabile pentru a învăţa diferite tipologii de oameni. O altă aplicaţie este în interpretarea diferitelor date pentru a determina gradul de necesitate şi eficienţă; de exemplu sunt folosite în exploatarea zăcămintelor de minerale, gaz şi petrol, dar și în recunoaşterea limbajului natural. Şi nu în ultimul rând, acest tip de SE sunt folosite în proiectarea și planificarea activităţilor şi proceselor, aplicaţii în care SE înlocuiesc unele activităţi ale experţilor pentru alegerea echipamentelor celor mai bune pentru o situaţie particulară în funcţie de costuri, caracteristici tehnice etc.

Un tip particular de SE de clasificare sunt cele care fac diagnoza tehnică. Apariţia şi dezvoltarea acestor SE a fost influenţată de utilizarea lor principială în special în medicină unde se dorea o clasificare cât mai precisă a parametrilor și caracteristicilor unei afecţiuni pentru a da un diagnostic precis pacienţilor. Cel mai fidel exemplu este primul SE în medicină, MYCIN. Acest sistem era utilizat în diagnosticarea infecţiilor bacteriene, cunoștințele fiind reprezentate sub formă de reguli, care făceau legăturile dintre simptome și posibilele afecţiuni. Cunoştinţele folosite erau cu precădere empirice, fiind numite și „cunoștințe de

suprafaţă” [3]. Dezavantajul în utilizarea cunoştinţelor de suprafaţă constă în faptul că este foarte dificil să se explice raţionamentele făcute. Modul de operare al SE MYCIN se baza pe următoarele două caracteristici: strategia de control înainte și căutare exhaustivă (completă) a soluţiilor. Ulterior au fost dezvoltate SE care să poată utiliza și „cunoștințe profunde”, însă acestea au dezavantajul că sunt greu de construit deoarece trebuie să se folosească reguli care au un caracter cât mai general pentru a se putea implementa pe mai multe tipuri de maşini. Cu scopul de a elimina neajunsurile introduse de utilizarea exclusivă a unui tip de cunoștințe (suprafaţă sau profunde), sunt dezvoltate SE care utilizează atât cunoștințe de suprafaţă, cât şi profunde.

Sistemele Expert de control includ toate SE care presupun controlul, monitorizarea şi optimizarea proceselor. Aceste SE sunt caracterizate prin introducerea noţiunii de „timp”, parametru necesar fiind că acestea lucrează în timp real, şi au rolul de a supraveghea buna evoluţie a diferitelor procese tehnologice. Modul de lucrul al acestor SE este următorul: cu ajutorul informaţiile (date şi semnale) primite de la dispozitivele de măsură (senzori și traductoare) ataşate echipamentelor procesului analizat, se fac diverse raţionamente pentru adaptarea parametrilor procesului și menţinerea constantă a acestora pe toată durata actului tehnologic.

Dificultatea în cazul SE de control constă în interpretarea noului parametru. Astfel, aceste SE trebuie să „măsoare” intervalele de timp pentru a genera declanşarea unei acţiuni la un anumit moment astfel încât să fie îndeplinite condiţiile necesare, iar evoluţia procesului să fie conform datelor de intrare. În acest sens, cunoștințele, de cele mai multe ori, sunt reprezentate sub formă de tabele cu date, din care SE selectează strategiile ce vor fi aplicate la momentele următoare în funcţie de valorile mărimii controlate.

Denumite şi SE de anticipare [3], SE de prognoză se caracterizează printr-o alocare a resurselor în funcţie de anumite restricţii. Astfel se urmăreşte anticiparea unui rezultat pe baza cunoştinţelor prezente și a evoluţiilor trecute. În rezolvarea problemelor și afişarea rezultatelor se folosesc factori de probabilitate care indică gradul în care o soluţie este posibilă.

Acest tip de SE este utilizat în ordonarea şi planificarea producţiilor, în meteorologie etc.

Modul în care sunt construite diferitele SE nu permite utilizarea lor în alt scop decât în cel pentru care au fost concepute. Adică, un SE de clasificare nu poate fi folosit pentru a se face diagnoza, însă poate fi adaptat pentru realizarea

unor activităţi asemănătoare. Sintetizarea tipurilor de SE este ilustrată în figura 2.4.

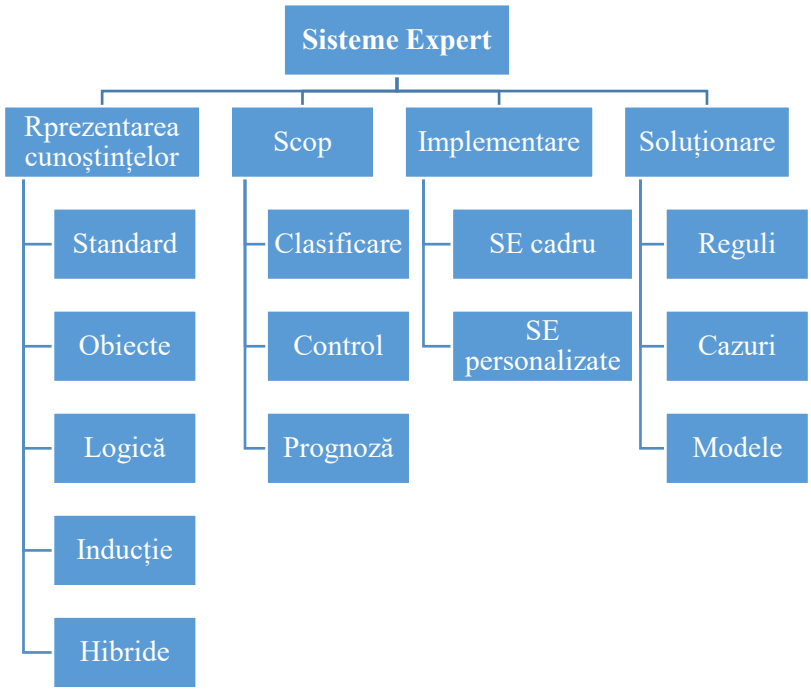


Fig. 2.4. Tipuri de sisteme expert

Dacă se iau în considerare trei caracteristici de bază ale SE, adică reprezentarea cunoștințelor, metoda de raționament și structura motorului de inferență, atunci SE se clasifică în următoarele categorii de bază:

1. Sisteme expert bazate pe reguli

Aceste SE sunt cele mai comune, datorită simplității lor. Ele utilizează reguli de tipul ”DACĂ-ATUNCI” pentru reprezentarea cunoștințelor, și raționamentul de înlănțuire înainte sau înapoi în interiorul motorului de inferență.

2. Sisteme expert bazate pe cadre

Aceste sisteme modelează mai ușor entități structurate (de exemplu, dispozitive sau subsisteme). Ele utilizează cadre (obiecte cu atribute), și astfel organizează cunoștințele ierarhic.

3. Sisteme expert fuzzy

Sistemele expert fuzzy sunt bune pentru aplicațiile care utilizează intuiția expertă. Ele folosesc logica fuzzy cu variabile lingvistice (de exemplu, „tensiune joasă”) și gestionează cunoștințe vagi sau imprecise.

4. Sisteme expert bazate pe cazuri

Aceste SE utilizează cazuri anterioare pentru a rezolva probleme noi. În consecință, în baza de cunoștințe sunt incluse memoria cazurilor, recuperarea, adaptarea și stocarea acestora. Un exemplu tipic din categoria aceasta de SE este diagnosticarea unei defecțiuni prin reamintirea unor defecțiuni similare din trecut.

5. Sisteme expert bazate pe modele

Sistemele expert bazate pe modele sunt mai mult analitice decât euristice. Ele conțin un model funcțional al sistemului (de exemplu, ecuații sau modele bloc). Raționamentul folosit în acestea se bazează pe simularea sau interpretarea comportamentului sistemului.

6. Sisteme expert Blackboard

Aceste SE utilizează o structură centrală de date („blackboard") unde contribuie diferite surse de cunoștințe. Ele sunt potrivite pentru rezolvarea problemelor în mai mulți pași sau colaborativă.

7. Sisteme expert în timp real

Acestea sunt concepute să funcționeze în condiții de constrângere de timp. Ele necesită motoare de inferență rapide și baze de reguli optimizate. Cele mai comune aplicații sunt cele de control sau protecție.

8. Sisteme expert distribuite

Sistemele expert distribuite sunt utile pentru sisteme la scară largă, cum ar fi rețelele inteligente sau substațiile distribuite. Acestea cuprind mai multe module expert care funcționează în locații sau subsisteme. Aceste subsisteme fac schimb de cunoștințe și inferențe parțiale.

9. Sisteme expert încorporate

Un exemplu de aplicație acestui tip de SE este detectarea locală a defecțiunilor într-un releu de protecție. Aceste sisteme sunt implementate în controlere hardware sau microprocesoare. Ele necesită memorie limitată și execuție rapidă.

10. Sisteme expert cu siguranțe sau hibride

Aceste sisteme combină două tipuri de sisteme expert, de exemplu SE bazat pe reguli împreună cu logică fuzzy, sau SE bazat pe cadre plus sistemele în timp real.

11. Sisteme expert bazate pe web

Aceste sisteme expert sunt accesibile printr-o interfață web. Ele sunt ușor de actualizat și disponibile pentru mai mulți utilizatori. De exemplu un sistem expert de consultanță online pentru dimensionarea instalațiilor electrice.

2.4 Intrebări şi exerciţii

1. Într-un Sistem Expert controlul este deţinut de către
   Motorul de Inferenţă sau Modulul de Interfaţă Grafică cu Utilizatorul

2. În cadrul unui Sistem Expert Baza de Cunoştinţe şi Motorul de Inferenţă sunt net delimitate
   Adevărat sau Fals

3. Care componente intră în structura de bază a unui SE
   Baza de Cunoştinţe sau Modulul Explicativ sau Modulul de Interfaţă cu Inginerul de Cunoştinţe sau Motorul de inferență sau Modulul Dinamic sau Modulul de Achiziţie a Cunoştinţelor sau Modulul de Interfaţă cu Utilizatorul

4. Componentele auxiliare din structura unui sistem expert sunt următoarele
   Baza de Cunoştinţe sau Modulul Explicativ sau Modulul de Interfaţă cu Inginerul de Cunoştinţe sau Motorul de inferență sau Modulul Dinamic sau Modulul de Achiziţie a Cunoştinţelor sau Modulul de Interfaţă cu Utilizatorul

5. Sistemul expert care are în componența sa elemente ce depind de parametru timp sunt
   SE de control / SE de prognoză // SE de clasificare

3

Cunoștințe și achiziția lor

Acest capitol descrie în amănunt problema achiziției şi natura cunoștințelor cu care lucrează Sistemele Expert. De asemenea se prezintă actorii care iau parte la procesul de achiziție a cunoștințelor, precum și factorii care influențează achiziția cunoștințelor în vederea dezvoltării bazei de cunoștințe a unui SE.

3.1 Introducere

Incă de la începuturile utilizării tehnicilor de IA, în particular a SE, specialiştii din domeniu au observat faptul că eficacitatea acestor programe inteligente depinde în primul rând de cunoștințele care le posedă, şi în al doilea rând de mecanismele de inferenţă utilizate. În consecinţă, şi-au îndreptat atenţia înspre înțelegerea naturii cunoștințelor, reprezentarea şi implicit achiziţia lor, respectiv codificarea lor în limbajul maşinii.

Cunoștințele sunt componentele cunoașterii, care se obțin din *date*, *informații* și *experiență* prin procese de cercetare, reflexie, acumulare, execuție și interacțiune. Cunoașterea poate să evolueze în înțelepciune, atunci când cunoștințele acumulate sunt folosite pentu a găsi soluții de rezolvare a unor probleme dificile sau noi.

Datele reprezintă caracteristici fundamentale asociate cu evenimente, obiecte, ființe vii sau persoane. Acestea sunt de obicei derivate din observații sau înregistrări factuale și pot fi discrete sau statice. Cel mai adesea, datele sunt exprimate folosind simboluri - numerice și/sau lingvistice. Deși datele își propun să reflecte realitatea obiectivă, ele nu sunt întotdeauna exacte, iar verificarea corectitudinii lor poate fi dificilă. Prin urmare, în multe cazuri, devine necesar să se aplice abordări și metodologii specifice pentru a valida sau rafina datele pentru o utilizare ulterioară. Din această perspectivă, datele pot fi empirice (de exemplu, măsurători zilnice ale presiunii atmosferice) sau non-empirice (de exemplu, numele zilelor săptămânii).

Toate datele sunt *informații*, afirmație care nu este adevarată în sens invers. Intr-adevăr, sunt informații considerate ca fiind date procesate, acestea având o importanță mai mare în luarea deciziilor decât datele simple. Procesarea datelor presupune folosirea unor procese de agregare, calcul, corecție etc. Comparativ cu datele, *informația* are un scop și un sens, astfel datele într-un anumit context pot fi considerate informație. Mai departe, informațiile pot să fie procesate, accesate, generate, stocate etc. Pe de altă parte, informațiile pot să fie de diferite tipuri și să aibă caracteristici diverse. Mai exact, informațiile pot fi sensibile (private), calitative și cantitative.

Informațiile sensibile sunt considerate date care este necesar a fi protejate în vederea accesării neautorizate, pentru a asigura intimitatea și siguranța unei persoane sau a unei organizații. Acestea se împart în trei categorii: personale, de afacere și guvernamentale.

Informațiile calitative se referă la caracteristici, descrieri sau afirmații care nu pot fi ușor măsurate sau exprimate numeric. Spre deosebire de datele cantitative, care se bazează pe cantități măsurabile, datele calitative se concentrează pe calitățile, atributele sau esențele lucrurilor - adesea tratând semnificații, interpretări și evaluări subiective. Astfel de informații sunt obținute de obicei prin surse non-numerice, cum ar fi texte scrise, limba vorbită, interviuri, răspunsuri la sondaje deschise, fotografii, desene, înregistrări video și observații. Acestea subliniază cum este ceva, mai degrabă decât cât de mult există din el. Informațiile calitative sunt adesea dependente de context și pot varia în funcție de factori culturali, emoționali sau perceptivi. De exemplu, opiniile oamenilor despre un serviciu public, simbolismul dintr-o pictură sau limbajul corpului observat într-o întâlnire înregistrată sunt toate forme de date calitative. Aceste informații sunt de obicei transmise sau documentate folosind medii precum text (de exemplu, rapoarte, narațiuni), figuri (de exemplu, diagrame, schițe), imagini (de exemplu, fotografii, infografice), hărți (de exemplu, hărți etnografice sau tematice) sau videoclipuri. Analiza acestor date implică interpretarea modelelor, temelor sau semnificațiilor, mai degrabă decât efectuarea de calcule matematice.

Informațiile cantitative se referă la date care pot fi măsurate, numărate sau exprimate numeric. Acestea se ocupă de cantități - răspunzând la întrebări precum „cât”, „câte”, „cât de des” sau „în ce măsură”. Acest tip de informații sunt obiective, structurate și pot fi supuse analizei statistice sau matematice. Datele cantitative sunt de obicei colectate prin metode precum măsurători, experimente, sondaje cu întrebări închise, senzori, instrumente sau proceduri de numărare. Valorile pe care le produc pot reprezenta cantități specifice, intervale

sau spectre ale unei anumite caracteristici sau fenomene - cum ar fi înălțimea, temperatura, frecvența, durata sau costul. Aceste informații sunt deosebit de utile atunci când sunt necesare precizia, comparabilitatea și generalizabilitatea. Ele pot fi organizate în tabele, diagrame sau grafice, iar analiza lor implică adesea medii, procente, tendințe sau corelații. Datorită naturii lor numerice, informațiile cantitative permit comparații, predicții și concluzii bazate pe dovezi între populații sau în timp. Cu toate acestea, deși dezvăluie cât de mult există din ceva, adesea aceste date nu explică de ce sau cum se produce acel fenomen - domenii explorate de obicei prin date calitative.

Informațiile stau la temelia cunoașterii, care se consideră ca fiind o caracteristică a oamenilor. Cunoașterea asociată unei persoane arată înțelegerea acesteia într-un domeniu sau legată de un subiect, și poate fi obținută prin studiu și experiență. Dacă datele și informațiile pot fi asociate unui grup mai mic sau mai mare de oameni, cunoașterea se asociază doar unei persoane, care prin eforturi personale a ajuns să stăpânească cunoștințele. Astfel, se poate afirma că, cunoașterea este personală, subiectivă și nominală.

In 1999, K.M. Wigg menționa legat de cunoaștere că aceasta poate să existe în afara minții umane, și sugera utilizarea unor programe executabile (denumite agenți) capabile să manipuleze cunoștințele. Acesta descria cunoștințele ca fiind adevăruri și credințe, perspective și concepte, judecăți și așteptări, metodologi și abilități care puteau fi atribuite oamenilor și agenților.

Analizând aspectele prezentate în paragrafele anterioare, se poate afirma că mintea umană este cea mai importantă sursă de cunoaștere. Alte surse de cunoștințe des accesate sunt rezultatele auditurilor, rapoarte media, studii de caz și propia experiență.

In continuare se prezintă diferite clase și categorii de cunoștințe.

3.2 Tipuri de cunoștințe

Cunoștințele din orice domeniu de activitate se împart în două mari clase:

- cunoștințe publice – acestea cuprind definiții, teorii și metodologii publicate în jurnale, articole de cercetare, cărți etc. Cunoștințele publice sunt accesibile publicului larg și sunt ușor de accesat.
- cunoștințe private – acestea sunt deținute de expertii umani, care se caracterizează prin reguli personale obținute în timp, denumite și cunoștințe euristice. Cunoștințele private nu pot fi accesate de publicul larg, și doar în cercuri închise.

Cunoștințele pot să fie atașate unui domeniu de activitate, ele cuprinzând informații generale sau mai aprofundate despre domeniul de activitate.

Cunoștințele pot fi euristice atunci când fac referință la informații obținute prin experință și practică. In rezolvarea problemelor, cunoștințele euristice se referă la găsirea unor soluții folosind reguli obținute prin experimentarea inteligentă a diferitelor metode, ci nu prin aplicarea unor reguli clare, sau a unor modele matematice bine definite. Un exemplu de regulă euristică este ”regula mâinii drepte” a lui John Ambrose Fleming inventată în secolul al XIX-lea.

Metacunoștințele sunt definite drept cunoștințe despre cunoștințe. Cu alte cuvinte, se referă la conștientizarea și înțelegerea modului în care cunoștințele sunt structurate, utilizate, clasificate, verificate sau transferate. În loc să se ocupe de fapte sau date directe, metacunoașterea se concentrează pe caracteristicile, domeniul de aplicare, originea, fiabilitatea, relevanța și aplicarea cunoștințelor în sine. Aceasta include cunoașterea:

- Ce fel de cunoștințe sunt utilizate (factuale, procedurale, conceptuale etc.).
- De unde provin cunoștințele (date, experiență, inferență, autoritate).
- Cât de fiabile sau incerte sunt cunoștințele.
- Cum pot fi combinate sau reutilizate cunoștințele.
- Cum să gestionăm sau să organizăm cunoștințele pentru o utilizare eficientă.

O altă clasificare a cunoștințelor se poate face în funcție de modul cum sunt ele folosite în rezolvarea problemelor. In această situație, cunoștințele pot fi:

- Condiționale - acestea oferă informații care se referă la restricții, condiții sau cerințe preliminare. Un exemplu de astfel de cunoștințe este: Receptoarele de joasă tensiune pentru a funcționa conform producătorului au nevoie de tensiune în intervalul 220 - 240 V.
- De utilitate - acestea oferă informații legate de utilitatea unor obiecte, dispozitive etc. Exemplu: Sistemul de securitate se declanșază automat, dar are și un buton manual prin care se poate acționa.
- De acțiune - acestea fac referire la ceea ce urmează să se întâmple, în funcție de informația prezentă.
- De scop - acestea oferă informații legate de obiective, scop sau utilitatea finală a unei probleme.

In funcție de natura lor, cunoștințele pot fi explicite sau implicite. Cunoștințele explicite sunt ușor de obținut sau extras din diferite surse precum: cărți, media, rapoarte etc., iar apoi sunt ușor de codificat în diferite moduri. Pe de altă parte, cunoștințele implicite sunt mai greu de extras și codificat, deoarece sunt în strânsă legatură cu experiența și modul de reprezentare a informațiilor de către o persoană.

Daca se ia în considerare timpul, cunoștințele pot fi permanente, statice sau dinamice. Cunoștințele permanente se referă la informații care nu se schimbă niciodată, precum legile fizice. Cunoștințele statice, pe de altă parte, oferă informații care se schimbă foarte rar, ele fiind constante pentru o anumită perioadă de timp, precum legile sau procedurile. Iar la final, cunoștințele dinamice se caracterizează printr-o schimbare continuă, precum prețurile.

O sinteză a tipurilor de cunoștințe este ilustrată cu ajutorul diagramei din figura 3.1.

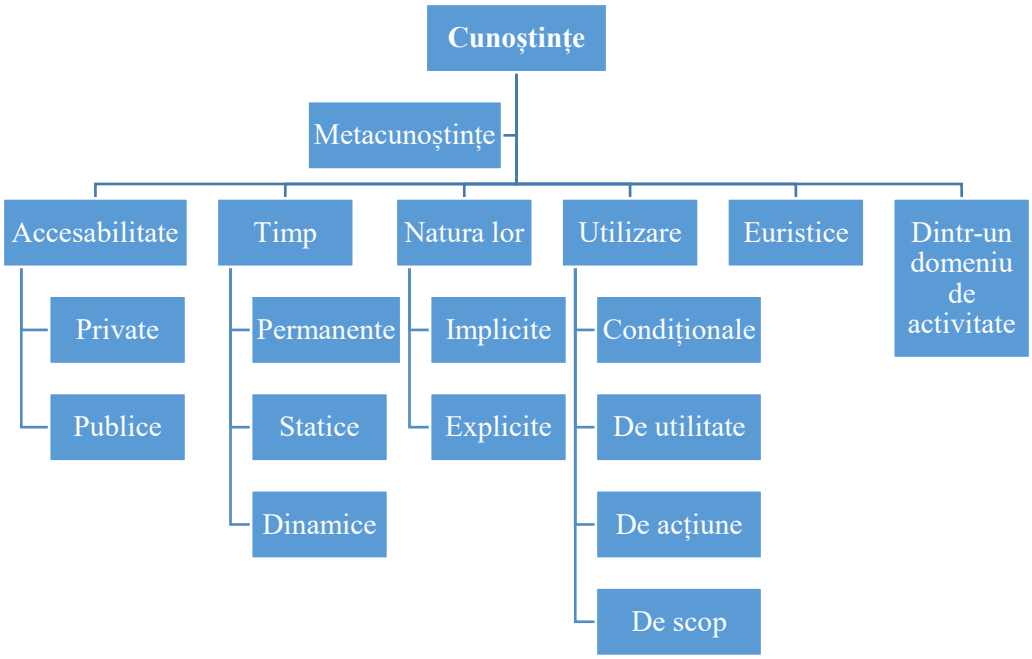


Fig. 3.1. Tipuri de cunoștințe

Indiferent de tipul cunoștințelor, acestea este de dorit să aibă mai multe caracteristici: naturalețe, transparență, modularitate, utilitate, explicitate, acuratețe și să se poată accesa ușor și eficient. Naturalețea este caracteristica care face cunoștințele să fie ușor de reprezentat în forma lor nativă, originală. Transparența unei caracteristici stocate o face pe aceasta ușor de identificate. O

cunoștință care este modulară poate fi stocată pe părți componente, astfel scade efortul de stocare, iar apoi este recuperată în întregime, oferind un grad ridicat de utilitate. Acuratețea unei cunoștințe se referă la abilitatea ei de a oferi toate informațiile corecte pentru rezolvarea unei probleme. O cunoștință explicită este ușor de reprezentat și extras.

3.3. Componentele cunoștințelor

Cunoașterea este înțeleasă în mod obișnuit ca fiind formată din două componente fundamentale, și anume componenta declarativă și componenta procedurală. Aceste componente reflectă ceea ce se știr și modul în care se folosește ceea ce se știe.

Componenta declarativă a cunoașterii este cunoscută și sub numele de cunoaștere declarativă. Ea se referă la cunoștințe descriptive care pot fi enunțate sau scrise explicit. Aceasta include fapte, concepte și afirmații care descriu lumea sau un anumit domeniu. Cunoașterea declarativă este compusă în principal din informații și afirmații factuale, care pot viza obiecte, relații, proprietăți sau evenimente. O modalitate mai precisă de a ne referi la instrumentele utilizate în reprezentarea cunoștințelor declarative este prin intermediul faptelor și regulilor. Cunoașterea declarativă este, în general, mai ușor de dobândit și de codificat. Odată colectate, acestea pot fi structurate și stocate într-o bază de cunoștințe într-o manieră formală sau semi-formală, făcându-le direct utilizabile de către sistemele bazate pe cunoștințe (de exemplu, sisteme expert sau motoare de reguli).

Componenta procedurală a cunoștințelor este cunoscută sub numele de cunoștințe procedurale. Ea reprezintă know-how-ul, adică capacitatea de a efectua sarcini sau de a aplica metode pentru a rezolva probleme. Aceasta implică o secvență de acțiuni, pași sau strategii care ghidează procesele de luare a deciziilor și de rezolvare a problemelor. Acest tip de cunoștințe este dinamic, sensibil la context și adesea dificil de articulat sau formalizat în comparație cu cunoștințele declarative. Cunoștințele procedurale includ de obicei abilități și capacități de a efectua procese (de exemplu, diagnosticarea unei defecțiuni într-un sistem energetic), metode și strategii utilizate pentru atingerea obiectivelor, și rezultate așteptate care rezultă din aplicarea unei secvențe de acțiuni. Spre deosebire de cunoștințele declarative, cunoștințele procedurale sunt mai greu de extras de la experți, deoarece acestea rezidă adesea în intuiție, experiență sau comportament antrenat. O modalitate obișnuită de reprezentare a cunoștințelor

procedurale este prin reguli euristice - reguli generale sau strategii care ghidează acțiunile în condiții de incertitudine.

Faptele sunt unități de bază de informație, care afirmă că ceva este adevărat. Ele sunt seturi de observații, simboluri sau afirmații în starea lor brută. Exemplu de fapte: Frecvența tensiunii alternative este 50 Hz. Valoarea tensiunii de fază în rețelele de joasă tensiune este 230 V.

Regulile exprimă relațiile sau conexiunile logice dintre fapte. Ele conțin condiții și acțiuni, mai exact sunt consecințele condițiilor sau acțiuni condiționate. Câteva exemple de reguli sunt: Dacă lampa nu funcționează, atunci lampa este stricată. Dacă forma de undă este nesinusoidală, atunci regimul este deformant.

Euristicele reprezintă o metodă de a arată legătura dintre date și soluții pentru o situație particulară a unei probleme. Euristicile sunt cunoștințe implicite, de cele mai multe ori găsindu-se în cunoașterea experților umani. Acestea sunt greu de caracterizat, modelat și implementat. Un exemplu unde se poate observa euristice este: Dacă tensiunea este nesinusoidală, atunci și curentul este nesinusoidal.

Figura 3.2. prezintă succint cele prezentate în paragrafele anterioare.

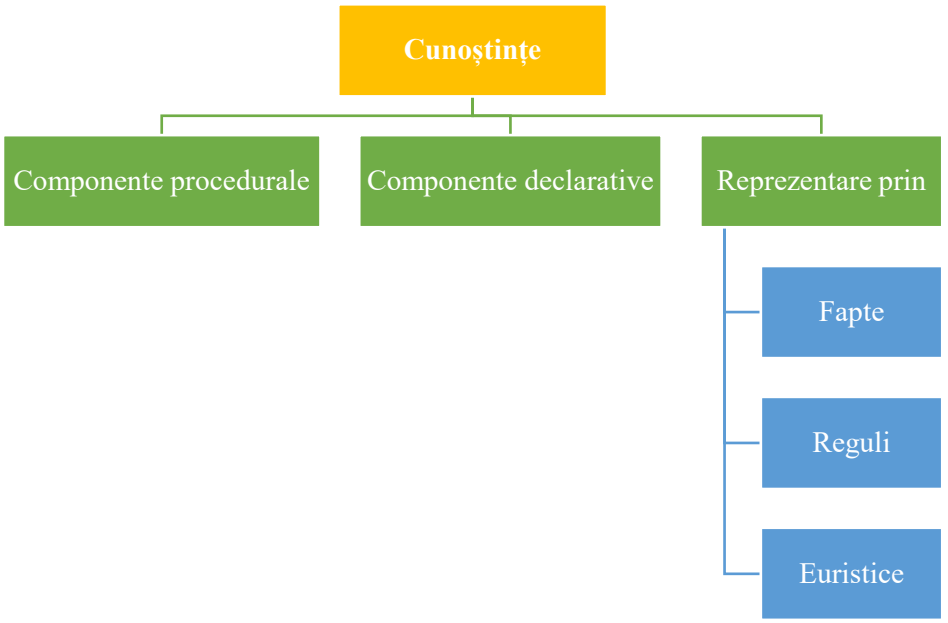


Fig. 3.2 Componentele cunoștințelor

În cadrul SE complexe, achiziţia şi codificarea cunoștințelor se face cu ajutorul unei componente specializate, numită Modulul de Achiziţie de Cunoștințe (MAC).

Modulul de Achiziţie de Cunoștințe este componenta unui SE care are rolul în achiziţia cunoştinţelor și popularea BC cu fapte sau reguli noi.

Complexitatea noilor SE determină utilizarea cunoştinţelor de la mai mulţi experţi umani. Astfel că, procesul de realizare a MAC devine tot mai complicat și presupune activităţi de colaborare dintre mai mulţi programatori (ingineri de cunoștințe). Noua situaţie, ridică o nouă provocare: achiziţia şi codificare în limbajul euristic al SE a cunoştinţelor de la toţi experţi umani, având în vedere că modul în care ei percep problema de rezolvat este diferit. Problema se acutizează dacă experţii umani sunt din domenii diferite de activitate, iar inginerul de cunoștințe trebuie să „unească" cunoştinţele într-un tot.

În dezvoltarea MAC apar bariere şi în ceea ce priveşte interacţiunea dintre experţii umani şi inginerul de cunoștințe. Transferul datelor între specialiştii care interacţionează în procesul de realizare a SE este ilustrat prin schema din figura 3.3:

- IG este cel care prin utilizarea diferitelor tehnici achiziţionează cunoștințele şi le transformă într-o formă recunoscută de calculator;
- ED, este expertul uman care prin expertiză transmite cunoștințele sale inginerului de cunoștințe;
- SE cuprinde cunoștințele din BC.

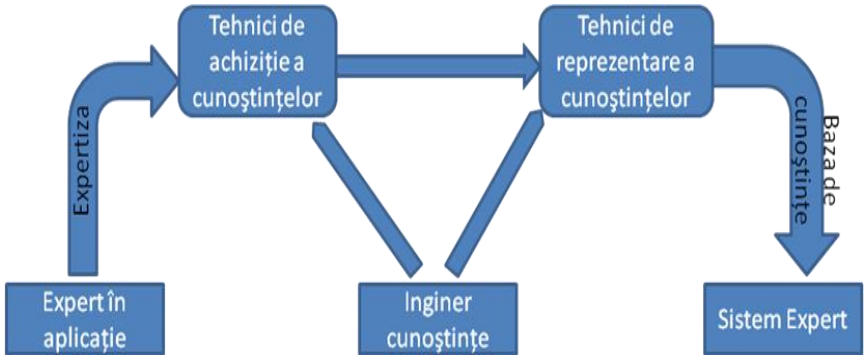


Fig. 3.3 Schema transferului de cunoștințe [2]

Expertiza din domeniu a expertului de aplicaţie este o colecţie de definiţii, relaţii, fapte specializate, euristici, proceduri, strategii şi ipoteze.

În achiziţia cunoştinţelor, respectiv dezvoltarea MAC apar trei preocupări: implicarea personalului uman corespunzător, utilizarea metodelor corespunzătoare pentru achiziţia datelor și aplicarea tehnicilor adecvate pentru codificarea cunoștințelor [1]. În sub-capitolele următoare sunt prezentate aceste aspecte, şi anume: ingineria cunoștințelor, achiziţia cunoştinţelor și modalităţi de dezvoltare a MAC al unui SE.

Inginerul de cunoştinţe

Prima etapă în achiziţia cunoștințelor atât în faza de proiectare a SE, cât și în faza de exploatare o reprezintă alegerea personalului care va lucra la proiectul SE. Persoanele care sunt necesare în etapa de proiectare și dezvoltare a unui SE sunt: experţii umani din domeniul problemei SE, programatori care au cunoştinţe și abilităţi în dezvoltarea SE şi administratori de proiect, dar nu în ultimul rând şi viitorii utilizatori ai SE. De altfel, studiile au arătat că un SE este cu atât mai acceptat cu cât utilizatorii se implică în dezvoltarea lui.

Experţii în domeniu sunt cei care au cunoștințele de specialitate legate de problema de rezolvat, cunoştinţe care se acumulează prin învăţare, experimentări şi experienţă. Cunoștințele expertului uman sunt transmise programatorilor pentru a le codifica în limbajul programului. În literatura de specialitate apare dese ori întrebarea dacă ar trebui să se apeleze la un singur expert sau la mai mulţi experţi umani. Experienţa celor care au realizat SE a arătat că avantajele utilizării mai multor experţi sunt mai multe, comparativ cu apelarea la un singur expert uman. Printre aceste avantaje se numără: bagajul de cunoștințe este mai mare, şi exclude mult posibilitatea incursiunii unor greşeli şi existând mai mulţi experţi, timpul necesar achiziţiei poate fi mai uşor distribuit.

Administratorii de proiect, sau managerii, sunt cei care oferă orientare strategică în realizarea Sistemului Expert. De asemenea, aceştia pot să îndeplinească și alte atribuţii precum: definirea scopului și a problemei iniţiale, şi pot să facă parte din rândul viitorilor utilizatori, care au o imagine clară asupra ceea ce doresc și aşteaptă de la Sistemul Expert.

Programatorii care au cunoștințe în realizarea Sistemelor Expert, poartă numele de ingineri de cunoștințe. Sub-domeniu din ştiinţa calculatoarelor cu care se ocupă aceştia este definită ca ingineria cunoaşterii.

Inginerul de cunoștințe are următoarele atribuţii [1]:

- Identificare – definirea corectă a problemei şi determinarea caracteristicilor sale;
- Conceptualizare – determinarea conceptelor care să sprijine reprezentarea cunoștințelor;

- Formalizare – alegerea unor metode de reprezentare a cunoştinţelor și a mecanismului de inferenţe;
- Implementare – reprezentarea propriu-zisă a cunoștințelor în formalismul ales (reguli, reţele semantice, cadre, obiecte etc.) și codificarea algoritmilor;
- Testare – verificarea cunoștințelor și validarea sistemului.

În practică, sunt situaţii în care un inginer de cunoștințe este şi managerul proiectului, el coordonând activitatea altor ingineri de cunoștințe care se ocupă cu achiziţia cunoștințelor și dezvoltarea SE. Pe de altă parte, inginerul de cunoștințe poate să facă parte din organizaţia care a comandat SE, adică viitori utilizatori.

3.4. Achiziţia cunoștințelor

Achiziţia cunoştinţelor este procesul prin care datele de la diferite surse, de obicei experţi umani, sunt extrase, structurate și organizate într-o formă accesibilă calculatorului. Aceste date, programul le transformă ulterior într-o formă specială folosind limbajul intern pentru a le putea gestiona cu scopul de a soluţiona problema.

Achiziţia cunoştinţelor de la experţii umani este un proces iterativ şi eterogen, care de cele mai multe ori se realizează prin purtarea unui dialog între inginerul de cunoștințe și experţilor umani, mai exact chestionarea experţilor umani legat de [1]:

- Experienţa personală prin realizarea unor probleme din trecut;
- Metode personale şi/sau tipuri de expertize pentru problema studiată;
- Cunoaşterea personală a motivelor care permite alegerea unor metode sau a unei expertize pentru o problemă dată.

Exemplu

Un Sistem Expert dedicat mentenanţei preventive a transformatoarelor de putere. Întrebări posibile ce pot fi utilizate pentru a obţine informaţii de la expertul uman sunt:

Ce zgomot se aude când transformatorul nu funcţionează corect ?

Ce miros se simte dacă transformatorul a funcţionat de mult ?

Care este valoarea concentraţiei de H_2 din ulei peste care este afectată funcţionarea corectă a acestuia ?

Ce substanţă prezentă în uleiul de răcire indică apariţia unui defect ?

Chestionarea experţilor umani în vederea obţinerii informaţiilor se poate face prin mai multe metode: interviu, interviu direcţionat, observaţii etc. Aceste metode sunt enumerate şi prezentate în tabelul 3.1.

Tabelul 3.1 Metode de achiziţie a cunoştinţelor de la experţii umani [1]

Metodă	Caracterizare	Observaţie
Interviu	· Este cea mai simplă şi utilizată metodă de chestionare · Se desfăşoară prin întrebări directe adresate expertului de către inginerul de cunoștințe · Întrebările pot fi scrise de la început, dacă inginerul de cunoștințe cunoaşte problema, sau elaborate pe parcurs	· Exemplu: Cum aţi rezolva problema X ? · Dezavantajele sunt legate de psihologia umană şi de modul în care expertul uman îşi structurează informaţiile: expertul este plictisit să explice procesul de soluţionare și omite amănunte care pot fi importante, foloseşte un limbaj de specialitate (jargou din domeniu), în descrierea procesului uită anumite amănunte etc. · Este o metodă care necesită multă răbdare şi timp · Pentru a se nu pierde informaţii utile, conversaţia cu expertul uman poate fi înregistrată
Interviu direcţionat	· Este o metodă de chestionare mai eficientă decât interviul simplu, deoarece acesta este direcţionat înspre un scop · Timpul de obţinere a informaţiilor este mai mic, iar introducerea subiectivismului în răspunsuri este mult diminuat	· Chestionarele pot fi foarte bine realizate, și în acest fel să oblige experţii să îşi organizeze cunoștințele şi să dea răspunsuri mai sistematice · Pot fi intervievaţi mai mulţi experţi o dată

	· Interviul se poate desfăşura prin intermediul unui chestionar care este completat de experţi, sau prin întrebări puse de inginerul de cunoștințe după un model clar stabilit	
Observare	· Este o metodă mai practică, în sensul că se desfăşoară prin observarea expertului uman la locul său de muncă în timp ce rezolvă problema, fără a interveni asupra activităţii sale · Informaţiile pot fi obţinute prin înregistrarea video a expertului şi apoi urmărirea acesteia pentru extragerea acţiunilor sale, respectiv de către inginerul de cunoștințe care îl urmăreşte pe expert și ia notiţe	· Avantajul în această metodă este în primul rând legată de expertul uman, care este în mediul lui, și deci nu intervine stresul sau alte aspecte legate de realizarea unui interviu · Inginerul de cunoștințe poate să observe cu uşurinţă complexitatea problemei şi etapele rezolvării ei · Dezavantajul este că nu se pot obţine informaţii legate de raţionamentul urmărit de expertul uman, având în vedere că această metodă nu presupune deranjarea expertului
Analiza de protocol	· Este denumită și metoda „gândirii cu vocea tare” (thinking aloud) · Expertului uman i se prezintă nişte situaţii particulare, acesta trebuie să prezinte modul în care va acţiona, va rezolva aceste sub-probleme · Expertul este înregistrat, iar pe baza acestor înregistrări se dezvoltă protocolul de	· Această metodă nu este corespunzătoare pentru toate tipurile de probleme · Este o metodă contestată de unii specialişti care amintesc excepţiile care pot să apară în practică

	rezolvare a problemelor specifice · Prin intermediul protocoalelor dezvoltate se urmăreşte obţinerea modului în care expertul uman raţionează rezolvarea problemei şi tipul cunoştinţelor folosite	
Analiza de grid a repertoriului	· Iniţiatorul ei este George Kelly în 1955, şi presupune mai multe etape prin care se urmăreşte clădirea modelului mental al expertului în rezolvarea problemei · Se începe cu un interviu, în care se cere expertului să identifice trei obiecte din domeniul de expertiză, și apoi atributele comune și diferite. Procesul continuă până când au fost identificate toate obiectele adiacente domeniului său de expertiză	· Oferă o imagine asupra modelului mental al expertului, incluzând şi subiectivitatea acestuia · Dezavantajul apare, când sunt multe obiecte definite de expert, ceea ce poate să ducă la confuzie având în vedere numărul mare de comparaţii ce trebuie făcute · Pentru unele obiecte este greu să se găsească caracteristicile comune sau diferite · Este o metodă care poate necesita mult timp

Complexitatea noilor SE determină implicarea unei echipe de ingineri de cunoştinţe, care lucrează pe baza unui proiect a SE. De asemenea, aceasta presupune chestionarea mai multor experţi umani din domeniu, de la care trebuie extrase cunoștințele. În această situaţie, achiziţia cunoştinţelor se face prin activităţi de colaborare între membrii echipei proiectului. Printre aceste activităţi se numără [1]:

- Brainstorming – un grup de ingineri de cunoştinţe și experţi umani care discută cu scopul de a descoperii noi idei şi metode de implementare a acestora;
- Nominal group technique – implică un grup mic de experţi care discută probleme contradictorii şi păreri privind rezolvarea acestora. Este o metodă eficientă pentru a identifica problemele, explorarea soluţiilor şi stabilirea priorităţilor;
- The Delphi technique – se folosesc mai multe chestionare date experţilor umani, pentru a le cumula cunoștințele, modul de raţionament și opiniile. Rezultatele chestionarelor sunt făcute cunoscute tuturor experţilor implicaţi;
- Focus group interviews – este o metodă folosită în cercetările din domeniul vânzărilor. Se aplică prin cehstionarea intensă a unui grup de consumatori cu scopul de identifica şi descrie caracteristicile unui produs;
- The voting technique – presupune dialogul cu mai mulţi experţi din domeniu, prezentarea acestora o problemă cu mai multe soluţii, experţii trebuind să voteze una dintre soluţiile prezentate;
- Group repertory grid analysis – este varianta în grup a analizei de grup a repertoriului, prin care se coroborează analiza de grup a repertoriului mai multor experţi, dezvoltându-se astfel un model mental a experţilor dintr-un domeniu;
- Group support systems – reprezintă calculatoarele și sistemele de comunicare prin care se facilitează dialogul dintre personalul care se implică în achiziţia cunoștințelor.

3.5. Metodologia de dezvoltare a Modulului de Achiziţie de Cunoștințe

Încă de la începuturile SE, specialiştii au observat că pentru a obţine o BC corespunzătoare, trebuie ca procesul de achiziţie a cunoştinţelor să fie la rândul lui bine gândit și structurat. Lipsa cooperării dintre inginerii de cunoștințe și experţii umani, a unui plan de lucru şi a organizării sortesc eşecului achiziţia adecvată a cunoștințelor.

Metodologia de dezvoltare a Modulului de Achiziţie de Cunoștințe, respectiv achiziţia datelor presupune parcurgerea unor etape bine stabilite: identificarea şi planificare, extragerea, analiza și verificarea cunoștințelor. În figura 3.4 este ilustrată schema de principiu a metodologiei de achiziţie a cunoştinţelor.

Etapa de identificare și planificare este primordială, deoarece acum, managerul proiectului SE decide asupra domeniului problemei, a obiectivelor, a personalului (ingineri de cunoștințe, experţi umani, utilizatori), tipul SE, metodele aplicate pentru achiziţia cunoștințelor și a achiziţie de cunoștințe propriu-zisă. Această etapă se realizează prin discuţii între membrii proiectului şi cu personalul implicat şi se finalizează cu obţinerea planului de lucru privind achiziţia cunoștințelor și a componentelor SE.

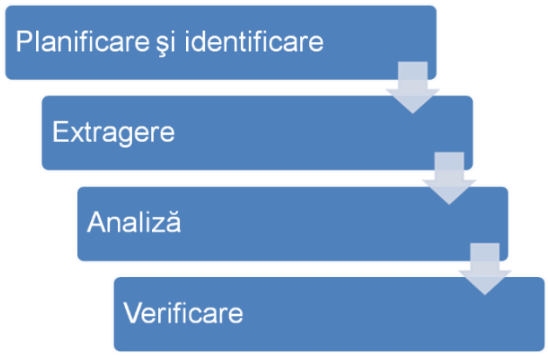


Fig. 3.4 Schema de principiu a metodologiei de achiziţie a cunoștințelor

Extragerea cunoştinţelor constă în obţinerea informaţiilor de la experţii umani printr-un număr mai mare sau mai mic de sesiuni de dialog, în care inginerii de cunoștințe vor aplica diferite metode de achiziţie a datelor (prezentate în sub-capitolul precedent). Informaţiile obţinute în această etapă pot fi memorate pe format electronic, filmate sau înregistrate, ele reprezentând datele brute pe care se vor baza inginerii de cunoștințe pentru a dezvolta BC și MAC.

Etapa de analiză a cunoștințelor presupune filtrarea informaţiilor din etapa de extragere în vederea obţinerii unor date care sunt în corespondenţă cu obiectivele și scopul proiectului. De asemenea tot în această etapă se dezvoltă o BC rudimentară, respectiv un prototip al SE care va fi utiliza în etapa de testare pentru a verifica cunoştinţele achiziţionate.

Ultima etapă în achiziţia cunoștințelor este etapa de verificare, în urma căreia se decide dacă achiziţia cunoștințelor este finalizată, sau se continuă. Alte rezultatele posibile ale testării cunoștințelor obţinute sunt:

- Reformularea problemei;
- Redefinirea conceptelor;
- Reproiectarea structurilor de cunoștințe;
- Rafinarea cunoştinţelor.

Modulul de Achiziție a Cunoștințelor este o componentă funcțională esențială în arhitectura unui sistem expert smart. Scopul său principal este de a culege, interpreta și codifica cunoștințele astfel încât acestea să poată fi stocate și utilizate de BC a sistemului. În termeni mai simpli, acest modul servește ca o punte între sursa de cunoștințe (de obicei experți umani, ingineri sau surse automate) și formatele de reprezentare a cunoștințelor utilizate de sistem.

Dezvoltarea acestui modul implică proiectarea de mecanisme și instrumente care pot extrage cunoștințe de la experți umani (prin interviuri, chestionare, dialoguri sau observații), colecta date și semnale de la dispozitive, senzori sau software implicate în procesul fizic sau domeniul monitorizat (de exemplu, senzori de temperatură, sisteme SCADA, instrumente de laborator)., interpreta, și codifica cunoștințelor.

Interpretarea și structurarea cunoștințele dobândite se poate face folosind metode formale precum - reguli (declarații if-then), ontologii, cadre, și rețele semantice.

Codificarea cunoștințele se face într-un format lizibil de mașină (de exemplu, reguli logice, date structurate, reguli de producție), astfel încât SE să poată raționa, deduce și lua decizii pe baza acestora.

În contextul unui SE smart, Modulul de Achiziție a Cunoștințelor nu este responsabil pentru raționament sau luarea deciziilor. În schimb, rolul său este axat pe datele de intrare, și anume asigură că cunoștințele relevante, corecte și utilizabile sunt introduse în sistem și devin disponibile pentru Motorul de Inferență și alte componente. Acest proces este esențial deoarece calitatea și structura cunoștințelor afectează direct performanța și fiabilitatea întregului sistem expert.

Precum s-a precizat pe scurt în paragraful precedent, Modulul de Achiziţie de Cunoștințe are rolul de a prelua datele din exterior și a le transmite SE. Datele pot fi preluate direct de la inginerul de cunoștințe, sau de la diverse aparate de măsură, senzori sau chiar utilizatori care indică starea problemei la un moment dat. În prima situaţie, inginerul de cunoștințe va folosi un limbaj cvasi-natural sau chiar limbajul recunoscut al SE pentru a modifica sau completa BC, și anume baza de reguli. În ceea ce priveşte, datele obţinute de la dispozitive externe SE sau utilizatori, acestea de obicei transmit date legate de rezolvarea unei probleme particulare, adică cunoştinţe din baza de fapte a BC.

În achiziţia cunoștințelor este nevoie de o colaborarea continuă dintre inginerul de cunoștințe și expertul în domeniu.

Cunoştinţele cuprinse în baza de reguli pot să fie reprezentate prin diferite metode, în funcţie de problema studiată, natura cunoștințelor şi complexitatea lor. Cunoștințele cuprinse în baza de reguli pot fi de trei feluri: compilate, calitative și cantitative.

Cunoştinţele compilate se referă la cele transmise de expert, sau luate din cărţi, standarde etc. Cunoștințele care se bazează pe cazuri trecute din domeniul de specialitate şi experienţă în general intră în categoria celor calitative. și în final, cunoștințele cantitative sunt cele care au la bază teorii matematice, tehnici numerice etc.

Pe de altă parte, cunoştinţele pot fi privite ca declarative (legate de proprietăţile fizice ce ţin de domeniul de specialitate a problemei de rezolvat) şi procedurale (ţin de tehnicile de rezolvare a problemei studiate) [1].

Abordarea declarativă a adoptat în scrierea programelor de aplicaţii separarea logică între toate componentele care intervin în modelele de calcul al fiecărui obiect de parametrii [1]. Din acest punct de vedere, în dezvoltarea și arhitectura SE se identifică în mod obișnuit trei niveluri de generalizare. Fiecare nivel definește modul în care sunt structurate cunoștințele și raționamentul și cât de flexibil sau specific domeniului este SE. Aceste niveluri sunt deosebit de relevante atunci când se alege sau se proiectează un SE pentru o aplicație specifică.

1. Nivelul general

La nivel general, formalismul logic și mecanismele de raționament sunt extrem de abstracte și aplicabile pe scară largă. Instrumentele și metodele dezvoltate la acest nivel pot fi reutilizate în multe tipuri de probleme și domenii, indiferent de conținutul specifiic. Scopul acestui nivel este de a oferi un cadru universal sau un sistem de suport care poate fi ulterior specializat. Caracteristic acestui nivel este reprezentarea cunoștințelor independentă de domeniu (de exemplu, logică de ordinul întâi, inferență bazată pe reguli), motoare de inferență generice și mecanisme de gestionare a cunoștințelor și arhitecturi flexibile concepute pentru a se adapta specializărilor viitoare.

Câteva exemple sunt shell-uri (cadre) sau platforme precum CLIPS, JESS sau Drools. Cadrele oferă instrumente de raționament general fără cunoștințe de domeniu încorporate.

Trebuie menționat faptul că acest nivel nu este specific problemei, ci servește drept fundament pentru construirea unor sisteme expert mai concentrate.

2. Nivelul specific

Nivelul specific de generalizare introduce cunoștințe specifice domeniului și structuri logice legate de o clasă de probleme sau un tip de obiecte. La acest nivel, SE este adaptat la un grup de aplicații conexe, dar nu la o singură sarcină. Scopul acestui nivel este rezolvarea unei categorii de probleme (de exemplu, toate defecțiunile transformatoarelor sau toate diagnosticele cardiovasculare). Caracteristicile acestui nivel sunt bazele de cunoștințe populate cu fapte și reguli legate de un domeniu definit și raționamentul care încă este reutilizabil în cadrul domeniului, dar nu în domenii fără legătură.

Un exemplu de SE care folosește un nivel de generalizare specific este un sistem expert cadru cu logică încorporată pentru sisteme de protecție electrică.

Acest nivel echilibrează generalitatea și aplicabilitatea, astfel se găsește frecvent în SE de tip suport, care sunt adaptabile într-un domeniu.

3. Nivel individual

Nivelul individual este cel mai specific nivel de generalizare. Acest nivel implică sisteme expert care sunt codificate fix sau dedicate complet rezolvării unei anumite probleme sau scenarii. Scopul acestui nivel de generalizare este rezolvarea unei singure probleme bine definite. Caracteristic SE care folosesc nivelul individual de generalizare sunt baza de cunoștințe și raționamentul construite personalizat pentru un singur obiect, sistem sau situație. Aceste sisteme expert nu pot fi reutilizate sau adaptate fără o reprogramare substanțială. Ele sunt optimizate pentru performanță și precizie într-un context restrâns definit.

Două exemple de astfel de SE sunt - un sistem de securitate construit pentru a diagnostica o singură mașină într-o fabrică, și un sistem de detectare a defecțiunilor programat pentru a monitoriza un anumit model de transformator în condiții de sarcină cunoscute.

Aceste sisteme sunt foarte eficiente pentru sarcina lor, dar le lipsește flexibilitatea și necesită refaceri abundente pentru a rezolva diferite probleme.

3.6. Intrebări și exerciții

1. Sistemele expert au o componenta care poate sa realizeze calcule matematice.
   Adevărat sau Fals

1. Achiziţia cunoştinţelor se poate realiza de către inginerul de cunoştinţe şi de alte dispozitive dedicate.

Adevărat sau Fals

2. Inginerul de cunoştinţe poate să aibă atribuţii şi manageriale în dezvoltarea proiectului Sistemului Expert.

Adevărat sau Fals

4

Baza de cunoștințe
Reprezentarea cunoștințelor

Prezentul capitol prezintă caracteristicile generale şi importanţa bazei de cunoștințe în structura SE, precum și principalele metode de reprezentare a cunoștințelor.

4.1. Aspecte generale

Baza de cunoștințe (BC) este componenta Sistemului Expert care conţine totalitatea informaţiilor din domeniul de specialitate al problemei studiate. Aceasta este prima care este dezvoltată de către inginerul de cunoștințe și comunică cu Motorul de Inferenţe, Modulul Explicativ şi Modulul de Achiziţie a Cunoştinţelor (figura 2.1).

Informaţiile cuprinse în BC sunt denumite simplu cunoștințe, iar modul de reprezentare lor în limbajul calculatorului este o ramură intens studiată a Inteligenţei Artificiale. Corespondenţele cunoştinţelor din lumea reală în limbajul Sistemelor Expert se fac în funcţie de natura și complexitatea acestora.

Baza de Cunoștințe interacţionează cu MAC bidirecţional: BC îi comunică modulului cunoștințele care le posedă și starea lor la un moment dat, iar acesta din urmă îi furnizează cunoștințele în limbajul SE legate de lumea reală.

O relaţie reciprocă are BC şi cu Modulul Explicativ. Astfel, BC îi oferă acestuia cunoștințele necesare pentru a oferii utilizatorului informaţiile cerute, iar Modulul Explicativ la rândul lui poate să acţioneze asupra BC prin intermediul instrucţiunilor date de către utilizator.

Baza de Cunoştinţe are o legătură uni-direcţională cu Motorul de Inferenţă, deoarece acesta preia informaţiile cuprinse în BC și le manipulează pentru a găsi soluţia la problema studiată, fără a acţiona asupra structurii BC. Pe de altă parte, precum se observă în figura 2.1, MI influenţează BC prin intermediul MAC, respectiv MD.

Baza de cunoștințe cuprinde informaţiile sub formă de reguli de inferență și de fapte. Totalitatea regulilor compun Baza de reguli, iar totalitatea faptelor Baza de fapte. În figura 4.1 este ilustrată structura BC, alături de un exemplu din domeniul electroenergetic.

Regulile se referă la operaţiile care pot fi efectuate asupra obiectelor conţinute în baza de date. Regulile se modifică mai rar, ca urmare, de exemplu, a unor schimbări în regulamentele tehnice. În esenţă, regulile constituie un ansamblu complet şi necontradictoriu de cunoștințe necesare rezolvării unei probleme [1].

Faptele sunt componentele BC care arată starea obiectelor la un moment dat. În exemplul următor se poate observa un exemplu de regulă şi fapt din domeniul electroenergetic.

Exemplu

Regulă:
Dacă Întreruptorul I este deschis, Atunci deschide separatorul S.

Fapt:
La momentul t întreruptorul I este închis / Întreruptorul I este închis.

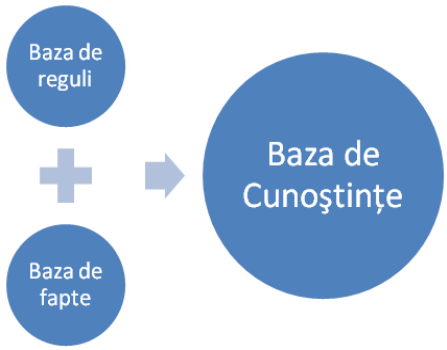


Fig. 4.1 Structura Bazei de Cunoştinţe

O bază de cunoștințe bine concepută are adesea o structură ierarhică sau modulară. Ea conține

- Informații despre domeniu (date specifice, fapte) - Acestea sunt informațiile de bază, verificabile, specifice domeniului pentru care este conceput SE. Ele formează fundamentul factual al cunoștințelor și sunt adesea statice sau direct observabile. Faptele sunt obiective și direct

măsurabile și adesea stocate sub formă de perechi atribut-valoare sau liste. Ele pot fi actualizate dinamic în timpul utilizării sistemului.

- Reguli generale (reguli de inferență) - Acestea sunt relații logice, adesea cauză-efect, codificate ca declarații DACĂ-ATUNCI sau condiție-acțiune. Acestea permit sistemului să deducă fapte noi din cele existente. Ele sunt reprezentate folosind reguli de producție sau declarații logice. Aceste reguli sunt deterministe (întotdeauna adevărate dacă este îndeplinită condiția). Mai mult, conform lor se realizează raționamentul pentru rezolvarea problemei.
- Euristică (intuiție expertă) - Acestea sunt reguli sau tehnici informale, bazate pe experiență, utilizate de experți umani atunci când nu este disponibilă o procedură algoritmică strictă. Ele reprezintă „reguli generale" și sunt mai puțin formale și pot să nu fie întotdeauna demonstrabile logic. Ele sunt specifice domeniului problemei de rezolvat.
- Definiții (cadre sau obiecte) - Aceasta se referă la cunoștințe structurate despre entități din domeniu - cum ar fi mașini, simptome sau procese - descrise prin atribute și valori. Acest lucru ajută sistemul să „înțeleagă" ce este fiecare obiect sau concept. Caracteristic acestora este faptul că se utilizează în reprezentare bazată pe cadre sau orientată pe obiecte. Astfel, sunt definite clase, subclase și proprietăți, și de asemenea permit moștenirea și clasificarea.
- Meta-reguli (reguli despre reguli) - Acestea sunt reguli care controlează sau ghidează utilizarea altor reguli. Ele ajută la alegerea, ordonarea, prioritizarea sau explicarea modului în care ar trebui aplicate regulile. Ele reglează strategia de inferență (de exemplu, înlănțuirea înainte vs. înapoi) și cresc flexibilitatea și inteligența procesului de raționament. Acestea sunt utile în sisteme complexe cu reguli conflictuale.

  Exemple

  DACĂ se aplică două reguli, ATUNCI alegeți-o pe cea cu o certitudine mai mare.

  DACĂ se potrivesc mai multe reguli, ATUNCI preferați-o pe cea care utilizează date mai recente.

Caracteristicile unui BC de bună calitate sunt corectitudinea datelor, consistența datelor (pentru a evita concluzii contradictorii), date complete, transparența datelor implementate și posibilitatea modificării facile a acestora.

4.2. Reprezentarea cunoștințelor

Reprezentarea cunoștințelor se referă la utilizarea unor simboluri recunoscute de calculator pentru a face corespondenţe cu lumea reală. Simbolurile utilizate sunt în funcţie de limbajele de programare, iar corespondenţele trebuie să asigure posibilitatea realizării unor raţionamente între simbolurile utilizate. În acest sens, aceste corespondenţe sunt caracteristici ale obiectelor, faptelor, fenomenelor şi proceselor care sunt concludente pentru problema de rezolvat, şi care le diferenţiază de alte obiecte, fenomene etc. de acelaşi tip.

Cunoștințele reale sunt caracterizate de mai multe trăsături care de multe ori pun în dificultate inginerii de cunoștințe, și anume faptul că sunt imprecise, incomplete, parţiale, și subiective. Astfel, în transpunerea cunoştinţelor de la subiecţii umani la programele de calculator trebuie avută în vedere natura cunoştinţelor reale şi selectate caracteristicile care oferă informaţiile necesare şi suficiente.

În cadrul unui program, cunoștințele sunt memorate sub forma unor piese de cunoaştere care descriu fapte, fenomene, procese şi evenimente, care alcătuiesc un model al lumii reale din domeniu analizat. Acest model este recunoscut de program și prin intermediul lui programul operează proceduri de organizare, clasificare, căutare și recunoaştere.

Reprezentarea cunoştinţelor în BC trebuie să aibă o anumită arhitectură care conţine trei niveluri [2]:

- Nivelul intern – descrie modul în care cunoștințele sunt stocate fizic în memoria sistemului. Acesta include structurile de date de nivel scăzut, indexarea, gestionarea memoriei și metodele de acces necesare sistemului pentru a recupera și manipula eficient cunoștințele. Caracteristici principale - complet ascuns utilizatorilor, se concentrează pe codificarea, compresia, legătura și viteza de recuperare a datelor, definește modul în care faptele, regulile, cadrele sau rețelele semantice sunt codificate (de exemplu, în date binare, liste, arbori, tabele hash) și optimizate pentru spațiu și performanță.
- Nivelul conceptual – definește structura abstractă, logică a bazei de cunoștințe, așa cum este văzută de dezvoltatorii de sisteme, inginerii de

cunoștințe sau experții în domeniu. Descrie ce cunoștințe există, cum sunt organizate și relațiile dintre elementele de cunoștințe, fără a expune implementarea fizică internă. Caracteristici principale - independent de detaliile de stocare, prezintă entitățile, atributele, relațiile și constrângerile acestora, oferă o vedere unificată a bazei de cunoștințe și susține logica de inferență, cum ar fi înlănțuirea regulilor, ontologiile, cadrele sau rețelele semantice.

- Nivelul extern – definește vizualizări sau scheme specifice utilizatorului ale bazei de cunoștințe. Fiecare utilizator sau grup accesează doar un subset relevant al bazei de cunoștințe, adaptat sarcinilor, rolurilor sau privilegiilor sale. Acest nivel acceptă interacțiunea personalizată cu sistemul de cunoștințe, îmbunătățind utilizabilitatea. Caracteristici principale - pot exista simultan mai multe vizualizări externe, vizualizările sunt filtrate, simplificate sau personalizate, facilitează interfețele specifice domeniului, interfețele grafice sau sistemele de interogare și protejează cunoștințele sensibile sau irelevante de accesarea de către utilizatori neautorizați.

Conceptul de bază în reprezentarea cunoştinţelor este cel de entitate. ***O entitate*** este un obiect al lumii reale care are anumite trăsături (***atribute***) care îl diferenţiază de alte entităţi, sau îl caracterizează ca o entitate a unui grup de entităţi.

În reprezentarea cunoștințelor se disting trei clase de metode [2]:

- Metode logice – privesc cunoștințele şi relaţiile dintre ele ca o serie de enunţuri adevărate. În cazul acestor metode sunt folosite reguli aplicate direct asupra componentelor BC. Dezavantajul acestor metode constă în soluţii nesatisfăcătoare de sistematizare a BC prin necorespondenţa reprezentării cunoaşterii despre acţiuni, precum și a reprezentării regulilor euristice;
- Metode relaţionale (structurate) – sunt metodele în care cunoştinţele sunt reprezentate mai mult prin relaţiile dintre ele sub forma de reţele și graf-uri. Aceste metode permit organizarea cunoștințelor în funcţie de omogenitatea acestora şi conduc la clase și sorturi. Metodele relaţionale reprezintă relaţiile dintre entităţi ca pe nişte elemente ale cunoaşterii, spre deosebire de cele logice în care relaţiile dintre entităţi se deduc;

- Metode procedurale – cunoștințele sunt reprezentate sub formă de proceduri ce permit obţinerea stărilor la momentele specifice pornind de la stările iniţiale sau intermediare.

În domeniul electroenergetic reprezentarea cunoștințelor se poate face prin utilizarea următoarelor metode: reguli de producţie, reţele semantice, cadre Minski, logica formală, programarea orientată pe obiecte, ontologii, logică fuzzy, euristică, reprezentare bazată pe constrângeri, arbori de decizie, rețele bayesiene, raționament bazat pe cazuri, scripturi, rețele petri, grafuri și modele topologice, meta-reguli, și modele relaționale / baze de date. În continuare sunt descrise metodele menţionate prin intermediul caracteristicilor generale, precum şi a avantajelor şi dezavantajelor.

4.3. Reguli de producţie

Reprezentarea cunoștințelor sub forma regulilor de producţie este una dintre cele mai utilizate metode de reprezentare a cunoştinţelor. Acest fapt a apărut datorită faptului că mulţi experţi memorează informaţiile și experienţele sub forma reguli personalizate. În consecinţă, această metodă este apropiată de modul de memorare şi raţionare uman.

Structura unei reguli este de forma:

Partea de condiţie → Partea de acţiune.

Această sintaxă poate fi interpretată prin expresia:

DACĂ ***partea de condiţie*** este îndeplinită**,**
ATUNCI ***partea de acţiune*** se execută**.**

Exemplu
DACĂ întreruptorul I este deschis,
ATUNCI deschide separatorul S.

Mecanismul regulilor de producţie a fost împrumutat din teoria limbajelor formale, şi anume logica propoziţiilor.

Partea de condiţie a unei reguli poartă numele şi de premisă, care trebuie îndeplinită pentru ca acţiunea să aibă loc. Partea de acţiune poate să corespundă declanşării unei acţiuni sau a unei noi reguli.

În cadrul BC, regulile de producţie sunt grupate sub forma unor ***sisteme de producţie***. Sistemele de producţie (figura 4.2) sunt metode de natură procedurală în care cunoștințele sunt de trei categorii [1]:

- Cunoștințe declarative – cunoștințele sunt grupate sub forma unor structuri de date memorate într-o colecţie numită context;
- Cunoștințe procedurale – cunoştinţele sunt grupate sub forma unor perechi de tipul condiţie – acţiune, numite reguli de producţie, care formează baza de reguli;
- Cunoştinţe strategice sau de control – cunoştinţele sunt reprezentate sub forma unor reguli care sprijină decizia asupra secvenţei de acţiuni în procesul de soluţionare a problemei analizate.

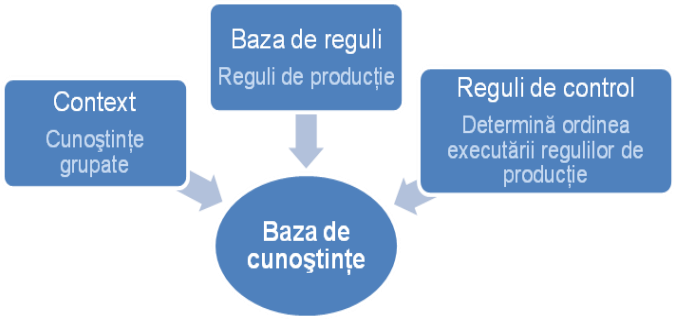


Fig. 4.2 Structura BC prin utilizarea sistemelor de producţie

Utilizarea sistemelor de producţie în reprezentarea cunoștințelor oferă următoarele avantaje [1]:

- Modularitatea proprie fiecărei reguli – orice regulă poate fi privită ca o entitate independentă de celelalte. Îndeplinirea premisei determină activarea ei. Dacă o regulă nu are legătură cu o altă regulă (nu este condiţionată de activarea altei reguli) atunci, o adăugare, ştergere sau modificare nu implică modificări asupra celorlalte reguli;
- Modularitate în realizarea formalismului de rezolvare a problemei – toate regulile pot fi văzute ca elemente ale unui ansamblu, care se combină și se adună pentru a da un răspuns la problema studiată. Ordinea în care regulile sunt introduse în Baza de Cunoștințe nu este importantă;
- Caracter natural de exprimare – studiile au arătat că experţii umani de cele mai multe ori își formulează cunoștințele într-o manieră asemănătoare regulilor;

- Accesibilitatea bazei de reguli – depinde de modul în care au fost structurii date bazei de reguli. Aceasta facilitează mentenanţa şi manipularea cunoştinţelor.

Regulile de producție sunt de mai multe tipuri, și anume reguli de diagnosticare (Exemplu - DACĂ temperatura uleiului de transformator > 90°C ATUNCI raportează supraîncălzirea), reguli de control (Exemplu - DACĂ cădere de tensiune > limită ATUNCI crește puterea reactivă), reguli de planificare (Exemplu - DACĂ întrerupere de tensiune ATUNCI inițiază secvența de detașare), reguli de inferență (Exemplu - DACĂ condiția A ȘI condiția B ATUNCI concluzionează rezultatul C) și reguli euristice (Exemplu - DACĂ problema este necunoscută ȘI simptomele ≈ X ATUNCI încearcă soluția Y). Aceste aspecte sunt sintetizate în figura 4.3.

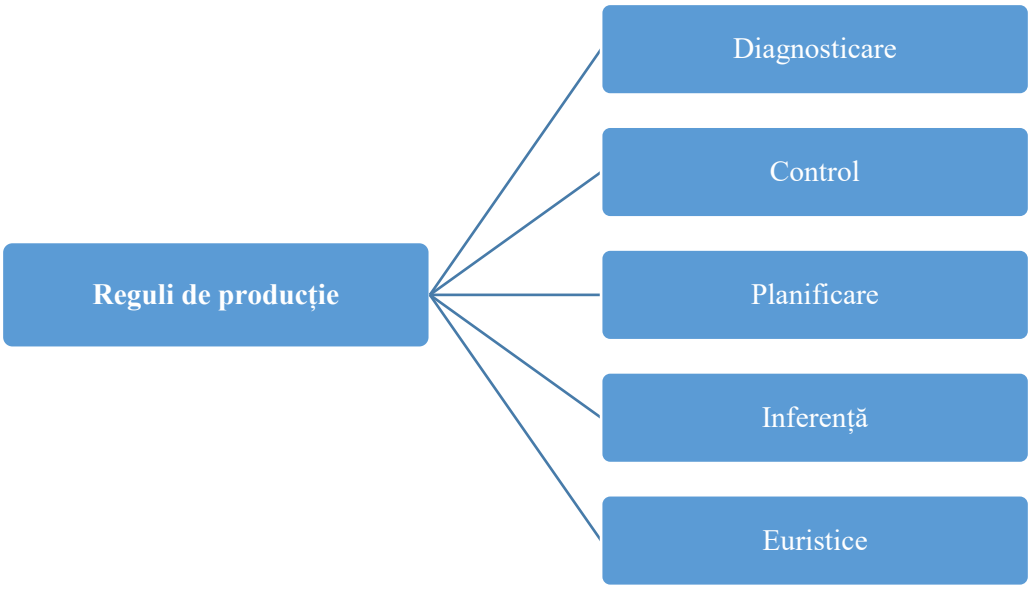


Fig. 4.3. Tipuri de reguli de producție

Motorul de inferență care lucrează în SE a căror BC folosesc regulile de producție utilizează strategia de control înainte (înlănțuirea înainte) sau strategia de control înapoi (înlănțuirea înapoi).

Instrumente și limbaje care suportă regulile de producție sunt:

- CLIPS — shell de sistem expert utilizat pe scară largă, bazat pe reguli de producție.

- Jess — motor de reguli pentru Java, sistem de reguli de producție.
- Drools — Sistem de gestionare a regulilor de business în Java.
- Prolog (cu unele adaptări) — programare logică, deși mai declarativă decât regulile de producție.

Trebuie reţinut faptul că modul în care sunt introduse cunoștințele în baza de reguli depinde de limbajul de programare utilizat. Aşa precum se va vedea la sfârşitul subcapitolului în cadrul Activităţii 1, în care se va prezenta o comparaţie între limbajul LISP şi PROLOG.

Dezavantajele reprezentării prin reguli de producţie sunt [1]:

- Imposibilitatea de a prevedea o desfăşurare optimă pentru o secvenţă de acţiuni. Datorită modularităţii regulilor, testarea tuturor regulilor și a variantelor de parcurgere a acestora pentru a soluţia optim problema studiată ar necesita mult timp;
- În situaţiile în care se imprimă o anumită ordine de parcurgere a regulilor influenţează o anumită concluzie, soluţie care nu este optima.

Pentru a elimina aceste dezavantaje, în unele situaţii se folosesc reţele de producţie, în care se dă o anumită ordine și ierarhie a regulilor. De asemenea, regulile din baza de reguli pot să fie aşezate sub forma unor graf-uri; prin aceasta eliminându-se din modularitate, dar facilitează găsirea unui optim.

4.4 Logica formală

Logica formală sau matematică este metodă de reprezentare a cunoştinţelor preferată de mulţi specialişti deoarece este lipsită de ambiguităţi. Caracteristica principală a unei BC descrisă prin logica formală este faptul că este compusă doar din formule și raţionamente logice care descriu universul problemei studiate. Astfel, BC nu are o organizare proprie, ele nu există decât prin intermediul proprietăţilor lor. Avantajul utilizării acestei metode este posibilitatea de a obţine rezultate noi folosind regulile de inferenţă: modus prones, rezoluţie, modus tollens etc. În tabelul 4.1 sunt prezentate raţionamentele cu care lucrează logica formală, precum și semnificaţiile lor. Unul dintre dezavantajele modelării cunoștințelor folosind logica este că apare dificultatea în găsirea corespondenţelor adecvate lumii reale, care să fie recunoscute de calculator prin intermediul formulelor.

Logica formală cuprinde trei ramuri [1]:

- Logica (calculul) propoziţiilor – studiază legăturile dintre propoziţii fără a ţine seama de structura lor internă;

- Logica predicatelor (propoziţiilor de ordinul întâi) – studiază legăturile dintre afirmaţiile relative la una sau mai multe variabile dintr-un domeniu de definiţie. Afirmaţiile au proprietatea că sunt adevărate numai pentru valori specifice ale variabilelor, aceste afirmaţii purtând numele de predicate;
- Logica cu mai multe grade de adevăr – este o dezvoltare a logicii predicatelor.

Logica propoziţiilor este un limbaj formal care utilizează propoziţii, care pot să ia valorile de adevărat = 1 sau fals = 0. Propoziţiile sunt construite cu ajutorul unui alfabet, reguli de sintaxă, axiome și o regulă de deducţie. Operaţiile dintre propoziţii sunt realizate prin utilizarea unor operatori logici: negaţia, conjuncţia, disjuncţia, disjuncţia exclusivă, implicaţia și echivalenţa. În tabelul 4.1 sunt prezentaţi operatorii logici și modul lor de operare.

Tabelul 4.1. Operatorii logici folosiţi în logica propoziţiilor

Operator logic	**Simbol**	**Reprezentare**	**Semnificaţie**
Negaţie	$\neg$	$\neg P$	**P** | **$\neg$ P** A | F F | A
Conjuncţie	$\wedge$	$P \wedge Q$	**P** | **Q** | **$P \wedge Q$** A | A | A A | F | F F | A | F F | F | F
Disjuncţie	$\vee$	$P \vee Q$	**P** | **Q** | **$P \vee Q$** A | A | A A | F | A F | A | A F | F | F
Disjuncţie exclusivă	$\forall$	$P \forall Q$	**P** | **Q** | **$P \forall Q$** A | A | F A | F | A F | A | A

			F F F
Implicaţie	→	P→Q	**P** **Q** **P→Q** A A A A F F F A A F F A
Echivalenţă	↔	P↔Q	**P** **Q** **P↔Q** A A A A F F F A F F F A

P, Q – propoziţie, A – adevărat, F - fals

Logica propoziţiilor de ordinul întâi (logica predicatelor) este mai complexă decât logica propoziţiilor, deoarece urmăreşte eliminarea dezavantajului introdus de aceasta din urmă, şi anume incapacitatea de a lucra cu unele ansambluri de elemente. Noţiunea de bază a acestui limbaj formal este predicatul, o afirmaţie relativă la una sau mai multe variabile care are proprietatea că pentru valori specifice ale variabilelor este adevărată sau falsă [1].

Combinarea predicatelor în cadrul propoziţiilor duce la crearea de noi propoziţii. Astfel, acestea pot fi propoziţii: particulare, particulare universalizate, particulare existenţiale, universale, existenţiale, existenţiale universalizate şi universale existenţializate. În tabelul 4.2 sunt descrise raţionamentele pe care se bazează logica propoziţiilor şi logica predicatelor.

Tabelul 4.2. Raţionamente folosite în logica propoziţiilor

Raţionament	**Reprezentare**	**Observaţii**
Modus ponens	Dacă P Şi P→Q Atunci Q	
Modus tollens	Dacă P→Q	

	Şi $\neg Q$ Atunci $\neg P$	
Reducere la absurd	Dacă $\neg P \rightarrow Q$ Şi $\neg P \rightarrow \neg Q$ Atunci P	
Adjuncţie	Dacă P Şi Q Atunci $P \wedge Q$	
Tranzitivitatea implicaţie	Dacă $P \rightarrow Q$ Şi $Q \rightarrow R$ Atunci $P \rightarrow R$	P, Q, şi R sunt propoziţii
Trecerea de la echivalenţă la implicaţie	Dacă $P \leftrightarrow Q$ Atunci $P \rightarrow Q$ Şi $Q \rightarrow P$	
Trecerea de la două implicaţii reciproce la echivalenţă	Dacă $P \rightarrow Q$ Şi $Q \rightarrow P$ Atunci $P \leftrightarrow Q$	

Logica predicatelor, la care acestea pot să ia mai multe valori de adevăr constituie o extensie a acesteia, care permite tratarea inexactităţilor prezente în cunoștințele utilizate de experţii umani dintr-un domeniu.

Avantajele logicii, claritate şi rigurozitate, de multe ori sunt și punctele slabe, deoarece gândirea umană nu este foarte riguroasă, raţionamentele fiind mai suple.

Logica formală este o metodă de reprezentare a cunoștințelor care nu poate fi folosită în cazul unor cantităţi mari de informaţii, deoarece lipsa de organizare a BC, face ca pe măsură ce îi cresc dimensiunile, să apară probleme în ceea ce priveşte coerenţa şi utilizarea ei. Pentru a se elimina aceste puncte slabe, au fost dezvoltate logici neclasice, numite și logici ne-monotone: logici multivalente, logici de incertitudine și logici modale.

Într-o bază de cunoștințe care utilizează logica formală, cunoștințele sunt codificate ca:

- Fapte (axiome): Afirmații care sunt întotdeauna adevărate
  De exemplu: Generator(G1) sau VoltageAt(Bus1, 400)
- Reguli (scheme de inferență): Implicații generalizate
  De exemplu: ∀x ∀y (Suprasarcină(x) ∧ Conectat(x, y) → Alarmă(y))

- Obiective (interogări): Afirmații care trebuie dovedite sau la care trebuie să răspundă motorul de inferență
  De exemplu: ∃x (Deschis(x) ∧ DefectDetectat(x))

Limbajele dedicate care suportă folosirea logicii formale în dezvoltarea BC sunt Prolog, CLIPS (cu module logice), și OWL (Web Ontology Language).

4.5. Reţele semantice

Reţelele semantice sunt o metodă a reprezentare a cunoştinţelor superioară reţelelor de producţie, și au apărut ca o consecinţă a modului de surprindere a structurilor relaţionale de mare complexitate. Acestea apar sub forma unor graf-uri complexe alcătuite din noduri care sunt legate prin arce, care arată relaţiile dintre noduri. În figura 4.4 este descris modul în care se reprezintă grafic o reţea semantică alcătuită din două noduri și arcul ce le leagă. Simbolurile asociate nodurilor şi arcelor poartă numele de etichete, iar arcul fiind reprezentat printr-o săgeată se numeşte arc orientat, în consecinţă o reţea semantică este un graf etichetat și orientat. Trebuie precizat faptul că etichetele nodurilor sunt unice pentru fiecare nod, în comparaţie cu etichetele arcelor care pot să fie asociate la mai multe arce care leagă noduri diferite. De asemenea, două noduri pot fi legate de un singur arc.

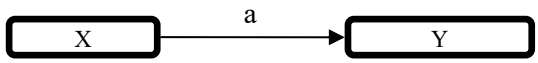


Fig. 4.4 Reprezentarea grafică a unei reţele semantice

Nodurile sunt folosite pentru reprezentarea de obiecte, concepte, atribute, evenimente și stări. Arcele sunt cele care arată relaţiile între noduri, şi pot fi de mai multe feluri [2]:

- Relaţii epistemice – descriu semnificaţia unui concept cu ajutorul unor concepte mai puţin generale;
- Relaţii compoziţionale – fac legătura între concepte și componentele lor;
- Relaţii de apartenenţă – descriu legătura de apartenenţă a unui nod la diferite clase sau mulţimi;
- Relaţii cauzale – fac legătura între noduri de tip obiect şi noduri de tip acţiune;
- Relaţii modale – sunt între componente și modalităţile de ataşare a acestora la concepte.

Reţelele semantice pot să fie organizate ierarhic, ceea ce le permite ca unele atribute să fie moştenite de la tipuri generale la tipuri specializate [1]. Relaţiile ierarhice se fac prin intermediul unor arce de tip „ESTE” (ISA, is a) care arată apartenenţa unui obiect la o clasă, respectiv a unor arce de tip „Un fel de” (AKO, a kind of) care arată incluziunea unei submulţimi la o mulţime.

Integrarea rețelelor semantice în BC a unui sistem expert se poate face alături de reguli de producție - pentru controlul operațional și declanșarea acțiunilor, Cadre - pentru proprietăți detaliate ale componentelor, și logică fuzzy sau modele numerice - pentru toleranță sau incertitudine. Rețeaua semantică poate acționa ca hartă centrală pentru navigarea în cunoștințe și interogarea relațiilor în timpul raționamentului. Aceste se explică astfel - rețeaua semantică servește ca o hartă a relațiilor, un ghid pentru parcurgerea logică, o bază pentru interogare și un instrument de suport pentru raționamentul dinamic. În acest fel, nu este doar o stocare de date - este o structură activă de cunoștințe, permițând sistemului expert să „înțeleagă” și să ia decizii cu privire la sistemul energetic pe care îl supraveghează.

Avantajele utilizării rețelelor semantice în reprezentarea cunoștințelor sunt următoarele:

- Reprezentare clară a relațiilor - rețelele semantice conectează vizual și logic entitățile folosind legături etichetate. Beneficiu - modelare mai ușoară a configurațiilor din lumea reală ale sistemelor electrice.
- Suportă raționamentul intuitiv - structura imită modelele cognitive umane pentru înțelegerea sistemelor complexe. Beneficiu - facilitează inteligența artificială explicabilă - raționamentul sistemului este ușor de urmărit și justificat.
- Interogare și navigare eficiente - nodurile și muchiile permit trecerea rapidă de la un concept la cele conexe. Beneficiu - permite diagnosticarea în timp real, analiza impactului sau localizarea defecțiunilor.
- Modular și scalabil - pot fi adăugate noduri și legături noi fără a perturba structura existentă. Beneficiu - asigură flexibilitatea și mentenabilitatea bazei de cunoștințe în timp.
- Suportă ierarhii și moștenire - nodurile pot fi structurate în clase și subclase. Beneficiu - promovează reprezentarea compactă a cunoștințelor și evită redundanța.

- Util pentru diagnosticarea defecțiunilor și raționamentul cauzal - relațiile pot fi parcurse înapoi pentru a deduce cauza sau înainte pentru a prezice efectul. Beneficiu - sprijină depanarea precisă și rapidă în medii cu miză mare, cum ar fi substațiile sau centrele de control.
- Se integrează bine cu algoritmii bazați pe grafuri - mulți algoritmi (de exemplu, calea cea mai scurtă, analiza dependențelor, extinderea arborelui de defecțiuni) pot opera pe rețele semantice. Beneficiu - permite un raționament computațional mai profund fără a remodela cunoștințele.
- Facilitează reutilizarea și partajarea cunoștințelor - deoarece nodurile reprezintă entități și concepte din lumea reală, rețeaua poate fi reutilizată în diferite SE. Beneficiu - îmbunătățește interoperabilitatea și reduce costurile inginerești.
- Bun pentru documentație și instruire - structura vizuală și logică a unei rețele semantice poate fi utilizată pentru documentația educațională și inginerească. Beneficiu: Ajută la reducerea decalajului dintre sistemele expert și experții umani.
- Permite integrarea cu ontologii și web-ul semantic - rețelele semantice sunt fundamentale pentru ontologii (de exemplu, OWL), permițând schimbul de cunoștințe bazat pe web. Beneficiu - sprijină automatizarea de generație următoare, diagnosticarea inteligentă și interoperabilitatea.

Limitările care apar în folosirea rețelelor semantice în reprezentarea cunoștințelor derivă din faptul că sunt slabe în reprezentarea cunoștințelor procedurale complexe (mai potrivit pentru fapte declarative) și probleme de scalabilitate în sisteme mari, fără structurare ierarhică.

4.6. Cadre Minsky

Cadrele (frames sau frameworks) au fost introduse de către Marvin Minsky, ele fiind o metodă avansată de reprezentare a cunoştinţelor care reunesc metodele de reprezentare procedurale, sistemele de producţie și reţelele semantice. În literatura de specialitate, cadrele sunt folosite și sub denumirea de frame-uri (structuri împachetate) [2].

Un cadru cuprinde trei tipuri de componente: obiectul căruia îi este asignat un nume prin intermediul identificatorului, forma (atributul) – arată o categorie

de caracteristici și faţeta care descrie obiectul cu ajutorul unor perechi de simbol – valoare. Utilizând un formalism meta-lingvistic, un cadru poate fi prezentat astfel [3]:

```
<cadru> =                    (<identificator_cadru>
<descriere_formă>)
<descriere_formă> =          ((<identificator_formă>
<descriere_faţetă>),
        .....
(<identificator_formă> <descriere_faţetă>)).
<descriere_faţetă> =         ((<identificator_faţetă>
<descriere_dată>),
        .....
(<identificator_faţetă> <descriere_dată>)).
<descriere_dată> =           (<valoare>)
(<descriere_date> <valoare>)
<valoare>  =  (<date>  <date>  <eticheta>  <mesaje>
```

Utilizarea cadrelor și reprezentarea cunoștințelor cu această metodă depinde de limbajul de programare folosit, dar acestea prezintă anumite caracteristici de genul primitivelor semantice utilizate, precum şi legăturile care apar în descrierea problemei. Astfel, categoriile de caracteristici, formele care intră în structura cadrului pot fi de diferite tipuri [3]:

- Forme utilizate la descrierea piesei de cunoaştere;
- Forme ce exprimă reprezentări legate de utilizarea cadrului;
- Forme corespunzând prezumţiilor considerate adevărate;
- Forme ce indică reorientarea continuării în situaţia eşecului.

Relaţiile care apar între forme sunt și ele de mai multe tipuri: de generalizare (specifică concepte cu definiţii mai puţin restrictive), de specializare (specifică concepte cu descrieri ce satisfac o condiţie dată), de apartenenţă (indică clasa din care face parte obiectul prin care acesta moştenește atributele clasei părinte), de compoziţie (indică obiectele care alcătuiesc obiectul principal) și proprietăţile (proprietăţi specifice obiectului).

Exemplu

Reprezentarea cadru pentru o universitate cu profil tehnic [3]:

cadru: universitate tehnică
forma: generalizare
faţeta: institut de învăţământ superior
forma: specializare
faţeta: pregătire tehnică
forma: apartenenţă
faţeta: universitate
forma: adresa
faţeta: strada
faţeta: număr
faţeta: cod poştal
faţeta: oraş
faţeta: ţara
forma: compoziţie
faţeta: conducere
valoarea: rector
valoarea: prorector ştiinţific
valoarea: prorector administrativ
valoarea: secretar ştiinţific
faţeta: compartimente
valoarea: electric
valoarea: mecanic
valoarea: chimic
forma: proprietăţi
faţeta: pregătire
valoarea: curs de zi
valoarea: curs seral
valoarea: curs postuniversitar.

Avantajele folosirii cadrelor Minsky în reprezentarea cunoștințelor sunt următoarele:

1. Reprezentare structurată - organizează cunoștințele clar prin obiecte, atribute (sloturi) și valori (fillere).
2. Modularitate și reutilizabilitate - fiecare cadru poate fi reutilizat pentru componente similare (de exemplu, transformatoare, întrerupătoare de circuit) și modificat independent.

3. Mecanism de moștenire - subcadrele pot moșteni sloturi și valori de la cadrele părinte, reducând redundanța.
4. Combină cunoștințele declarative și procedurale - permite stocarea atât a faptelor (declarative), cât și a acțiunilor/metodelor (procedurale) în cadrul aceleiași structuri.
5. Valori implicite și așteptate - ajută la gestionarea informațiilor incomplete folosind valori implicite și așteptări.
6. Format lizibil de om - inginerii și experții în domeniu pot înțelege și modifica cu ușurință bazate pe cadre.
7. Eficient pentru modelarea specifică domeniului - ideal pentru reprezentarea componentelor și scenariilor stereotipe ale sistemului energetic.
8. Suportă raționamentul cu declanșatoare - poate invoca automat proceduri atunci când sunt îndeplinite anumite condiții din sloturi.

Punctele slabe ale cadrelor Minsky sunt expresivitate limitată pentru logica complexă (nu este potrivită pentru reprezentarea relațiilor logice extrem de dinamice sau abstracte), modelare mai dificilă a cunoștințelor neorientate pe obiecte (metodele bazate pe cadre nu sunt ideale pentru modelarea proceselor sau relațiilor care nu se încadrează într-o structură clară obiect-atribut), lipsa unui mecanism standard de inferență (spre deosebire de regulile de producție, cadrele se bazează pe reguli sau daemoni atașați extern pentru inferență, ceea ce face ca raționamentul să fie mai puțin uniform), probleme de scalabilitate în sistemele mari (pe măsură ce numărul de cadre și interacțiuni între sloturi crește, devine mai greu de gestionat consistența și performanța), conflicte de moștenire (dacă nu este gestionată cu atenție, moștenirea sloturilor poate duce la ambiguitate sau conflicte de valori) și mai puțin transparentă decât sistemele bazate pe reguli (logica din spatele concluziilor poate să nu fie la fel de ușor de urmărit ca în cazul regulilor de producție explicite).

4.7. Programare orientată pe obiecte

Programarea orientată pe obiecte reprezintă o etapă avansată în reprezentarea cunoștințelor. Structura de bază în această metodă este obiectul, care cuprinde toate atributele şi funcţiile (metodele) care acţionează sau au legătură cu el. În domeniul programării, obiectele sunt incluse în clase (classes) şi poartă numele de instanţe ale acestora.

Programarea orientată pe obiecte se caracterizează prin faptul că un program conţine o colecţie de obiecte care interacţionează între ele, în comparaţie cu programele procedurale care conţin liste de instrucţiuni. Această abordare a problemelor este asemănătoare cu modul în care lumea reală este percepută de oameni, şi oferă avantajele de a crea programe mai uşor de înţeles, depanat şi extins.

Principiile de bază ale programării orientate pe obiecte sunt [4]:

1. Abstractizarea – Este posibilitatea ca un program să ignore unele aspecte ale informaţiei pe care o manipulează, adică posibilitatea de a se concentra asupra esenţialului. Fiecare obiect în sistem are rolul unui "actor" abstract, care poate executa acţiuni, îşi poate modifica și comunica starea şi poate comunica cu alte obiecte din sistem fără a dezvălui cum au fost implementate acele facilităţi. Procesele, funcţiile sau metodele pot fi de asemenea abstracte, și în acest caz sunt necesare o varietate de tehnici pentru a extinde abstractizarea.
2. Încapsularea – numită și ascunderea de informaţii: Asigură faptul că obiectele nu pot schimba starea internă a altor obiecte în mod direct (ci doar prin metode puse la dispoziţie de obiectul respectiv); doar metodele proprii ale obiectului pot accesa starea acestuia. Fiecare tip de obiect expune o interfaţă pentru celelalte obiecte care specifică modul cum acele obiecte pot interacţiona cu el.
3. Polimorfismul – Este abilitatea de a procesa obiectele în mod diferit, în funcţie de tipul sau de clasa lor. Mai exact, este abilitatea de a redefini metode pentru clasele derivate. De exemplu pentru o clasă Figura putem defini o metodă arie. Dacă Cerc, Dreptunghi, etc. vor extinde clasa Figura, acestea pot redefini metoda arie.
4. Moştenirea – Organizează şi facilitează polimorfismul şi încapsularea, permiţând definirea și crearea unor clase specializate plecând de la clase (generale) deja definite - acestea pot împărtăşi (şi extinde) comportamentul lor, fără a fi nevoie de a-l redefini. Aceasta se face de obicei prin gruparea obiectelor în clase şi prin definirea de clase ca extinderi ale unor clase existente. Conceptul de moştenire permite construirea unor clase noi, care păstrează caracteristicile şi comportarea, deci datele și funcţiile membru, de la una sau mai multe clase definite anterior, numite clase de bază, fiind posibilă redefinirea sau adăugarea unor date şi funcţii noi. Se utilizează ideea: "Anumite obiecte sunt similare, dar în acelaşi timp diferite". O clasă moştenitoare a uneia sau

mai multor clase de bază se numeşte clasă derivată. Esenţa moştenirii constă în posibilitatea refolosirii lucrurilor care funcţionează.

Precum s-a precizat în paragraful anterior, clasele sunt ierarhizate, unele moştenind caracteristicile altora. Un grup de obiecte de la un nivel mai înalt are un număr mai mic de atribute comune decât cel de la un nivel inferior. Atributele unui grup de obiecte de la nivel mai înalt sunt moştenite la niveluri inferioare.

Exemplu

Un sistem electroenergetic poate fi privit ca ansamblu a tuturor dispozitivelor, instalaţiilor şi echipamentelor care pot fi simplu catalogate ca echipament. Astfel liniile, transformatoarele, generatoarele, consumatorii, dispozitivele de protecţie etc. sunt fiecare definite prin intermediul unor clase, având caracteristicile lor. Adică, conceptul de linie de transport poate fi descris cu ajutorul clasei „Linie", iar totalitatea liniilor din sistemului electroenergetic sunt instanţe ale acestei clase. În figura următoare este descrisă grafic structura unui sistem electroenergetic, şi relaţiile dintre clase și instanţele lor.

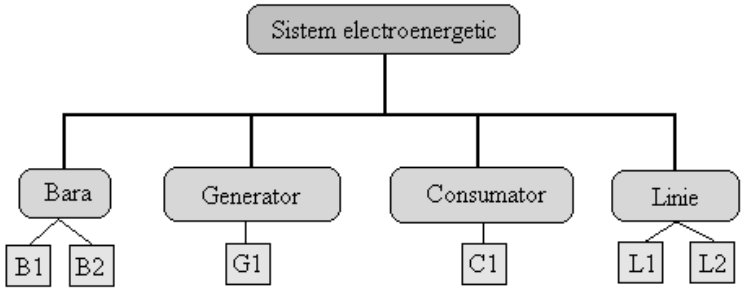


Fig. 4.5. Reprezentarea grafică a unui sistem electroenergetic

Exemplu

În continuare se dă un exemplu a clasei „Linie".

Clasa: Linie

flow	circulaţie de putere
max	capacitate de transport
x	reactanţa liniei
status	starea (în funcţiune, deconectată, defectă)

Metode specifice

energize	pune linia în funcţiune

interrupt	deconectează linia
fault	linia este defectă
overload check	verifică supraîncărcarea

Avantajele folosirii obiectelor și claselor în reprezentarea cunoștințelor în cadrul sistemelor expert rezultă din caracteristicile acestora. Acestea sunt:

- Modularitate – fiecare componentă a sistemului energetic este un obiect separat, îmbunătățind claritatea și mentenabilitatea.
- Reutilizabilitate – clasele pot fi reutilizate în diferite aplicații sau scenarii (de exemplu, flux de energie, protecție).
- Scalabilitate – sistemele mari pot fi extinse fără a restructura întreaga bază de cunoștințe.
- Mapare naturală – se potrivește îndeaproape cu modul în care inginerii gândesc deja despre sisteme ca componente care interacționează.
- Încapsulare – ascunde complexitatea și separă cunoștințele de control.
- Comportament dinamic – metodele permit luarea activă a deciziilor și schimbările de stare în cadrul obiectului.

Comparativ, dezavantajele acestei metode de reprezentare a cunoștințelor rezultă din complexitatea în implementare, deoarece este nevoie de expertiza în programare și un design adecvat pentru a gestiona interacțiunile. Un alt dezavantaj apare din cauza necesității utilizării mai intense a resurselor, care apare din cauza că sistemele expert bazate pe obiecte pot utiliza mai multă memorie și procesare, în special în aplicațiile în timp real. Integrarea mai dificilă a regulilor și suprafața de depanare sunt alte puncte slabe ale utilizării claselor și obiectelor în reprezentarea cunoștințelor. Într-adevăr, spre deosebire de sistemele de producție, raționamentul bazat pe reguli nu este nativ decât dacă este încorporat în metode, iar urmărirea logicii prin obiecte care interacționează și moștenire poate fi dificilă.

4.8. Ontologii

O ontologie este o specificație formală și explicită a unei conceptualizări partajate. Aceasta definește concepte (clase), relațiile între concepte, atribute (proprietăți), constrângeri și instanțe (indivizi sau obiecte). În sisteme expert, reprezentarea prin implicarea ontologiilor permit sistemelor să raționeze despre cunoștințele domeniului folosind modele semantice structurate.

În sistemele energetice, ontologiile formalizează elemente precum „transformator”, „bus”, „linie”, „releu”, „defect” etc. și inter-relațiile acestora.

O ontologie este de obicei compusă din clase, relații, proprietăți, axiome și instanțe. Clasele sunt conceptele cheie ale ontologiilor, care sunt în legătură unele cu altele prin intermediul relațiilor, și au atribute / proprietăți caracteristice. Regulile sau contrângerile precum „Un întrerupător trebuie să fie conectat la două magistrale” intră în categoria axiomelor. Instanțele sunt obiecte specifice ale ontologiilor.

Exemplu de obiect este:

Transformator_1 cu o putere nominală de 100 MVA.

Ontologiile pot fi reprezentate în OWL (Limbaj de ontologie web), RDFS (Schema RDF) și Logicele de descriere (Descriptions logics - fundamentul logic formal).

În sistemele expert pentru sisteme energetice, ontologiile oferă o bază solidă semantică pentru organizarea și interpretarea cunoștințelor din domeniu, permițând:

1. Modelarea domeniului - reprezintă componentele sistemelor electroenergetice, subsistemele și stările operaționale.
   Exemplu: Întrerupător ⟶ hasState ⟶ Deschis | Închis (Întrerupătorul are starea deschis sau închis).
2. Interoperabilitate - permit integrarea între sisteme (de exemplu, SCADA, EMS – Energy Management System, DMS – Demand Management System) prin comunicare comună. Această capabilitate este utilă pentru modelele bazate pe IEC 61970/61968 (CIM).
3. Raționament - inferența logică asupra stării sistemului energetic.
   Exemplu: Dacă Linia A este întreruptă și Întrerupătorul B este deschis → declanșează reconfigurarea.
4. Gestionarea alarmelor – clasificarea alarmelor din punct de vedere semantic (de exemplu, încălcări de tensiune, supraîncărcări ale transformatoarelor).
5. Diagnosticarea defecțiunilor – utilizarea relațiilor ontologice și a inferenței pentru a identifica cauzele principale.
6. Suport decizional - asistarea operatorilor în alegerea acțiunilor de restaurare sau comutare. În plus, regulile bazate pe ontologii pot evalua opțiunile semantic.

Exemplu ontologie

Class: Transformer
SubClassOf: PowerComponent
Properties:
Clasă: Transformator
SubClasăA: ComponentăPutere
Proprietăți:
hasRatedPower→float [MVA]
hasVoltageLevel→VoltageLevel

Class: Bus
SubClassOf: Node
RelatedTo: Transformer (via connectedTo)

Class: Alarm
Properties:
hasCause→PowerComponent
hasSeverity→{Low, Medium, High}
hasTimestamp → DateTime
arePutereNominală→float [MVA]
areNivelTens→NivelTens

Clasă: Magistrală
SubClasăA: Nod
RelațieCu: Transformator (via conectatCU

Clasă: Alarmă
Proprietăți:
AreCauză→ComponentăPutere
AreSeveritate→{Joasă, Medie, Mare}
areDurata→ Timp

Avantajele utilizării ontologiilor în reprezentarea cunoștințelor în sistemele expert sunt următoarele:

- Standardizarea cunoștințelor din domeniu.
- Reutilizabilitatea între diferite aplicații și proiecte.
- Interoperabilitatea între sisteme eterogene.
- Facilitează achiziția și întreținerea cunoștințelor.
- Căutare și interogare semantică.
- Suportă atât raționament determinist, cât și probabilist.

Dezavantajele și provocările folosirii ontologiilor în reprezentarea cunoștințelor din BC sunt legate de efortul inițial de modelare ridicat, necesitatea expertizei în domeniu și de abilități de inginerie ontologică, problemele de scalabilitate pentru sisteme foarte mari (pot fi atenuate cu design modular), dificultatea de sincronizare cu datele în timp real și necesitatea alinierii ontologiilor la integrarea între domenii.

Instrumente pentru dezvoltarea de ontologii sunt Protégé (Stanford) – Cel mai utilizat editor de ontologii gratuit, TopBraid Composer, OntoStudio, OWL API și SPARQL – Pentru interogarea ontologiilor.

În concluzie, în sistemele expert pentru sisteme energetice, ontologiile oferă un model structurat bazat pe logică al domeniului, o fundație pentru raționament avansat și luarea deciziilor, o punte pentru interoperabilitate în rețelele electrice smart și sistemele energetice smart. Acestea sunt esențiale în sistemele complexe în care claritatea semantică, raționamentul automat și consecvența domeniului sunt critice.

4.9. Logica fuzzy

Logica fuzzy joacă un rol esențial în sistemele expert pentru sistemele energetice, dând abilitatea SE să gestioneze imprecizia, incertitudinea și adevărurile parțiale, care sunt comune în rețelele electrice și în procesele decizionale din lumea reală.

Întrucât logica fuzzy a fost prezentată în detaliu în cadrul suportului de curs a disciplinei Aplicații ale IA în managementul energiei, aici se va insista doar asupra legăturii dintre SE și logica fuzzy în reprezentarea cunoștințelor.

Implicarea logicii fuzzy în dezvoltarea SE se motivează prin faptul că sistemele energetice sunt complexe, neliniare și funcționează adesea în medii cu date vagi sau incomplete. Logica fuzzy ajută în rezolvarea problemelor prin folosirea raționamentului cu măsurători imprecise (de exemplu, „tensiunea este ușor scăzută”), imitarea expertizei operatorului și gestionarea cunoștințelor subiective (de exemplu, ce constituie „supraîncărcare aproape completă”?)

Structura unui sistem expert fuzzy cuprinde în plus față de un SE clasic două blocuri în plus și anume blocul de fuzzificare și blocul de defuzzificare.

Față de regulile de inferență clasice, regulile fuzzy de inferență conțin variabile și valori fuzzy, precum se observă în exemplele de mai jos.

Exemplu

DACĂ (curentul de linie este mare) ȘI (tensiunea este mică)
ATUNCI (riscul este mare)
DACĂ (temperatura transformatorului este foarte mare)
ATUNCI (opriți transformatorul)
DACĂ (puterea reactivă este ușor mică)
ATUNCI (creșteți ușor ieșirea băncii de condensatoare)

Logica fuzzy poate fi integrată cu structuri semantice, rezultând metode hibride de reprezentarea a cunoștințelor. Astfel, sunt ontologiile fuzzy - extinderea ontologiile clasice pentru a gestiona concepte imprecise (de exemplu, „sarcină mare", „ușor sub tensiune") și cadre fuzzy - definirea sloturi cu valori fuzzy (de exemplu, slot: voltageLevel = {low: 0.3, normal: 0.6, high: 0.1}).

Punctele slabe și forte ale utilizării logicii fuzzy în dezvoltarea BC a sistemelor expert sunt sintetizate în diagrama din figura 4.6.

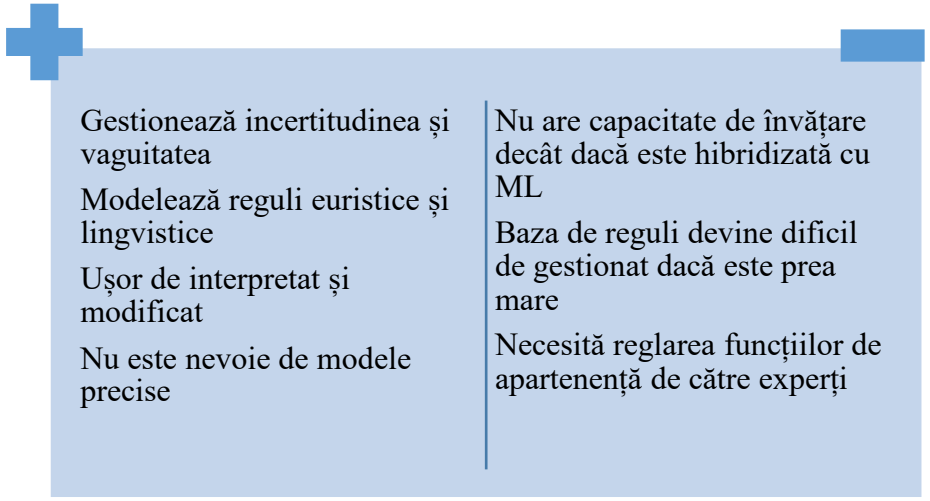


Fig. 4.6. Punctele slabe și forte ale logicii fuzzy în SE

În concluzie, despre logica fuzzy în sistemele expert se poate specifica următoarele - logica fuzzy permite sistemelor expert să emuleze raționamentul uman expert, să gestioneze date imprecise de la senzori, să îmbunătățească procesul decizional în condiții de incertitudine și să sporească reziliența strategiilor de control și protecție. Logica fuzzy este deosebit de puternică acolo unde modelele precise nu sunt disponibile sau sunt insuficiente, iar regulile bazate pe experiență sunt cea mai bună sursă de cunoștințe disponibilă.

4.10. Euristici

Euristicile sunt esențiale pentru SE aplicate pentru rezolvarea problemelor din sistemele energetice, unde sunt necesare decizii rapide, bazate pe experiență, în medii complexe, incerte și sensibile la timp. În acest context, euristicile reprezintă cunoștințele practice, empirice, pe care experții umani le utilizează, în

special atunci când metodele analitice sau numerice sunt prea lente sau impracticabile.

Euristicile sunt tehnici bazate pe experiență, utilizate pentru a rezolva probleme, a lua decizii sau a descoperi soluții rapid atunci când o căutare exhaustivă este impracticabilă. Nu se garantează că sunt optime sau corecte, dar sunt eficiente în practică. În sistemele expert, euristicile iau adesea forma unor reguli simplificate, îndrumări sau strategii intuitive. În sistemele energetice, euristicile sunt derivate din experiența operatorului, standardele de proiectare sau modelele istorice de incidente.

În sistemele expert pentru sistemele energetice, euristicile sunt utilizate pentru a simplifica deciziile complexe (Exemplu - Dacă două linii sunt supraîncărcate, deschideți-o pe cea cu cea mai mică marjă de sarcină), pentru a controla direcția raționamentului și de a rezolva inferențele (Exemplu - Dacă defectul nu este eliminat după 100 ms, reînchideți întrerupătorul după 300 ms), pentru a completa datele lipsă (Exemplu - Presupunând că priza transformatorului este în poziția nominală dacă datele sunt învechite), pentru a prioritiza opțiunile (Exemplu - Restaurați alimentatoarele de înaltă tensiune înaintea celor de joasă tensiune) și permit sistemului expert să imite judecata umană și să acționeze rapid fără un calcul complet.

Euristicile sunt reprezentate sub forma unor reguli de inferență

DACĂ condiție, ATUNCI acțiune (CU factorul de inferență).

Exemplu

DACĂ (tensiune < 0,9 pu) ȘI (sarcina este mare)
ATUNCI (se emite comanda de comutare a băncii de condensatoare)
CU încredere 0,8

Mai multe exemple de reguli euristice sunt prezentate în tabelul 4.3.

Tabelul 4.3. Exemple de reguli euristice

Domeniul energetici	Regula
Alegerea alarmelor	DACĂ (tensiunea scade ușor ȘI revine rapid) ATUNCI (ignorați ca perturbație tranzitorie)
Deconectare sarcină	DACĂ (frecvență < 48,5 Hz ȘI există deficit de generare) ATUNCI (deconectați mai întâi sarcinile industriale)

Diagnosticare defecțiuni	DACĂ (întrerupătorul se declanșează ȘI zona releului se potrivește cu locația defectului) ATUNCI (defect intern probabil)
Planificare restaurare	DACĂ (pană de curent detectată ȘI distribuție automată disponibilă) ATUNCI (porniți distribuția automată și restaurați mai întâi alimentatorul spitalului)

Euristicile utilizate în sistemele expert sunt de obicei extrase de la experți umani (operatori, planificatori, ingineri de protecție), manuale, manuale operaționale, ghiduri de utilități, studii de caz și jurnale de incidente, precum și experimente de simulare. De asemenea, pot fi învățate adaptiv (de exemplu, prin învățare automată) și apoi codificate manual sau automat.

Caracteristicile sistemelor expert care utilizează euristicilor în reprezentarea cunoștințelor sunt următoarele:

- Iau rapid decizii, întrucât euristicile ocolesc calculele complexe.
- Reține cunoștințele experților având în vedere că reflectă intuiția și experiența operatorilor umani.
- Lucrează cu date incomplete întrucât euristicile oferă decizii rezonabile în condiții de incertitudine.
- Sunt flexibile și modificabile - regulile pot fi actualizate în funcție de nevoile în continuă evoluție ale grilei.

Comparativ, limitări sistemelor expert bazate pe euristici se referă la faptul că acestea nu sunt optime, deoarece se poate ajunge la decizii suboptimale sau incorecte în cazuri rare, sunt dificil de validat (performanța depinde de context și de calitatea regulilor), pot fi asociate cu un blocaj în achiziția de cunoștințe (extragerea euristicilor utile de la experți poate fi dificilă) și oferă o generalizare slabă, întrucât o euristică poate să nu funcționeze bine în scenarii noi.

Instrumentele și platformele de implementare a SE care suportă euristici în reprezentarea cunoștințelor din BC sunt CLIPS / JESS (shell-uri pentru sisteme expert bazate pe reguli), MATLAB (pentru logică de control bazată pe euristică), Drools (motor de reguli bazat pe Java) și instrumente bazate pe Python (PyKnow, Experta).

În concluzie, euristicile aduc intuiția expertă în sistemele expert pentru rezolvarea problemelor din sistemele energetice. Acestea sunt cel mai bine utilizate atunci când sunt necesare decizii rapide, modelele matematice nu sunt

disponibile, datele de intrare sunt incomplete și este nevoie de expertiză, dar experții nu sunt disponibili. Acestea sporesc capacitatea sistemului de a imita deciziile umane în condiții complexe și incerte, în special atunci când sunt combinate cu logică fuzzy, ontologii sau structuri orientate pe obiecte.

4.11. Reprezentarea bazată pe constrângeri

Reprezentarea cunoștințelor bazată pe constrângeri este o altă metodă puternică utilizată în sistemele expert pentru aplicațiile sistemelor energetice, în special atunci când problema implică optimizare, configurare sau alocare de resurse. Reprezentarea cunoștințelor bazată pe constrângeri permite sistemului expert să raționeze eliminând opțiunile nevalide și satisfacând cerințele specificate, în loc să se bazeze exclusiv pe logica procedurală sau regulile euristice.

Reprezentarea cunoștințelor bazată pe constrângeri presupune ca cunoștințele să fie itroduse ca un set de condiții sau restricții care trebuie îndeplinite de orice soluție validă. Astfel, o constrângere este o afirmație declarativă despre ceea ce este permis sau necesar într-o soluție. Nu spune cum să se realizeze acest lucru, ci doar ce trebuie să fie adevărat.

În sistemele energetice, constrângerile pot fi de diferite tipuri:

- Constrângeri fizice - ecuații ale fluxului de putere (Legea lui Ohm, legile lui Kirchhoff).
- Constrângeri operaționale - limite de tensiune, limite termice ale liniilor, interval de frecvență.
- Constrângeri logice - „O linie trebuie să fie energizată înainte de a putea transporta energie”;
- Constrângeri de securitate - criterii de contingență N-1.
- Constrângeri de programare - „Întreținerea pe Linia A nu trebuie să se suprapună cu Linia B”.
- Constrângeri topologice - barele colectoare ale substației trebuie conectate în configurații specifice.

Un sistem expert bazat pe constrângeri implică de obicei variabile, domenii, contrângeri și motorul de inferență. Variabilele sunt mărimi cărora li se pot atribui valori (de exemplu, tensiuni, stări ale întrerupătorului). Domeniile sunt valori posibile pentru fiecare variabilă (de exemplu, [0,95–1,05 pu] pentru tensiune). Constrângerile sunt relații dintre variabile care restricționează combinațiile valide, iar motor de inferență găsește soluții care satisfac toate

constrângerile. Acest model este adesea denumit problemă de satisfacție a constrângerilor.

O problemă de satisfacere a constrângerilor este definită ca

Un set de variabile

$$X=\{x_1,x_2,..,x_n\} \tag{4.1}$$

Fiecare variabilă x_i are un domeniu D_i.

Un set de constrângeri

$$C=\{C1,C2,...,Ck\} \tag{4.2}$$

restricționează valorile pe care variabilele le pot lua simultan.

Exemplu

Variabile:

$V\_bus_1 \in [0.95, 1.05]$ pu

Tap_Transformer1 $\in \{\pm5\%, \pm10\%, \pm15\%\}$

Constrângeri:

$|V\_bus_1 - V\_bus_2| < 0.1$ pu

$Transformer_1$ output voltage = $V\_bus_1 \times Tap\_Transformer_1$

Contrângerile pot fi scrise în diferite forme:

- Expresii matematice

  $P\_gen_1 + P\_gen_2 = P\_load + P\_losses$

  $|V\_i - V\_j| \leq \Delta V\_max$

- Constrângeri logice (declarative)

  Dacă Linia_AB nu funcționează Atunci Putere_AB = 0

  Dacă Feeder_1 alimentează spitalul Atunci Nu oprire Feeder_1

- Grafuri de contrângeri

  Noduri: Variabile (de exemplu, tensiuni de magistrală)

  Margini: Constrângeri (de exemplu, Legea lui Ohm)

Instrumentele și limbajele care pot fi utilizate pentru a dezvolta sisteme expert cu constrângeri sunt Programarea Logică cu Constrângeri, care combină reguli logice cu rezolvarea constrângerilor, Prolog (un limbaj declarativ pentru exprimarea și rezolvarea contrarelor), MATLAB + Optimization Toolbox care poate rezolva constrângerile sistemului energetic, CPLEX și Drools Planner

(OptaPlanner), care este un rezolvitor de constrângeri bazat pe Java pentru programare și dispecerizare.

Avantajele și dezavantajele utilizării sistemelor expert bazate pe constrângeri sunt sintetizate în figura 4.7.

Reprezentarea cunoștințelor bazată pe constrângeri este crucială pentru sistemele expert dedicate sistemelor energetice atunci când scopul implică optimizare, verificare a fezabilității, reconfigurarea sistemului, dispecerizarea sarcinii și impunerea securității. Aceasta completează alte metode de reprezentare a cunoștințelor, oferind o modalitate formală, structurată și consistentă din punct de vedere matematic de a defini și impune cerințele operaționale.

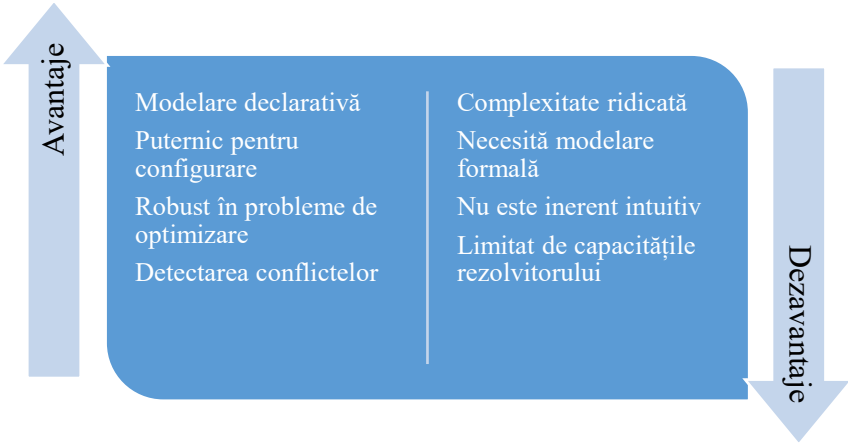


Fig. 4.7. Avantajele și dezavantajele utilizării constrângerilor

4.12. Arbori de decizie

Arborii decizionali sunt o formă intuitivă și structurată de reprezentare a cunoștințelor care poate fi aplicată eficient în sistemele expert dedicate problemelor din sistemele energetice. Aceștia modelează deciziile și posibilele lor consecințe folosind o structură arborescentă, ceea ce îi face ideali pentru sarcini de diagnosticare, clasificare și luare a deciziilor.

Un arbore decizional este un model ierarhic utilizat pentru a reprezenta reguli sau logica decizională într-un format ramificat. Acesta constă dintr-un nod rădăcină care este punctul de plecare și reprezintă principala condiție de decizie sau de intrare, noduri interne, care sunt puncte de decizie bazate pe variabile de

intrare (de exemplu, tensiune, curent), muchii/ramificații definite ca rezultate posibile ale unui test (de exemplu, „tensiune < 0,9 pu”) și noduri frunză care sunt rezultatele sau acțiunile finale (de exemplu, „sarcină redusă”, „avertisment de problemă”).

Arborii decizionali sunt ideali atunci când este nevoie de o logică clară și ușor de urmărit, sistemul trebuie să își explice deciziile, există nevoia de a clasifica erori, de a declanșa alarme sau de a recomanda acțiuni, iar datele sunt discrete sau categorice, dar pot fi și discretizate. În sistemele expert, arborii decizionali pot emula procesul de luare a deciziilor al unui expert uman folosind alegeri sistematice binare/multivivale.

Figura 4.8. ilustrează arborele decizional care descrie logica protecției la supracurent.

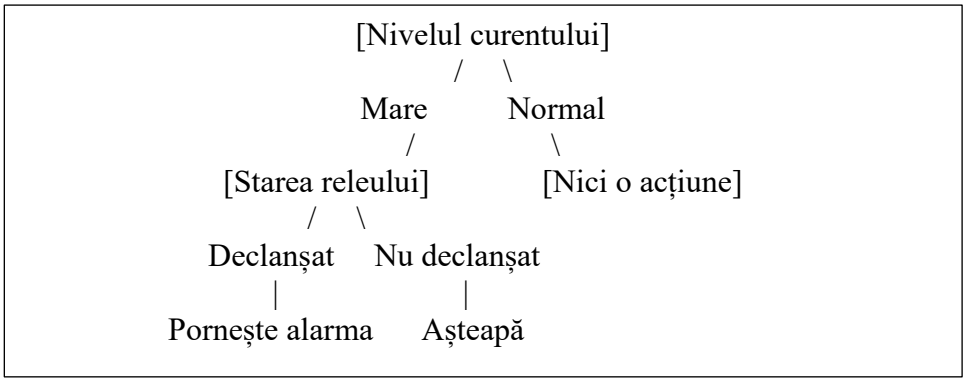


Fig. 4.8. Exemplu arbore decizional

În reprezentarea cunoștințelor se pot folosi trei tipuri de arbori decizionali, și anume: clasici (decizie asupra unei clasificări), de regresie (prezice valoarea numerică de la ieșire) și hibrizi (ieșiri mixte).

Arborii decizionali sunt stocați în baza de cunoștințe ca reguli de inferență DACĂ-ATUNIC imbricate, structuri de date grafice (modele de obiecte arbore/graf), structuri XML sau JSON (pentru interpretabilitate) sau tabele de date (în matrici decizionale tabelare). În sistemele expert bazate pe reguli, fiecare cale de la rădăcină la frunză poate fi convertită într-o regulă.

Construirea arborilor de decizie se poate face manual (bazată pe cunoștințe), utilizând informații de la experți care traduc regulile intuitive în condiții de ramificare, sau automat (bazată pe date) utilizând algoritmi de învățare automată precum ID3, C4.5 / C5.0, CART (arbori de clasificare și regresie) și CHAID. Aceștia din urmă învață arbori din date istorice SCADA, relee sau senzori.

Avantajele sistemelor expert bazate pe arbori decizionali, și dedicate problemelor din sistemele energetice rezultă din factul că aceste sisteme expert sunt lizibile și interpretabile, au o inferență rapidă, pot funcționa cu date lipsă, plus faptul că extrageree de reguli și suport pentru abordări sunt bazate atât pe logică, cât și pe învățare.

Contrar, limitările acestor sisteme expert se referă la faptul că acestea pot să suporte o supra-adaptare, soluția găsită este limitată la raționament ierarhic, sunt dificil de întreținut manual pentru arbori mari și nu au gestionare incertitudinii.

Instrumentele pentru implementarea acestor sisteme expert sunt MATLAB Classification Learner, Python (scikit-learn), Weka / Orange / KNIME, R (rpart, party) și CLIPS / JESS.

În concluzie, arborii decizionali sunt instrumente puternice în sistemele expert în rezolvarea problemelor de diagnosticare, clasificare și consultanță în sistemele energetice. Transparența și formatul lor intuitiv îi fac ideali pentru simularea deciziilor operatorilor umani, structurarea logicii într-un mod compact și explicabil și servirea drept punte între cunoașterea manuală și învățarea automatizată. Atunci când sunt combinați cu logica fuzzy sau ontologii, arborii decizionali devin și mai puternici în gestionarea incertitudinii și a raționamentului semantic în rețelele inteligente sau în sistemele de protecție.

4.13. Rețele Bayesiene

Rețelele bayesiene, cunoscute și sub denumirea de rețele de credințe sau rețele Bayes, sunt o formă puternică de reprezentare probabilistică a cunoștințelor utilizată în sistemele expert pentru aplicații în sisteme energetice, în special acolo unde incertitudinea și raționamentul probabilistic sunt esențiale.

O rețea bayesiană este un graf aciclic direcționat în care nodurile reprezintă variabile aleatoare (de exemplu, citiri ale senzorilor, stări ale sistemului, cauze ale erorilor), muchiile reprezintă dependențele probabilistice dintre variabile, iar fiecare nod conține un tabel de probabilități condiționate care definește probabilitatea nodului, având în vedere părinții săi. Pe scurt, rețelele bayesiene permit raționamentul în condiții de incertitudine, folosind teorema lui Bayes pentru a actualiza convingerile pe baza unor dovezi noi.

Rețelele bayesiene sunt utilizate în sistemele expert dedicate sistemelor energetice, deoarece sistemele energetice sunt afectate de date incomplete sau incerte ale senzorilor, echipamente care se pot defecta probabilistic, acțiuni ale operatorilor care au consecințe incerte; iar rețelele bayesiene sunt ideale pentru

diagnosticarea defecțiunilor, validarea senzorilor, evaluarea riscurilor, mentenanța predictivă și raționamentul cauzal.

Rețelele bayesiene se pot caracteriza de următoarele trăsături:

1. Probabilistice – acestea au capabilitatea de a modela incertitudinea naturală.
2. Grafice – acestea au o structură intuitivă și vizuală.
3. Modulare – acestea pot cuprinde noi cunoștințe prin extinderea grafului.
4. Susține inferența - se pot deduce cauze (diagnostic) sau efecte (predicție).
5. Prietenoase cu datele și experții - poate fi construit din date, reguli de experți sau ambele.

Structura unei rețele beyesiene care poate fi folosită într-un sistem expert dedicat diagnozei defectelor apărute la transformatoarele de putere este ilustrată în figura 4.9.

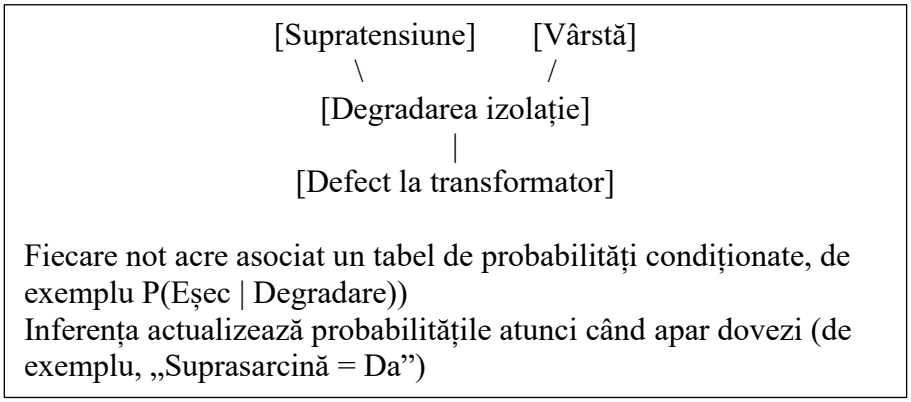


Fig. 4.9. Exemplu rețea Bayesiană

Rețelele bayesiene pot fi integrate în baza de cunoștințe a sistemelor expert construite manual din cunoștințe de specialitate, învățate din date istorice (folosind algoritmi precum K2, Hill-Climbing etc.) sau codificate folosind structuri de grafuri, tabele de probabilități condiționate și motoare de inferență probabilistice (de exemplu, Netica, PyMC, bnlearn).

Punctele forte ale rețelelor bayesiene se referă la faptul că gestionează incertitudinea, suportă raționamentul, sunt scalabiles, suportă învățarea și funcționează cu date lipsă sau parțiale. Comparativ, limitările acestora au legătură cu faptul că acestea necesită tabele cu propabilități condiționate precise, au un cost computațional ridicat, trebuie proiectat cu atenție pentru a reflecta relațiile cauzale și poate necesita seturi de date mari și curate.

Instrumente și biblioteci care suportă dezvoltarea de sisteme expert bazate pe rețele bayesiene sunt Netica pentru construirea și vizualizarea rețelelor bayesiene, GeNIe/SMILE, Python, limbajul R și Hugin Expert.

În concluzie, rețelele Bayesiene aduc capacități puternice de raționament probabilistic sistemelor expert din sistemele energetice. Structura lor grafică, capacitatea de a gestiona incertitudinea și modelarea bazată pe date le fac neprețuite pentru sarcini precum diagnosticarea defecțiunilor, estimarea riscurilor și asistența decizională în timp real. În rețelele electrice smart în continuă evoluție, unde datele și incertitudinea sunt abundente, rețelele bayesiene pot ajuta la integrarea inteligenței artificiale cu cunoștințele fizice și operaționale într-un mod matematic valid.

4.14. Reprezentarea bazată pe cazuri

Reprezentarea cunoștințelor bazată pe cazuri este o metodă în care experiențele anterioare (cazurile) sunt utilizate pentru a rezolva probleme noi. În contextul sistemelor expert pentru aplicații în sisteme energetice, această abordare modelează raționamentul într-un mod similar cu cel al experților umani, prin reamintirea și adaptarea soluțiilor anterioare.

Reprezentarea bazată pe cazuri implică stocarea cazurilor anterioare (perechi problemă-soluție), recuperarea cazurilor similare atunci când apare o problemă nouă, adaptarea vechii soluții pentru a se potrivi noii probleme și învățarea din noua experiență prin stocarea ei ca un caz nou. Urmează principiul: „Dacă a funcționat înainte, încearcă din nou cu adaptare”.

Un caz tipic are forma prezentată în următoarele rânduri

Caz = {
Descrierea problemei,
Context,
Soluție,
Rezultat/Evaluare
}

Dacă se particularizează pentru o problemă clară, un exemplu de caz se poate observa în continuare.

Exemplu

{
"Problemă": "Supraîncălzirea transformatorului substației sub sarcină parțială",
"Context": "Temperatura ambiantă = 38°C, Sarcină = 55%, Ventilație = defectă",
"Soluție": "Activați sistemul de ventilație de rezervă și redistribuiți sarcina",

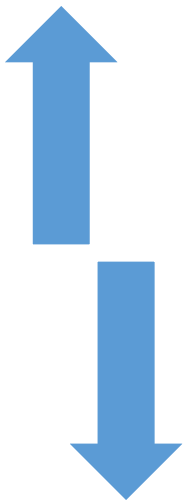
"Rezultat": "Temperatura redusă în 15 minute"
}

Folosirea cazurilor în dezvoltarea sistemelor expert și utilizarea lor pentru a determina soluția optimă presupune realizarea a patru acțiuni:

1. Recuperare - găsirea celor mai similare cazuri anterioare.
   Scopul acestei etape este de a identifica și extrage cazurile stocate anterior din baza de date care sunt cele mai similare cu problema sau situația curentă. În sistemele energetice un astfel de sistem expert va funcționa astfel: când apare o defecțiune (de exemplu, o scădere de tensiune), sistemul caută în baza de date cazuri istorice cu simptome similare. Similitudinea este determinată prin potrivirea caracteristicilor cheie, cum ar fi nivelurile de tensiune, locația, abaterile de frecvență, tipul defecțiunii, condițiile meteorologice etc. Tehnicile utilizate sunt metricile de similaritate (euclidiană, cosinus, distanță ponderată), schemele de indexare (arbori de decizie, tabele de caracteristici, hărți hash) și potrivirea bazată pe ontologie sau fuzzy (pentru atribute vagi sau incerte).
2. Reutilizare - aplicarea soluției din cazul recuperat.
   Scopul etapei este de a reutiliza soluția sau planul de acțiune din cazul (cazurile) recuperat(e) pentru a aborda situația actuală. În rezolvarea problemelor din sistemele energetice, sistemul expert preia răspunsul asociat cu cazul potrivit, de exemplu, redirecționarea energiei, ajustarea compensării reactive sau inițierea deleției de sarcină, și îl aplică ca soluție provizorie. Gestionarea mai multor cazuri se poate realiza prin utilizarea celei mai bune potriviri, combinarea soluțiilor (agregare ponderată sau fuzzy) sau sugerarea de acțiuni alternative (opțiuni de decizie clasificate).
3. Revizuire - modificarea soluției dacă este necesar.
   Scopul etapei este de a testa soluția în contextul actual și adaptarea ei dacă nu este în întregime aplicabilă. În sistemele energetice, sistemele expert în această etapă simulează sau oferă feedback-ul în timp real pentru a fi utilizate în evaluarea dacă soluția reutilizată funcționează. Dacă există diferențe (de exemplu, durata defectului sau setările de protecție ușor diferite), sistemul expert ajustează soluția, eventual implicând un operator uman sau o simulare. Aceste revizuiri se pot realiza prin monitorizare și feedback în timp real, intervenția operatorului, reguli de inferență sau euristici și simularea sistemului energetic sau modelare geamăn digital.
4. Memorare - stocarea noului caz pentru utilizare ulterioară.

Scopul etapei este învățarea din noua experiență prin stocarea ei ca un caz nou în baza de cazuri. Astfel, cazul final revizuit, inclusiv caracteristicile problemei, acțiunile întreprinse și rezultatul final, este stocat în baza de date. Acest lucru permite învățarea continuă, îmbunătățirea și adaptarea sistemului expert la noi tipuri de probleme. Informațiile stocate sunt condiții de intrare (măsurători, date despre defecțiuni), acțiuni întreprinse, rezultate (reușite/eșuate, eficacitate), și context cu marcaj temporal (sezon, condiție de sarcină etc.)

Punctele slabe și forte ale reprezentării bazate pe cazuri sunt enumerate în figura 4.10.



- Imită expertiza umană
- Se adaptează la situații noi
- Învață în timp
- Lucrează cu cunoștințe incomplete
- Efort redus de modelare
- Specific domeniului

- Complexitatea indexării cazurilor
- Adaptarea soluțiilor
- Explozie a cazurilor
- Calitatea cunoștințelor
- Lipsa generalizării

Fig. 4.10. Avantajele și dezavantajele utilizării cazurilor

Instrumentele și framework-urile care suportă dezvoltarea sistemelor expert prin cazuri sunt următoarele: jCOLIBRI, myCBR, CAKE (Case-Based Knowledge Engine) și biblioteci Python precum scikit-learn, CBRpy sau implementări personalizate.

În concluzie, reprezentarea cunoștințelor bazată pe cazuri în sistemele expert dedicate sistemelor energetice este deosebit de eficientă pentru gestionarea defecțiunilor, asistența operațională și planificarea întreținerii. Acestea excelează acolo unde regulile pot fi prea rigide și unde excepțiile și variabilitatea din lumea reală domină. Pe măsură ce sistemele energetice evoluează către rețele electrice smart cu o complexitate crescândă, învățarea din istoricul operațional real prin raționament bazat pe cazuri devine din ce în ce mai valoroasă.

4.15. Reprezentarea bazată pe scenarii

Scenariile sunt un tip de reprezentare structurată a cunoștințelor utilizată pentru a modela secvențe stereotipe de evenimente în contexte specifice. Ele sunt extensii ale cadrelor Minsky, punând accentul pe aspectele temporale și procedurale ale cunoștințelor. Deci, un scenariu descrie ce se întâmplă de obicei, în ce ordine și cine este implicat.

Un scenariu include următoarele informații: condițiile de intrare (când este declanșat scenariu), recuzitele (obiecte/dispozitive necesare), rolurile (agenți, de exemplu, operatori, dispozitive), scenele (pași sau acțiuni ordonate) și rezultatele (rezultatul(ele) așteptat(e)).

Exemplu

Scenariu de declanșare a liniei de înaltă tensiune

Condiții de intrare: Declanșare releu de protecție detectată pe linia de 110 kV.

Roluri: Releu de protecție, sistem SCADA, tehnician de teren.

Elemente: Întrerupător de circuit, indicatori de stare.

Scene:

Declanșarea releului de protecție

SCADA înregistrează defectul

Operatorul confirmă alarma

Linia este izolată

Tehnician trimis pentru inspecție

Rezultat: Defectul este rezolvat și linia este restabilită.

Scenariile sunt excelente atunci când sistemul expert trebuie să analizeze evenimente care se desfășoară în timp, astfel ele sunt ideale pentru diagnosticarea procedurală, asistența operatorilor, predicția evenimentelor și simulatoarele de instruire.

Avantajele utilizării scenariilor sunt următoarele:

- Înmagazinează cunoștințe procedurale - ideal pentru sarcini operaționale.
- Codifică secvențe temporale - excelent pentru modelarea evenimentelor de sistem în timp.
- Se pot compune scenarii modulare și reutilizabile pentru diferite defecțiuni sau subsisteme.
- Se aliniază cu raționamentul uman - reflectă modul în care operatorii gândesc în etape sau scenarii.

- Îmbunătățește explicabilitatea - proces de raționament ușor de urmărit și auditat.
- Suportă simularea/instruirea - acționează ca manuale digitale pentru reluarea scenariilor.

Comparativ, atunci când se folosesc scenarii în sisteme expert pentru reprezentarea cunoștințelor, apar următoarele riscuri din cauza limitărilor acestei metode, și anume: structura rigidă, generalizarea dificilă, neprobabilistica, efortul de achiziție a cunoștințelor și scalabilitate limitată.

Scenariile pot fi implementate folosind sisteme de producție cu secvențiere temporală, motoare de flux de lucru, limbaje de scripting (de exemplu, Python, RuleML cu extensii temporale) și framework-uri orientate pe obiecte (unde fiecare scenă este o metodă). Unele instrumente acceptă scripting cu diagrame de stare sau rețele Petri pentru controlul proceselor.

În concluzie, scenariile din sistemele expert pentru sisteme energetice aduc structură și predictibilitate operațiunilor în care sincronizarea, secvența și corectitudinea procedurală sunt critice. Deși acestea sunt mai puțin adaptive decât alte reprezentări, acestea sunt extrem de eficiente în camerele de control, sistemele de protecție și simulatoarele de antrenament.

4.16. Rețele Petri

O rețea Petri este un instrument de modelare grafică și matematică utilizat pentru a reprezenta sisteme distribuite, concurente, asincrone, nedeterministe și paralele. Rețelele Petri sunt deosebit de utile pentru modelarea comportamentului dinamic al sistemelor energetice, precum operațiunile de comutare, coordonarea protecției, propagarea defectelor sau procesele de flux de lucru în sistemele energetice.

Rețelele Petri sunt grafuri bipartite compuse din locuri (reprezentând stări sau condiții), tranziții (reprezentând evenimente sau acțiuni) și jetoane (reprezentând starea curentă sau marcajele). Mai exact, structura unei rețele Petri conține locul, care este starea, condiția sau situația în care se află echipamentul (de exemplu, „Întrerupător deschis”), evenimentul sau operațiunea de tranziție (de exemplu, „Releu de declanșare declanșat”), jetonul, care reprezintă starea activă (de exemplu, „Defect prezent”) și arcul - conexiune de la loc la tranziție sau invers.

Exemplu

Rețea Petri pentru izolarea defecțiunilor transformatoarelor

Locuri: P1: Funcționare normală
P2: Defecțiune detectată
P3: Declanșare întrerupător inițiată
P4: Transformator izolat
Tranziții: T1: Declanșare releu
T2: Întrerupător deschis
Jetoane: Inițial în P1
→ Se mută în P2 când este detectată o defecțiune
→ T1 se declanșează, jetonul către P3
→ T2 se declanșează, jetonul către P4

Aceasta modelează secvența exactă a acțiunilor de protecție în cazul unei defecțiuni.

Utilizarea rețelelor Petri în reprezentarea cunoștințelor din sistemele expert dedicate problemelor din sistemele energetice este motivată de faptul că rețelele Petri sunt excelente pentru modelarea comportamentului sistemelor secvențiale și concurente, a logicii de control distribuite, a evenimentelor în cascadă, a interblocărilor și mecanismelor de siguranță și a automatizării fluxului de lucru.

Punctele slabe și forte ale sistemelor expert bazate pe rețele Petri sunt descrise în tabelul 4.4.

Tabelul 4.4. Avantajele și dezavantajele SE bazate pe rețele Petri

Avantaje	Dezavantaje
Gestionează concurența și conflictele, întrucât modelează componente multiple care acționează în paralel	Structură statică - rețelele Petri de bază nu pot gestiona cunoștințe dinamice decât dacă sunt extinse
Semantică formală, ele au la bază un algoritm matematic riguros și analizabil	Nu gestionează incertitudinea - nu are raționament probabilistic sau fuzzy decât dacă este modificat
Claritate vizuală, întrucât acestea au o logică de operare ușor de diagramat și de înțeles	Dificil pentru cunoștințe abstracte - nu este potrivit pentru raționament simbolic, conceptual sau bazat pe reguli
Urmărirea stării, având în vedere că jetoanele permit reprezentarea stării în timp real	Probleme de scalabilitate - rețelele complexe pot deveni mari și greu de gestionat

Analiza blocajelor, ceea ce duce la previnirea operațiunile conflictuale	Generalizare slabă - fiecare secvență trebuie modelată explicit
Util pentru validare, deoarece se testează logica sistemului înainte de implementare	
Potrivit pentru automatizare, ele se integrează cu secvențe PLC și SCADA	

Instrumentele software care suportă implementarea rețelelor Petri sunt instrumentele CPN pentru modelarea rețelelor Petri colorate pentru controlul sistemului energetic, Editor de rețele Petri independent de platforma PIPE2 pentru proiectare PLC bazată pe rețele Petri pentru automatizarea substațiilor, și logică hibridă de tip rețea Petri MATLAB/Simulink, plus Stateflow pentru sisteme integrate în rețele inteligente

În concluzie, rețelele Petri sunt un instrument grafic și matematic puternic utilizat pentru reprezentarea și analiza comportamentului dinamic al sistemelor cu evenimente discrete. În contextul sistemelor expert aplicate în rezolvarea problemelor caracteristice sistemelor energetice, rețelele Petri sunt deosebit de potrivite pentru modelarea proceselor care implică concurență, sincronizare, secvențiere și partajare a resurselor - caracteristici adesea întâlnite în operațiunile cu energie electrică.

4.17. Grafuri și metodologii topologice

Grafurile sunt structuri matematice formate din noduri (vârfuri) conectate prin muchii (legături). Topologia se referă la studiul și reprezentarea modului în care aceste noduri sunt interconectate, punând accentul pe conectivitate și relații mai degrabă decât pe distanțele metrice. În sistemele energetice, graficele modelează în mod natural rețeaua de magistrale, linii, transformatoare și consumatori, adică topologia rețelei electrice.

Grafurile și elementele grafice au corespondenți în sistemele energetice, astfel nodul (vortex) grafic poate fi o bară colectoare, o substație, generatoarea sau consumatori, marginea (legătura) poate fi linie electrică, transformator sau generator și atribute (parametrii electrici precum impedanșa, capacitatea, starea de funcțiune).

Implicarea grafurilor în reprezentarea cunoștințelor caracteristice sistemelor energetice este motivată de patru aspecte importante:

1. Sistemele energetice sunt inerent conectate în rețea; modelele grafice reflectă conectivitatea reală a sistemului.
2. Rezonanța numerică bazată pe grafice acceptă raționament structural, localizarea defectelor, reconfigurarea rețelei și analiza impactului.
3. Permite analize topologice precum conectivitatea, găsirea traseului, detectarea insulelor.
4. Util pentru vizualizare și înțelegere intuitivă.

Grafurile folosite pentru reprezentarea cunoștințelor din sistemele energetice se împart în șase categorii: (i) grafuri nedirecționate, care modelează linii în care direcția circulația de putere nu este fixă, (ii) grafuri direcționate, care modelează direcția circulației de putere sau semnalele de control, (iii) grafuri ponderate, în care muchiile poartă ponderi (de exemplu, impedanță, capacitate, distanță), (iv) grafuri dinamice, dedicate situațiilor în care topologia rețelei se schimbă în timp (comutare, defecte), (v) multi-grafuri, caracterizate prin muchii multiple între aceleași noduri (linii paralele), și (vi) hipergrafuri, care modelează dispozitive cu mai multe terminale (de exemplu, magistrale care conectează >2 linii).

Avantajele și dezavantajele utilizării grafurilor în reprezentarea cunoștințelor sunt prezentate succint în figura 4.11.

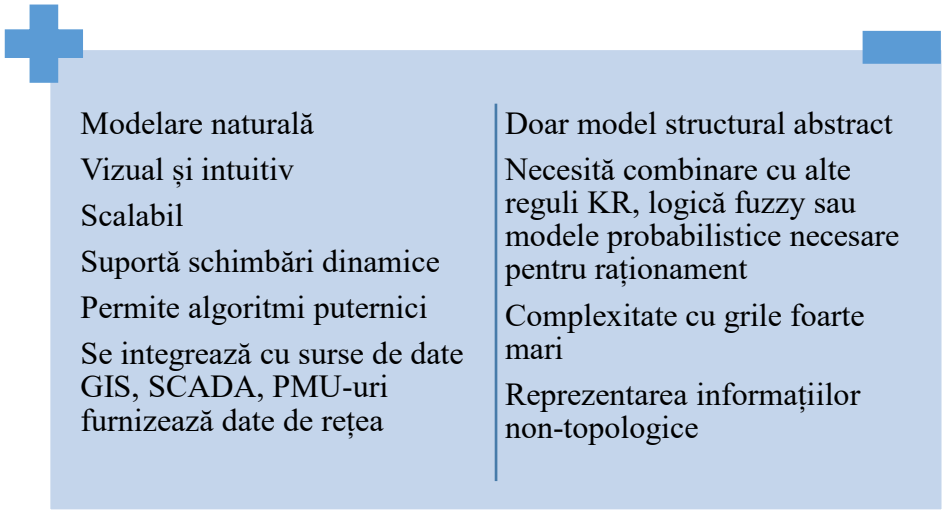


Fig. 4.11. Avantajele și dezavantajele utilizării grafurilor

Instrumente informatice și librări care suportă implementarea de grafuri pentru reprezentarea cunoștințelor în cadrul sistemelor expert sunt NetworkX

(Python), Graph-tool (Python), Gephi, Cytoscape, PowerWorld Simulator și MATPOWER Analiza fluxului de putere și a rețelelor (MATLAB).

În concluzie se pot afirma următoarele: grafurile și reprezentarea cunoștințelor bazate pe topologie formează fundamentul modului în care sistemele expert înțeleg și manipulează structura rețelei electrice. Acestea oferă coloana vertebrală pentru raționamentul despre conectivitate, defecțiuni și reconfigurarea sistemului, adesea combinate cu alte metode de reprezentare a cunoștințelor pentru diagnosticare completă și suport decizional.

4.18. Meta-reguli

Meta-regulile sunt reguli despre reguli, astfel ele guvernează cum, când și ce reguli din sistemul expert sunt aplicate sau modificate. Ele operează la un nivel superior regulilor standard ale domeniului și controlează procesul de raționament, strategia de inferență sau gestionarea bazei de cunoștințe. Meta-regulile ghidează fluxul de control, rezolvarea conflictelor și mecanismele de învățare ale sistemului expert.

Meta-regulile sunt folosite pentru a controla ordinea sau prioritatea declanșării regulilor, a gestiona adaptarea dinamică a regulilor (activare, dezactivare), a implementa strategii de rezolvare a conflictelor atunci când mai multe reguli se potrivesc, a permite raționamentul despre raționament (inferență la nivel meta), a susține explicarea și depanarea comportamentului sistemelor expert și pentru a gestiona incertitudinea și excepțiile din baza de cunoștințe.

Avantajele utilizării meta-regulilor în sistemele expert sunt următoarele:

- Îmbunătățește flexibilitatea sistemului - adaptează raționamentul în funcție de contextul operațional.
- Îmbunătățește mentenabilitatea - gestionează mai eficient bazele de reguli mari, în evoluție.
- Sprijină luarea deciziilor complexe - gestionează elegant cunoștințele concurente sau conflictuale.
- Permite raționamentul la nivel meta - raționează despre procesul de raționament în sine.
- Facilitează încrederea utilizatorului - prin explicații și inferență controlată.

Limitările și provocările legate de utilizarea meta-regulilor în sistemele expert sunt următoarelor: complexitate adăugată (proiectarea meta-regulilor necesită expertiză și planificare atentă), cost suplimentar de performanță

(raționamentul la nivel meta poate încetini inferența), dificultate de depanare (erorile din meta-reguli pot cauza un comportament neașteptat al sistemului) și achiziționarea de cunoștințe (obținerea meta-regulilor de la experți este mai dificilă decât obținerea regulilor de domeniu).

Exemplu

META-REGULĂ: PRIORITIZEAZĂ_REGULILE_DE_SIGURANȚĂ
DACĂ reguli_mai_multiple_gata_de_activare ATUNCI
reguli de activare legate de protecție și siguranță pe primul loc
CONTRAR
reguli de activare legate de optimizare sau diagnosticare

META-REGULĂ: DEZACTIVARE_ÎN_TIMPUL_ÎNTREȚINERII
DACĂ modul_sistem = mentenanță ATUNCI
dezactivează toate regulile de comutare automată

META-REGULĂ: ACȚIUNE_EXPLICAȚIE
CÂND se_declanșează_regula ATUNCI
înregistrează explicația și notifică operatorul

Exemple de meta-reguli din cadrul sistemelor expert dedicate sistemelor energetice sunt prezentate în tabelul 4.5.

Tabelul 4.5. Exemple de meta-reguli

Tip de meta-regulă	Descriere	Exemplu în sistem energetic
Meta-regulă de activare a regulii	Activează sau dezactivează dinamic anumite reguli de domeniu	Dezactivează regulile de optimizare a generării în timpul situațiilor de urgență
Meta-regulă de rezolvare a conflictelor	Definește prioritatea între regulile concurente	Dacă se detectează atât supraîncărcarea transformatorului, cât și defectul releului, se gestionează mai întâi supraîncărcarea
Meta-regulă de control al inferenței	Controlează metoda de căutare sau înlănțuire (înainte/înapoi)	Folosește înlănțuirea înainte în timpul funcționării normale,

		înapoi în timpul analizei defecțiunilor
Meta-regulă de actualizare a bazei de cunoștințe	Declanșează învățarea sau adaptarea regulilor	Dacă o regulă eșuează în mod repetat, se semnalează pentru revizuire
Meta-regulă de explicație	Generează explicații sau urmărește declanșarea regulii	Furnizează justificarea operatorului pentru acțiunea de comutare

Tehnicile de implementare ale meta-regulilor în cadrul sistemelor expert sunt enumerate și explicate în continuare:

- Meta-sisteme bazate pe reguli - meta-reguli codificate ca reguli într-un strat de control separat.
- Îmbunătățiri ale sistemului de producție - meta-reguli integrate în motorul de inferență.
- Meta-programare logică - meta-reguli scrise în limbaje meta-logice (de exemplu, Prolog).
- Euristici de control - meta-reguli procedurale în limbaje de programare procedurală.

În concluzie, meta-regulile reprezintă un mecanism puternic de reprezentare a cunoștințelor și de control în sistemele expert pentru sistemele energetice, permițând gestionarea sofisticată a unor baze de reguli complexe, în evoluție și dinamice. Acestea asigură alinierea raționamentului sistemelor expert cu prioritățile operaționale, cerințele de siguranță și condițiile în schimbare.

4.19. Modele de relații

Modelele de relații reprezintă cunoștințele ca un set de relații (sau predicate) care descriu modul în care entitățile (obiecte, concepte) sunt conectate sau interacționează. Această abordare este înrădăcinată în bazele de date relaționale și logica predicatelor, concentrându-se pe exprimarea faptelor și relațiilor într-un format structurat, tabelar sau logic. În sistemele expert, modelele de relații organizează cunoștințele din domeniu ca tupluri (înregistrări) care descriu entități și relațiile acestora, permițând interogarea, inferența și raționamentul.

Conceptele de bază ale modelelor de relații sunt următoarele:

1. Entități (obiecte) - componente din lumea reală, cum ar fi magistrale, transformatoare, sarcini.
2. Atribute - proprietăți ale entităților (de exemplu, nivelul tensiunii, capacitatea).
3. Relații - asocieri între entități, cum ar fi conectivitatea, proprietatea sau cauzalitatea.
4. Tupluri - rânduri care reprezintă instanțe ale relațiilor (de exemplu, (Magistra1, conectat_la, Magistra2)).
5. Schema - definiții ale relațiilor și tipurilor de atribute.

Utilizarea modelelor de relații în reprezentarea cunoștințelor în sistemele expert este motivată de următoarele aspecte: modelele de relații sunt o metodă de reprezentare clară și formală a cunoștințelor din domeniu ca fapte, ele facilitează interogarea structurată (similar cu interogările bazelor de date), modelele permit inferențe logice bazate pe relații, acestea susțin integrarea cu baze de date relaționale, adesea utilizate în sistemele energetice și ele pot exprima interdependențe complexe esențiale în analiza sistemelor energetice.

Avantajele și dezavantajele folosirii modelelor de relații sunt prezentate în figura 4.12.

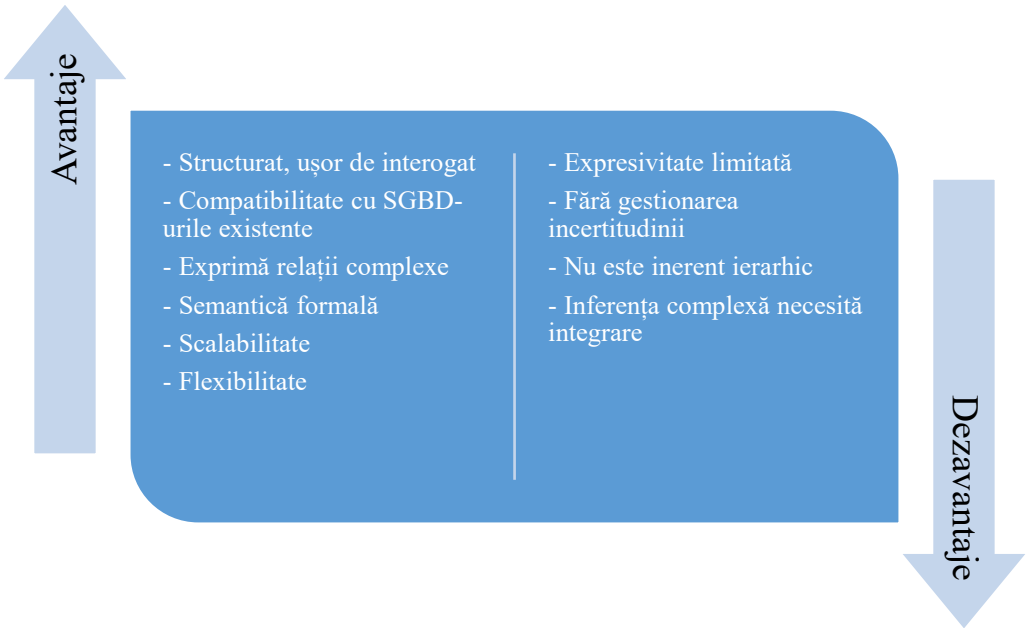


Fig. 4.12. Avantajele și dezavantajele modelelor de relații

Exemple de verificare a modelelor de relații folosite în sistemele expert dedicate sistemelor electroenergetice:

Găsiți toate magistralele conectate la Magistrala_A

SELECT magistrala2 FROM conectat_la WHERE magistrala1 = 'Magistrala_A'

Verificați transformatoarele supraîncărcate peste 90%

SELECT transformator FROM supraîncărcare WHERE valoare > 90

Identificați toate releele de protecție asociate cu transformatoarele supraîncărcate

SELECT releu FROM protejează WHERE transformator IN

(SELECT transformator FROM supraîncărcare WHERE valoare > 90)

Instrumentele și platformele informatice utilizate pentru implementarea modelelor de relații sunt Oracle, MySQL, PostgreSQL utilizat în utilități, programarea logică folosind Prolog cu predicate relaționale, tripletele RDF în Web-ul Semantic ca fapte relaționale, limbajul de interogare, Datalog pentru baze de date relaționale în sisteme expert și SPARQL.

În concluzie, modelele de relații oferă o modalitate fundamentală și structurată de a reprezenta cunoștințele statice și relaționale ale sistemelor energetice în sistemele expert. Deși sunt excelente pentru organizarea cunoștințelor factuale și permiterea interogărilor eficiente, acestea sunt adesea combinate cu alte tehnici de reprezentare a cunoștințelor pentru gestionarea cunoștințelor dinamice, incerte sau procedurale în sistemele expert dedicate sistemelor energetice.

4.20. Reprezentarea bazată pe baze de date.

Bazele de date sunt colecții structurate, concepute pentru a stoca, organiza și gestiona eficient volume mari de date și informații. În sistemele expert, bazele de date servesc drept strat de stocare a bazei de cunoștințe, conținând date factuale, istorice și operaționale utilizate pentru raționament și luarea deciziilor. Spre deosebire de metodele de reprezentare a cunoștințelor pur simbolice (cum ar fi regulile sau cadrele), bazele de date pun accent pe stocarea, recuperarea și integritatea datelor, interacționând adesea cu motorul de inferență a sistemului expert.

În dezvoltarea BC sunt folosite diferite tipuri de baze de date, și anume baze de date relaționale, baze de date de tip serie temporală, baze de date grafice, baze de date orientate pe obiecte, baze de date NoSQL și baze de date distribuite. Bazele de date relaționale stochează date structurale în tabele; ele fiind utilizate

pe scară largă pentru SCADA și alte date active. Bazele de date cu serii temporale sunt specializate pentru date temporale secvențiale de tipul măsurători PMU (Phasor Measurements Units). Topologia rețelei cu noduri și margini se înmagazinează în baze de date grafice. De exemplu, baza de date topologică va conține informații despre liniile electrice, transformatoare și conexiunea dintre acestea. Stocarea obiectelor complexe, împreună cu caracteristicile lor este realizată prin intermediul bazelor de date dedicate. Bazele de date distribuite sunt utilizate în rețele electrice smart cu surse de date dispersate geografic. Bazele de date NoSQL reprezintă o schemă flexibilă flexibilă pentru seturi de date semi-structurate sau mari.

Utilizarea bazelor de date în reprezentarea cunoștințelor în cadrul sistemelor expert este motivată prin următoarelor:

- Stocare și recuperare eficientă a seturilor de date mari și complexe.
- Suport pentru analiza datelor istorice pentru tendințe și diagnosticare.
- Oferirea accesului la date în timp real (fluxuri de date SCADA, PMU).
- Asigurarea consistenței și integrității datelor.
- Interfațarea cu motoarele de inferență pentru a furniza informații.
- Permiterea actualizărilor bazei de cunoștințe și învățarea din date noi.

Integrarea bazelor de date în cadrul sistemelor expert presupune anumite acțiuni și aspecte caracteristice. Astfel, bazele de date funcționează adesea ca stocare backend pentru baza de cunoștințe a sistemului expert. Toate datele factuale - cum ar fi specificațiile echipamentelor, topologia sistemului energetic sau datele istorice de funcționare - sunt stocate în formate structurate. Acest lucru permite motorului de inferență să acceseze informații consistente și actualizate atunci când aplică reguli sau strategii de raționament. Mai mult, în sistemele energetice moderne, monitorizarea în timp real este crucială. Bazele de date sunt frecvent conectate la sisteme SCADA, senzori și PMU pentru a colecta și stoca continuu date. Sistemul expert accesează aceste date în timp real pentru a evalua starea de sănătate a sistemului, a detecta anomalii sau a iniția acțiuni de control pe baza condițiilor actuale de funcționare. Bazele de date stochează, de asemenea, evenimente trecute, istoricul defecțiunilor sau intervențiile operatorilor, ceea ce le face resurse valoroase pentru sistemele expert care utilizează raționamentul bazat pe cazuri. Prin recuperarea și analizarea datelor istorice, sistemul poate identifica situații trecute similare și poate sugera soluții dovedite. Unele sisteme expert includ componente de învățare care se adaptează sau se îmbunătățesc în timp. Bazele de date pot stoca reguli nou învățate, ponderi

ajustate, feedback de la utilizatori sau modele de sistem actualizate. Acest lucru permite rafinarea continuă a sistemului expert fără reprogramare manuală. Interfețele cu utilizatorul pentru sistemele expert se bazează adesea pe baze de date pentru a afișa sau colecta informații. Operatorii pot interoga baza de date pentru a vizualiza starea sistemului, a introduce date noi sau a revizui rapoartele de diagnosticare. Integrarea asigură că toate fluxurile de date dintre interfață și sistemul expert rămân sincronizate. Integrarea bazelor de date cu sistemele expert din sistemele energetice asigură accesul rapid al sistemului la date precise, structurate și voluminoase. Această cuplare strânsă susține funcționarea în timp real, diagnosticarea, analiza defecțiunilor și învățarea adaptivă, făcând din bazele de date o componentă indispensabilă a sistemelor expert practice în domeniul energiei.

Avantajele și dezavantajele utilizării bazelor de date în reprezentarea cunoștințelor în cadrul sistemelor expert sunt sintetizate în figura 4.13 .

În concluzie, despre utilizarea bazelor de date în reprezentarea cunoștințelor se pot afirma următoarele - bazele de date reprezintă o fundație esențială pentru sistemele expert din domeniul energetic, oferind stocarea structurată, fiabilă și scalabilă a unor cantități vaste de date din domeniu. Atunci când sunt combinate cu motoarele de inferență și alte metode de reprezentare a cunoștințelor, acestea permit raționament eficient, diagnosticare și suport decizional.

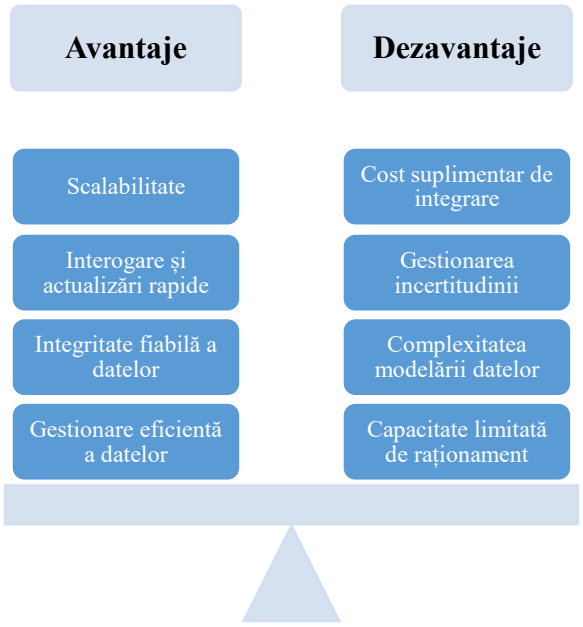


Fig. 4.13. Avantajele și dezavantajele bazelor de date

4.21. Întrebări și exerciții

Exercițiu

Formulaţi sub forma unor reguli de producţie următoarele enunţuri:

1. Un calculator utilizat în cercetarea ştiinţifică sau în proiectarea asistată trebuie să aibă un coprocesor matematic, o frecvenţă mare a ceasului şi un plotter.
2. La conectarea la reţea a unei centrale, se verifică dacă aceasta este în sincronism cu reţeaua, în caz contrar se sincronizează şi se conectează la sistemul electroenergetic.
3. În situaţia în care încărcările laturilor sunt prea mari, se va alege o metodă de reconfigurare a reţelei prin reducerea încărcărilor laturilor.

Întrebări

1. Baza de Cunoştinţe este compusă din:

Baza de reguli sau baza dinamică sau baza de fapte

2. Regulile din Baza de Cunoştinţe nu se pot modifica pe parcursul utilizării SE

Adevărat sau Fals □

3. Avantajele utilizării regulilor de producţie sunt

□ modularitate / □ caracter natural de exprimare / □ simplitate

2. Cea mai avansată metodă de reprezentare a cunoștințelor este

Sistemele de producție / Rețelele semantice / Programarea orientată pe obiecte

3. Cadrele Minski și rețelele semantice se aseamănă în modul de reprezentare a cunoștințelor.

Adevărat / Fals

5

Baza de cunoștințe
Reguli și fapte

Acest capitol continuă descrierea bazei de cunoștințe, mai exact prin prezentarea caracteristicilor bazei de fapte.

5.1. Aspecte generale

Baza de fapte reprezintă partea dinamică a BC; ea conţine informaţii relative cu privire la problema studiată. În multe SE, baza de fapte, datorită compoziţiei sale este privită ca o bază de date. De altfel, făcându-se analogie cu programele convenţionale, baza de fapte constituie datele de intrare ale problemei având în vedere că arată situaţia curentă a acesteia.

Faptele care compun baza de fapte sunt nişte date normale sau cantitative, care arată starea unei mărimi, care contează în procesul decizional, la un moment dat. Aceste fapte se modifică pe măsură ce stările obiectelor cărora le sunt asociate se schimbă.

Structura unui fapt este de forma [2]:

<Obiect>, <Relaţie>, <Valoare>

în care:

<Obiect> este obiectul a cărui stare poate fi stabilită printr-o semnalizare, verificare sau deducţie logică;

<Relaţie> corespunde unei relaţii între obiecte sau între obiecte şi diferite stări în care se poate afla acestea;

<Valoare> se referă la starea pe care o poate lua un obiect sursă.

Exemplu

<Întreruptorul I>, <Este în poziţia>, <Conectat>

Un SE poate să aibă acces la un bagaj mare de fapte referitoare la problema studiată, cantitate care depinde de domeniul de specialitate și de natura problemei. O parte din datele din baza de fapte sunt introduse direct de către utilizator cu ajutorul dialogului prin interfaţa cu utilizatorul, însă pe de altă parte aceste informaţii sunt obţinute şi încărcate automat prin citirea datelor direct din sistem. Datele citite pot să fie obţinute de la diferite echipamente de măsură, traductoare şi senzori care apoi sunt preluate de către SE prin intermediul Modulului de Achiziţie a Cunoștințelor și traduse în limbajul calculatorului.

5.2. Caracteristicile faptelor din electroenergetică

În electroenergetică, în funcţie de problema de rezolvat şi subdomeniul abordat faptele pot să fie de tipul închis-deschis pentru un echipament de comutaţie, o alarmă de ieşire din funcţiune pentru o instalaţie, o valoare sau o mărime electrică care a ieşit din limitele admisibile. Sistemele electroenergetice sunt caracterizate de un mare dinamism, de aceea este foarte important ca baza de fapte să fie mereu actualizată, chiar și în timp real în cazul SE de control și mentenanţă a echipamentelor şi instalaţiilor.

Caracteristicile faptelor folosite de sistemele expert dedicate sistemelor energetice sunt următoarele:

- Dinamismul - faptele pot fi actualizate frecvent pe baza măsurătorilor, a intrărilor operatorului sau a fluxurilor de date externe. Acest dinamism depinde de tipul problemei de rezolvat, dacă este sistem de control, clasificare sau prognoză. Astfel, în cazul sistemelor expert de control, faptele prezintă un dinamism mult mai accentuat decât în cazul sistemelor expert de clasificare.
- Structuralitatea – faptele sunt stocate în formate structurate, cum ar fi perechi atribut-valoare, tabele sau modele orientate pe obiecte. Din nou, modul în care sunt structurate faptele depinde de tipul sistemului expert.
- Accesibilitatea - motorul de inferență trebuie să poată interoga și actualiza rapid faptele.
- Actualizarea - faptele noi pot fi deduse din reguli sau inserate din intrările senzorilor/operatorului.
- Sensibilitatea la conflicte - motorul de inferență poate verifica dacă există fapte contradictorii sau inconsecvențe.

5.3. Funcțiile faptelor în baza de cunoștință

În cadrul sistemelor expert și a bazei de cunoștințe, baza de fapte îndeplinește anume funcții clar definite.

Funcțiile bazei de fapte în sistemele expert dedicate problemelor caracteristice sistemelor energetice sunt:

1. Permite raționamentul

   Oferă partea „DACĂ" a regulilor DACĂ-ATUNCI: Motorul de inferență potrivește condițiile din reguli cu faptele.

2. Facilitează monitorizarea

   Sistemele expert poate monitoriza starea sistemului prin compararea constantă a faptelor cu pragurile normale de funcționare.

3. Acceptă diagnosticarea

   În caz de alarme sau defecțiuni, sistemul analizează modelele faptice (de exemplu, căderi de curent/tensiune) pentru a deduce cauza.

4. Stimulează luarea deciziilor

   Pentru restaurare sau reconfigurare: Pe baza stării actuale a sistemului (fapte), regulile determină cea mai bună acțiune.

5. Ajută la învățare (în sistemele expert adaptive)

   În unele sisteme, baza de fapte stochează noi modele de date pentru raționament bazat pe cazuri sau module de învățare automată.

În tabelul 5.1 se dau exemple de fapte din diferite subdomenii ale electroenergeticii.

Tabelul 5.1. Exemple de cunoștințe cuprinse în baza de fapte a BC unui SE

Cunoştinţă	**Fapt**	**Observaţii**
Se măsoară şi transmite valoarea curentului din sub – reţea	Curentul I are valoarea de 250 A	Domeniul dispozitivelor de protecţie
Senzorul ataşat nivelului maxim admis în lacul de acumulare a indicat o depăşire a acestuia	Senzorul ataşat nivelul admisibil a apei are valoarea Adevărat	Domeniul generării energiei electrice

Se măsoară tensiunea pe tronsonul 1-2 a reţelei, aceasta indică valoarea de 370 V	Tensiunea $U_{1\text{-}2}$ are valoarea de 370 V	Domeniul transportului energiei electrice (menţinerea tensiunii în limitele admisibile)
În urma analizei compoziţie uleiului din sistemul de răcire al transformatorului s-a identificat substanţa X în concentraţie de x%	În uleiul de răcire al transformatorului subsanţa X este în concentraţie de x%	Domeniul mentenaţei sistemului electroenergetic

Provocările care apar în manipularea faptelor caracteristice sistemelor energetice sunt următoarele:

- Volumul de date - gestionarea eficientă a unei cantități mari de date în timp real.
- Inconsistența - gestionarea unor fapte contradictorii din cauza erorilor senzorilor sau a întârzierilor de comunicare.
- Scalabilitatea - baza de fapte trebuie să se scaleze în funcție de dimensiunea și complexitatea grilei.
- Validitatea temporală - faptele trebuie adesea marcate temporal și să expire după o fereastră de timp.

În concluzie, baza de fapte dintr-un sistem expert pentru sisteme energetice este depozitul de date statice și în timp real care reprezintă starea actuală a rețelei electrice. Aceasta interacționează direct cu baza de reguli prin intermediul motorului de inferență pentru a sprijini raționamentul inteligent, diagnosticarea, detectarea defecțiunilor și deciziile operaționale. Corectitudinea, promptitudinea și caracterul complet al acesteia sunt esențiale pentru eficacitatea sistemului expert.

5.4. Regulile de inferență

În contextul sistemelor expert dedicate sistemelor energetice, baza de reguli de inferență, adesea numită pur și simplu bază de reguli, este o componentă centrală a bazei de cunoștințe. Astfel, baza de reguli codifică expertiza specifică

domeniului problemei de rezolvat sub formă de afirmații logice, permițând sistemului expert să obțină informații sau decizii noi din faptele cunoscute.

Baza de reguli de inferență este o colecție structurată de reguli de producție (cunoscute și sub denumirea de reguli DACĂ - ATUNCI, reguli condiție-acțiune sau clauze Horn) care reprezintă cunoștințele procedurale ale sistemului expert.

DACĂ (condiție) este comparată cu baza de fapte.

ATUNCI (acțiune/concluzie) este executată sau afirmată atunci când condiția este adevărată.

Observație

Forma unei reguli de inferență este aceeași ca o regulă de producție doar atunci când reprezentarea cunoștințelor în sine este construită cu reguli de producție. În alte tipuri de reprezentare a cunoștințelor (rețele semantice, logică, cadre, ontologii), regulile de inferență sunt exprimate diferit - nu ca reguli de acțiune DACĂ-ATUNCI, ci ca mecanisme logice, structurale sau relaționale.

Structura unei reguli de inferență în cazul logicii formale este:

$P \rightarrow Q$	(dacă P, atunci Q)
P	(fapt)
$\therefore Q$	(concluzie, de Modus Ponens)

În cazul rețelelor semantice, inferența se realizează prin moștenire, tranzitivitate sau traversare a relației.

Dacă (X este un Y) și (Y are proprietatea Z), atunci (X are proprietatea Z)

În cazul cadrelor Minski, inferența este bazată pe sloturi și structurală.

Dacă un cadru „Pompă” moștenește de la „Mașină” și „Mașina” are o putere nominală implicită de 10 kW, atunci, dacă nu se specifică altfel, și pompa are 10 kW.

Structura unei inferențe în cazul folosirii ontologiilor este bazată pe subclase, constrângeri de proprietăți și axiome logice.

Dacă toate întrerupătoarele de circuit sunt dispozitive

și X este un întrerupător de circuit

→ atunci X este un dispozitiv

Baza de reguli este partea bazei de cunoștințe responsabilă pentru furnizarea logicii pentru raționamentul deductiv: transformarea faptelor cunoscute în fapte

noi, acțiuni, alarme, diagnostice sau sugestii noi. Baza de reguli conține euristici și strategii care ghidează luarea deciziilor.

Regulile de inferență sunt :

1. Declarative – acestea capturează cunoștințele expertului sub formă de instrucțiuni logice de tip „dacă-atunci”.
2. Modulare - acestea sunt de obicei independente și pot fi actualizate individual.
3. Transparente – acestea sunt ușor de înțeles, auditat și explicat (spre deosebire de modelele de tip „cutie neagră”).
4. Executabile – ele sunt utilizabile direct de motorul de inferență pentru luarea deciziilor.
5. Extensibile - regulile noi pot fi adăugate fără a reprograma întregul sistem.

Regulile de inferență care intră în componența bazei de reguli a bazei de cunoștințe a sistemelor expert pot fi reguli deterministe, reguli probabilistice și reguli de rezolvare a conflictelor. Regulile deterministe conduc întotdeauna la aceeași concluzie dacă condiția este îndeplinită.

Exemplu

DACĂ curentul de linie > 600 A ATUNCI alarmă de suprasarcină

Regulile probabilistice (cu factori de certitudine) se utilizează atunci când cunoștințele sunt incerte sau aproximative.

Exemplu

DACĂ nivelul vibrațiilor este moderat ATUNCI (70% șanse de defecțiune a rulmentului)

Regulile de rezolvare a conflictelor definesc prioritățile atunci când mai multe reguli sunt declanșate simultan.

Exemplu

Preferă regulile care afectează siguranța față de cele care afectează costul

Provocările și considerațiile care trebuie avute în vedere în dezvoltarea bazei de reguli se referă la conflictul dintre reguli (mai multe reguli se pot declanșa simultan; este necesară o strategie de rezolvare a conflictelor), redundanța

regulilor (regulile suprapuse pot face sistemul ineficient sau imprevizibil), completitudinea regulilor (lipsa regulilor poate duce la eșecul acoperirii situațiilor critice) și întreținerea regulilor (menținerea setului de reguli la zi cu configurațiile și reglementările grilei în evoluție).

În concluzie, baza de reguli de inferență este inima mecanismului de raționament din sistemele expert. În sistemele energetice, aceasta codifică cunoștințele experte și logica operațională ca un set de reguli DACĂ-ATUNCI. Aceste reguli permit sistemului expert să diagnosticheze defecțiunile, să controleze tensiunea, să restabilească sistemele și să asiste operatorii în gestionarea rapidă și precisă a comportamentelor complexe ale rețelei.

5.5. Întrebări și exerciții

1. Modul de reprezentare a cunoştinţelor depinde de complexitatea, natura şi domeniul problemei de soluţionat.
□ Adevărat / □ Fals

2. Cadrele Minsky au la bază atât elemente ale sistemelor de producţie, cât şi a reţelelor semantice
Adevărat □ / Fals □

3. Starea problemei la un moment dat poate fi obţinută din
□ Baza de fapte / □ Baza de reguli / □ Modulul de interfaţă cu utilizatorul

6

Motorul de inferențe

Prezentul capitol prezintă caracteristicile generale, rolul şi funcţionarea Motorului de Inferenţă.

6.1. Introducere

Motorul de Inferenţă (MI) în esenţa lui este un sistem rezolutiv care se bazează pe strategii de control, algoritmi de căutare și alte metode mai mult sau mai puţin formale pentru a soluţiona problema. Astfel, înţelegerea acestei componente a unui SE presupune analiza strategiilor de control și căutare, precum şi a celorlalte aspecte care se întâlnesc în dezvoltarea lui. În consecinţă, prezentarea MI a fost divizată în două părţi: prima parte este cuprinsă în structura acestui capitol şi următorul, și conţine descrierea strategiilor de control, respectiv a modului în care trebuie alese acestea; a doua parte este tema capitolelor următoare, în care se discută despre algoritmii de căutare și metodele de rezolvare a conflictelor dintre regulile selectate.

Un sistem rezolutiv conţine totalitatea componentelor unui sistem de Inteligenţă Artificială având ca obiectiv rezolvarea problemei studiate [1]. Pe de altă parte, obiectivul rezolvării oricărei probleme este de a furniza o determinarea pentru fiecare din necunoscutele acesteia. Procesul rezolvării unei probleme începe de la datele și condiţiile formulate prin enunţ. La sistemele expert bazate pe cunoaşterea universului iniţial, i se adaugă piese de cunoaştere din baza de cunoștințe, care sunt selectate fie prin referire directă la enunţ, fie prin inferenţe logice cu premise formulate în enunţ.

Legăturile logice care se formează între premise și concluzii determină procesul care se numeşte în literatura de specialitate şi raţionament (reasoning), care se bazează pe faptele furnizate prin enunţ și pe cunoștințele existente în BC.

Motorul de Inferenţă numit şi Mecanismul de Inferenţă, este cel care dă funcţionalitate unui SE, și se bazează pe inferenţe. Conform definiţiei din DEX,

inferenţa este o operaţie logică de trecere de la un enunţ la altul şi în care ultimul enunţ este dedus din primul, adică este procesul prin care se trage o concluzie. Într-un sens mai restrâns, privită ca un caz particular al unui algoritm de căutare, inferenţa este un proces de deducţie care porneşte de la condiţiile iniţiale sau finale şi, prin aplicarea succesivă a unor reguli, ajunge la starea dorită [2].

Din punct de vedere informatic, MI este un program general care implementează mecanismul prin care se construiesc raţionamentele. În figura 1 este ilustrată structura unui SE, accentuându-se MI și legăturile acestuia cu celelalte componente ale SE.

Motorul de Inferenţă are legături bidirecţionale cu Modulul Explicativ și Modulul Dinamic. Cu primul dintre acestea, Modulul Explicativ, comunică permanent pentru a putea furniza utilizatorului raţionamentele după care s-a rezolvat problema, utilizatorul la rândul lui putând să dea instrucţiuni SE şi să influenţeze MI într-un grad mai mare sau mai mic în funcţie de modul în care a fost dezvoltat SE. Modulul Dinamic este componenta care înmagazinează datele intermediare apărute în etapele de rezolvare a problemei studiate, astfel că trebuie să existe mereu o legătură între acesta și MI. O legătura unidirecţională este între MI și BC, şi anume despre BC spre MI. Explicaţia acestei legături este următoarea: BC transmite MI regulile din baza de reguli şi datele iniţiale prin intermediul bazei de fapte, date pe care MI le procesează pentru a se atinge obiectivul; MI nu are nici o influenţă asupra baze de reguli, deci nu poate să modifice BC.

În următoarele subcapitole se prezintă rolul MI și componenţa lui în ceea ce priveşte strategiile de căutare şi algoritmii de căutare care poate să îi conţină; modul în care funcţionează MI și modalităţile în care poate fi soluţionată o problemă.

6.2. Rolul Motorului de Inferenţă

Motorul de Inferenţă este inima oricărui SE, care alimentat de BC construieşte raţionamente în mod dinamic decizând ce reguli trebuie declanşate și în ce ordine.

Motorul de inferenţă prelucrează cunoștințele din baza de reguli şi fapte folosind diferite procedee. El constituie mecanismul de raţionament având ca sarcină exploatarea regulilor şi generarea răspunsurilor la întrebările puse de utilizatori.

MI este elementul efectiv de prelucrare a SE, care pornind de la fapte, acţionează reguli corespunzătoare din BC, efectuând asociaţiile și legăturile necesare într-o manieră ce duce la soluţia problemei puse [3].

Rolul MI în cadrul SE este de a da sens regulilor de inferenţă, legându-le între ele, astfel încât să fie puse întrebările necesare pentru a obţine răspunsurile corecte.

Motorul de Inferenţă conţine o strategie de control, care este aleasă în funcţie de tipul problemei de studiat. Acestei strategii de control, i se adaugă unul sau mai mulți algoritmi de căutare (tehnici de căutare), care parcurg BC și combină regulile pentru a găsi soluţia optimă. În acest context, rolul MI este de a controla strategia de control și algoritmii de căutare pentru a manipula eficient cunoştinţele din BC. O sinteză a celor expuse în acest subcapitol este ilustrată în figura 6.1.

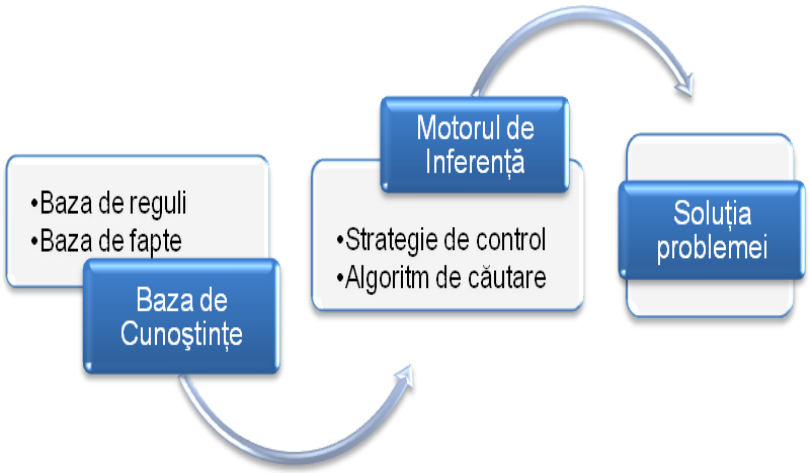


Fig. 6.1 Motorului de Inferenţă în cadrul unui Sistem Expert

În concluzie, MI aplică raționament logic asupra faptelor și regulilor din baza de cunoștințe, conduce procesul decizional în sistemul expert, potrivește datele de intrare (faptele) cu reguli pentru a genera concluzii, recomandări sau diagnostice și controlează fluxul raționamentului folosind fie strategia de control înainte (bazată pe date), fie strategia de control înapoi (bazată pe obiective).

6.3. Funcţionarea Motorului de Inferenţă

Funcţionarea MI este exemplificată în figura 6.2, unde se poate observa:

- Sistemul de achiziţie de date transmite SE informaţiile care ajung în baza de fapte;
- Utilizatorul poate să transmită la un moment dat informaţii legate de situaţia problemei la un moment dat;
- MI primeşte din baza de fapte datele privind starea problemei la un moment dat;
- MI accesează baza de reguli de unde selectează regulile aplicabile pentru situaţia problemei folosind strategiile de control și tehnicile de căutare corespunzătoare;
- MI foloseşte o metodă de selecţie pentru a rezolva conflictul dintre regulile aplicabile, în final alegând o regulă;
- Regula aleasă determină alte acţiuni, respectiv schimbă starea bazei de fapte, prin adăugarea noii stări a problemei.

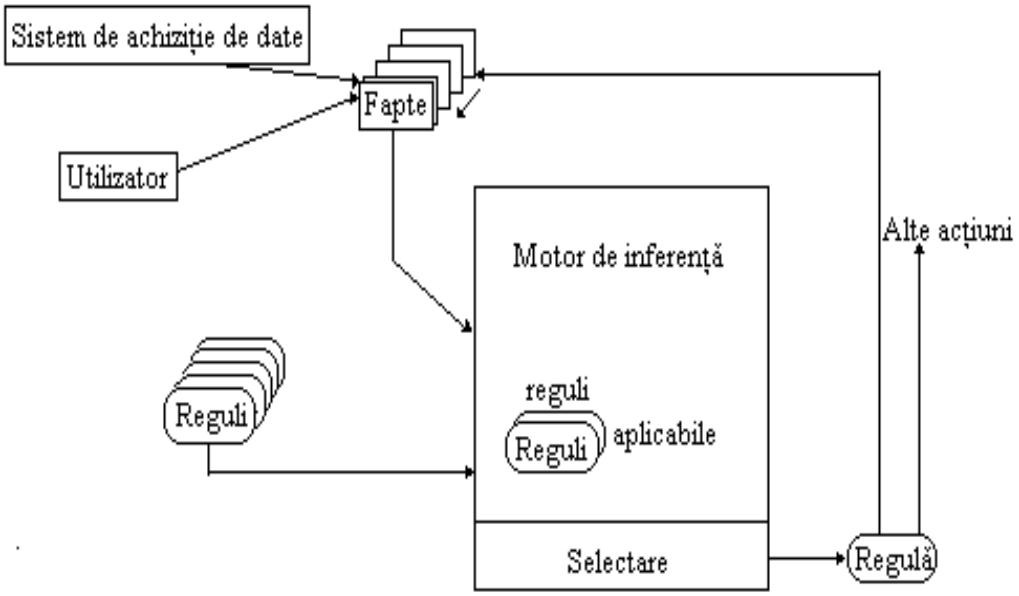


Fig. 6.2 Funcţionarea Motorului de Inferenţă [3]

Funcţionarea MI poate fi descrisă şi în alt mod, precum apare în [2]. Acest mod este expus în figura 6.3; el presupunând patru activităţi efectuate de MI:

- Suprapunerea bazei de reguli peste baza de fapte pentru identificarea regulilor aplicabile

Aceasta este etapa de potrivire a tiparelor, în care motorul de inferență examinează baza de fapte (memoria de lucru) și caută reguli în baza de reguli ale căror condiții (părți DACĂ) sunt îndeplinite de faptele actuale. Compară sistematic antecedentele fiecărei reguli cu datele cunoscute pentru a detecta care reguli sunt aplicabile în prezent, adică eligibile pentru execuție. Acest pas determină ce reguli logice sunt candidate pentru raționament în contextul actual.

- Selectarea regulilor care se aplică
  Dacă se constată că se aplică mai multe reguli, motorul de inferență folosește o strategie de rezolvare a conflictelor pentru a determina care regulă (reguli) ar trebui declanșate primele. Pot fi utilizate mai multe criterii: specificitatea (regulile mai specifice pot avea prioritate), prioritatea sau ponderea atribuită regulilor și noutatea sau caracterul recent al faptelor. Ordinea regulilor sau valorile de importanță - aceasta asigură că raționamentul urmează o progresie controlată și logică și evită declanșarea arbitrară a regulilor.
- Aplicarea regulilor
  Odată ce o regulă este selectată, motorul de inferență execută partea de acțiune a regulii (partea ATUNCI). Aceasta duce de obicei la adăugarea de noi fapte la baza de fapte, modificarea sau eliminarea faptelor existente și declanșarea de acțiuni, cum ar fi furnizarea unei recomandări, a unui diagnostic sau a unei decizii. Aceasta este faza în care sistemul își dezvoltă dinamic cunoștințele prin actualizarea stării sale interne.
- Verificarea criteriului de oprire
  După aplicarea unei reguli, motorul de inferență verifică dacă procesul de raționament ar trebui să continue sau să se oprească. Criteriile de oprire pot include: atingerea unui obiectiv (de exemplu, sistemul a găsit o soluție), nu mai sunt aplicabile reguli, executarea unui număr predefinit de cicluri și atingerea unui prag de încredere (în sistemele probabilistice sau fuzzy). Acest pas asigură că procesul de inferență nu rulează la nesfârșit și se oprește atunci când raționamentul s-a finalizat cu succes sau a ajuns la un punct mort logic.

Aceste patru activități permit mecanismului de inferență să reproducă raționamentul uman, ghidând sistemul expert printr-un ciclu structurat de recunoaștere a cunoștințelor relevante, aplicarea lor semnificativă și determinarea momentului în care s-a ajuns la o concluzie satisfăcătoare.

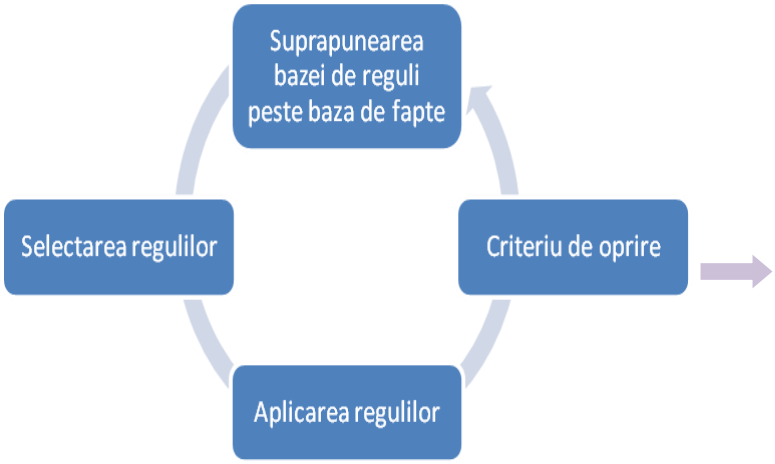


Fig. 6.3 Funcţionarea Motorului de Inferenţă [2]

6.4. Ciclul de bază al Motorului de Inferenţă

Ciclul de bază al MI presupune realizarea activităţilor din figura 6.4: selecţia, filtrajul, rezolvarea conflictelor și execuţia. Tot în această figura se observă legătura dintre etape şi rezultatele obţinute la sfârşitul fiecăreia. În cele ce urmează se descriu fiecare etapă a ciclului de bază.

Selecţia

Selecţia este etapa în care se extrag din Baza de Cunoștințe acele reguli şi fapte care sunt în legătură cu subdomeniul de rezolvare a problemei. Adică se constituie o sub-mulţime pornind de la baza de reguli şi fapte iniţiale pentru a scurta timpul de căutare din următoarele etape. În această etapă trebuie avut în vedere faptul ca MI să nu creeze privilegii pentru anumite reguli şi fapte, lucru care ar putea să elimine posibile soluţii.

Etapa de selecţie este justificată și necesară în cazul unei BC mari, cu multe informaţii, care cuprinde date din mai multe domenii de cunoaştere.

Precum se observă în figura 6.4, în etapa de selecţie se iau regulile din baza de reguli şi sunt luate regulile posibile, iar din baza de fapte sunt puse în sub-mulţimea corespunzătoare faptele selecţionate.

Filtrajul

Regulile posibile şi faptele selecţionate care rezultă din etapa de selecţie, ajung în etapa de filtraj, unde suferă mai multe acţiuni, după care sunt transmise mai departe regulile declanşabile (aplicabile).

Etapa de filtraj este denumită şi etapa de compararea cu modelul (pattern matching), deoarece în această etapă regulile posibile (mai specific premisele regulilor) sunt comparate cu faptele ce caracterizează problema de rezolvat pentru a rezulta regulile aplicabile. În urma acestei etape este redus substanţial numărul regulilor din baza de reguli, putând să fie selectate una, mai multe sau nici o regulă. Dacă nu rezultă nici o regulă, atunci este o situaţie de „eşec", pe care SE trebuie să o semnaleze sau în care utilizatorul trebuie să răspundă la o serie de întrebări pentru a se completa datele problemei şi a se continua procesul de soluţionare a acesteia. În cazul când sunt mai multe reguli aplicabile, atunci se trece la etapa de rezolvare a conflictelor.

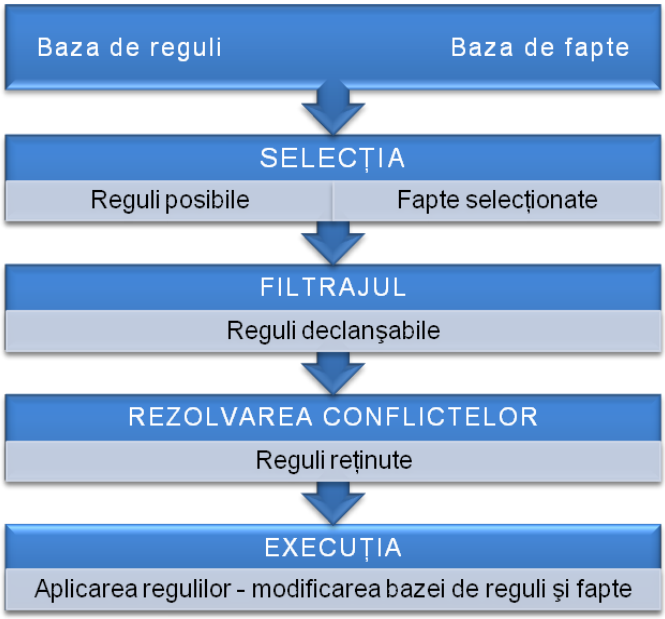


Fig. 6.4 Schema logică a ciclului de bază a Motorului de Inferenţă

Rezolvarea conflictelor

Etapa de rezolvare a conflictelor este necesară atunci când după etapa de filtraj au fost alese mai multe reguli aplicabile. Regula aleasă este cea care va fi executată, astfel că trebuie acordată foarte mare atenţie acestei etape, mai ales, că rapiditatea de găsire a soluţiei optime depinde în foarte mare măsură de aceasta.

Criteriile de alegere a regulii de executat sunt foarte variate, dintre acestea cele mai populare sunt: prima regulă din mulţimea regulilor aplicabile, regula cea

mai specializată, regula cea mai productivă, regula cea mai utilizată, și nu în ultimul rând metodele euristice. În acest subcapitol nu se va insista pe o descriere a criteriilor de rezolvare a conflictelor, deoarece este un capitol dedicat acestui subiect. Trebuie însă subliniat faptul că, de multe ori este dificil să se precizeze cu exactitate care criteriu trebuie ales, deoarece alegerea este condiţionată de starea problemei la un moment dat şi de meta-reguli (adică acele date care ne spun cum au fost construite regulile și cum funcţionează ele).

Execuţia

Etapa de execuţie este ultima a ciclului de bază, şi constă în aplicarea regulii alese. Consecinţa acestei etape este adăugarea în baza de fapte a unei sau mai multor fapte care schimbă starea problemei aducând-o mai aproape de starea finală. Există situaţii când aplicarea unei reguli semnifică [3]:

- Alegerea altei sau altor reguli;
- Apelarea unor proceduri externe SE (de obicei, baze de date externe sau procesoare de tabele);
- Un dialog cu utilizatorul prin care sunt puse întrebări acestuia pentru a rezolva o situaţie de eşec.

În procesul de rezolvare a unei probleme, MI execută mai multe cicluri de bază, până la realizarea criteriului de oprire.

Motoarele de inferență sunt deterministe, adică utilizează reguli logice stricte (adevărat/fals), comune în sistemele clasice bazate pe reguli, și incerte/probabilistice, întrucât ele utilizează tehnici precum factorii de certitudine, inferența bayesiană sau logica fuzzy pentru a raționa în condiții de incertitudine - important în sistemele expert din lumea reală.

Un motor de inferență bun poate explica de ce a pus o întrebare (urmărire inversă), poate arăta cum a ajuns la o decizie (urmărire directă) și poate crește transparența și încrederea utilizatorilor.

În sistemele expert de control (de exemplu, sistemele energetice), motorul de inferență poate necesita raționament în timp real, gestionarea întreruperilor și activarea regulilor declanșate de evenimente.

6.5. Întrebări și exerciții

1. Motorul de Inferenţă comunică cu:

□Modulul Explicativ / □Interfaţa Grafică cu Utilizatorul / □Baza de Cunoştinţe

2. În rezolvarea problemelor Motorul de Inferenţă efectuează un singur ciclu de bază.

Adevărat □ / Fals □

3. Etapele din ciclul de bază a unui MI care pot fi omise sunt

Selecția / Execuția / Rezolvarea conflictelor / Filtrajul

7

Motorul de inferențe
Strategii de control

Prezentul capitol cuprinde descrierea Motorului de inferență și a strategiilor de control aferente.

7.1. Aspecte generale

Motorul de Inferenţă are nevoie de o strategie de control după care să fie parcursă BC, deoarece fără o strategie adecvată, un SE ar avea nevoie de timpi mari de lucru pentru a soluţiona o problemă.

Strategia de control reprezintă un set de norme care ghidează căutarea regulilor, care pot fi aplicabile pentru rezolvarea unei probleme. Strategiile de control folosite, care se bazează pe raţionamentul direct (sunt furnizate suficiente cunoștințe inițiale în BC pentru a soluţiona problema), și care sunt descrise în prezentul capitol sunt:

- Strategia de control înainte - o abordare bazată pe date, în care raționamentul pornește de la fapte cunoscute și aplică reguli pentru a deduce fapte noi până când se atinge un obiectiv.
- Strategia de control înapoi – o abordare bazată pe obiective, care începe cu o ipoteză și funcționează invers prin reguli pentru a găsi fapte justificative.
- Strategia de control mixt - o metodă flexibilă de raționament care combină strategii directe și inverse, comutând între ele după cum este necesar, în funcție de context.
- Strategia de control circumstanţial - o abordare dinamică în care strategia de control este selectată sau ajustată în timp real, pe baza condițiilor specifice ale problemei sau a stării sistemului.

Regulile cuprinse în BC pot să fie reprezentate sub forma unor arbori de inferenţă care au ca noduri premisele (condiţiile), iar ca ramuri arce care conectează diferitele premise. Arcele pot să fie ramuri de tip ŞI, respectiv SAU în funcţie de operatorul logic care se află între diversele reguli.

Procedeul de inferenţă poate fi vizualizat ca un drum de-a lungul arcelor arborelui, în acest fel strategiile de control aplicabile sunt mai uşor de înţeles și urmărit. Trebuie precizat faptul că pentru a parcurge un arc de tip ȘI trebuie ca toate arcele parcurse înaintea lui să fi fost tot de tip ŞI.

7.2. Strategia de control înainte

Strategia de control înainte, numită și modul de raţionament deductiv sau raționament de înlănțuire directă (figura 7.1) este o tehnică de inferenţă care se poate aplica atât regulilor (cea mai utilizată variantă), dar și grafurilor. Această strategie porneşte de la starea iniţială de fapte, care este descrisă la începutul prin enunţ și prin aplicarea de raţionamente se generează noi fapte până la obţinerea răspunsului corespunzător obiectivului problemei.

Această strategia de control foloseşte o metodologie de parcurgere a datelor de la fapte înspre obiectiv, din această cauză este aplicabilă în cazul problemelor la care se cunosc foarte bine datele de intrare.

În momentul parcurgerii bazei de fapte, și în etapa de selectare a regulilor aplicabile, vor fi alese acele reguli pentru care faptele din enunţ se află în corpul premiselor regulilor. Pentru faza de rezolvare a conflictelor se pot aplica mai multe criterii (care sunt prezentate într-un capitol următor dedicat), în funcţie de starea problemei la acel moment.

> Criteriul de rezolvare a conflictelor trebuie ales cu atenţie în funcţie de starea problemei la momentul respectiv, deoarece influenţează performanţele SE în găsirea rapidă a soluţiei optime.

Eficienţa strategiei de control înainte depinde de numărul de operaţii care le face MI, astfel este de dorit scăderea numărului acestora prin selectarea unui criteriu de rezolvare a conflictelor adecvat.

Prin aplicarea regulilor selectate apar date şi fapte noi care sunt introduse în baza de fapte, și prin urmare se schimbă starea problemei. Procesul de căutare se va încheia în momentul în care s-a parcurs toată baza de reguli aplicabile. Se consideră “succes” momentul când scopul este adăugat la baza de fapte, iar

“eşec” când nu se mai poate aplica nici o regulă. Oprirea căutării are loc în momentul în care în baza de fapte apare scopul urmărit.

În figura 7.1. este ilustrată schema logică a strategiei de control înainte.

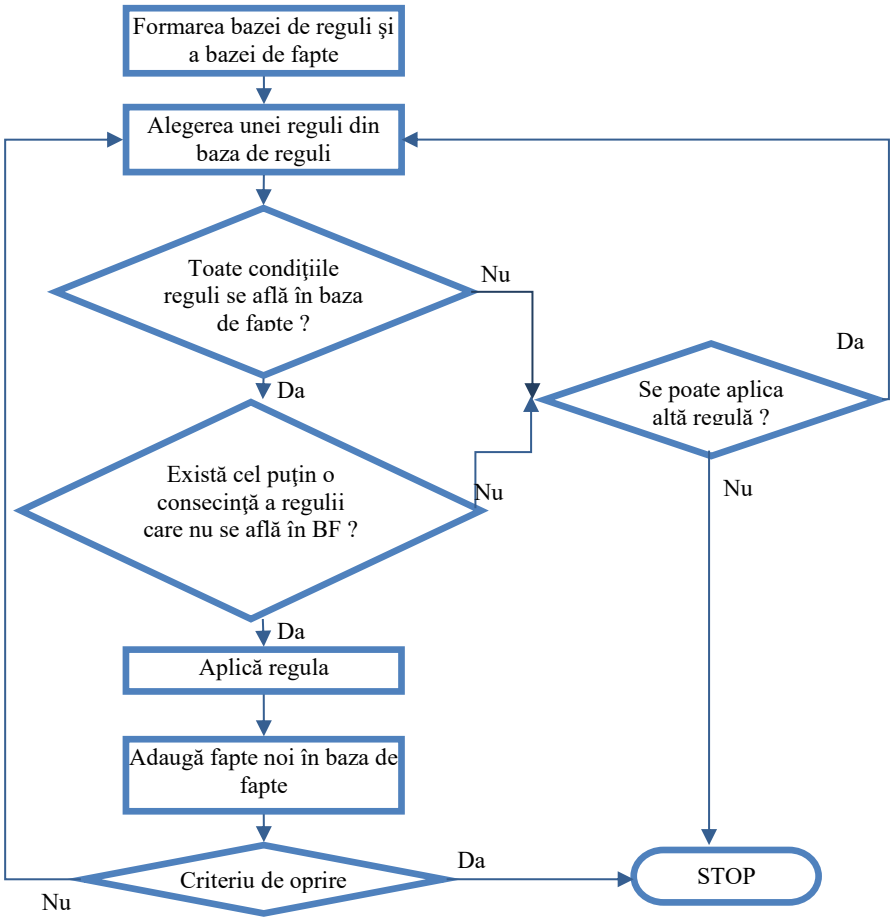


Fig. 7.1 Schema logică a strategiei de control înainte [2]

Exemplu

Se consideră o BC reprezentată prin următoarele reguli de producţie:

Regulile R1...R9:

R1: dacă B și D și E atunci F

R2: dacă D şi G atunci A

R3: dacă C și F atunci A

R4: dacă B atunci X

R5: dacă D atunci E
R6: dacă A şi X atunci H
R7: dacă C atunci D
R8: dacă X şi C atunci A
R9: dacă X și B atunci D

Semnificaţia regulii R1 este: dacă B și D și E sunt adevărate atunci reiese F.

Faptul de început este caracterizat prin cunoaşterea lui B și C, iar obiectivul este H.

Prin aplicarea strategiei de control înainte, se porneşte de la premisele care conţin cunoscutele. În exemplul dat, se va folosi ca criteriu de rezolvarea a conflictelor, prima regulă aplicabilă în ordinea în care apar ele. Astfel, regulile selectate la fiecare etapă sunt:

1. R4 și R7, se alege R4, după care se cunoaşte X;
2. R7, R8 și R9, se alege R7, după care se cunoaşte D;
3. R5, R8 și R9, se alege R5, după care se cunoaşte E;
4. R1, R8 și R9, se alege R1, după care se cunoaşte F;
5. R3, R8 și R9, se alege R3, după care se cunoaşte A;
6. R6, R7 și R8 se alege R6 a cărui acţiune este H, obiectivul problemei.

Considerând faptul că regulile au o structură arborescentă (există o rădăcină şi ramuri, depind unele de altele), deducerea rezultatului poate fi reprezentată prin schema din figura 7.

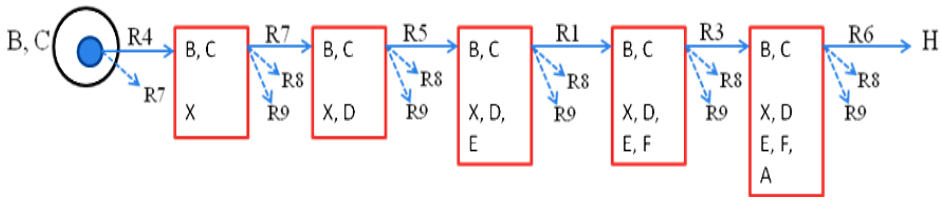


Fig. 7.2 Reprezentarea strategiei de control înainte

Strategia de control înainte are două mari avantaje, care sunt necesare în unele aplicaţii:

1. Dă posibilitatea generării tuturor soluţiilor posibile;

2. Îmbogăţeşte la fiecare ciclu al MI baza de fapte, ceea ce simplifică deducţiile ulterioare.

Marele dezavantaj al acestei strategii de control este faptul că există posibilitatea ca să nu se găsească soluţia optimă a problemei, ci doar o soluţie posibilă.

7.3. Strategia de control înapoi

Strategia de control înapoi, care apare în literatura de specialitate şi ca modul de raţionament inductiv (raționament de înlănțuire inversă), porneşte de la obiectivul problemei care prin aplicarea regulilor de descompunere se transformă în sub-probleme de complexitate mai mică [1]. În consecinţă, strategia de control înapoi utilizează un raţionament centrat pe scop: se începe de la scop, şi se caută faptele care permit atingerea acestuia.

Această strategie se aplică atât sistemelor care folosesc reguli, cât şi celor care folosesc graf-uri şi arbori. În cazul sistemelor bazate pe reguli, în timpul procesului de selecţie a regulilor, la un anumit pas sunt alese acelea care au în concluzii scopul specificat iniţial. Procesul se repetă până toate sub-scopurile obţinute sunt demonstrate, sau în urma etapei de filtraj mulţimea regulilor declanşabile este vidă, adică este "eşec".

Diferenţa dintre strategia de control înainte și strategia de control înapoi constă în faptul că, în cazul primei strategii baza de fapte este modificată, iar starea problemei se schimbă în funcţie de noile fapte. Prin contrast strategia de control înapoi lasă nealterată baza de fapte iniţială, deoarece foloseşte doar baza de reguli, și anume consecinţele regulilor.

Principiul care stă la baza strategiei de control înapoi este următorul [3]:
Fiind dată o stare iniţială sau un obiectiv care trebuie atins, se analizează toate regulile ale căror consecinţe coincid cu acel obiectiv şi se reţin condiţiile corespunzătoare acestor reguli, care formează noi obiective intermediare.

Pentru a se înţelege mai bine modul de funcţionare a strategiei de control înapoi s-a introdus exemplul următor.

Exemplu

Se consideră o BC reprezentată prin următoarele reguli de producţie:

Regulile R1...R9:

R1: dacă B și D și E atunci F
R2: dacă D şi G atunci A
R3: dacă C și F atunci A
R4: dacă B atunci X
R5: dacă D atunci E
R6: dacă A şi X atunci H
R7: dacă C atunci D
R8: dacă X şi C atunci A
R9: dacă X și B atunci D

Datele iniţiale sunt B şi C, iar scopul este H.

În exemplu dat se poate observa că procesul se încheie în momentul când toate condiţiile unei dintre regulile analizate la un moment dat se regăsesc în baza de fapte iniţială a problemei.

Metodologia de funcţionare a MI în cazul strategiei de control înapoi poate fi reprezentată sub forma unui algoritm care are următoarea sintaxă [3]:

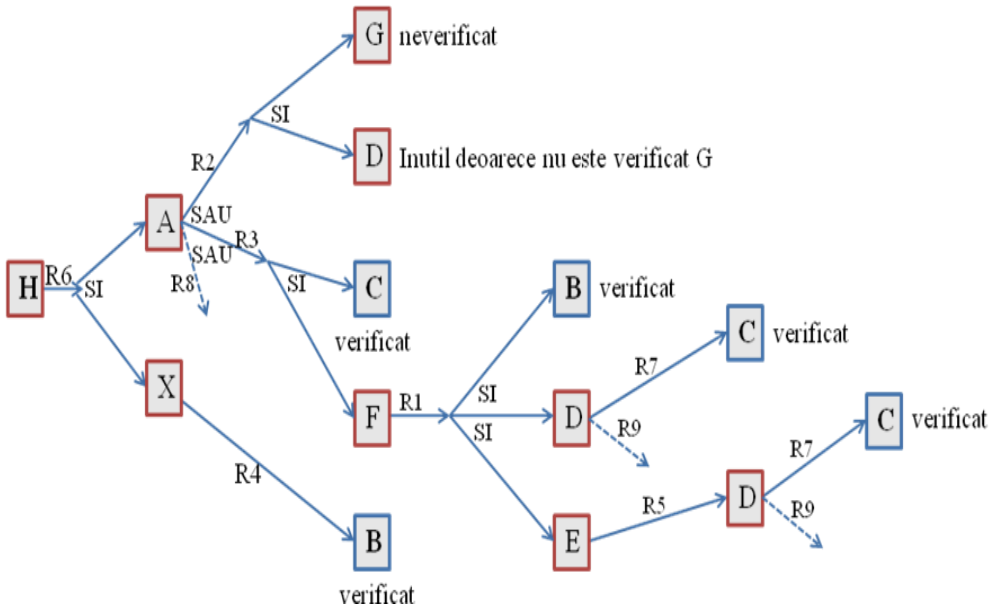


Fig. 7.3 Reprezentarea arborescentă a strategiei de control înapoi

Modul în care funcţionează acest algoritm reiese și din următorul exemplu, în care se dau regulile din BC, faptele iniţiale și obiectivul.

Exemplu

Baza de reguli:

r1: dacă A şi B şi C şi D atunci X
r2: dacă E și F și G atunci X
r3: dacă F1 și F2 şi F3 atunci A
r4: dacă F1 și F4 atunci B
r5: dacă F5 și F6 atunci B
r6: dacă F6 și F7 şi F8 atunci C
r7: dacă F9 și F10 atunci C
r8: dacă F11 şi F12 atunci D
r9: dacă F12 și F13 și F14 atunci E
r10: dacă F12 şi F15 atunci E
r11: dacă F13 şi F14 și F15 atunci F
r12: dacă F15 şi F13 și F17 atunci G
r13: dacă F16 şi F18 atunci G.

Baza de fapte: F1, F2, F4, F5, F6, F9, F10, F11, F12, F14, F15, F17, F18.
Obiectivul este: X.

Etapele parcurse de SE pentru rezolvarea problemei sunt:

1) O = {X} – pasul 1
2) X nu aparţine BF – pasul 2
3) R1= {r1, r2} – pasul 3
4) Se alege r1 (pasul 5) și se reformează O={A, B, C, D}
5) R1= {r3}, R2= {r4, r5}, R3= {r6, r7} şi R4= {r8} – pasul 3
6) Se aleg r3, r4, r6 și r8 și se reformează O ={F1, F2, F3, F4, F6, F7, F8, F11, F12}
7) O ={F3, F7, F8} – pasul 2.1
8) R1= {Ø}, R2= {Ø}, R3= {Ø} – se revine la un pas anterior, pasul 5
9) Se alege regula r2, O ={E, F, G}
10) R1= {r9, r10}, R2= {r11}, R3= {r12, r13}
11) Se aleg regulile r9, r11 și r12, O ={F12, F13, F14, F15, F17}
12) O = {Ø} – pasul 2.1

13) S-a identificat o soluţie a problemei – pasul 2.2

1. Iniţializarea mulţimilor stărilor obiectiv final / intermediar, notată *O*, cu obiectivul problemei: $O = \{Sf\}$
2. Criteriul de oprire:
    a. Se consideră fiecare element *oi* din mulţimea *O* şi se verifică dacă acesta se găseşte în baza iniţială de fapte *BF*. Dacă *oi* aparţine *BF*, *oi* se elimină din mulţimea *O*;
    b. Dacă la finalul acestui pas mulţimea *O* este vidă, înseamnă că s-a identificat o cale între starea finală *Sf* şi starea iniţială *Si*; algoritmul se întrerupe. În caz contrar, se trece la pasul 3.
3. Selectarea regulilor care conţin obiectivele intermediare *oi* printre consecinţele lor. Pentru fiecare element *oi*, se determină câte o mulţime *Ri* a regulilor care conţin pe *oi* printre consecinţe;
4. Verificarea posibilităţii de continuare a procesului de propagare înapoi. Dacă toate mulţimile *Ri* sunt vide, adică obiectivul intermediar *oi* nu poate fi produs cu bazele de fapte şi reguli existente, se revine la unul sau mai mulţi paşi anteriori (back-propagation), formându-se o altă mulţime *O* şi se trece la pasul 2. În caz contrar, se trece la pasul 5;
5. Modificarea mulţimii obiectivelor intermediare. Se alege câte o regulă din fiecare mulţime *Ri* şi, folosind condiţiile acestor reguli,

Avantajele sistemelor care folosesc strategia de control înapoi sunt [1]:

- Proces de raţionare este interactiv;
- Dacă este necesar sau toate posibilităţile au fost explorate, atunci sistemele recurg la întrebări adresate utilizatorilor;

- Arborele de căutare ataşat procesului este deseori mai puţin adânc decât în cazul strategie de control înainte.

Marele dezavantaj care apare la utilizarea strategiei de control înapoi este faptul că poate să apară blocajul, deoarece se intră într-o buclă infinită.

7.4. Strategia de control mixt

Strategia de control mixt, combină într-un fel strategiile de control înainte și înapoi, în încercarea de a elimina dezavantajele introduse de utilizarea individuală a fiecăreia dintre ele şi a profita de avantajele ambelor metode.

Strategia de control mixt este o abordare hibridă a raționamentului care combină atât strategia de control înainte (bazată pe date), cât și strategia de control înapoi (bazată pe obiective) într-un singur proces de inferență. Sistemul expert poate comuta dinamic între cele două, în funcție de cerințele problemei, interacțiunea utilizatorului sau rezultatele intermediare.

Această metodă corespunde într-o mare măsură modului de raţionare uman şi condiţiilor concrete în care în cazul unei probleme caracterizată de o arborescenţă cu adâncime mare se cunosc premisele iniţiale şi concluziile, dar nu se cunosc datele intermediare.

Strategia de control mixt este o metodă de parcurgere a regulilor prin care se utilizează descompunerea problemei iniţiale în sub-probleme (a obiectivului principal în sub-obiective) care apoi sunt rezolvate printr-un control înainte.

Un motor de inferență mixtă poate începe prin înlănțuirea inversă pentru a verifica un obiectiv specific; în timpul procesului de raționament, ar putea realiza că unele sub-obiective intermediare sau fapte necesare nu sunt cunoscute, apoi trece la înlănțuirea directă pentru a deriva acele fapte lipsă din datele disponibile. Odată ce faptele necesare sunt găsite, poate relua înlănțuirea inversă pentru a finaliza verificarea obiectivului. Acest ciclu înainte și înapoi continuă până când se ajunge la o concluzie finală sau sistemul stabilește că obiectivul nu poate fi îndeplinit.

Exemplu

Un sistem expert pentru diagnosticarea și restaurarea defecțiunilor

Sistemul folosește conexiunea inversă pentru a testa ipoteza „Există o defecțiune la alimentatorul F1”.

În timp ce încearcă să demonstreze acest lucru, constată că sistemul are nevoie de mai multe date de la senzori în timp real despre căderile de tensiune și stările întrerupătoarelor.

Trece la conexiunea directă pentru a procesa informațiile disponibile din sistemele SCADA, deduce că întrerupătoarele B2 și B3 s-au declanșat și identifică posibile zone de defect.

Apoi reia raționamentul invers pentru a finaliza etapele de izolare a defecțiunilor și restaurare.

Criteriile de schimbare sunt atunci când un subobiectiv necesar în lanțul invers nu are un fapt cunoscut, se poate declanșa lanțul direct, sau atunci când lanțul direct produce un obiectiv în concluziile sale, lanțul invers îl poate valida sau extinde, sau atunci când interacțiunea utilizatorului dezvăluie obiective sau fapte noi, respectiv pe baza meta-regulilor sau a priorităților regulilor (de exemplu, restaurarea are o prioritate mai mare decât explorarea).

Pentru a permite raționamentul mixt, sistemul are nevoie de obicei de:

1. O bază de reguli care să suporte ambele tipuri de înlănțuire.
2. Un controler de raționament pentru a decide când să se schimbe strategiile.
3. Un mecanism de rezolvare a conflictelor care poate gestiona prioritățile regulilor în ambele moduri.
4. Un strat opțional de meta-reguli pentru a guverna logica de comutare.

Avantajul strategiei de control mixt constă într-un spaţiu de memorie internă redus, deoarece arborele ataşat este de cele mai multe ori mic, respectiv timpul de găsire a soluţiilor este scurt, fiindcă sunt multe în considerare doar sub-scopurile care au o şansă de verificare. Alte avantaje sunt flexibilitate sporită (se adaptează la structura problemei și la datele disponibile), eficiență îmbunătățită (evită declanșarea inutilă a regulilor prin schimbarea strategiilor atunci când una devine ineficientă), susține rezolvarea problemelor complexe (ideal pentru sistemele care necesită atât diagnosticare (înapoi), cât și planificare a acțiunilor (înainte)), și adaptabilitate în timp real (poate reacționa la noile date primite în timpul inferenței, util în sistemele energetice și monitorizare).

Dezavantajele strategiei de control mixt sunt:

- Complexitate crescută - logica de control pentru comutarea între strategii poate fi dificil de proiectat și întreținut.
- Mai greu de urmărit - căile de raționament pot deveni mai complicate, afectând explicabilitatea.
- Consumă intensiv de resurse - poate consuma mai multă memorie și putere de procesare decât strategiile pure.

În concluzie, strategia de control mixt este o metodă puternică de raționament care combină punctele forte ale înlănțuirii directe și inverse, permițând sistemelor expert să fie mai flexibile, să gestioneze informații incomplete sau în evoluție și să rezolve o clasă mai largă de probleme. Aceasta este potrivită pentru domenii complexe precum sistemele energetice, diagnosticul medical și controlul inteligent, unde atât raționamentul reactiv, cât și cel orientat spre obiective sunt esențiale.

7.5. Strategia de control circumstanţial

În această strategie, motorul de inferență nu folosește o direcție de control fixă. În schimb, monitorizează situația sau mediul actual, evaluează circumstanțele sau condițiile (de exemplu, tipul de sarcină, disponibilitatea datelor, urgența, prioritatea) și selectează cea mai potrivită metodă de control în consecință. Poate comuta între raționament direct, invers sau mixt pe baza unor criterii definite fie explicit (prin reguli sau meta-reguli), fie implicit (prin analiză euristică sau algoritmică).

Strategia de control circumstanţial este cea mai nouă metodă implementată în MI a SE care au o interfaţă grafică prietenoasă şi pot uşor comunica cu utilizatorul. Această strategie se caracterizează prin faptul că aplicarea regulilor de inferenţă este autorizată atât de valoarea de adevăr a condiţiei, dar și de caracteristicile relaţionale derivate din situaţia actuală a obiectelor problemei. Datorită acestui fapt, care face ca rezolvarea să fie independentă de sensul în care sunt parcurse regulile, se pot aplica și alte strategii de control: înainte, înapoi sau mixt.

Sistemul evaluează continuu contextul în timpul inferenței. De exemplu:

- Dacă sunt disponibile multe informații, dar nu se cunoaște niciun obiectiv specific, astfel se utilizează înlănțuirea directă.
- Dacă utilizatorul solicită un obiectiv specific, se trece la înlănțuirea inversă.
- Dacă datele sosesc în timp real (de exemplu, de la senzori), atunci se adoptă înlănțuirea directă reactivă.
- Dacă sunt implicate obiective multiple și date parțiale, se selectează raționamentul mixt.
- În controlul circumstanțial, aceste decizii nu sunt predefinite, ci sunt ghidate de factori declanșatori situaționali.

Sistemul expert include de obicei un modul de monitorizare, care urmărește condițiile interne și externe, un selector de strategii, prin care se evaluează condițiile și alege cel mai bun mod de raționament și meta-reguli sau evaluatori euristici, care ajută la determinarea schimbărilor de strategie.

Sistemele care se bazează pe această strategie, au nevoie de date inițiale reduse, deoarece răspunsurile sunt legate de context, și nu este nevoie să fie analizate răspunsuri complexe. Un mod simplu de realizarea a acestor sisteme se bazează pe întrebări puse atunci când MI le foloseşte şi nu pe o cantitate mare de informaţii standard introduse iniţial care ar încărca memoria inutil. Astfel, în funcţie de necesităţile SE, între utilizator și SE are loc un dialog inteligent bazat pe ceea ce are nevoie sistemul [1].

Avantajele și dezavantajele strategiei de control circumstanțial sunt sintetizate în figura 7.4.

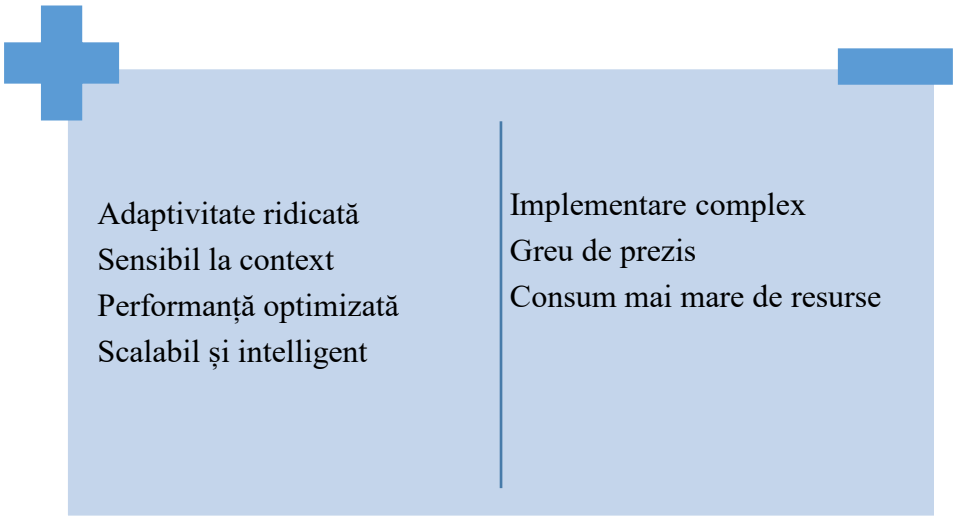


Fig. 7.4 Avantajele și dezavantajele strategiei de control circumstanțial

În concluzie, strategia de control circumstanțial este cea mai inteligentă și mai conștientă de context metodă de control din sistemele expert. Aceasta alege și schimbă strategiile de inferență pe baza circumstanțelor în timp real, făcând sistemul expert mai flexibil, mai receptiv și mai eficient în medii dinamice. Este utilă în special în domenii complexe și în timp real, cum ar fi sistemele Power.

7.6. Alegerea strategiei de control

În dezvoltarea Motorului de Inferenţă este foarte important să se aleagă strategia de control adecvată, care să asigure parcurgerea eficientă a Bazei de Cunoștințe. Aşa cum expertul uman ştie ce cunoștințe să utilizeze într-o situaţie dată, SE trebuie să aibă o strategie de control, de parcurgere a datelor. Această strategie se decide în prima etapă a dezvoltării MI, fără de care SE nu ar funcţiona corespunzător. Pe de altă parte, în etapa următoare se decid metodele de alegere a regulilor care sunt declanşabile la un moment dat. Deciziile luate în cadrul acestei etape influenţează performanţele SE.

Un mod de alegere a strategiei de control se bazează pe utilizarea meta-cunoștințelor [1]; adică pe folosirea cunoştinţelor care oferă informaţii despre modul în care sunt reprezentate cunoștințele, funcţiile, relaţiile etc.

În alegerea strategiei de control în funcţie de tipul problemei trebuie să se aibă în vedere următoarele:

- Strategia de control înainte este adecvată pentru problemele la care sunt disponibile multe date de intrare și nu sunt disponibile date legate de starea finală a problemei. pe de altă parte, această strategie este ineficientă în cazul problemelor complexe;
- Strategia de control înapoi este recomandată în cazul problemelor la care se cunoaşte obiectivul final şi obiectivele intermediare; această metodă este eficientă pentru problemele care au informaţii incomplete sau când se poate aranja un dialog cu utilizatorul;
- Strategia de control mixt este mai greu de implementat pentru că presupune descompunerea problemelor și apoi aplicarea raţionamentului deductiv, dar ea se poate aplica cu succes în cazul majorităţii problemelor;
- Strategia de control circumstanţial este caracteristică noilor SE și aceasta implică utilizarea și altor strategii de control.

Controlul dirijat prin date este cea mai populară metodă de alegere a strategie de control, deoarece este uşor de realizat şi se recomandă a se utiliza atunci când sunt informaţii multe și nu se ştie nimic despre scopul ce trebuie atins []1.

Literatura de specialitate arată faptul că nici o strategie de control ne este bună în orice situaţie. Din această cauză noile SE au MI care sunt dezvoltate pe mai multe tipuri de strategii în funcţie de starea problemei la un moment dat.

7.7. Întrebări şi exerciţii

Exerciţiu

Să se precizeze ordinea de parcurgere a regulilor, dacă se cunoaşte X şi C, iar obiectivul este A, folosind strategia de control înapoi.

Baza de reguli:

Regulile R1...R9:

R1: dacă B şi D şi E atunci F
R2: dacă D şi G atunci A
R3: dacă C şi F atunci A
R4: dacă B atunci X
R5: dacă D atunci E
R7: dacă A şi X atunci H
R7: dacă C atunci D
R8: dacă X şi C atunci A
R9: dacă X şi B atunci D

Întrebări

1. Regulile de producţie din baza de reguli pot fi reprezentate sub altă formă.
□ Da (Sub ce formă ?)/ □ Nu

2. Avantajele strategiei de control înapoi sunt:
□ Proces de raţionare interactiv / □ SE rezolvă singur problema, fără intervenţia utilizatorului / □ Arborele de căutare este mai mic

3. Metoda de rezolvare a conflictelor se alege la începutul dezvoltării Motorului de Inferenţă şi este valabilă pentru toate situaţiile de conflict între reguli.
4. □ Adevărat / □ Fals

8

Motorul de inferențe
Metode de rezolvare a conflictelor

Prezentul capitol continua capitolul precedent prin descrierea modului de funcționare a Motorului de inferență, mai exact pocesul de rezolvare al conflictelor.

8.1. Aspecte generale

Rezolvarea conflictelor este procesul prin care se alege regula dintre cele selectate ca aplicabile la un moment dat, care va aplicată. În acest proces, de cele mai multe ori se alege prima regulă aplicabilă, însă această alegere poate să ducă la o stare care nu este cea optimă, iar alegerea altei reguli să conducă la o stare a bazei de fapte care este mai aproape de starea dorită decât starea pe care ar produce-o celelalte reguli aplicabile.

Procesul de rezolvarea conflictelor poate fi exemplificat prin intermediul diagramei logice din figura 8, unde se poate observa că procedura presupune trei paşi [2]:

1. Stabilirea regulilor care pot fi aplicate;
2. Alegerea regulii, sau regulilor, adecvate pentru a fi aplicată;
3. Aplicarea regulilor alese.

În etapa de alegere a regulii, sau regulilor, care se va aplica se pot folosi diferite metode de selectare. Dintre acestea cea mai utilizată este selectarea primei reguli aplicabile. Însă această metodă de multe ori determină creşterea timpului de atingere a obiectivului problemei (acest lucru se observă în cadrul secţiunii de Activitate dat în acest capitol). Alte metode de rezolvare a conflictelor sunt: selectarea regulii celei mai specializate, selectarea regulii celei mai productive și selectarea după o tehnică euristică.

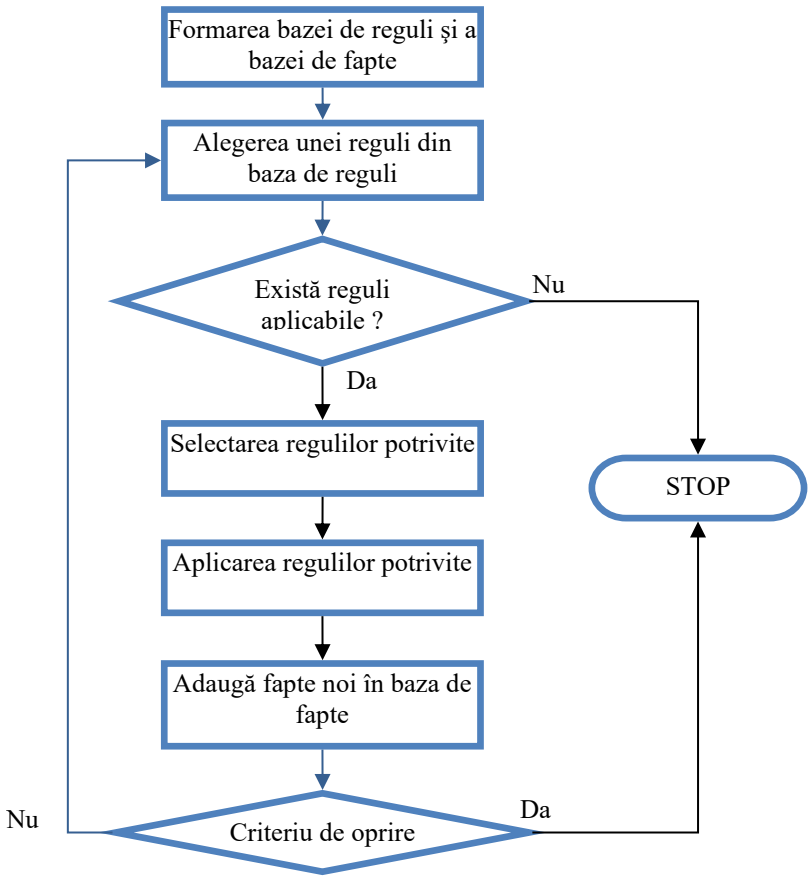


Fig. 8.1 Schema logică a procesului de rezolvare a conflictelor dintre reguli [2]

8.2. Selectarea regulii celei mai specializate

Această metodă de rezolvare a conflictelor dintre reguli este denumită şi metoda regulii cu cele mai multe condiţii. Denumirea este justificată, având în vedere că ea funcţionează astfel:

Dintre două reguli X şi Z, se alege ce care conţine în porţiunea Dacă (If) numărul maxim de condiţii.

Exemplu

Regula X: Dacă (A,B,C,D) Atunci (R,S,T)

Regula Y: Dacă (A,C) Atunci (P,Q,T)

Dintre cele două reguli, se va alege regula X, deoarece este mai specializată, adică are mai multe condiţii în premisă.

8.3. Selectarea regulii celei mai productive

Selectarea regulii celei mai productive, este întâlnită în literatura de specialitate și sub denumirea de selectarea regulii cu cele mai multe consecinţe; ea pleacă de la ideea că mai mult înseamnă mai bine. În consecinţă, dintre două reguli X şi Y va fi aleasă regula care are mai multe argumente în partea de aplicaţie, adică în porţiunea Atunci (Then).

Regula X: Dacă (A,B,C,D) Atunci (R,S)
Regula Y: Dacă (A,C) Atunci (P,Q,T)
Dintre cele două reguli, se va alege regula Y, deoarece este mai productivă, adică are mai multe argumente la aplicaţie.

8.4. Selectarea după o regulă euristică

Selectarea după o regulă euristică presupune alegerea regulii care aduce problema cel mai aproape de scopul dorit. Ea nu impune o anumită funcţie implicată, ci lasă la latitudinea programatorului acest lucru. În consecinţă, la aplicarea acestei metode trebuie definită o funcţii care să măsoare distanţa dintre două stări ale sistemului, în funcţie de care se va construii criteriul de rezolvare a conflictelor dintre reguli.

Modul de selectare a regulii care va fi aplicată dintre mulţimea regulilor aplicabile este influenţat de starea problemei, astfel de-a lungul procesului de soluţionare a unei probleme pot fi aplicate mai multe metode de rezolvare a conflictelor.

Exemplu

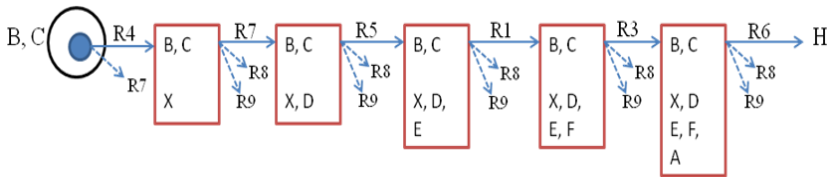
Să se realizeze secvenţa de parcurgere a regulilor pentru a se atinge obiectivul dorit. Se va aplica selectarea regulii celei mai productive, şi se va face o comparaţie între metodele de selectare aplicate. În continuare se dă baza de reguli şi obiectivul, precum și arborele decizional în cazul aplicării primei dintre regulile aplicabile și a regulii celei mai specializate.

Se consideră o BC reprezentată prin următoarele reguli de producţie:
Regulile R1...R9:

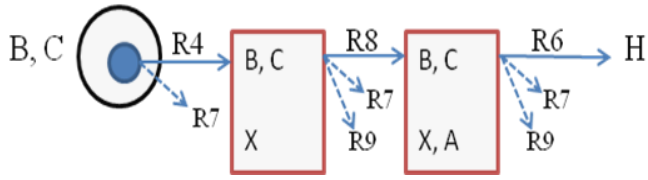
R1: dacă B și D și E atunci F
R2: dacă D şi G atunci A
R3: dacă C și F atunci A
R4: dacă B atunci X
R5: dacă D atunci E
R6: dacă A şi X atunci H
R7: dacă C atunci D
R8: dacă X şi C atunci A
R9: dacă X și B atunci D

Se cunoaşte B şi C, iar obiectivul este H.

Selectarea primei reguli aplicabile



Selectarea regulii celei mai specializate



8.5. Intrebări și exerciții

Exercițiu

Să se realizeze secvenţa de parcurgere a regulilor pentru a se atinge obiectivul dorit. Se va aplica selectarea regulii celei mai productive, şi se va face o comparaţie între metodele de selectare aplicate.

Regulile R1...R9:

R1: dacă B şi D şi E atunci F
R2: dacă D şi G atunci A
R3: dacă C şi F atunci A
R4: dacă B atunci X
R5: dacă D atunci E

R7: dacă A şi X atunci H
R7: dacă C atunci D
R8: dacă X şi C atunci A
R9: dacă X şi B atunci D

Întrebări

1. Cea mai populară metodă de rezolvare a conflictelor este:
   Prima regulă aplicabilă / Regula cea mai productivă / Regula cea mai specializată

2. În exemplu dat, regula cea mai specializată a dat rezultate mai bune decât prima regulă aplicabilă.
   Adevărat / Fals

3. Metodele de rezolvare a conflictelor euristice au caracteristic faptul că sunt dedicate problemei de rezolvat.
   Adevărat / Fals

9

Motorul de inferențe
Algoritmi de căutare

Prezentul capitol reprezintă ultimua etapă în prezentarea Motorului de Inferență, și anume descrierea celor mai populari algoritmi de căutare folosiți în cadrul Sistemelor Expert.

9.1. Introducere

Un SE trebuie în primul rând să fie eficace, şi apoi eficient. Pentru a îndeplini prima condiţie are nevoie de toate elementele componente (Baza de Cunoștințe, Motorul de Inferenţă etc.) și de datele necesare (transmise de către expertul uman). Pe de altă parte, pentru a fi eficient trebuie să aibă în componenţa sa elemente auxiliare (modulul dinamic, modul de achiziție a cunoștințelor etc.) care să îi crească performanţele.

Algoritmul este definit ca ansamblu de reguli şi operatori pentru efectuarea unui sistem de operaţii într-o ordine dată în vederea rezolvării unei probleme de un anumit tip. În acest context, un algoritm de căutare cuprinde totalitatea operatorilor și regulilor necesare pentru a căuta în spaţiul regulilor BC în vederea găsirii acelor reguli care sunt în concordanţă cu starea problemei la un moment dat.

Regulile cuprinse în BC pot să fie reprezentate sub forma unor arbori de inferenţă care au ca noduri premisele (condiţiile), iar ca ramuri arce care conectează diferitele premise. Arcele pot să fie ramuri de tip ŞI, respectiv SAU în funcţie de operatorul logic care se află între diversele reguli.

Procedeul de inferenţă poate fi vizualizat ca un drum de-a lungul arcelor arborelui, în acest fel strategiile de control aplicabile sunt mai uşor de înţeles și urmărit. Trebuie precizat faptul că pentru a parcurge un arc de tip și trebuie ca toate arcele parcurse înaintea lui să fi fost tot de tip ŞI.

În cadrul unui graf, algoritmul de căutare este un ansamblu de reguli scrise într-o anumită ordine care arată modul în care trebuie parcurs graful, astfel are rol în evitarea conflictelor în timpul stabilirii regulii de declanşat dintre cele multe selectate.

În timpul funcţionării MI pentru rezolvarea unei probleme se recomandă utilizarea mai multor algoritmi de căutare în funcţie de starea problemei şi de datele disponibile la momentul curent.

9.2. Căutarea exhaustivă

Metoda cea mai simplă, dar în acelaşi timp cea mai ineficientă este căutarea exhaustivă (în totalitate), denumită și căutarea prin încercări repetate (trial-and-error searching). Acest algoritm presupune parcurgerea tuturor regulilor fără nici o regulă până când se atinge scopul propus.

Cunoscând starea iniţială a problemei şi cea finală, adică obiectivul, se generează aleatoriu un set de transformări care se aplică succesiv stării iniţiale şi stărilor care rezultă din aceasta, apoi se verifică dacă s-a atins sau nu starea finală. În acest sens, este garantată găsirea unei soluţii la problema de rezolvat. Pe de altă parte, timpul de lucru poate fi mare şi nu oferă o eficienţă foarte ridicată, doar un ajutor adus strategiei de control.

Algoritmul căutării exhaustive poate avea următoarea formă:

```
Stabileşte starea iniţială
        While (stare ≠ starea obiectiv)
        { selectează un operator aplicabil în această stare pe care îl aplică;
          stare = operator(stare);
        }
```

Fig. 9.1. Algoritmul de căutare exhaustivă

Căutarea exhaustivă poate fi aplicată cu succes în cazul problemelor simple, care sunt caracterizate de un spaţiu al stărilor de dimensiuni mici [2].

Exerciţiu - exemplu

Se consideră o BC reprezentată prin următoarele reguli de producţie:

Regulile R1...R9:

R1: dacă B și D și E atunci F
R2: dacă D şi G atunci A
R3: dacă C și F atunci A
R4: dacă B atunci X
R5: dacă D atunci E
R6: dacă A şi X atunci H
R7: dacă C atunci D
R8: dacă X şi C atunci A
R9: dacă X și B atunci D

Datele iniţiale sunt B şi C, iar scopul este H. Enunţaţi ordinea în care sunt parcurse conform algoritmului căutării exhaustive.
Se vor parcurge regulile în ordinea în care seunt scrise în BC.

9.3. Căutarea în adâncime

Această metodă de căutare se aplică în special în cazul graf-urilor şi a arborilor, şi face parte din categoria algoritmilor de căutare prin examinare semantică a nodurilor.

Căutarea în adâncime, denumită şi căutarea în profunzime sau căutarea în lungime (depth-first search) este cea mai frecvent folosită metodă de căutare în aplicaţiile practice. Algoritmul căutării în adâncime funcţionează astfel: la fiecare pas se generează o succesiune a nodurilor care definesc o cale a grafului. Dacă ultimul nod al acestei căi este starea finală dorită, atunci acea care indică modul de parcurgere a nodurilor pentru determinarea soluţiei, dacă nu este starea finală atunci se va reveni la ultimul nod care nu a fost parcurs şi se va forma o cale cu acesta.

Căutarea în adâncime poate fi implementată pe calculator în mod direct prin utilizarea unor proceduri de parcurgere înainte și revenire. Astfel se parcurge garful de la un nivel la altul de-a lungul arcelor. În prima etapă se defineşte nodul de start, rădăcina, apoi se continuă să se parcurgă nodurile pe niveluri până este atins nodul obiectiv.

Un exemplu de funcţionare a acestei metode este dat în continuare.

Exemplu

Se dă graful din figură. Starea iniţială este A, iar cea finală E. Dacă se începe parcurgerea cu B atunci este parcurs drumul alcătuit din

săgeţile indicate cu 1, 2, 3 și 4. Figura 9.2. ilustrează parcurgerea regulilor.

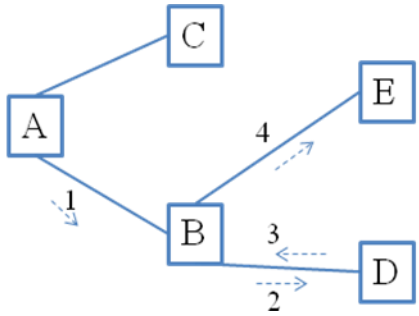


Fig. 9.2. Căutarea în adâncime

Căutarea în adâncime este de preferat comparativ cu căutarea în lăţime, deoarece utilizează memorie mai puţină, având în vedere că la un moment dat se memorează doar nodurile corespunzătoare căii analizate.

9.4. Căutarea în lăţime

Căutarea în lăţime (căutarea în grosime, breadth-first search) este o metodă de parcurgere a unui graf, ea făcând parte din parcurgerea semantică a nodurilor. Această metodă este mai puţin populară decât căutarea în lăţime, dar este adecvată problemelor a căror reprezentare arborescentă a regulilor arată o adâncime mare (ramuri lungi).

Modul de lucru al căutării în lăţime este următorul: porneşte de la nodul rădăcină și parcurge garful pe nivel, după ce au fost parcurse toate nodurile corespunzătoare unui nivel, se trece la nivelul următor (inferior).

În continuare, se dă un exemplu de modul în care funcţionează căutarea în lăţime pentru un graf format din 5 noduri aranjate pe trei niveluri.

Exemplu

Se dă graful din figură. Starea iniţială este A, iar cea finală E. Dacă se începe parcurgerea cu B atunci este parcurs drumul alcătuit din săgeţile indicate cu 1, 2, 3, 4 şi 5 pentru a se ajunge la starea E. Se poate observa, că în cazul grafului din figură numărul de paşi necesari pentru a determina soluţia este mai mare decât în cazul căutării în lărgime. Figura 9.3. subliniază regulile parcurse în cadrul algoritmului de căutare în lățime.

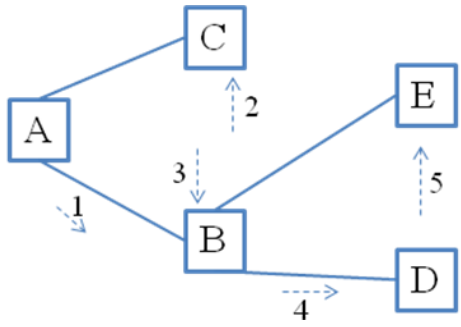


Fig. 9.3. Căutarea în lățime

Exercițiu - exemplu

Se consideră graful din figura 9.4. Să se indice căile de parcurgere a nodurilor şi arcelor pentru a se atinge obiectivul format din nodul 13. Se vor folosi cele două metodă de examinare semantică a nodurilor și se va face comparaţie între rezultatele obţinute.

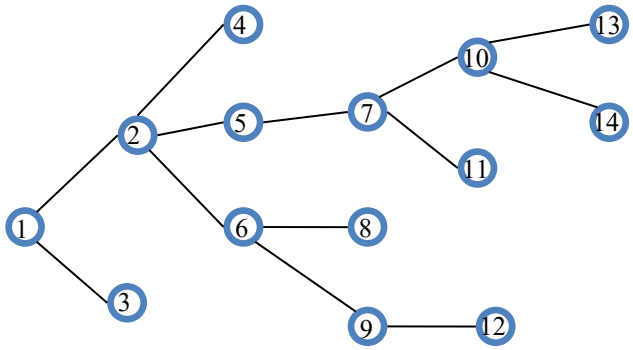


Fig. 9.4. Graful corespunzător regulilor din BC

9.5. Căutarea soluţiei optime

Algoritmii de căutare prezentaţi anterior nu iau în considerare influenţa drumului parcurs asupra rezultatului problemei. Pentru a elimina acest dezavantaj s-a introdus căutarea soluţiei optime prin care fiecărui operator i se asociază o funcţie de valoare, de cost sau de distanţă cu ajutorul căreia se cuantifică consecinţele aplicării unui operator. La alegerea drumului de parcurs într-un graf se vor alege stările produse de acei operatori a căror condiţii de aplicabilitate sunt îndeplinite.

Un algoritm de căutare a soluţiei optime este după costul minim. În acest caz, fiecărui arc care leagă două noduri îi este atribuită o valoare care arată costul

parcurgerii lui în rezolvarea problemei, sau mai exact cât de departe se află acel nod de nodul final. Astfel, în determinarea soluţiei se va alege calea care are costul mai mic. Acest lucru este evidenţiat și în exemplul următor.

Exemplu

Pentru graful din figura 9.5, s-au asociat arcelor valori care arată costul alegerii arcului corespunzător. Parcurgerea după acest algoritm va determina soluţia G1.

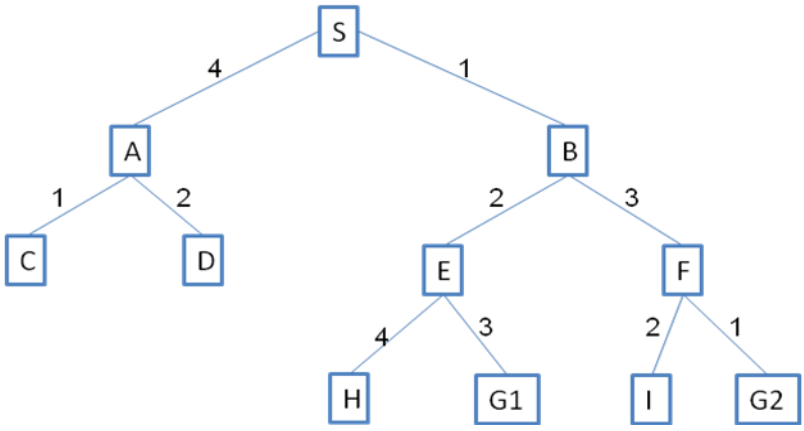


Fig. 9.5. Graf – căutarea după costul minim

În cazul unui graf finit va exista mereu un drum optim de la nodul de start la nodul obiectiv. Acest lucru poate fi obţinut prin aplicarea a două teoreme [1]:

- Într-un graf finit există un optim de la nodul de start la nodul obiectiv pe care procedura de căutare îl găseşte;
- La momentul în care nodul *n* este expandat căutarea optimă asigură drumul optim până la *n*.

Acest algoritm de căutare poate să ducă la eşec dacă nu sunt alese bine funcţiile asociate arcelor grafului, astfel în această direcţie trebuie avut mare grijă.

Metoda pasului optim este o îmbinare între căutarea în adâncime și în lăţime, care se focalizează pentru găsirea soluției optime.

9.6. Metode euristice

Metodele euristice de căutare se bazează pe determinarea la fiecare moment al căutării a depărtării sau apropierii de starea finală, obiectivul problemei.

Conform definiţie oferite de DEX, „un procedeu euristic serveşte la descoperirea unor cunoştinţe noi". Alte definiţii vorbesc despre procesul prin care se obţin fapte noi. În ceea ce priveşte algoritmii de căutare, metodele euristice determină creşterea eficienţei căutării unei soluţii, fără a garanta determinarea ei în orice condiţii.

În dezvoltarea algoritmilor euristici de căutare de cele mai multe ori se pleacă de la nişte aproximări raţionale, deduceri care acoperă o gamă mai largă de probleme. În etapa următoare trebuie să se particularizeze algoritmul pentru problema specifică de rezolvat. Acest lucru se poate face cu ajutorul unor funcţii, numite funcţii euristice.

O funcţie euristică specifică la un moment dat starea problemei, adică cât de departe este momentul curent faţă de obiectivul problemei. Astfel, pentru a se construi o funcţie euristică trebuie să existe un model după care să se pornească pentru calcularea acestei distanţe. Pentru eficientizarea metodei este nevoie pe lângă distanţa de la starea curentă la starea finală şi de costul care îl presupune parcurgerea acestei distanţe. În consecinţă, construirea funcţiei euristice este un compromis între costul evaluării funcţiei în sine și precizia cu care această funcţie determină distanţa reală dintre starea curentă şi starea finală [1].

Modul cel mai general de definire a unei funcţii euristice se face prin intermediul a două funcţii care:

- Evaluează costul deplasării din starea iniţială în starea curentă;
- Estimează costul deplasării din starea curentă în starea finală.

Din aceste funcţii (funcţia euristică și cele 2 care o compun) doar funcţia care evaluează deplasarea curentă este o funcţie reală, celelalte două sunt doar nişte estimări. De acest fapt depinde foarte mult eficienţa metodei euristice.

Printre metodele euristice de căutare care sunt implementabile în cadrul SE se numără [1]:

- Metoda căutării euristice în adâncime – se aseamănă cu căutarea în adâncime simplă doar că apelează la o funcţie euristică pentru a eficientiza metoda de bază. Astfel, la alegerea nodului următor se ia în considerare valoarea funcţiei euristice ataşat fiecăruia.

- Metoda căii spre obiectiv – principiul metodei constă în alegerea acelor reguli care par să conducă către un anumit obiectiv, care la rezolvarea unei probleme va fi obiectivul ei.
- Metoda urcării pantei – este o metodă eficientă în cazul problemelor care au un spaţiu mare al soluţiilor. Principiul acestei metode este următorul: într-un spaţiu cu trei dimensiuni, algoritmul metodei încearcă să deplaseze un punct care descrie soluţia problemei, astfel încât aceasta să ocupe locul poziţia cea mai înaltă, asemănătoare vârfului unei coline;
- Metoda călirii simultane – este o combinaţie între metoda urcării pantei și a selecţiei aleatorii. Principiul constă în aplicarea unui salt în spaţiu de căutare atunci când s-a găsit un maxim local, cu speranţă că acest salt va determina găsirea unui valori superioare maximului local;
- Metoda căutării spectrale – avantajul constă în posibilitatea găsirii soluţiei optime globale, ci nu a unui optim local ca în cazul celorlalte metode. Metoda este asemănătoare cu metoda urcării pantei, diferenţa constă în faptul că verificarea noii stări a problemei se face pe mai multe direcţii, ci doar pe una ca în cazul metodei amintite;
- Algoritmul A* - este o îmbinare între metoda costului minim și a pasului optim. Această metodă foloseşte pentru fiecare nod care se verifică o funcţie cost globală, care evaluează deplasarea între nodul iniţial și final, ca suma dintre două funcţii care conţin o valoarea reală și una estimată;
- Metoda programării dinamice – urmăreşte construirea unei funcţii euristice ideale care să asigure identificarea unei soluţii folosind căutarea în adâncime, fără a fi necesară însă aplicarea unor tehnici de backtracking (revenire).

În următoarele paragrafe vor fi prezentate două metode euristice de căutare, și anume metoda pasului optim și metoda programării dinamice.

Metoda pasului optim combină avantajele oferite de căutarea în adâncime și căutarea în lăţime. Schema de principiu a acestei metode este ilustrate în figura 9.6.

Metoda pasului optim este asemănătoare celei a costului minim, cu deosebirea că aceasta metodă se foloseşte în locul costului minim (funcţia deplasării din starea iniţială în starea de la momentul curent) funcţia care estimează deplasarea din starea curentă în starea finală. Pentru a se evidenţia

modul de operare al acestei metode este dat următorul exemplu, în care se dă un arbore la care pentru fiecare nod sunt date valorile funcţiei de cost.

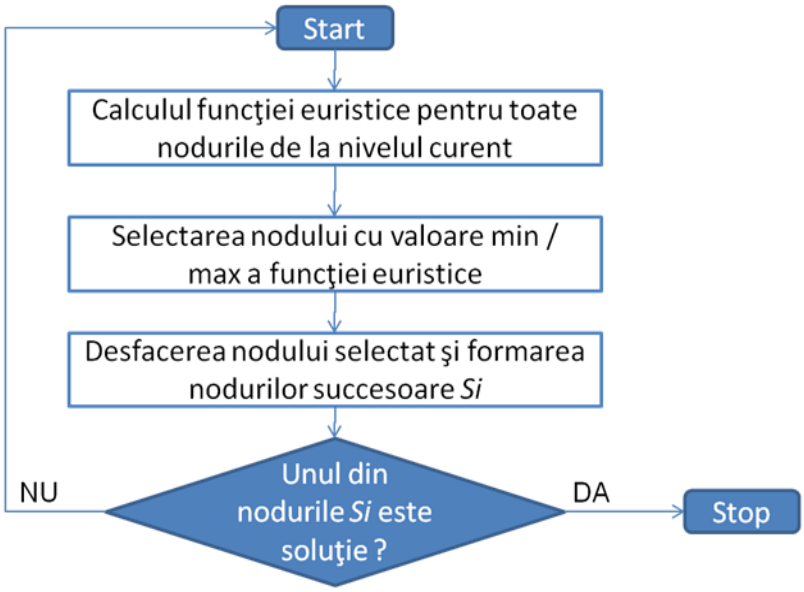


Fig. 9.6. Schema de principiu a căutării pasului optim

Exemplu

Se dă arborele din figură. Valorile funcţiei euristice sunt enumerate în continuare.

Valorile funcţiei euristice:

h(n1) = 930
h(n2) = 870
h(n3) = ***
h(n4) = 560
h(n5) = 430
h(n6) = ***
h(n7) = 680
h(n8) = ***
h(n9) = 350
h(n10) = ***
h(n11) = ***
h(n12) =180
h(n13) = 0
h(n14) = ***

Ordinea în care sunt alese nodurile este: n1, n2, (n4, n5, n6) n5, n9, (n11, n12) n12, (n13, n14) n13

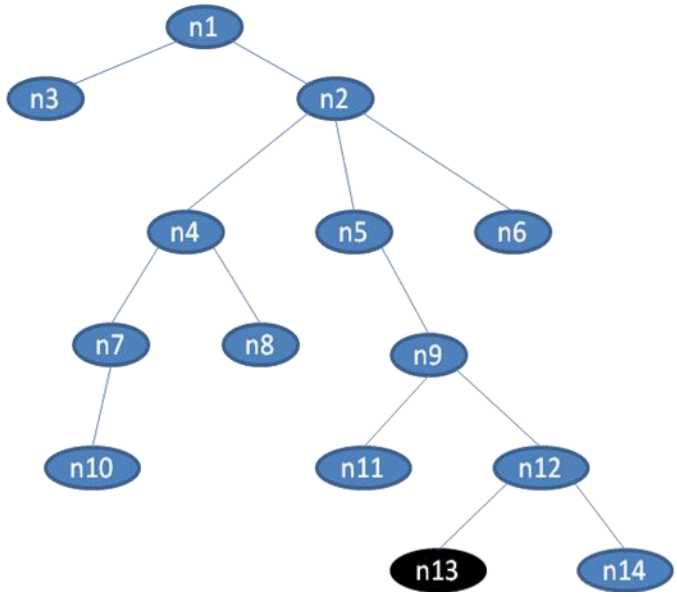


Fig. 9.7. Căutarea pasului optim

Metoda programării dinamice nu foloseşte o funcţie euristică propriu-zisă conform definiţiei date în paragrafele anterioare, ci o funcţie care este asociată fiecărui nod și reprezintă valoarea minimă a costului asociat căii care asigură deplasarea de la nodul respectiv la nodul final cu un cost minim [1]:

$$D(n) = 0 \text{ } (n \text{ este nod final}) \text{ / } \min[d(n,m)]+D(m) \text{ (în caz contrar).}$$

Distanţele $D(n)$ se determină la început, prin utilizarea grafului iniţial. Dacă de la un nod nu există drum către nodul final obiectiv, atunci distanţa va lua valori foarte mari, astfel posibilitatea să fie aleasă această cale este inexistentă.

Se dă arborele din figura ce urmează. Precum se poate observa, fiecărui nod îi este asociată o distanţă a cărui valoare depinde de apropierea acelui nod de nodul final.

D(14) = 500
D(13) = 0
D(12) = 200
D(11) = 1200
D(10) = 1700
D(9) = 400
D(8) = 1400
D(7) = 1300
D(6) = 1700
D(5) = 500
D(4) = 1100
D(3) =1400
D(2) = 1000
D(1) = 1100

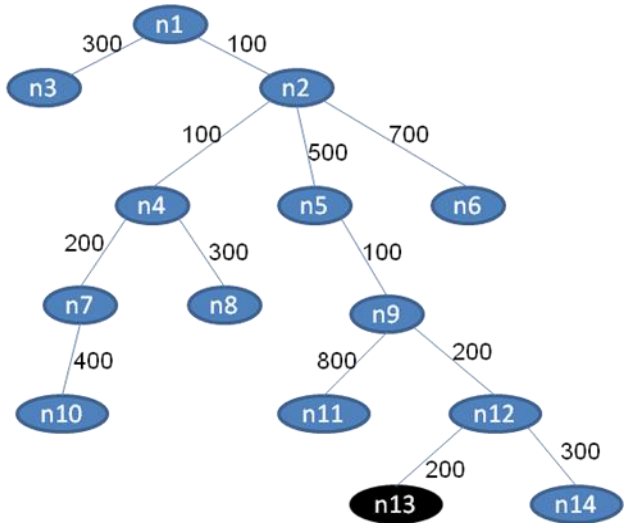


Fig. 9.8. Căutarea programării dinamice

Ordinea în care va fi parcurs arborele este: n1, n2, n5, n9, n12, n13. Trebuie specificat că în situaţiile în care sunt mai multe noduri legate de nodul curent atunci pentru fiecare dintre ele se calculează distanţa până la nodul final, alegându-se nodul care are valoarea minimă.

La finalul celor patru capitole dedicate motorului de inferență, se pot face următoarele afirmații recapitulative.

Algoritmii de căutare sunt componente fundamentale ale motorului de inferență din sistemele expert. Rolul lor principal este de a explora spațiul stărilor (faptelor), regulilor sau deciziilor posibile pentru a ajunge la o soluție validă sau optimă. Acești algoritmi diferă în ceea ce privește informațiile pe care le utilizează, direcția explorării și eficiența lor.

Algoritmii de căutare pot fi clasificați în general în trei categorii principale:

- Algoritmii de căutare neinformați, care funcționează fără nicio cunoaștere specifică domeniului. Aceștia explorează spațiul de căutare sistematic, folosind metode precum căutarea în adâncime, căutarea în lățime sau căutarea exhaustivă. Aceștia sunt ușor de implementat și garantează caracterul complet, dar sunt adesea ineficienți în ceea ce privește timpul și memoria.
- Căutare soluțiilor celor mai bune, sunt algoritmi, care urmăresc să găsească soluția optimă pe baza criteriilor de cost sau lungime, fără a

utiliza neapărat euristici. Tehnici precum căutarea cu cost uniform și ramificarea și limitarea se încadrează în acest grup. Aceștia sunt potriviți pentru problemele în care toate soluțiile posibile trebuie evaluate pe baza costului și garantează rezultate optime atunci când sunt implementați corect.

- Algoritmii euristici de căutare utilizează cunoștințe specifice problemei sau funcții euristice pentru a ghida căutarea mai eficient. Exemplele includ căutarea euristică în profunzime, metoda căii țintă, urcarea dealurilor, căutarea spectrală, A* și programarea dinamică cu aproximare euristică. Aceste metode prioritizează direcțiile promițătoare în spațiul de căutare, adesea obținând o convergență mai rapidă și capacitatea de a evita optimele locale atunci când sunt proiectate cu caracteristici stocastice sau multidirecționale.

Mulți algoritmi euristici sunt, de asemenea, hibrizi, combinând metode deterministe (cum ar fi urcarea dealurilor sau traversarea în profunzime) cu aleatorietatea sau direcționalitatea pentru a evita capcanele locale și a îmbunătăți șansele de a găsi un optim global.

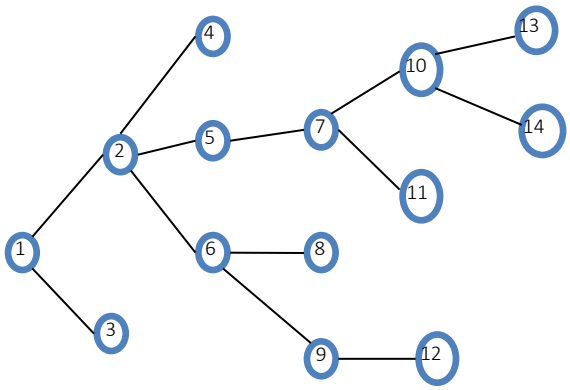
Strategia de control (cum ar fi înlănțuirea înainte sau înapoi) definește direcția de aplicare a regulilor, în timp ce algoritmul de căutare guvernează modul în care este traversat spațiul de posibilități. Împreună, acestea modelează comportamentul de raționament al sistemului expert.

În concluzie, selectarea algoritmului de căutare adecvat depinde de natura problemei, de dimensiunea și complexitatea spațiului de soluții și optimalitatea sau eficiența este preocuparea principală. Metodele euristice, în special cele care încorporează învățarea sau programarea dinamică, sunt deosebit de puternice în sistemele expert mari și complexe, unde căutarea exhaustivă nu este fezabilă.

9.7. Întrebări şi răspunsuri

Exerciţiu

Se consideră graful din figura de mai jos. Să se indice căile de parcurgere a nodurilor şi arcelor pentru a se atinge obiectivul format din nodul 13. Se vor folosi cele două metodă de examinare semantică a nodurilor (parcurgerea în adâncime și în lățime) şi se va face comparaţie între rezultatele obţinute.



Întrebări

1. Algoritmul de căutare prin care se parcurge întreaga mulţime a regulilor este căutarea.
   Exhaustivă / □ Euristică / □ Soluţiei optime

2. Căutarea în lăţime este mai utilizată pentru că are nevoie de mai puţină memorie.
   □ Adevărat / □ Fals

3. Metodele de căutare euristice folosesc o funcţie euristică pentru a determina la fiecare moment depărtarea sau apropierea de soluţia problemei.
   □ Adevărat / □ Fals

4. Algoritmii de căutare care se adaptează la cerințele problemei sunt:
   Căutarea în adâncime / Căutarea în lățime / Căutarea soluției optime

10

Modulul de interfață cu utilizatorii
Modelarea utilizatorilor

Prezentul capitol descrie Modulul de interfață cu utilizatorul, o componentă esențială a oricărui sistem expert.

10.1. Aspecte generale

Interfaţa grafică cu utilizatorul (interfaţa cu utilizatorul) reprezintă elementul component al unui program cu care utilizatorul intră în contact, deci comunică cu programul. Astfel, importanţa ei este justificată, având în vedere că de ea depinde acceptarea iniţială a programului.

Un SE este un program informatic care încorporează în structura sa cunoștințele unui expert uman (unor experţi umani) specialist în domeniul său de activitate, și le gestionează astfel încât utilizatorii sistemului care doresc informaţii din acel domeniu restrâns al expertului să poată să le găsească prin intermediul programului.

Interfaţa cu utilizatorul a unui SE poartă numele de Modulul de Interfaţă cu Utilizatorul (MIU) ce faţă de un program convenţional are anumite caracteristici specifice.

În prezentul capitol se descriu aspectele generale legate de interfaţa cu utilizatorul, cu particularizare asupra interfeţelor din cadrul SE. De asemenea, se prezintă importanţa și rolul MIU.

Interfaţa cu utilizatorul, denumită şi interacţiunea om – maşină, se referă la modul în care o maşină (un program informatic, un dispozitiv, o instalaţie etc.) comunică cu utilizatorii. În prezent, tendinţa în acest domeniu este ca interfaţa să fie uşor de înţeles, învăţat şi folosit. În acest sens au fost dezvoltate un număr mare de metode, dintre care multe sunt în stadiu de prototip, sau sunt inspectate și evaluate pentru a fi transmise publicului larg.

Interfaţa grafică cu utilizatorul, este acea variantă specială a unei interfeţe cu utilizatorul care se caracterizează prin faptul că interacţiunea cu operatorul uman se face prin intermediul unor elemente grafice (ferestre, icoane etc.). Alte variante de interfeţe cu utilizatorul utilizate în domeniul programelor informatice sunt interfeţele care se bazează pe text, pe comenzi etc., în care comunicarea dintre calculator și utilizator se realizează cu ajutorul tastaturii sau a mouse-ului.

Trebuie specificat faptul că, în literatura de specialitate, în definirea interfeţei cu utilizatorul, se face referire la toate elementele componente ale unei maşini cu care aceasta comunică cu utilizatorii săi. În acest document, termenul de interfaţă cu utilizatorul este folosit pentru a deservi componenta unui program informatic, cu care acesta intră în contact cu utilizatorii.

Sistemele Expert sunt sisteme capabile să ofere soluţii la probleme specifice dintr-un domeniu restrâns de activitate umană, sau consultaţii, într-un mod și la un nivel ridicat de expertiză asemănător experţilor umani din domeniu.

Modul în care sunt construite SE le oferă anumite avantaje specifice, care le diferenţiază de programele convenţionale de gestiune a datelor. Un Sistem Expert conţine un Modul Explicativ prin care sistemul oferă utilizatorului explicaţii legate de modul în care s-a rezolvat problema, un Modul de Interfaţă cu Inginerul de Cunoștințe prin care acesta poate modifica datele din Baza de Cunoștințe, introduce date, un Modul de Interfaţă cu Utilizatorul etc., aşa cum este ilustrat în figura 2.1.

Precum se observă în arhitectura complexă a unui SE din imaginea anterioară, componenta care asigură dialogul dintre utilizator și sistem este Modulul de Interfaţă Grafică cu Utilizatorul (Modulul de Interfaţă cu Utilizatorul - MIU). Dialogul dintre cele două părţi (utilizator şi program) se realizează în limbaj cvasinatural, ce este recunoscut de ambele părţi. De altfel, programul foloseşte un limbajul intern cu care face calculele și inferenţele, iar utilizatorul limbajul natural.

Studiile în domeniul SE au arătat faptul că aprecierea utilizatorilor asupra unui SE depinde de măsura de asemănare între modul de reprezentare a informaţiilor de către SE şi modelul pe care utilizatorul îl consideră corespunzător pentru rezolvarea problemei. Acest fapt este de asemenea în strânsă legătură cu modul în care toate informaţiile sunt făcute cunoscute utilizatorului, adică de interfaţa cu utilizatorul.

Interfaţa cu utilizatorul a unui SE poate să fie foarte sofisticată:

- Să integreze dicţionare;

- Să fie capabilă să realizeze analize lexico-sintactice, verificări semantice, corecţii ortografice;
- Să gestioneze prescurtări
- etc.

Pe de altă parte, prin esenţa ei, interfaţa cu utilizatorul a unui SE este complexă, având în vedere că trebuie să ofere o gamă atât de largă de informaţii (starea problemei la fiecare moment, explicaţie asupra soluţiei găsite etc.)

10.2. Importanţa MIU

Recunoaşterea și acceptarea precoce a unui Sistem Expert de către utilizatori este în legătură directă cu eficienţa şi imaginea sistemului, a caracteristicilor interfeţei şi a implicării utilizatorilor.

Istoria SE a început în laboratoarele universităţilor, când MIU nu era o componentă primordială, căruia să i se dea mare importanţă. Într-adevăr, acest modul era dezvoltat doar pentru a asigura cerinţele minime de interacţiune cu programul. Ulterior, trecerea SE de la nivelul de experiment la nivelul utilizării industriale, crescând în acest fel numărul utilizatorilor și a pretenţiilor lor, MIU a căpătat o mare importanţă, și s-a pus tot mai mult accentul pe el. În această direcţie, s-au făcut cercetări pentru a se determina aspectele care influenţează acceptarea SE. Aceste cercetări au arătat că pentru a creşte acceptarea utilizatorilor, interfaţa cu utilizatorul a unui SE trebuie să fie uşor de învăţat și prietenoasă, și de asemenea să se integreze perfect la locul de muncă. Pe de altă parte, majoritatea utilizatoriilor SE au un bagaj de cunoștințe mai mult sau mai puţin mare din domeniul SE, și doresc să se implice în rezolvarea problemei, și să înţeleagă modul în care a rezultat soluţia găsită [1].

Considerând rezultatele studiilor şi a experienţei acumulate în anii de exploatare a SE, MIU este considerată componenta principală care intervine în acceptarea timpurie a unui SE, iar rolul ei este considerat ca atare. În consecinţă, noile SE cuprind interfeţe care au fost dezvoltate independent de celelalte componente, după ce s-au analizat toate aspectele care influenţează eficienţa.

Importanța MIU poate fi sintetizată astfel:

- El permite utilizatorilor fără expertiză tehnică sau specifică domeniului să beneficieze de sistemul expert.
- O interfață bine concepută asigură o interacțiune fluidă, reducând erorile și frustrarea utilizatorilor.

- Prezentarea clară și logică a informațiilor consolidează încrederea utilizatorilor în rezultatele sistemului.
- MIU optimizează procesul decizional prin minimizarea interacțiunilor inutile și prezentarea promptă a rezultatelor.

10.3. Rolul MIU

Modulul de Interfaţă cu Utilizatorul, aşa cum se observă şi în figura 2.1, este în legătură directă cu exteriorul, și anume cu utilizatorul, respectiv Modulul Explicativ (ME) ca parte internă a SE. Prin intermediul Modulului Explicativ interacţionează cu Motorul de Inferenţă (MI) şi Baza de Cunoștințe (BC). În această direcţie, rolul MIU este de a comunica MI cererile utilizatorului, rezultatul fiind furnizat înapoi celui din urmă.

Modulul de interfață utilizator îndeplinește următoarele funcții principale:

- Gestionarea intrărilor - capturează interogările, observațiile sau faptele utilizatorului într-o formă structurată sau nestructurată (de exemplu, prin meniuri, formulare, limbaj natural sau prompturi de comandă).
- Prezentarea ieșirilor - modulul afișează concluziile, explicațiile și recomandările sistemului într-un mod clar și ușor de înțeles. Pot fi incluse materiale vizuale precum diagrame, tabele sau căi de diagnosticare pentru a îmbunătăți înțelegerea.
- Gestionarea interacțiunii - ghidează utilizatorii prin dialoguri, ajutându-i să furnizeze informațiile necesare prin adresarea întrebărilor adecvate. Interfața se adaptează în funcție de contextul sesiunii și de calea de inferență a sistemului.
- Suport pentru explicații - multe sisteme expert includ o „facilitate de explicații” prin intermediul interfeței utilizator care permite utilizatorului să pună întrebări de tipul „De ce?” și „Cum?” - clarificând raționamentul din spatele concluziilor sau necesitatea anumitor intrări.
- Personalizare și control - utilizatorilor li se pot oferi opțiuni de control, cum ar fi alegerea nivelului de detaliu în răspunsuri, comutarea între moduri (începător/expert) sau ajustarea preferințelor sistemului.

Sistemele Expert recente, ridică noi probleme de imagine, deoarece se doreşte ca aceeaşi interfaţă cu utilizatorul să îndeplinească rolul şi de achiziţie de date, adică să interacţioneze și cu inginerul de cunoștințe. Astfel, există SE care deja sunt concepute să faciliteze achiziţia enunţului problemei şi a anumitor date

de intrare (achiziţia datelor se poate face direct prin intermediul dispozitivelor specializate, sau prin introducerea de către utilizator / inginerul de cunoștințe).

10.4. Caracteristicile MIU a unui SE

Sistemele Expert au necesităţi mai speciale în ceea ce priveşte interfaţa cu utilizatorul. Cerinţele care trebuie să le îndeplinească interfaţa unui SE derivă din faptul că aceasta este în legătură directă cu modulul explicativ, iar în unele situaţii şi cu modulul de achiziţie a cunoștințelor. Aşa cum s-a mai subliniat, studiile au arătat faptul că acceptarea precoce a unui SE este direct influenţată de interfaţa cu utilizatorul, respectiv de modul în care sunt afişate explicaţiile.

Într-un SE, MIU trebuie să asigure echilibrul dintre comunicare, control şi acces [2], aşa cum este descris prin graficul din figura 10.1.

Precum se observă în figura 10.1., problema comunicării se referă la dialogul dintre sistem și utilizator, în particular la cine iniţiază dialogul în situaţiile cheie ale procesului de soluţionare. În continuare se dă un exemplu pentru a înţelege acest aspect.

În ceea ce priveşte controlul, interfaţa prezintă elemente care se referă la cine ia deciziile în situaţiile critice. La exemplul 1, cine este cel care ia deciziile ? În comparaţie cu acesta, există SE care lucrează cu utilizatori neexperimentaţi, la care sistemul este cel care ia deciziile.

Accesul se referă la toate elementele unei interfeţe care permit utilizatorului să înţeleagă tot mecanismul și raţionamentul prin care s-a rezolvat problema. La interfaţa unui SE aceste elemente cuprind: ferestre de comandă, meniu Help etc.

MIU poate să fie construit ca un program independent, care apoi să fie ataşat SE. În altă ordine de idei, complexitatea și cerinţele care trebuie să le îndeplinească noile SE impun ca acestea să nu fie rezultatul muncii doar a unui inginer de cunoștințe, ci a unei echipe compusă din specialişti care crează anumite componente ale sistemului final.

O interfață cu utilizatorul modernă are următoarele atribute comune:

1. Cuprinde pictograme, formulare și meniuri.
2. Are capacități de procesare a limbajului natural (NLP – Natural Language Processing) pentru o interacțiune mai intuitivă.
3. Conține o intrare/ieșire bazată pe voce în sisteme avansate.
4. Oferă posibilitatea de înregistrare în jurnal și istoricul sesiunilor în scopuri de auditare sau revizuire.

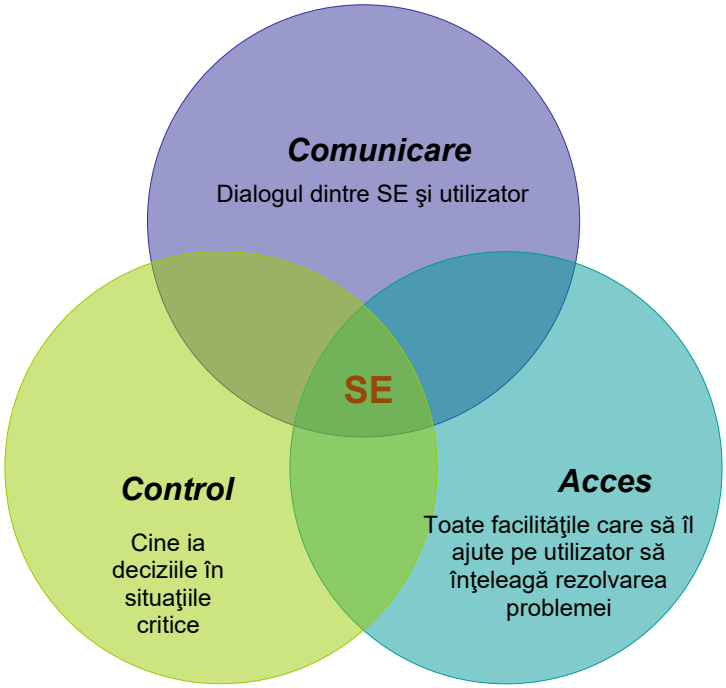


Fig. 10.1 Caracteristicile MIU a unui SE

Un MIU dezvoltat bine este esențial pentru transformarea unui sistem expert dintr-un procesor pasiv de cunoștințe într-un instrument interactiv de asistență decizională. Designul și funcționalitatea sa au un impact direct asupra satisfacției utilizatorului, eficacității sistemului și succesului general al sistemului expert.

10.5. Metodologii de realizare a MIU

Realizarea interfeţei grafice cu utilizatorul al unui SE poate să fie o muncă mai mult sau mai puţin laborioasă. Inginerul de cunoştinţe poate să construiscă un SE de la zero, în situaţia în care acesta va dezvolta fiecare componentă a SE în funcţie de diferite criterii. O altă variantă care poate fi abordată este utilizarea unui SE cadru (Shell Expert System – Sistem Expert scoică), care are anumite componente pre-definite (Motorul de Inferenţă, Interfaţa cu Utilizatorul), celelalte trebuind să fie dezvoltate de inginerul de cunoștințe (Baza de Cunoștințe).

Sistemele Expert Cadru se caracterizează printr-un nivel conceptual de reprezentare a cunoștințelor, respectiv de realizare a inferenţelor, la care inginerul de cunoștințe care l-a creat a dezvoltat raţionamentul motorului de inferenţă, modulul de interfaţă cu utilizatorul şi modulul de achiziţie a cunoştinţelor. sistemele expert cadru sunt dezvoltate pentru a răspunde necesităţilor unor

categorii largi de utilizatori, respectiv a rezolva mai multe tipuri de probleme. Pentru a răspunde acestor deziderate, MIU are de suferit, având un caracter general, care nu poate servi la maxim o anumită problemă.

Utilizarea Sistemelor Expert Cadru oferă avantaje în ceea ce priveşte introducerea cunoștințelor şi afişarea rezultatelor, și rapiditatea cu care este finalizat Sistemul Expert. Marele dezavantaj care apare în utilizarea acestor SE suport, în ceea ce priveşte interfaţa cu utilizatorul este faptul că acestea având o interfaţă predefinită defavorizează SE în interacţiunea cu utilizatorii.

În concluzie, despre dezvoltarea MIU atunci cînd se lucrează cu un sistem expert cadru se pot afirma următoarele:

- Sistemul exprt cadru include interfețe bazate pe formulare, meniuri sau în stil întrebări-răspunsuri.
- Prin folosirea sistemului expert cadru se obține un prototip rapid, astfel, abordarea este utilă în etapele incipiente de dezvoltare sau în scopuri experimentale.
- Acesată abordare presupune un efort redus de programare, întrucât SE cadru oferă un modul de interfață gata de utilizare.
- Interfața dezvoltată este constrânsă de capacitățile sistemului expert cadru.

În continuare sunt descrise caracteristicile generale şi particularităţile de realizare a interfeţelor grafice bazate pe manipulare directă, respectiv pe dialogul cu utilizatorul.

Manipularea directă

O interfață de manipulare directă permite utilizatorilor să interacționeze direct cu reprezentările vizuale ale datelor sau elementelor sistemului, mai degrabă decât prin comenzi abstracte sau intrări tastate. Caracteristicile tipice includ reprezentări grafice (de exemplu, diagrame de circuite, modele de flux de sarcină), interacțiune de tip point-and-click sau drag-and-drop, feedback imediat privind acțiunile utilizatorului, capacitate de anulare/refacere și interacțiune sensibilă la context (meniuri clic dreapta, date pop-up).

Provocările care apar atunci când se dorește folosirea manipulării directe pentru MIU sunt legate de faptul că dezvoltare este complexă, astfel necesită mai mult timp și resurse decât interfețele utilizator bazate pe text, apoi performanța în timp real (trebuie să fie receptivă pentru a asigura siguranța) și consecvența cu logica expertului, întrucât acțiunile interfeței trebuie să se alinieze cu raționamentul bazat pe reguli sau pe modele.

În concluzie, aspectele generale legate de manipularea directă sunt:

- Controlul permanent al utilizatorului asupra obiectelor care sunt reprezentate în limbajul maşină,
- Control foarte lin şi versatil,
- Caracteristica - nu există un dialog propriu-zis între calculator şi utilizator,
- Puncte slabe - memorarea datelor, vizualizarea rezultatelor şi tranziţia dintre diferite niveluri ale cunoaşterii,
- Formă mai accentuată sau mai subtilă de comunicare între utilizator şi calculator.

Interfeţe bazate pe dialogul cu utilizatorul

Interfeţele grafice cu utilizatorul bazate pe dialogul (text sau verbal) cu acesta reprezintă cea mai mare categorie de interfeţe om-maşină. Acestea se caracterizează printr-un nivel ridicat de comunicare dintre calculator și utilizator, comunicare care se realizează prin mai multe moduri: verbal, instrucţiuni text, răspunsuri, întrebări etc.

Idealul urmărit în dezvoltarea interfeţelor bazate pe dialog este ca acestea să se adapteze în funcţie de tipul utilizatorului. Aceasta este și cea mai mare provocare, deoarece presupune modelarea utilizatorului specific, ceea ce ridică probleme de ergonomie cognitivă, psihologie, ergonomia muncii etc. În consecinţă, interfaţa cu utilizatorul trebuie să se muleze pe diferite moduri de comunicare, și să aibă posibilitatea de a oferi unui tip de utilizator mai multe modalităţi de dialog.

O altă caracteristică a acestor interfeţe, care este sublinită în literatura de specialitate, este că aceste interfeţe ar trebui să poată gestiona evenimente specifice interacţiunii cu utilizatorul şi să le interpreteze corespunzător în cadrul dialogului cu acesta. Această caracteristică ridică interfaţa la nivelul de interfaţă inteligentă, deoarece presupune capacitatea de a recunoaşte situaţiile și modelul utilizatorului.

Metodologia de realizare a interfeţelor bazate pe dialog porneşte de la modelarea utilizatorului specific. De altfel, toate interfeţele bazate pe dialog sunt create pentru a deservi un model de utilizator. Scopul modelării utilizatorului este de a determina consecinţele comportamentului acestuia, respectiv cum să se achiziţioneze informaţia, să se reprezinte și cum să se afişeze rezultatele (soluţia şi raţionamentul).

Modelarea utilizatorului este un proces lung de determinare a trăsăturilor comune unui grup ţintă de utilizatori, în care trebuie să se aibă în vedere [1]:

- Vârsta medie a utilizatorilor;
- Genul;
- Locul de muncă;
- Caracteristicile şi condiţiile în care se lucrează;
- Cunoștințele deţiune de utilizator;
- Etc.

În domeniul SE majoritatea programelor au MIU bazate pe dialogul cu utilizatorul. Trebuie făcută observaţia, că acest aspect se subînţelege având în vedere existenţa Modului Explicativ. Ca și în cazul programelor convenţionale, provocarea în acest domeniu, este acela de a determina din etapa iniţială caracteristicile utilizatorului, necesităţile și ceea ce aşteaptă el de la Sistemul Expert. Aceasta presupune o analiză detaliată a utilizatorului pentru a dezvolta modelul utilizatorului specific.

În dezvoltarea interfeţelor grafice se utilizează şi modele de dialog, modelarea scopului respectiv recunoaşterea planului de lucru. Modelele de dialog se referă la cine iniţiază dialogul într-un moment de răspântie în rezolvarea problemei. Astfel sunt interfeţe la care dialogul este iniţiat de utilizator, de sistem sau combinat [1].

La SE la care interfeţele grafice sunt construite pe dialogul bazat pe utilizator, interacţiunea dintre SE și utilizator este mereu iniţiată de utilizator. Specialiştii, au observat că acest tip de dialog este preferat de către utilizatorii experimentaţi din domeniu de specialitate a problemei. Astfel în cazul acestor SE, utilizatorilor le este dată mai multă libertate de alegere, ei fiind cei care iniţiază dialogul şi decid asupra rezolvării problemei. În contrast, în cazul utilizatorilor ne-experimentaţi, care au mai puţine cunoștințe legate de problema de rezolvat, interfeţele grafice ale SE se bazează pe dialogul iniţiat de sistem. Acest fapt de realizează prin întrebarea utilizatorului despre acţiunile ce trebuie urmate în anumite momente ale procesului de rezolvare.

În practică, majoritatea SE folosesc în MIU un tip de dialog combinat, în care iniţiarea comunicării poate fi schimbată între SE şi utilizatori. Într-adevăr, sunt momente în rezolvarea problemei în care intervine utilizatorul şi transmite SE instrucţiuni, şi alte momente când SE a obţinut anumite valori care necesită o redirecţionare sau o clarificare a procesului de soluţionare.

Interfețe avansate

Interfeţele avansate reprezintă acea categorie de interfeţe grafice cu utilizatorul care se bazează pe noile descoperiri din domeniu, și respectiv pe principiul dialogului şi satisfaceri maxime a utilizatorului. Printre acestea se numără [1]:

- Interfeţele inteligente;
- Sistemele help active și pasive;
- Sistemele hypertext;
- Interfeţele adaptive;
- Sistemele Expert cooperative;
- Sisteme bazate pe limbajul natural
- Etc.

O caracterizare succintă a acestor tipuri de interfeţe avansate este realizată în tabelul 10.1.

Tabelul 10.1. Interfeţe grafice cu utilizatorul avansate [1]

Tip interfaţă	**Caracteristici generale**	**Observaţii**
Interfaţă inteligentă	Poate să reacţioneze în mod anticipativ, asistând în mod cooperant aplicaţia proprie-zisă şi utilizatorul	
Interfeţele adaptive	Sunt capabile să îşi modifice funcţionarea în funcţie de utilizator	Dezvoltate pentru un anumit tip de utilizator
Hypertext	Un text care conţine scurtături la alte texte, imagini sau tabele	Intră în componenţa altor interfeţe avansate
Sistemele Help active	Își focalizează căutarea pe utilizator (caracteristici și cunoștințe)	Sisteme de căutare on-line
Sisteme bazate pe limbajul natural	Conţin componente capabile să recunoască limbajul natural și să realizeze acţiuni accesate vocal	Implementate în sistemele de securitate

Sisteme Expert cooperative	Sisteme Expert care funcţionează pe o interacţiune sporită cu utilizatorul, astfel căutarea soluţiilor este dirijată de utilizator	

Interfeţele inteligente reprezintă ultima realizare în domeniul interacţiunii om – maşină, şi se caracterizează prin [1]:

- Imagine prietenoasă;
- Formă accesibilă;
- Stil simplu;
- Claritate;
- Adaptare în funcţie de caracteristicile utilizatorului (exemplul privind interfeţele inteligente).

Luând în considerare exemplul anterior, conceptul alternativ al interfeţei inteligente este de interfaţa adaptivă; dar acesta limitează atributele asociate acestor tipuri de interfeţe, care presupun schimbul „inteligent” de cunoștințe tehnice dintre om și maşină. Pe de altă parte, specialiştii din domeniu prezintă în literatura de specialitate preocupări în principal legate de atributele funcţionale ale interfeţelor inteligente. În acest sens, aceştia definesc acţiunile ce trebuie să le realizeze aceste interfeţe, cu scopul de a încuraja experimentările și minimiza erorilor, la care se adaugă și o „atitudine” prietenoasă faţă de greşelile utilizatorilor (figura 10.2).

Legătura dintre interfeţele inteligente și Sistemele Expert nu este tot timpul foarte evidentă, deoarece cele din urmă sunt considerate ca modele alternative ale primelor, având în vedere că interfaţa joacă un rol important în asistarea utilizatorului unui SE. De asemenea, diferite tipuri de SE au evoluat paralel cu interfeţele inteligente și au la rândul lor interfeţe avansate care le permit un dialog activ cu utilizatorii (exemplu: Sistemele Expert dedicate instruirii personalului). Pe de altă parte, unii ingineri de cunoștințe au apelat la interfeţe inteligente independente, care le-au ataşat ulterior programului final.

Tot în categoria interfeţelor inteligente sunt introduse şi unele Sisteme Expert care se bazează pe un nivel de transparenţă ridicat al modului de raţionament și a transmiterii informaţilor.

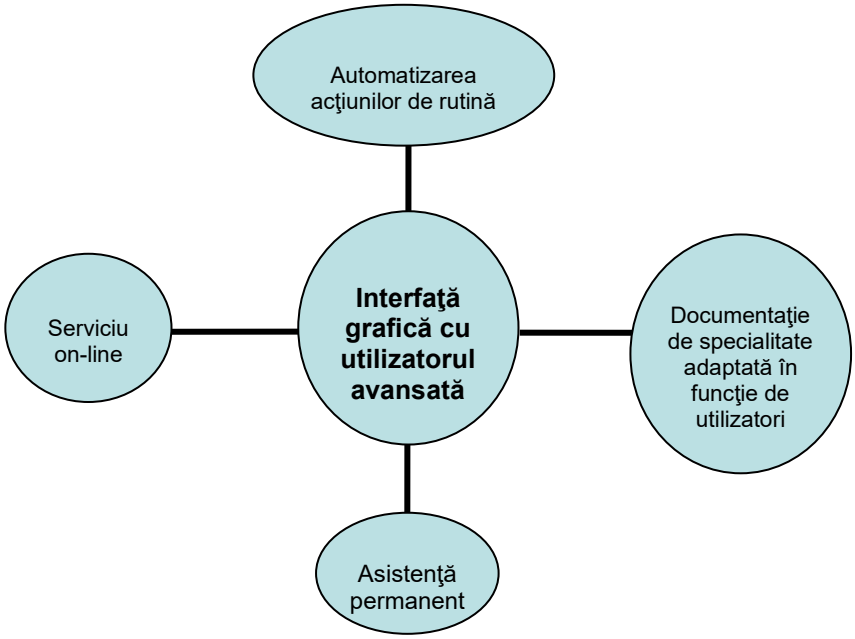


Fig. 10.2 Acţiunile interfeţelor prietenoase

O interfaţă inteligentă poate fi văzută ca o entitate „inteligentă" care mediază relaţia dintre doi sau mai mulţi agenţi care au informaţii incomplete și înţeleg parţial rolul şi caracteristicile celorlalţi participanţi.

Interfeţele WWW sunt noua revoluţie în ceea ce priveşte relaţia om – maşină, care a început odată cu anii 1990. Acestea se caracterizează printr-o grafică foarte prietenoasă şi accesibilă unui număr mare de utilizatori.

În acest fundal al globalizării în domeniul informaţiei, inginerii de cunoștințe ai Sistemelor Expert au realizat oportunitatea oferită de interfeţele Web, respectiv de conectarea Bazei de Cunoștințe la reţeaua de internet. Un avantaj urmărit de inginerii de cunoștințe este faptul că interfeţele Web cresc posibilitatea ca Sistemele Expert să fi utilizate de mai mulţi utilizatori ,și în acest fel acceptate mai uşor. De altfel, unele companii, înainte de a comercializa un program îl lansează Web cu un bagaj ales de informaţii pentru a vedea reacţia utilizatorilor.

În esenţă, un Sistem Expert care are o interfaţă Web, are aceleaşi componente ca un Sistem Expert care are o interfaţă simplă bazată pe text. În literatura de specialitate sunt date exemple de Sisteme Expert la care este schimbat Modulul de Interfaţă cu Utilizatorul cu interfeţe Web pentru a creşte accesabilitatea și utilizarea de către alte clase de utilizatori pentru care a fost construit [1].

Ultima tendinţă în domeniul interfeţelor WWW este combinarea acestora cu interfeţele de realitate virtuală, în special în cazul programelor inteligente dedicate instruirii personalului.

Interfeţele bazate pe realitatea virtuală permit utilizatorului să intre în realitatea expertului care îi oferă sfaturi şi cunoștințe. În acest sens se pot simula diferite situaţii, dezastre și deciziile ce trebuie luate.

10.6. Întrebări și exerciții

1. Modulul de Interfaţă cu Utilizatorul este în legătură directă cu Baza de Cunoştinţe şi Motorul de Inferenţă.

□ Adevărat/ □ Fals

2. Rolul Modului de Interfaţă cu Utilizatorul constă în trasmiterea utilizatorului

□ Explicaţii legate de mecanismul de rezolvare a problemei/ □ Starea Bazei de Cunoştinţe / □ Starea problemei la un moment dat.

3. Sistemele Expert care au un Modul de Interfaţă bazat pe dialogul cu utilizatorii consideră utilizatorul şi necesităţile sale elementul central

□ Adevărat / □ Fals

4. Modelarea utilizatorului specific depinde de următorii factori:
Genul / Vârsta / Preferințele culinare

5. În dezvoltarea MUI trebuie să se considere echilibrul dintre control, comunicare și acces.

□ Adevărat / □ Fals

11

Modulul explicativ
Tipuri de explicații și achiziția lor

Acest capitol descrie o parte esentiala a oricarui sistem expert, și anume modulul explicativ.

11.1. Introducere

Asemenea experţilor umani care se bazează pe cunoștințele lor acumulate și memorate cu care fac raţionamente influenţate de experienţă şi alţi factori subiectivi sau obiectivi, Sistemele Expert au o Bază de Cunoștințe (BC) unde sunt înmagazinate informaţiile, respectiv un Motor de Inferenţă (MI) care gestionează aceste cunoștințe pentru a soluţiona problema. Pe lângă cele două componente definitorii ale SE, acestea au şi alte elemente necesare: Modulul de Interfaţă cu Utilizatorul (MIU) și Modulul Explicativ (ME). Modulul de Interfaţă cu Utilizatorul este la fel de important la BC și MI, având în vedere faptul că acceptarea precoce a unui SE depinde în ceea mai mare măsură de acesta. Modulul Explicativ este o componentă a SE care le diferenţiază pe acestea de programele convenţionale de gestiune şi prelucrare a datelor. La aceste componente, aşa cum se poate observa din fig. 2.1, se adaugă și altele, care formează structura complexă a Sistemelor Expert moderne.

Acest capitol introductiv cuprinde informaţii generale legate de Sistemele Expert per ansamblu şi două dintre componentele acestora (care sunt și subiectul prezentului curs): Modulul Explicativ și Modulul de Achiziţie de Cunoștințe.

Structura de bază a unui SE cuprinde BC, MI, MIU și ME. Aceste componente au rol în eficacitatea sistemului, ele fiind necesare și definitorii. Pe lângă aceste elemente principale, Sistemele Expert recente cuprind alte elemente componente care au rol în creşterea eficienţei: Modulul de Achiziţie de Cunoștințe și Modulul Dinamic. Dacă luăm în considerare și bazele de date externe şi toate dispozitivele de achiziţie automată a datelor, lista se măreşte.

Modulul Explicativ este în legătură directă cu MIU, având în vedere că acesta este cel care intră în dialog cu utilizatorul, căruia îi trimite informaţiile necesare pentru a explica utilizatorilor funcţionarea SE și rezolvarea problemei. Tot în legătură directă este cu BC și MI, de unde îşi ia informaţiile pentru a popula baza de explicaţii. Având în vedere că pentru unii utilizatori este mai important decât BC, de multe ori ME este creat independent de BC şi MI (despre acest lucru se prezintă mai multe în capitolul 2 dedicat ME).

Modulul de Achiziţie de Cunoștințe (MAC) este o componentă auxiliară a unui SE, deoarece nu este necesară, însă creşte eficienţa SE în relaţia cu inginerul de cunoștințe. Această componentă poate lipsi şi atribuţiile ei preluate de alte componente sau de inginerul de cunoștințe. Totuşi, utilitatea acestui modul este de necontestat având în vedere că rolul lui este de a achiziţiona informaţiile și a le transforma pe acestea în limbajul calculatorului. Descrierea MAC este realizată în capitolul 3, unde se prezintă metodologia de realizare a MAC, caracteristicile programatorilor care realizează SE, numiţi ingineri de cunoștințe și achiziţia cunoştinţelor.

Abilitatea de a furniza explicaţii reprezintă o caracteristică definitorie a unui Sistem Expert. De altfel, utilizatorii sistemului judecă performanţele acestuia în funcţie de explicaţiile care le sunt furnizate şi modul în care se realizează acest lucru. Explicaţiile intră în componenţa elementului de bază al unui SE, și anume a ME.

Modulul Explicativ este componenta unui SE care furnizează utilizatorilor explicaţii legate de cunoștințele utilizate și raţionamentul folosit pentru a ajunge la o anumită recomandare sau decizie.

În dezvoltarea Sistemelor Expert structura şi imaginea Modulului Explicativ a evoluat odată cu acestea. Astfel, Sistemele Expert originare conţineau module explicative primitive, care răspundeau unor cerinţe de bază ale utilizatorilor, iar informaţiile care le conţineau erau derivate din Baza de Cunoștințe. Ulterior s-a observat faptul că importanţa explicaţiilor este mare, și s-a dorit eficientizarea acestora prin crearea unor BC care să răspundă atât atribuţiile privind rezolvarea problemei cât şi formarea explicaţiilor. Cercetările au arătat faptul că modul în care sunt reprezentate cunoștințele în BC influenţează construirea explicaţiilor. În această direcţie, tentativele nu au avut succes, deoarece aşteptările utilizatorilor au crescut, iar specialiştii SE au ajuns la concluzia că datele din BC nu sunt suficiente pentru a furniza utilizatorilor toate informaţiile dorite.

Studiile ulterioare au demonstrat faptul că este nevoie de un modul special, independent de restul componentelor SE, care să fie dedicat explicării soluţiei

găsite. Urmând această direcţie, au fost dezvoltate SE modulare, care conţineau elemente construite independent unele de altele și apoi unite într-un tot unitar. Astfel, Modulul Explicativ a ajuns să fie o componentă aparte a unui SE care de multe ori începe a fi construit înainte de achiziţia cunoștințelor, sau în paralel cu aceasta.

În concluzie, rolurile ME în cadrul și în munca cu un SE sunt următoarele:

- Oferă justificare pentru concluzii, cum ar fi diagnosticarea defecțiunilor, deciziile de detașare a sarcinii sau recomandările de control.
- Crește încrederea arătând operatorilor cum a ajuns sistemul expert la concluzia sa.
- Ajută la instruirea personalului mai puțin experimentat prin expunerea logicii din spatele deciziilor experților.
- Asistă la depanarea bazei de cunoștințe ajutând dezvoltatorii să înțeleagă ce reguli s-au declanșat și de ce.
- Sprijină auditarea și conformitatea în mediile reglementate.

11.2. Etapele dezvoltării ME

Dezvoltarea ME urmăreşte etape asemănătoare dezvoltării MIU, şi anume: achiziţia, implementarea şi validarea explicaţiilor. Acest fapt reiese și din introducerea capitolului unde s-a făcut comparaţie între cunoștințe și explicaţii, respectiv procesele de obţinere ale celor două. În figura 11.1 este ilustrată schema de principiu a dezvoltării Modulului Explicativ.

Prima etapă, cea de achiziţie, presupune utilizarea unor unelte și tehnici (automate sau manuale) diferite de cele folosite pentru achiziţia cunoştinţelor, adică dialogul cu experţii. În cazul unor SE, o mare parte din explicaţii pot fi obţinute din Baza de Cunoștințe și din Motorul de Inferenţe. Însă pentru a completa baza de explicaţii este nevoie de mai multe informaţii, în special pentru a furniza explicaţii legate de modul în care funcţionează SE (meta-cunoştinţe). Achiziţia explicaţiilor se realizează în principal prin dialogul cu utilizatorii, deoarece explicaţiile sunt de multe ori mai importante decât soluţia în sine, având în vedere faptul că utilizatorii, în special cei experimentaţi nu vor accepta rezolvarea problemei dacă nu înţeleg modul în care a fost găsită soluţia. Astfel, trebuie să se pună accent pe această etapă, care se poate realiza chiar înainte de achiziţia cunoștințelor.

În literatura de specialitate se atrage atenţia asupra faptului, că achiziţia explicaţiilor trebuie să fie diferită de achiziţia cunoștințelor deoarece, poate să

apară conflicte între planificarea și coordonarea procesului de obţinere a cunoștințelor și a explicaţiilor.

Fig. 11.1 Schema de principiu privind dezvoltarea ME

Odată ce explicaţiile au fost obţinute, acestea se implementează de către inginerul de cunoștințe în SE, mai exact în Modulul Explicativ (baza de explicaţii), și apoi transmise utilizatorilor prin intermediul MIU. În această variantă, SE este un prototip, abia după ce se vor valida explicaţiile și toate componentele lui va deveni un sistem funcţional.

Validarea explicaţiilor presupune utilizarea unor teste și proceduri diferite de cele folosite pentru validarea cunoştinţelor, și anume prezentarea utilizatorilor Modulul Explicativ ca un program individual care ulterior va fi inclus în SE, respectiv alături de celelalte componente ca parte integrantă a SE prototip.

11.3. Tipuri de explicaţii

Sistemele Expert originare își bazau Modulul Explicativ pe explicaţii care răspundeau la întrebările De ce ? şi Cum ? Ulterior, odată cu dezvoltarea SE, explicaţiile ME au devenit mai complexe, pe măsură ce doleanţele utilizatorilor au devenit mai impunătoare. De altfel, Modulul Explicativ este mai important decât Baza de Cunoștințe pentru unii utilizatori. Din acest punct de vedere, explicaţiile pot fi mai importante decât cunoştinţele pentru a realiza diferite sarcini (SE de clasificare).

În dezvoltarea ME, există mai multe metode de clasificare a explicaţiilor, care depind de natura întrebării care determină explicaţia: ce ?, de ce ?, cum ?, când ? Aceste explicaţii, pot să fie reprezentate prin simple propoziţii, ecuaţii, imagini, grafice etc., care depinde de informaţiile de care au nevoie utilizatorii: cunoștințe terminologice, descriptive din domeniu și modului de rezolvare a problemei.

Experienţa în dezvoltarea SE, și a Modulului Explicativ a arătat că utilizatorii SE au nevoie de informaţii despre procedurile, raţionamentele, scopul, controlul şi cunoştinţe în general. Aceste necesităţi, clasifică explicaţiile în următoarele tipuri, care în multe surse bibliografice apar menţionate doar primele trei tipuri [1]:

- De ce ? – dau informaţii care justifică o acţiune întreprinsă de SE;
- Cum ? – dau informaţii despre drumul parcurs în baza de cunoștințe care a dus la soluţia găsită;
- Strategice – dau utilizatorului informaţii despre meta-cunoștințe (controlul și strategiile folosite de sistem);
- Ce ? – concepute pentru a da informaţii despre obiectele şi variabilele folosite de SE; "Ce înseamnă obiectul sau variabila *x*"
- Ce dacă ? – permite reaplicarea raţionamentelor cu alţi parametrii ai variabilelor de intrare.

Sistemele Experte recente au introdus explicaţiile de tipul "feedforward" şi "feedback" ataşate celor trei explicaţii de bază – de ce, cum şi strategice – figura 11.2.

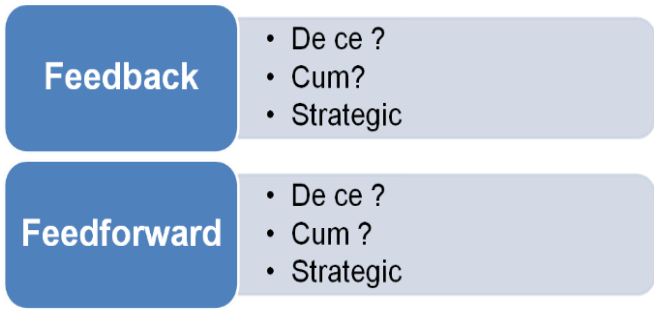


Fig. 11.2 Tipuri de explicaţii

Cele două tipuri de explicaţii se diferenţiază prin faptul că explicaţiile feedforward se referă la informaţii care sunt date utilizatorilor înainte de realizarea unei acţiuni, în contrast cu explicaţiile feedback care se focalizează pe ieşiri. Faţă de ultimul tip de explicaţii menţionate care pot să dea mai multe variante de răspuns, primele nu pot să prezinte o singură variantă. Combinând aceste două tipuri de explicaţii cu cele trei tipuri originare se obţin explicaţiile din tabelul 11.1.

Tabelul 11.1. Tipuri de explicaţii [1]

Tipul explicaţiei	Caracterizare	Exemplu
Feedforward de ce	– Justifică importanţa şi necesitatea informaţiei de intrare sau a unei proceduri ce urmează să se execute	– În etapa aceasta de rezolvare a problemei va fi utilizată metoda X, deoarece drumul parcurs până la obiectiv este mai scurt
Feedforward cum	– Detaliază modul în care informaţia de intrare se va utiliza şi procedura se va executa	
Feedforward strategic	– Clarifică modul în care informaţia de intrare care va fi utilizată este clasificată sau structurată, şi specifică modul cum fiecare intrare se integrează în planul de calcul	
Feedback de ce	– Prezintă importanţa şi clarifică implicaţiile unei concluzii găsite de SE	– Soluţia obţinută în această etapă va conduce la găsirea soluţiei optime
Feedback cum	– Prezintă drumul evaluărilor parcurse şi inferenţele folosite care au dus la soluţia găsită	
Feedback strategic	– Clarifică structura generală a metodei folosite de sistem pentru a ajunge la concluzia găsită şi specifică modul în care o anume o	

	inferenţă anume a determinat concluzia şi cum se integrează această inferenţă în întreg

11.4. Factorii care influenţează ME

În dezvoltarea ME trebuie să se aibă în vedere faptul că există mai mulţi factori care pot să îl influenţeze [1]:

- Tipul atribuţiilor (sarcinilor) care trebuie să le rezolve SE și contextul în care se află SE la un moment dat;
- Categoria explicaţiilor;
- Interfaţa şi modul de prezentare a explicaţiilor;
- Utilizatorii.

În dezvoltarea ME, tipul atribuţiilor şi contextul în care se află SE, adică tipul problemei de rezolvat sunt primii factori care trebuie luaţi în considerare. Tipul atribuţiilor care trebuie realizate de SE pot fi clasificate în multe categorii: atribuţii de analiză, atribuţii de sinteză, atribuţii de configuraţie, care depind de tipul problemei: clasificare, diagnoză, prognoză, control etc. Pe de altă parte, atribuţiile se împart în sub-categorii; de exemplu atribuţiile de analiză pot fi diagnoza şi prognoza.

Categoria explicaţiilor pot să influenţeze decisiv realizarea bazei explicaţiilor, unde explicaţiile se împart în două categorii în funcţie de tipul, respectiv conţinutul explicaţiilor. Tipul explicaţiilor se referă la cele trei clase: De ce ?, Cum ? şi Strategice. Conţinutul explicaţiilor depinde de clasa din care face parte. Astfel, explicaţiile De ce ? vor conţine mereu informaţii declarative legate de atribuţiile SE, cele Cum ? informaţii procedurale, iar cele Strategice vor prezenta date legate de modul în care funcţionează SE, adică meta-cunoştinţe [1].

Interfaţa cu utilizatorul şi modul de prezentare a explicaţiilor, mai exact facilităţile care le oferă MIU afectează modul în care se construiesc explicaţiile. Astfel, din acest punct de vedere al modului în care utilizatorul are acces la explicaţii prin intermediul MIU, acesta poate fi unul activ (utilizatorul nu cere explicaţii, ci SE le furnizează în funcţie de etapa la care se află), sau pasiv (utilizatorul cere explicaţiile).

Dintre cei patru factori care influenţează dezvoltarea ME, utilizatorii sunt cei care au cea mai mare importanţă. În considerarea acestui factor se au în vedere trei sub-factori: expertiza utilizatorilor (cunoștințele), diferenţele individuale dintre utilizatori și gradul de acceptare a SE de către utilizatori. În literatura de specialitate, se menţionează că expertiza utilizatorilor este factorul predominant, faţă de care se va dezvolta ME și se vor construi explicaţiile.

Modulul Explicativ este o componentă esenţială a unui Sistem Expert care trebuie dezvoltată în colaborare strânsă cu utilizatorii.

11.5. Întrebări și exerciții

1. Modulul Explicativ este o componentă auxiliară a unui Sistem Expert.
   □ Adevărat / □ Fals

2. Achiziţia cunoştinţelor se poate realiza de către inginerul de cunoştinţe şi de alte dispozitive dedicate.
   □ Adevărat / □ Fals

3. Inginerul de cunoştinţe poate să aibă atribuţii şi manageriale în dezvoltarea proiectului Sistemului Expert.
   □ Adevărat / □ Fals

12

Realizarea sistemelor expert

Acest capitol descrie etapele in realizarea sistemelor expert comerciale, dar care se pot aplica și în realizarea SE de laborator, a prototipurilor.

12.1. Introducere

Construcţia unui Sistem Expert este un proces laborios care presupune multe etape și implicarea unei echipe de specialişti care să culeagă cunoştinţele şi să le implementeze în limbajul maşinii. Pe de altă parte, noile pretenţii apărute pe piaţa programelor informatice presupun şi implicarea utilizatorilor.

În concluzie, timpul şi costul realizării unui Sistem Expert ridică un semn de exclamare legat de necesitatea unui astfel de program. De aceea, înainte de începerea procesului de construcţie a unui Sistem Expert trebuie ca dezvoltatorul să răspundă la o întrebare Este necesar un Sistem Expert ? Acest capitol răspunde la această întrebare, respectiv cuprinde aspecte introductive legate de realizarea Sistemelor Expert.

Sistemele Expert sunt programe extensive în construcţie şi mentenanţă, dar intensive în exploatare; adică utilizează cunoştinţe multe, dar într-un domeniu restrâns de activitate umană. Pe de altă parte, aceste sisteme oferă anumite avantaje rezultate din modul în care sunt construite. Comparativ cu expertul uman care este unic şi necesită un timp îndelungat de antrenare, SE poate fi dezvoltat în mai multe copii şi construit în câteva luni (timpul depinde de competenţa echipei proiectului Sistemului Expert).

Utilizarea unui SE reduce informaţiile care trebuie procesate de către utilizatorii umani, costurile de personal şi competenţele acestora. De altfel, SE recente sunt capabile să rezolve probleme care de multe ori depăşesc performanţele multor experţi umani. În plus, un SE este capabil să abordeze situaţiile apărute în rezolvarea unei probleme asemenea unor experţi umani, dar

nu este capabil să înveţe, deci să se bazeze pe experienţă, asemenea expertului uman.

12.2. Este Sistemul Expert soluția problemei ?

La această întrebare trebuie să răspundă orice manager de proiect care doreşte să soluţioneze o problemă prin intermediul Sistemelor Expert.

Utilizarea Sistemelor Expert în rezolvarea unor probleme este viabilă și abordabilă dacă următoarele afirmaţii sunt adevărate [1]:

- Problema nu este bine rezolvată în maniera clasică – nu există nici o metodă clasică cunoscută de rezolvare a problemei (programarea clasică – programe convenţionale) care să fie eficiente;
- Existenţa expertului uman – există cel puţin un expert uman disponibil, care cunoaşte bine domeniul de activitate studiat;
- Evoluţia cunoștințelor în domeniu este caracterizată de o dinamică rapidă - necesită frecvente schimbări.

În tabelul 12.1 este prezentată o comparaţie între abordarea clasică de rezolvare a problemelor şi utilizarea Sistemelor Expert. Pentru realizarea comparaţiei se iau în considerare tipul soluţiei de care este nevoie pentru a rezolva problema.

Tabelul 12.1. Comparaţie între metoda clasică și SE în rezolvarea problemelor [1]

Problema / Soluţia	**Precisă și stabilă**	**Precisă dar evoluează frecvent**	**Fluctuantă într-un domeniu bine stabilit**
Cunoscută	Programare clasică	SE uşor de actualizat ca urmare a evoluţiei	SE pentru că se poate adapta la fiecare problemă
Necunoscută	SE pentru găsirea soluţiei, apoi abandonat	SE pentru căutarea soluţiei, apoi adaptat pentru exploatare	SE pentru căutarea soluţiei, uşor de exploatat pentru că se adaptează la problema precisă

	în favoarea programării clasice		

Precum se observă din tabelul anterior, dacă o problema este precisă şi stabilă, dar soluţia este necunoscută, atunci se pretează utilizarea SE la început pentru a găsi o soluţie primară, apoi problema va fi transmisă unor programe convenţionale care pot să continue rezolvarea totală a problemei.

Sistemele Expert se utilizează cu succes în cazul cunoștințelor slab formalizate, în situaţiile în care nu se cunosc soluţiile, atunci când problema este fluctuantă, însă se cunoaşte bine domeniu de variaţie. De asemenea, Sistemele Expert se pretează la modificări locale numeroase, atât atunci când soluţia este cunoscută sau necunoscută.

Aceste programe este recomandat să se folosească într-un mediu de lucru caracterizat de o cotă a rotaţie de personal ridicată. Acest lucru permite memorarea experienţei acumulate de persoanele ce lucrează în domeniu (perenitatea informării).

În plus, tot din tabelul 1 reiese faptul că, Sistemele Expert lucrează cu probleme la care nu se cunosc variabilele de intrare, numărul de terminale, tipul lor, tipuri de comunicaţie numeroase, adică rezolvarea precisă a problemei.

Problemele care presupun multe calcule și nu prezintă acţiuni repetitive nu se pretează la rezolvarea cu Sistemele Expert.

12.3. Tipuri de Sisteme Expert

În dezvoltarea Sistemelor Expert trebuie să se ţină cont de tipul viitorului SE, deoarece fiecare tip are anumite caracteristici care presupune alt fel de abordare. În capitolul 2 s-au prezentat cele trei tipuri principale de SE [2]:

- Sisteme Expert de clasificare;
- Sisteme Expert de control;
- Sisteme Expert de prognoză.

În continuare se prezintă caracteristicile generale ale celor trei tipuri de SE, care au fost luate de la programe dezvoltate și utilizate cu succes.

Sistemele Expert de clasificare sunt cele mai utilizate programe inteligente; rolul lor fiind realizarea clasificării cauzelor posibile ale unei dis-funcţionări (cantitative, calitative, vizuale), deci sunt dedicate pentru diagnosticul tehnic. În

cazul acestor tipuri de SE, cunoştinţele sunt empirice, numite „cunoștințe de suprafaţă”. Dezavantajul utilizării acestor tipuri de cunoştinţe este că nu pot fi folosite pentru a justifica raţionamentul ce explică deducţiile realizate. Pentru a elimina acest punct slab s-au dezvoltat variante de SE care utilizează „cunoștințe profunde”. Dar aceste SE sunt dificil de construit, deoarece nu există metode adecvate de reprezentare a acestor informaţii detaliate [2].

Majoritatea SE de clasificare folosesc regulile de producţie pentru reprezentarea cunoştinţelor, doar că această metodă lucrează greu cu regulile de producţie deoarece această metodă este adecvată informaţiilor generale, astfel că se apelează la reprezentări sub forma descrierilor structură – funcţie. Pentru a uşura reprezentarea cunoștințelor, cele mai multe SE de clasificare folosesc atât cunoștințe de suprafaţă, cât şi profunde.

Sistemele Expert de clasificare utilizează un singur mod de raţionament, şi anume cel deductiv (strategia de control înainte). În plus, se aplică o căutare exhaustivă a soluţiilor, dar în această situaţie, timpul de rezolvare se prelungeşte.

Sistemele Expert de control se diferenţiază de cele de clasificare prin faptul că acestea trebuie să urmărească buna funcţionare a evoluţiei unui proces. Astfel, aceste SE realizează conducerea de procese. Pentru aceasta, ele cuprind obligatoriu un parametru cu ajutorul căruia vor supravehea datele sau semnalele provenite de la procesul controlat. Acest parametru, este timpul, iar toate celelalte mărimi care intră în controlul procesului depind de acesta, adică sunt lansate în execuţie sau nu în funcţie de valorile temporale [2].

Provocarea în cazul SE de control constă în interpretarea intervalelor de timp, şi realizarea unei structuri care să permită evoluţia datelor. Astfel, metoda de reprezentare a cunoştinţelor cea mai utilizată este tabelul cu date, iar selectarea strategiei depinde de mărimea controlată.

Principiul de funcţionare al acestor SE este următorul: o acţiune se va declanşa la momentul T, dacă toate condiţiile necesare sunt îndeplinite în acel moment. Din aceste Sisteme Expert se ramifică o categorie specială de SE care se folosesc la optimizarea proceselor cooperative.

Sistemele Expert de anticipare sunt folosite (aşa cum reiese din denumire) pentru anticiparea unui rezultat în funcţie de anumite restricţii şi resurse. Procesul de prognoză (anticipare) a soluţiilor unei probleme se bazează pe datele prezente și pe evoluţiile trecute ale acestora. Prognoza se poate realiza pe termen scurt, mediu sau lung. Cele mai utilizate SE de anticipare sunt cele pe termen scurt, deoarece implică mai puţine necunoscute ca cele pe termen mediu sau lung.

Este indispensabil în dezvoltarea unui Sistem Expert să se determine clasa de aplicaţie problemei de rezolvat, deoarece aceasta dictează structura datelor din Baza de Cunoştinţe și strategia de control abordată, precum şi toate activităţile adiacente acestora (achiziţia datelor, realizarea MIU etc.).

Un Sistem Expert (BC + MI) construit pentru un anumit scop (clasificare, control sau anticipare) nu poate să îşi schimbe specializarea. Astfel că un SE de prognoză nu va putea fi folosit pentru control sau clasificare şi invers.

12.4. Variante în construcţia Sistemelor Expert

În construcţia unui SE trebuie să se aibă în vedere că există mai multe variante: Sisteme Expert cadru sau Sisteme Expert specifice.

Dezvoltarea unui SE prin abordarea Sistemelor Expert cadru este cea mai simplă şi rapidă metodă de obţinerea a unui SE, deoarece această variantă presupune utilizarea unui program care are deja MIU, MI și structura BC. Rolul inginerului de cunoștințe este în popularea BC cu informaţiile specializate din domeniul problemei de rezolvat. Avantajele în această situaţie sunt: timp scurt de lucru şi obţinerea rapidă a unui SE funcţional. Dezavantajele constau în: strategia de control a datelor este impusă, astfel că trebuie să se acorde mare atenţie în alegerea SE cadru, MIU este simplist, neatractiv pentru utilizatori și nu este dedicat domeniului de studiu.

Sistemele Expert specifice sunt cele dedicate rezolvării unei probleme particulare, fiind în totalitate creat pentru aceasta. Astfel că, componentele sistemului sunt atent alese şi dezvoltate pentru a da randamentul maxim. Avantajul acestei abordări este că tot SE este specializat în rezolvarea problemei dorite. Însă costul acestui avantaj este timpul mare necesar creării acestui Sistem Expert.

În ultimii ani, pe fondul eficientizării procesului de construire a Sistemelor Expert, dezvoltatorii acestora folosesc ambele variante, şi anume:

- Sisteme Expert cadru în prima etapă în care dezvoltă macheta SE, și o testează pe piaţă dacă este fezabilă;
- Dacă reiese că dezvoltarea unui SE este necesară și profitabilă atunci se construieşte un Sistem Expert specific cu toate atuurile necesare pentru a satisface rigorile utilizatorilor.

12.5. Etapele în dezvoltarea Sistemelor Expert comerciale

Procesul de dezvoltare a SE a evoluat odată cu trecerea acestora de la nivelul de programe experimentale de laborator la programe inteligente comerciale folosite în rezolvarea multor probleme din activitatea umană. Dacă la început se punea accentul doar pe programe, iar construcţia acestora se reducea la dezvoltarea Bazei de Cunoștințe, Motorul de Inferenţă, Modulului Explicativ și a Modulului de Inferenţă cu Utilizatorul (structura de bază a unui Sistem Expert), ulterior s-a trecut atenţia asupra utilizatorilor și a profitabilităţii construcţiei Sistemelor Expert.

Procesul de realizare a unui SE cuprinde mai multe etape care se desfăşoară în paralel și secvenţial. Într-adevăr, obţinerea unui Sistem Expert comercial presupune construcţia programului inteligent eficace pe de o parte, și pe de altă parte a unui SE eficient. Eficacitatea unui SE se bazează pe structura acestuia, iar eficienţa pe elementele sale auxiliare. Dacă primul atribut se obţine prin dialogul cu experţii umani, adică achiziţia cunoştinţelor și realizarea deducţiilor logice, eficienţa se dobândeşte prin comunicarea cu viitorii utilizatori.

Construirea unui SE eficace este un proces care presupune mai multe etape care se realizează secvenţial, dacă sunt interdependente sau paralel dacă sunt dependente. Aceste etape au fost descrise în cursurile precedente:

- Achiziţia cunoştinţelor și construirea Bazei de Cunoștințe;
- Realizarea Motorului de Inferenţă;
- Achiziţia explicaţiilor şi dezvoltarea Modulului Explicativ;
- Realizarea Modulului de Interfaţă cu Utilizatorul;
- Construirea celorlalte elemente componente ale SE: Modulul de Achiziţie a Cunoştinţelor, Modulul Dinamic etc.

Obţinerea unui SE comercial se realizează în paralel cu construirea SE eficace şi eficient; iar procesul se poate împărţi pe trei perioade [2]:

- Studiu de fezabilitate;
- Realizarea prototipului;
- Dezvoltarea SE final, comercial.

În figura 12.1 este ilustrat graficul care descrie cele trei etape de obţinere a SE comercial, în paralel cu etapele de realizare ale SE.

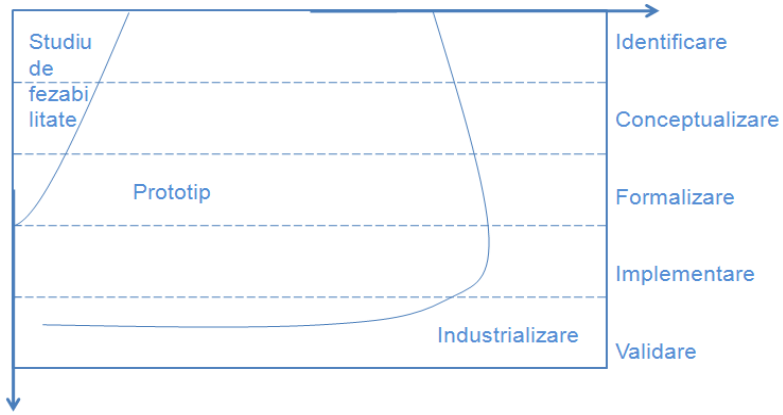


Fig. 12.1 Etapele procesului de construire și dezvoltare a unui SE

Precum se observă în figura 12.1, etapa de fezabilitate se realizează paralel cu etapa de identificare, conceptualizare şi formalizare a problemei. Pe de altă parte studiu de fezabilitate se desfăşoară în acelaşi timp cu realizarea prototipului, care la rândul ei se suprapune peste procesul de achiziţie a datelor și implementarea acestora. Validarea Sistemului Expert începe încă din starea de prototip a SE şi se încheie în etapa de industrializare a SE comercial.

În continuare, vor fi descrise etapele de obţinere a Sistemelor Expert comerciale.

Etapa studiului de fezabilitate

Etapa studiului de fezabilitate are drept scop obţinerea unor informaţii legate de necesitatea și profitabilitatea utilizării unui Sistem Expert în rezolvarea problemei dorite. Timpul dedicat unui studiu de fezabilitate este scurt de maxim o lună.

Această etapă presupune cercetări legate de identificarea, conceptualizarea şi formalizarea problemei de rezolvat prin dialogul cu viitorii utilizatori, cu experţilor din domeniu și prin studiu pieţei.

Studiu de fezabilitate se realizează cu ajutorul unei machete de Sistem Expert, care de cele mai multe ori se obţine prin utilizarea uneltelor suport (SE cadru).

În realizarea unei machete nu se iau în considerare aspecte legate de optimizarea timpului de răspuns şi a memoriei ocupate, sau de validarea formalismului utilizat; se utilizează reprezentări simple şi flexibile, nuclee existente, fără să urmărească validarea formalismului utilizat.

Caracteristicile unei machete sunt:

- Redă anumite cazuri particulare (tipice) ale problemei studiate;

- Este total independentă faţă de SE final;
- Cuprinde structura de bază a unui SE și un bagaj minimal de cunoștințe.

Etapa de prototip

Etapa de prototip permite verificarea rapidă, într-o perioadă de timp ce variază de la 2 la 4 luni, dacă o anumită variantă de SE este eficace [2].

Diferenţa dintre o machetă a unui SE şi un prototip asociat nu este tot timpul foarte clară, deşi există diferenţe între cele două. În tabelul 2 se prezintă o comparaţie între macheta și prototipul unui Sistem Expert.

Tabelul 12.2. Comparaţie între macheta și prototipul unui SE

Tipul SE **Caracteristica**	**Macheta**	**Prototip**
Relaţia cu SE final	Macheta este independentă de SE final	Prototipul cuprinde subansambluri ale SE final
Baza de Cunoştinţe	Reprezentarea cunoștințelor se realizează printr-o singură metodă, fără să se pună accentul pe eficienţa şi dimensiunile BC	În realizarea prototipului se încearcă mai multe metode de reprezentare a cunoştinţelor, iar BC se modifică, crescându-şi dimensiunile pe măsură ce dialogul cu experţii umani avansează
Motorul de Inferenţe	Se utilizează un singur tip de raţionament în MI, și nu se ia în considerare formalismul utilizat	Se utilizează mai variante de raţionamente logice, strategii de control și algoritmi de căutare, pentru a se ajunge la cea mai eficientă variantă
Problema rezolvată	Cuprinde doar anumite cazuri tipice ale problemei studiate	Prezintă soluţii la cazuri mai complexe ale problemei, care sunt testate şi comparate pe parcursul dezvoltării

Metoda abordată pentru dezvoltare	Se utilizează SE cadru	Se abordează SE cadru
Timpul de utilizare	Se dezvoltă rapid Se utilizează pe durata studiului de fezabilitate, care este de aproximativ 1 lună Se abandonează, atunci când prototipul cuprinde toate cazurile soluţionate cu ajutorul machetei	Se dezvoltă pe tot parcursul achiziţiei datelor Se utilizează între 2 şi 4 luni De cele mai multe ori se abandonează, şi se începe dezvoltarea unui SE comercial

La finalul etapei de prototip, managerul de proiect a SE, sau inginerul de cunoștințe trebuie să aibă o idee precisă asupra viitorului Sistem Expert. În consecinţă, prototipul va fi abandonat, preluându-se anumite părţi ale acestuia sau nici una, ci doar rezultatele obţinute privind reprezentarea cunoştinţelor și a inferenţelor aplicabile.

Abandonarea este justificată deoarece preluarea prototipului şi construirea pe bazele lui poate să afecteze eficacitatea şi eficienţa viitorului SE.

Etapa de industrializare

Etapa de industrializare este ultima în construirea SE comercial. La acest nivel se ştie foarte exact problema şi modul de rezolvare, respectiv metodele de reprezentare a cunoștințelor și a raţionamentul de gestiune a datelor.

Dezvoltarea Sistemului Expert final se poate realiza prin două modalităţi. Metoda clasică presupune descompunerea problemei în sub-probleme independente unele de altele. În comparaţie cu această metodă, abordarea de viitor implică verificări experimentale și luarea în considerare a tuturor programelor create înainte: machetă şi prototip.

Pornind de la experienţa cumulată odată cu dezvoltarea prototipului SE, și anume a cunoştinţelor culese şi a modului de rezolvare a problemei de către expertul uman, se dezvoltă SE comercial urmărindu-se satisfacerea condiţiilor de eficacitate şi eficienţă. Trebuie specificat faptul că extracţia cunoștințelor este lungă și delicată, ea consumă o mare parte din procesul de realizare a Sistemului Expert. De altfel, Baza de Cunoștințe evoluează pe întreg parcursul etapei de industrializare, acumulând date reale și complexe legate de problema de rezolvat.

Cunoștințele incluse în BC sunt verificate şi validate continuu, astfel că SE va deveni utilizabil înainte ca BC și obiectivul final fixat să fie realizat complet.

> Realizarea unui Sistem Expert comercial este un proces dinamic care poate să se schimbe pe măsură ce se descoperă noi doleanţe ale utilizatorilor, respectiv cunoștințele se modifică odată cu dialogul cu experţii umani.

12.6. Sisteme Expert Cadru

Sistemele Expert pot să fie construite de la bază ezultând Sisteme Expert specifice, sau să fie utilizate nişte programe suport denumite, Sisteme Expert cadru.

În construirea Sistemelor Expert comerciale, managerii proiectelor sistemelor apelează de cele mai multe ori la SE cadru pentru a crea machetele SE şi a le verifica fezabilitatea, urmând ca ulterior după ce s-a hotărât asupra caracteristicilor și atribuţiilor SE să se folosească limbajele de nivel înalt pentru a construi un Sistem Expert specific.

Sistemul Expert cadru este un program specializat, mai exact un instrument pentru dezvoltarea Sistemelor Expert, care conţine toate elementele necesare dezvoltării şi utilizării Sistemelor Expert [1].

Istoria Sistemelor Expert cadru a început cu SE MYCIN (dedicat diagnosticării infecţiilor sangvine), la care dezvoltatorii au observat că BC este independentă de MI. Astfel a rezultat EMYCIN, un program care nu are BC populată, ci cuprinde toate celelalte elemente componente. Ulterior, cu ajutorul EMYCIN a fost dezvoltat PUFF, un SE dedicat diagnozei afecţiunilor pulmonare. După această reuşită, au fost dezvoltate o mare varietate de SE cadru care sunt dedicate anumitor tipuri de probleme: monitorizare, clasificare, prognoză etc., sau anumitor domenii: economie, medicină, inginerie etc.

Caracteristicile Sistemului Expert cadru eficient

Sistemele Expert cadru conţin următoarele elemente componente:

- Baza de cunoștințe – această componentă nu este populată, dar este pregătită pentru a înmagazina cunoştinţele legate de problema de rezolvat;
- Motorul de Inferenţă – este o componentă care nu îşi modifică structura, ci va rămâne stabilă pe parcursul utilizării SE cadru. Noile variante de

SE cadru oferă posibilitatea utilizatorilor să îşi aleagă strategia de control, dintr-o listă predefinită;

- Interfaţa cu utilizatorul – componenta cu care utilizatorii intră în contact;
- Interfaţa cu inginerul de cunoștințe – interfaţa cu care SE cadru comunică cu dezvoltatorul SE particular;
- Modulul Explicativ – oferă utilizatorilor explicaţii cu referire la metoda de rezolvare a problemei;
- Interfaţa sistemului – componenta prin intermediul căreia se face legătura dintre sistem și programe externe acestuia, precum baze de date, algoritmi etc.;
- Modulul dinamic – partea din sistem unde se înmagazinează informaţiile recente.

În dezvoltarea SE prin utilizarea unui instrument suport, singura acţiune ce trebuie făcută este popularea Bazei de Cunoștințe din domeniul restrâns al problemei de rezolvat.

În prezent, pe piaţa SE există o multitudine de variante și opţiuni în alegerea SE cadru. Unele dintre ele sunt de dimensiuni mai mici, fiind dedicate anumitor domenii restrânse, respectiv instrumente suport de dimensiuni mari, care oferă mai multe facilităţi şi pot fi folosite într-un domeniu mai larg de activitate.

Experienţa specialiştilor şi evaluatorilor din domeniu Sistemele Expert cadru au evidenţiat o serie de caracteristici care diferenţiază SE cadru eficiente. Astfel, un SE cadru eficient trebuie să posede:

- Un grad de generalitate suficient pentru rezolvarea cel puţin unui tip de probleme, deoarece un sistem prea general posedă capacităţi neutralizate în detrimentul performanţelor;
- O metodă de reprezentare a cunoștințelor care să fie apropiat de modul în care gândeşte expertul; pe de altă parte să fie simplu şi universal;
- Un mijloc prin care inginerul de cunoștințe să aibă posibilitatea să ajungă la mecanismul de control;
- Capacităţi şi facilităţi de dialog elaborat pentru utilizatori și inginerul de cunoștințe.

În alegerea unui Sistem Expert cadru trebuie să se aibă în vedere toate caracteristicile necesare pentru dezvoltarea SE particular. Este de recomandat să se aleagă programe suport care au fost deja utilizate cu succes pentru rezolvarea

unor probleme similare. În figura 2 este ilustrată interfaţa cu utilizatorul al unui SE cadru.

Avantajele utilizării Sistemelor Expert cadru

Utilizarea Sistemelor Expert cadru în dezvoltarea Sistemelor Expert oferă următoarele avantaje: timp scurt de obţinere unui SE, performanţă ridicată și nu este nevoie de cunoștințe foarte aprofundate de programare.

Timpul scurt de obţinere a unui SE particular reiese din faptul că MI, MIU și ME sunt deja construite, rolul inginerului de cunoștințe fiind acela de a completa BC cu informaţiile particulare din domeniul problemei de rezolvat.

Având în vedere că toate componentele ale unui SE sunt deja construite, sau trebuie doar făcută legătura între ele, inginerul de cunoștințe nu trebuie să aibă cunoștințe aprofundate despre limbajul de programare folosit pentru crearea SE specifice.

Datorită acestor avantaje, SE cadru sunt folosite pentru dezvoltarea machetelor şi prototipurilor de Sisteme Expert.

Dezavantajul în utilizarea SE cadru este că nu se cunoaşte intimitatea sistemului, ceea ce poate să scadă performanţele sistemului particular.

12.7. Sistemele Expert specifice

Caracteristicile SE specifice

Un Sistem Expert specific se caracterizează prin faptul că este construit de la bază de către inginerul de cunoștințe prin utilizarea limbajelor de programare de nivel.

Majoritatea SE comerciale sunt Sistemele Expert specifice, ele având toate elementele componente și facilităţile necesare satisfacerii cerinţelor utilizatorilor. Astfel, structura unui SE specific este cea care cuprinde toate componentele principale şi auxiliare, precum este ilustrat în figura 3: Baza de Cunoștințe, Motorul de Inferenţe, Modulul Explicativ și Modulul de Interfaţă cu Utilizatorii, respectiv Modulul de Achiziţie de Cunoștințe, Modulul Dinamic şi Modulul de Interfaţă cu Inginerul de Cunoștințe. În pratică, în funcţie de cerinţele utilizatorilor, un SE specific poate să conţină și alte elemente componente, sau să îi lipsească unele dintre ele. În exemplu este prezentată interfaţa grafică a unui SE din domeniul medicinei.

În cazul SE specifice rolul inginerului de cunoștințe este de a crea şi dezvolta toate componentele SE. Având în vedere complexitatea unui astfel de program, SE comerciale sunt dezvoltate de către echipe de programatori.

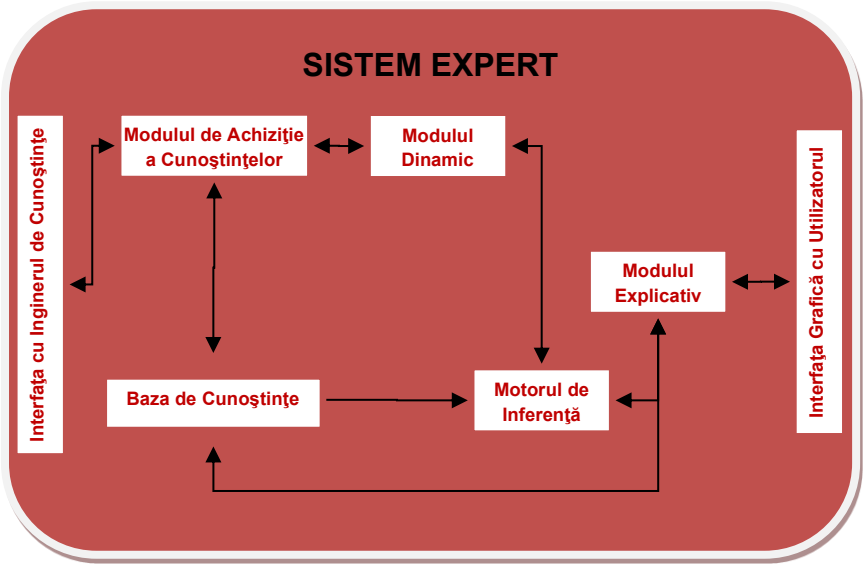


Fig. 12.3 Arhitectura complexă a Sistemelor Expert

Caracteristicile SE specifice sunt destul de variate, având în vedere multitudinea tipurilor de probleme rezolvate în prezent prin intermediul SE, respectiv a domeniilor abordate. Astfel că este dificil să se prezinte printr-o listă de elemente specifice. Pe de altă parte, toate SE specifice se caracterizează prin:

- Performanţă ridicată;
- Structură şi funcţionalitate dictate de către utilizatori;
- Supleţe deosebită.

Un Sistem Expert comercial este recomandat să fie un SE specific, deoarece doar prin crearea lui de la zero se poate obţine un program centrat pe utilizatori.

Avantajele SE specifice

Avantajul principal dat de către SE specific reiese din faptul că fiind dedicat rezolvării unei anumite probleme şi fiind centrat pe utilizatori, acesta conţine tote atuurile în a fi un program comercial și profitabil.

Dezavantajul SE specifice constă în faptul că este nevoie de un buget mare, timp îndelungat și mai mult personal pentru a le dezvolta.

12.8. Întrebări și exerciții

1. Abordarea Sistemelor Expert în rezolvarea unei probleme este viabilă dacă □ Există experţi umani în domeniu / □ Programarea clasică este eficace dar nu eficientă

2. Etapa de prototip este cea mai lungă etapă în realizarea unui Sistem Expert comercial
□ Adevărat / □ Fals

3. Marele avantaj al Sistemelor Expert specifice este că aceste programe sunt centrate pe utilizatori.
□ Adevărat / □ Fals

13

Instrumente pentru dezvoltarea Sistemelor expert

Prezentul capitol descrie principalele instrumente informatice folosite pana in prezent pentru dezvoltarea sistemelor expert.

13.1. Aspecte generale

Sistemele Expert au evoluat de la experimente de laborator ale univeristăţilor de prestigiu din Europa, Japonia şi S.U.A., la produse comerciale utilizate într-o gamă largă de domenii de activitate umană. Odată cu acestea s-au dezvoltat și instrumentele și limbajele de calculator folosite pentru construirea acestor programe.

În prezent, Sistemele Expert pot fi dezvoltate cu ajutorul limbajelor de programare de nivel înalt, respectiv a instrumentelor suport, denumite Sisteme Expert cadru.

Istoricul Sistemelor Expert este strict legat de naşterea IA ca ştiinţă şi sintaxă în anul 1956, când a fost prezentat la conferinţa de la Dartmouth College primul program de demonstrare a logicii propoziţiilor. În următorii ani după acest eveniment au apărut primele programe de demonstrare a teoremelor bazate pe logica propoziţiilor.

Etapa premergătoare apariţiei SE ca aplicaţii specifice a fost dezvoltarea unor programe informatice care să cuprindă principiile de bază ale acestora. Această etapă s-a realizat la puţini ani după conferinţa de la Dartmouth College din 1956, şi anume în anii 1960 – 1970, când s-au elaborat principiile majoritare în cercetarea arborescentă şi ideile de bază ale SE care se utilizează și în prezent.

Limbajele de programare specializate, sau programele inteligente au urmat evenimentelor prezentate anterior, precum se observă şi din figura 1. Astfel, în

anul 1959 a apărut primul limbaj de acest fel, numit LISP, dezvoltat de J. McCharty, urmând ca în 1962 acesta să publice primul manual LISP [1].

LISP (LISt Programming) este un limbaj care lucrează doar cu două entităţi: atomi şi liste; el fiind adoptat cu precădere de universităţile și companiile de cercetare din S.U.A. Listele sunt structurate în arborescentă binară. Dezavantajul limbajului de programare este că nu face deosebire între proceduri şi date, dar permite adăugarea de reguli sau de cunoștințe.

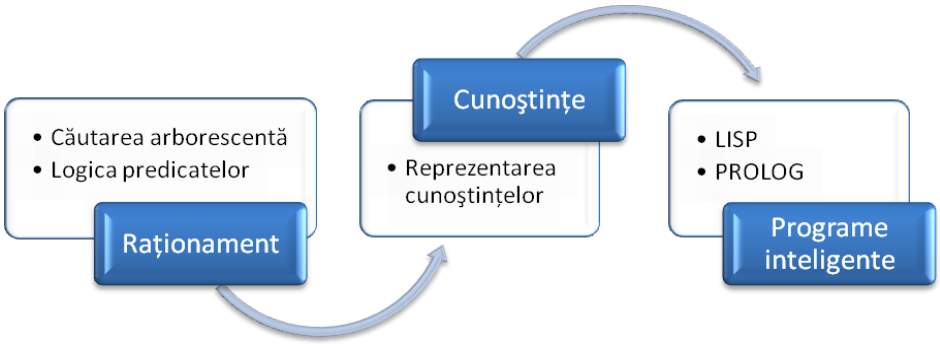


Fig. 13.1 Evoluţia programelor inteligente

La începutul anilor 1970 a apărut ideea de a utiliza logica predicatelor în realizarea limbajelor de programare. Prin urmare, între anii 1970-1975 a fost dezvoltat un nou limbaj de programare, PROLOG, bazat pe logica predicatelor de ordinul întâi. Acesta a fost elaborat de Colmerauer și Roussel, şi a fost intens folosit de atunci în special de cercetătorii din domeniu din Marea Britanie și Japonia. PROLOG (PROgramming in LOGic) are propria strategie de demonstrare şi este un instrument puternic de elaborare a SE.

După anul 1975 a avut loc o dezvoltare rapidă a programelor bazate pe tehnicile de IA pentru diferite utilităţi:

- Knowledge Acquisition System - KAS (1979);
- Expert (1979);
- Knowledge Engineering Environment - KEE (1983);
- Vp - Expert;
- C Language Integrated Production System - CLIPS (1984) etc. [8].
- Etc.

Unul dintre limbajele de programare care s-a impus este OPS, Official Production System, care a creat în 1975, de către Instructable Production System Project.

De-a lungul timpului au fost scrise diverse limbaje dedicate dezvoltării sistemelor de IA (LISP, PROLOG, CLIPS). Totodată au fost dezvoltate motoare de inferenţă specializate în aplicaţii specifice unui anumit domeniu și Sistemele Expert cadru.

Primul Sistem Expert cadru a fost dezvoltat din SE MYCIN, și a fost denumit EMYCIN, el cuprinzând o Bază de Cunoștințe nepopulată, Motorul de Inferenţe, Interfaţa grafică și Modulul Explicativ a vechiului MYCIN. După acest eveniment, au fost construite o mare varietate de sisteme suport dedicate anumitor tipuri de probleme, sau anumitor domenii de activitate, putându-se vorbi despre o piaţă a instrumentelor suport pentru dezvoltarea Sistemelor Expert specifice.

Clasificarea instrumentelor

Instrumentele cu ajutorul cărora se construiesc Sistemele Expert se împart în două mari categorii: limbaje de programare și sisteme cadru.

Limbajele de programare la rândul lor se împart în limbajele de nivel înalt care sunt dedicate scrierii Sistemelor Expert, precum: PROLOG, LISP şi OPS, şi limbajele convenţionale care au fost adaptate, dezvoltându-se dialecte ale acestora pentru scrierea SE: C++, CLIPS etc.

Sistemele cadru se împart în mai multe clase în funcţie de modalitatea de rezolvare a problemei, respectiv de domeniu căruia sunt dedicate. Astfel, în funcţie de metoda de rezolvare a problemei se identifică sisteme [2]:

- Bazate pe reguli - folosesc regulile de producţie pentru reprezentarea cunoștințelor. Motorul de Inferenţă a acestor sisteme foloseşte strategiile de control înainte, înainte sau mixtă. Aceste sisteme sunt cele mai răspândite instrumente suport și populare;
- Bazate pe cadre – reprezentarea cunoştinţelor se face prin intermediul cadrelor (frames). În cadrul acestor sisteme, gestiunea cunoştinţelor se face prin utilizarea unei tehnici aplicabilă asupra cadrelor, denumită transmiterea de mesaje (message passing). O altă metodă hibridă se obţine prin combinarea regulilor aplicabile sistemelor de producţie şi a metodei transmiterea mesajelor. În prezent, sistemele bazate pe cadre au evoluat, luând multe din caracteristicile programării orientate pe obiecte;
- Bazate pe exemple – aceste sisteme sunt denumite și sisteme bazate pe inducţie, deoarece aceste sisteme generează reguli pornind de la un

număr de exemple introduse de programator. Acest tip de sisteme cadru sunt recomandabil să se folosească în cazul problemelor la care există o bază mare de date cu exemple. Datorită modului de obţinere a bazei de reguli, în cazul utilizarea sistemelor bazate pe exemple nu este necesar să existe un expert uman din domeniu. Prin intermediul acestor sisteme cadru, se obţin SE specifice foarte rapid, iar BC poate fi uşor modificată și actualizată prin ştergerea sau adăugarea altor exemple reprezentative;

- Bazate pe logica fuzzy – aceste sisteme deservesc pentru dezvoltarea SE hibride. Prin intermediul acestor sisteme, reprezentarea cunoștințelor și deducţiile logice se realizează cu ajutorul numerelor și mulţimilor fuzzy, respectiv a operaţiilor corespunzătoare. Aceste sisteme devin tot mai populare, deoarece ele lucrează cu date caracteristice lumii reale;
- Bazate pe cazuri – aceste sisteme sunt cele mai recent introduse în domeniul ingineriei cunoștințelor. Aceste sisteme sunt diferenţiate de cele bazate pe exemple, dar în practică acestea sunt asemănătoare cu acestea, deoarece regulile sunt generate pe seama unor cazuri introduse de inginerul de cunoștințe.

O sinteză a categoriilor de instrumente și programe dedicate realizării Sistemelor Expert este prezentată prin intermediul diagramei din figura 2.

O altă modalitate de a clasifica instrumentelor cadru pentru dezvoltarea SE specifice este în funcţie domeniul de aplicaţie a acestora. Astfel, sunt instrumente cadru dedicate construcţiei SE din următoarele domenii [2]:

- Control – controlul diferitelor tipuri de procese;
- Proiectare – configurarea unor obiecte conform unor specificaţii;
- Diagnoză – detectarea cauzei unor funcţionări anormale sau a unor defecte;
- Instruire – caracterizarea şi instruirea, respectiv îmbunătăţirea activităţii şi a comportamentului personalului;
- Interpretare – indicarea unor situaţii pe baza unor date;
- Monitorizare – monitorizarea unor procese şi compararea situaţiilor reale cu aşteptările;
- Planificare – selectarea şi ordonarea unor activităţi în funcţie de anumite cerinţe, cu scopul de a atinge un obiectiv principal;
- Predicţie – prezicerea unor situaţii care pot să apară pe baza unor informaţii trecute;

- Soluţie de rezolvare a unui defect – recomandarea unei soluţii de rezolvare pentru a elimina un defect apărut sau o situaţie anormală;
- Programare – programarea resurselor și a timpului necesare realizării unei activităţi;
- Selecţie – identificarea opţiunii celei mai bune dintre o listă de posibilităţi;
- Simulare – modelarea interacţiunii dintre diferite componente ale unui sistem.

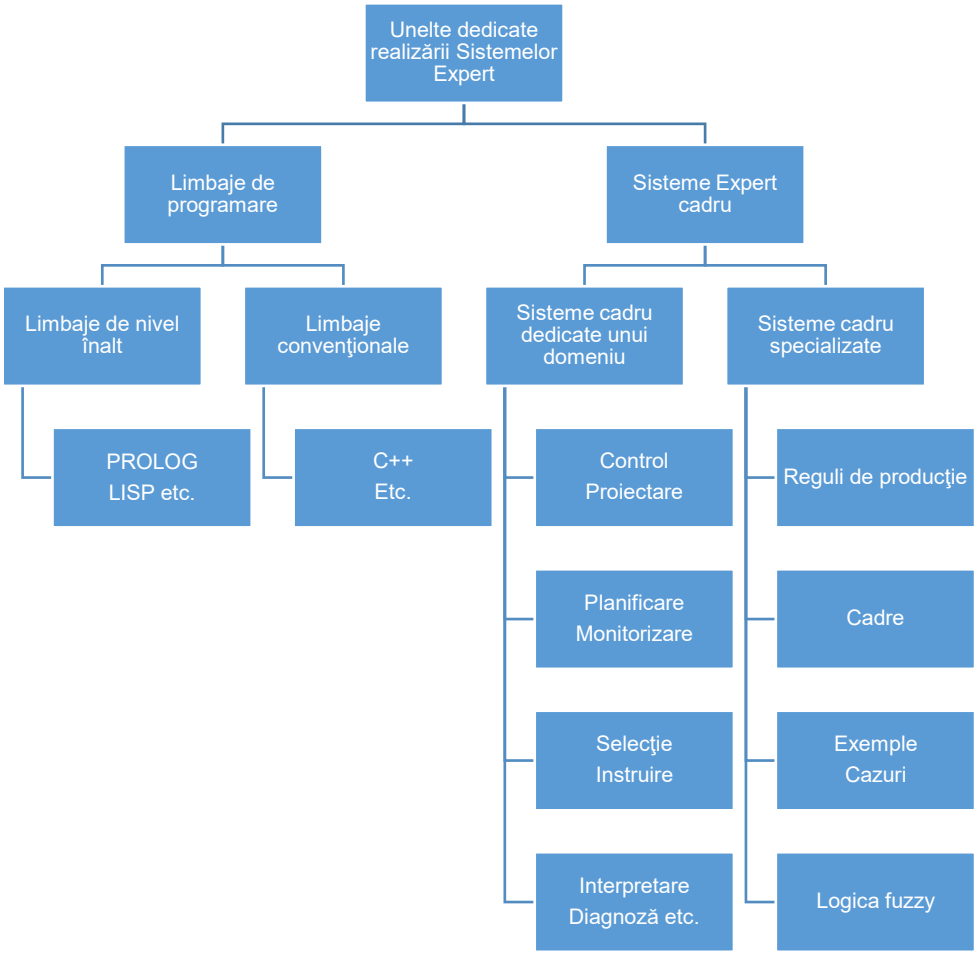


Fig. 13. 2 Clasificarea instrumentelor pentru construirea Sistemelor Expert

13.2. Medii de programare

Primul limbaj de programare creat cu scopul de a construi SE a fost LISP, de către cercetătorii din S.U.A., urmând apoi ca cei din Marea Britanie şi Japonia să dezvolte PROLOG cu acelaşi scop. Mai târziu a fost creat OPS de către universitatea Carnagie-Mellon University din S.U.A.

În ultimii ani, ingineri de cunoştinţe datorită dificultăţii cu care se lucrează cu limbajele de nivel înalt bazate pe logica propoziţiilor și a predicatelor, au apelat la limbajele convenţionale gen C și C++. Utilizarea limbajelor convenţionale le oferă programatorilor avantaje precum: experienţa de lucru, cu accesibilitate, cunoștințe mai puţine de programare avansată. Chiar dacă nu sunt dedicate dezvoltării Sistemelor Expert, limbajele convenţionale pot fi adaptate pentru construirea acestor programe.

În continuare, sunt prezentate caracteristicile generale, avantajele şi punctele slabe ale limbajelor de programare de nivel înalt: PROLOG, LISP, OPS şi CLIPS.

PROLOG

PROLOG, PROgramming in LOGic, este un limbaj de programare de nivel înalt care se bazează pe logica predicatelor de ordinul întâi. Acest limbaj a fost elaborat de Colmerauer și Roussel între anii 1970-1975.

PROLOG a fost unul dintre primele limbaje de programare logică, și rămâne cel mai popular din categoria sa, fiind preferat în defavoarea altora, mai ales că este gratuit și există multe aplicaţii drept exemplu. Iniţial, acest limbaj de programare a fost dezvoltat pentru a ajuta în procesarea limbajului natural, însă ulterior a ajuns să fie folosit și în alte domenii precum: Sistemele Expert, sisteme de control, jocuri etc.

PROLOG este un limbaj de programare declarativ, la care programele sunt create prin scrierea unor expresii care reprezintă regulile și faptele problemei.

În PROLOG, un program conţine două elemente, logica şi controlul. Logica se referă la „ce” face programul, iar controlul la „cum” o face. Pentru o mai bună înţelegere se poate face o analogie cu un program scris într-un limbaj convenţional: programul tradiţional conţine o serie de instrucţiuni care descriu acţiunile ce trebuie realizate secvenţial pentru a rezolva problema, un program în PROLOG conţine variabile și relaţiile dintre acestea (relaţii care leagă direct două variabile, sau relaţii care se pot deduce din altele) [4, 5]. În tabelul 1 se

poate observa diferenţa de conţinut şi sintaxă dintre un program în PASCAL și un program în PROLOG.

Tabelul 13.1 Analogie între un program în PASCAL și un program în PROLOG

Program	**PASCAL**	**PROLOG**

Conţinut	var x,y: real;
	begin
	write(‘numar1 = ‘);
	readln(x);
	write(‘numar2 = ‘);
	readln(y);
	if (x < y) then
	writeln(x)
	else writeln(y);
	end.
		predicates
		program
		mai_mare(real,real,real)
		clauses
		
		program :- write("n1="), readreal(X),
		write("n2="), readreal(Y),
		mai_mare(X,Y,Z),
		write(Z), nl.
		mai_mare(X,X,X).
		mai_mare(X,Y,Y):-X<Y.
		mai_mare(X,Y,X):-Y<X.
		goal
		program

Rol declarativ	Secţiunea *var* cuprinde variabilele	Secţiunea ***predicates*** cuprinde predicatele
Structură	Şir de *instrucţiuni*, executate în ordinea indicată în program	Colecţie de ***clause***, care descriu relaţia de ordine dintre variabile și logica programului Controlul – ordinea în care sunt definite și aranjate predicatele
Elementul de execuţie	Operatorul “<”	Predicatul “mai_mare”
Lansarea în execuţie	Momentul de start “begin” și de sfârşit “end”	Execuţia programului se face prin definirea scopului în secţiunea ***goal***. În exemplu, scopul este dat de predicatul program
Actualizarea variabilelor	*Atribuire*	***Unificare**** Variabila anonimă “_”

*Notă – Prin operaţia de unificare se analizează dacă două predicate pot fi identice. Dacă ele nu pot fi identice, procedura de unificare eşuează.

Variabilele sunt reprezentate prin nume simbolice, iar relaţiile dintre acestea sunt denumite clauze. Există două feluri de clauze: fapte şi reguli.

Structura sintetică a unui ***fapt*** este următoarea:

$$\boldsymbol{nume}(\boldsymbol{arg}_1, \boldsymbol{arg}_2, ..., \boldsymbol{arg}_n),$$

unde ***nume*** este un nume de predicat reprezentat printr-o succesiune de caractere alfabetice, cifre sau liniuţa de subliniere;

$\boldsymbol{arg}_1, \boldsymbol{arg}_2, ..., \boldsymbol{arg}_n$ sunt argumentele predicatului, care din punct de vedere sintactic pot fi nume de variabile sau nume de variabilă (un nume de variabilă este o succesiune de caractere).

Prima literă a numelui predicatului sau variabilei trebuie tot timpul să fie o literă mică.

O regulă este o structură sintactică de forma:

$$\boldsymbol{c}_{n+1} \textbf{ :- } \boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n.$$

unde: entităţile $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n, \boldsymbol{c}_{n+1}$ sunt de forma $\boldsymbol{nume}(\boldsymbol{arg}_1, \boldsymbol{arg}_2, \ldots, \boldsymbol{arg}_k)$, **nume**, $\boldsymbol{arg}_1, \boldsymbol{arg}_2, \ldots, \boldsymbol{argk}$ având aceeaşi semnificaţie ca în cazul unei fapte;

$\boldsymbol{c}_{n+1}$ se numeşte ***capul reguli***;

$\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n$ formează ***corpul reguli***.

Explicaţia unei clauze este de forma: “Dacă c_1 şi c_2 şi … c_n sunt adevărate, atunci c_{n+1} este adevărată.” Se observă din sintaxă că orice faptă şi orice regulă se încheie cu caracterul „ .”, iar „ ,” prezentă în corpul regulii suplineşte operatorul „şi”. Pentru operatorul „sau” se foloseşte „ ;”

Din punct de vedere sintactic, un program în PROLOG conţine unul sau mai multe module. Fiecare modul conţine mai multe secţiuni care sunt scrise într-o anumită ordine. În tabelul 13.2 sunt prezentate aceste secţiuni, în ordinea în care trebuie scrise.

Tabelul 13.2. Secţiunile unui modul al programelor în PROLOG

Nr.	Secţiune	Descriere sintactică	Observaţii
1.	Constante	***constants*** const1 = definiţie const2 = definiţie	În această secţiune se definesc constantele utilizate în program
2.	Tipul variabilelor	***domains*** tip_util1[tip_util2, ...] =tip1; tip2... lista1 = tip_element	În această secţiune se definesc tipurile variabilelor utilizate tip_util1, tip_util2, ..., lista1 sunt numele simbolice care desemnează tipurile de variabile definite de programator tip1; tip2 reprezintă o listă de tipuri standard sau definite de utilizator Exemplu: variabilele din lista1 sunt elemente de tipul tip_element
3.	Tipuri globale de elemente	***global domains***	În această secţiune, programatorul defineşte tipurile globale. Structura este asemănătoare structurii domains
4.	Date de baze	***database*** [-nume] ***global database*** [-nume]	În aceste secţiuni sunt definite bazele de cunoștințe dinamice.
5.	Predicatele	***predicates*** predicat1(tip_arg$_1$, tip_arg$_2$, ...,tip_arg$_n$)	Se definesc predicatele locale modulului. Fiecare predicat are un nume şi tipul argumentelor sale

		predicat2(tip_arg$_1$, tip_arg$_2$, ...,tip_arg$_k$)	
6.	Predicate globale	***global predicates*** predicat1(tip_arg$_1$, tip_arg$_2$, ...,tip_arg$_n$) predicat2(tip_arg$_1$, tip_arg$_2$, ...,tip_arg$_k$)	Se precizează predicatele globale și tipul argumentelor lor, precum și faptul că anumite argumente sunt de intrare sau de ieşire ar eventual, limbajul în care este scrisă procedura corespunzătoare predicatului
7.	Clauzele	***clauses*** predicat(arg$_1$, arg$_2$, ..., arg$_n$) predicat$_{k+1}$(...):- predicat$_1$(...), predicat$_2$(...), ..., predicat$_k$(...)	Secţiunea aceasta conţine definirea clauzelor.
8.	Scopul	***goal*** predicat1(...) [, predicat2(...), ..., predicatk(...)]	Scopul programului reprezintă o interogare privind faptele şi regulile definite în secţiunea clauses. Interogarea se poate realiza această secţiune, dar şi în fereastra de dialog. O formă de interogare o exclude pe cealaltă.

Notă – în paranteze drepte „[ ]” sunt elementele care nu este obligatoriu să fie introduse, sau utilizate

În PROLOG variabilele pot avea două clase tipuri: tipul compus și tipul elementar. Tipul elementar poate fi standard (tabelul 3) sau definit de utilizator. Tipurile definite de utilizator utilizează tot tipuri standard, la care utilizatorul le dă nume simbolice care au o anumită semnificaţie pentru problema de rezolvat.

Tipul complex cuprinde tipul compus și tipul listă. O listă este o înşiruire de elemente de acelaşi tip, separate prin listă și închise între paranteze pătrate. Tipul elementelor dintr-o listă poate să fie elementar sau complex. Un tip compus cuprinde un nume simbolic urmat de unul sau mai multe tipuri separate prin virgulă şi închise în paranteze rotunde.

LISP

LISP, LISt Processing, este o familie de limbaje de programare care îşi trag rădăcinile din al doilea cel mai vechi (primul limbaj de programare de nivel înalt care a fost utilizat pentru crearea primului SE este FORTRAN) limbaj de programare de nivel înalt, LISP care a fost folosit prima dată în 1958.

LISP a fost pentru prima dată creat cu scopul de a reprezenta notaţiile matematice în programele de calculator, de către John McCharty de la Massachusetts Institute of Technology, S.U.A. El a demonstrat că, cu ajutorul unor notaţii simple se pot crea algoritmi implementabili în calculatoare. Pentru reprezentarea acestor notaţii, McCharty a folosit expresii M, „M-expressions", care ulterior au fost transformate în expresii S. Caracteristica expresiilor M era că operatorii erau introduşi prin paranteze pătrate „[ ]", iar a expresiilor S erau parantezele rotunde „( )".

LISP a fost prima dată implementat pe un calculator IBM 704 în 1960 de către Steve Russel, când s-a obţinut primul compilator LISP, prin intermediul căruia se puteau verifica și executa aplicaţiile scrise în LISP. Programele scrise în acest limbaj primitiv aveau extensia „car" şi „cdr"; de altefl această extensie şi-au păstrat-o până astăzi aplicaţiile LISP. În următorul deceniu s-a încercat dezvoltarea acestui limbaj de programare, astfel au apărut mai multe dialecte LISP. Între anii 1980 şi 1990 mai mulţi cercetărtori şi specialişti au con-lucrat pentru a uni mai multe dialecte LISP, cu scopul de a obţine un limbaj care să beneficieze de avantajele acestor dialecte (ZetaLisp, NIL – New Implementation of Lisp etc.). În final a rezultat LISP-ul comun care este utilizat și în zilele noastre de o mulţime de ingineri de cunoştinţe, mai ales că există și un standard de operare common LISP - ANSI X3.226-1994 Information Technology Programming Language Common Lisp.

De la începuturile lui, LISP a suferit multe modificări, chiar radicale; el stând la baza multor dialecte care se folosesc astăzi. În prezent, cele mai populare

dialecte LISP sunt: LISP comun (Common LIPS), Scheme şi Clojure. În tabelul 3 se face o scurtă istorie a acestora.

Tabelul 13.3. Istoria familiei de programe LISP [6]

Dialect	**Caracteristici**
LISP 1	Prima implementare LISP
LISP 1.5	Prima versiune care a fost utilizată cu succes. Această variantă are anumite îmbunătăţiri faţă de LISP 1
Stanford LISP 1.6	Această versiune este succesoarea lui LISP 1.5 şi a fost dezvoltată de specialişti de la Stanford AI Lab. Această versiune stă la baza dialectelor Maclisp şi InterLisp.
MACLISP	Este descendentul direct a LISP 1.5, și a rulat pe sisteme Multics și PDP- 10
InterLisp	A fost dezvoltat de către BBN Technologies pentru a rula pe sisteme PDP- 10. Ulterior acest dialect a fost dezvoltat și implementat pe mai multe sisteme IT, precum maşinile Xerox Lisp
Standard Lisp și Portable Standard Lisp	Au fost folosite în mod intensive în special de către sistemele harware Alegra REDUCE (Computer Algebra System REDUCE)
ZetaLisp	A fost cunoscut și sub denumirea de Lisp Machine Lisp, care a fost folosit pe maşinile Lisp, și este descententul direct a MacLisp Acest dialect a avut o influenţă decisivă asupra Common LISP
Common Lisp (1984)	A fost dezvoltat pe baza dialectului MacLisp cu multe influenţe şi de la dialectul Scheme. Această versiune LISP a reprezentat încă de la apariţia lui dialectul LISP de bază, fiind acceptat de mulţi programatori drept standardul în domeniu, până la apariţia standardului ANSI Common Lisp (ANSI X3.226-1994)

Dylan	A fost prima versiune LISP care combina elemente ale dialectului Scheme şi a dialectului LISP comun orientat pe obiecte
IEEE Scheme	IEEE standard, 1178–1990 (R1995)
ANSI Common Lisp	Este un standard american care stă la baza lucrului cu dialectul LISP comun și a dezvoltării aplicaţiilor în acesta. În prezent, acesta este recunoscut și utilizat pe plan mondial
Clojure	Este un dialect modern a limbajului LISP, care este dedicat lucrului pe platforme Java

Un program în LISP este compus din expresii (LISP este un limbaj orientat pe expresii), între care nu se face distincţie. În momentul execuţiei, o expresie ia o valoare care va fi folosită în interiorul altor expresii.

O expresie în LISP este introdusă prin intermediul parantezelor rotunde „( )”. Aceasta reprezintă elementul caracteristic a acestui limbaj, care oferă aplicaţiilor dezvoltate o sintaxă foarte regulată, astfel că acestea sunt uşor de manipulat de către calculatoare.

Elementul de bază în LISP este lista. De altfel, chiar și funcţiile predefinite sunt reprezentate prin liste, ceea ce face mai uşoară gestiunea lor, ele fiind privite ca o mulţime ordonată de date. O listă este implementată prin elementele sale componente care sunt delimitate de paranteze şi despărţite prin spaţiu. În exemplul următor se prezintă mai multe exemple de liste.

(list '1 '2 'foo)	lista formată din elementele 1, 2 şi foo
(list 1 2 (list 3 4))	lista cuprinde 1, 2 şi lista 3 şi 4
(+ 1 2 3 4)	lista conţine o operaţie de adunare care va fi
()	evaluată, iar rezultatul este 10

În LISP sunt cuvinte predefinite care diferite roluri: ***list***, ***lambda***, ***defun***, ***cons***, ***Output***, ***append*** etc. [5] Aceste cuvinte apar între paranteze pe primul loc, urmate de restul expresiei. ***List*** este utilizat pentru crearea de liste, care conţin elemente de diferite tipuri de date (în exemplu 1 se poate observa acest fapt).

Lambda este un operator special folost pentru a lega variabilele de valorile corespunzătoare. O altă funcţie a acestui operator este de a crea şi introduce noi funcţii definite de utilizatori.

(lambda (arg) (+ arg 1)) leagă un argument de arg și returnează
argumentul + 1

(defun f (a) b...) defineşte o funcţie f cu argumentul a

Defun este utilizat pentru a introduce şi definii noi funcţii. Cuvântul ***cons*** este utilizat pentru a specifica faptul că o valoare este o constantă. Pentru a se afişa utilizatorului, adică pe ecran anumite date se utilizează ***Output***. Concatenarea a două liste se realizează prin cuvântul ***append***.

În LISP există puţine structuri de control, care sunt reprezentate de operatori speciali, prin intermediul cărora se „obligă" programul să realizeze anumite acţiuni.

OPS

OPS, Official Production System, reprezintă o familie de medii de programare dedicate construirii Sistemelor Expert. Prima variantă OPS a fost dezvoltată la sfârşitul anilor 1970 de către Charles Forgy de la universitatea Carnegie Mellon, S.U.A., care a obţinut un mediu de programare, care are la bază algoritmul RETE. OPS5 a fost versiunea care a fost folosită în dezvoltarea primului SE aplicabil, și anume R1/XCON dedicat configurării calculatoarelor.

OPS5 era un mediu de programare bazat pe reguli care oferea avantajul creării unor BC care conţineau sute sau chiar mii de reguli. Raţionamentul folosit în cadrul Motorului de Inferenţă era deductiv, strategia de control înainte. Prima versiune comercială a fost OPS4, în prezent s-a ajuns la versiunea OPS83. dacă la început s-a folosit LISP, apoi s-a trecut la BLISS pentru a creşte viteza de procesare a informaţiilor. În continuare se descriu caracteristicile şi structura unui program informatic dezvoltat în OPS5.

O aplicaţie în OPS5 cuprinde o secţiune unde sunt declarate datele care vor fi utilizate în secţiunile următoare, urmată de o secţiune de producţie, care cuprinde regulile de producţie pentru manipularea cunoştinţelor [4]. Datele dintr-un program OPS5 sunt înmagazinate într-o bază de date care este denumită memoria de lucru, iar reguliele sunt stocate în memoria de producţie. Pe de altă parte, aplicaţiile OPS5 rulează prin compararea elementelor din memoria de lucru cu regulile din memoria de producţie, de unde selectează şi execută regulile care se potrivesc. Acest ciclu, potrivire – selecţie – execuţie, se va opri când programul va parcurge întreaga memorie de lucru, sau întâlneşte o comandă explicită de oprire [5].

Datele din OPS5 se numesc atomi, care pot fi de două tipuri: numeric şi simbol. Datele numerice se împart în întregi, cu virgulă mobilă. În continuare, se dă un exemplu din cele două tipuri de date numerice și modul cum sunt implementate ele. Orice dată care nu este numrică este de tip simbol. Datele simbolice pot fi definite de utilizator sau predefinite.

Exemplu

Date numerice:

Întregi

decimalDigit ::= 0|1|2|3|4|5|6|7|8|9

integer ::= [+|-] *decimalDigit* {*decimalDigit*} [.]

Virgulă mobilă

exp::= e *integer*

float ::= [+|-] {*decimalDigit*} [.] *decimalDigit* {*decimalDigit*} [*exp*]

Exemplu numeric: întregi 25, -56

Virgulă mobilă 2.717, -3e+22

Date simbolice:

Predefinite NIL

Definite de utilizator Multime, etc.

Elementele din memoria de lucru sunt declarate în prima secţiune a aplicaţiei OPS5. Astfel, sunt declarate clase prin folosirea comenzii „literalize”, iar structura sintaxei este descrisă în alineatul următor, iar exemplu în Exemplu 2:

(literalize nume_clasa {atribut_clasa})

O clasă nu este obligatoriu să aibă atribute; această parte a sintaxei poate să lipsescă. Fiecare element al unei clase poate să aibă cel mult un atribut cu mai multe valori, care este denumit vector-atribut. Un vector-atribut se declară astfel:

(vector-attribute nume_vector {nume_vector}).

Precum se observă şi din exemplu, pentru o clasă se introduce numele și numele atributelor; deoarece OPS5 nu cere tipul atributelor, un anumit atribut poate lua valori de orice tip (numeric sau simbolic).

Exemplu

Definirea clasei student în OPS5

```
( literalize student
  nume
  nota        )
```

Prin utilizarea comenzii „literalize”, nu se alocă memorie fizică în memoria de lucru, ci doar se indică elementele cu care se va lucra. Pentru a realiza acest lucru, se foloseşte comanda „make”, care la fel se declară în paranteze rotunde.

În OPS5 variabilele se introduc prin intermediul caracterelor „< >” astfel:

< nume_variabilă >

A doua parte a unei aplicaţii OPS5 conţine regulile, care sunt denumite producţii. O producţie are forma:

LHS Conditions --> RHS Actions

iar sintaxa în interiorul programului este:

```
(p nume_producţie
   LHS
  -->
   RHS
)
```

Parte LHS (left-hand side) conţine condiţiile regulii; ea fiind prima care este verificată în procesul de potrivire-selecţie-execuţie pentru a se determina ce acţiuni să fie întreprinse. Astfel, condiţiile reprezintă singurul element de control în OPS5 [5].

Partea RHS (right-hand side) este cea care conţine acţiunile care se vor executa. Aceste acţiuni sunt reprezentate de operaţii de creere, modificare şi ştergere a elementelor memoriei de lucru, calculul unor mărimi aritmetice, forţarea programului să oprească executarea și realizarea de operaţii I/O. Există trei comenzi predefinite care corespund operaţiilor care sunt întreprinse asupra memoriei de lucru: creare, ştergere şi modificare. În tabelul 13.4 sunt descrise cele trei comenzi.

Tabelul 13.4. Comenzi predefinite în OPS5 [4, 5]

Comanda	**Cuvânt predefinit**	**Sintaxa**	**Exemplu**
Creare	make	(make *nume_clasa* {*nume_atribut valoare*})	(make student) (make student ^gpa 4.0) (make student ^id <id-2> ^major CS)
Ştergere	remove	(remove *număr_condiţie*)	(p example-1 (student ^gpa > 4.0) ; error data -->(remove 1))
Modificare	modify	(modify *număr_condiţie* {*nume_atribut* valoare})	(p example-3 (student ^gpa { > 3.5 <= 4.0 }) (student ^gpa > 4.0) ; error data -->(modify 1) (modify 2 ^gpa 4.0))
Calculare	compute	Este o acţiune prin care se cere programului OPS5 să realizeze nişte operaţii aritmetice	(compute 2 + 3 * 4 + 5)

La fel ca toate programele scrise în medii de programare dedicate SE, și programele OPS5 nu au o structură fixă, care să presupună o execuţie secvenţială a regulilor. De altfel nu este nici o structură de control predefinită, ci execuţia este direcţionată de procesul potrivire-selecţie-execuţie. În conscinţă, nu se va şti

în faza de proiectare a programului care va fi ordinea de execuţie a regulilor; aceasta este dictată de datele de intrare.

CLIPS

CLIPS, C Language Integrated Production System, este un mediu de programare specializat pentru construcţia Sistemelor Expert, care utilizează limbajului de programare C [6]. Acest mediu de programare a fost dezvoltat începând cu anul 1984 la Centru l Spaţial Johnson al NASA.

Prima versiune a mediului CLIPS a fost realizată în anul 1984, cu intenţia de a crea un mediu de programare care să aibă posibilitatea construirii Sistemelor Expert. CLIPS 3.0 este prima versiune comercială care a putut fi folosită de programatori externi centrului spaţial NASA începând cu vara anului 1986. În următorii ani, CLIPS a fost rescris de mai multe ori în ANSI-C și folosind programarea orientată pe obiecte, ajungându-se la versiunea 6.0 în 1995. De la o versiune la alta, CLIPS a evoluat, astfel dacă la început se baza pe strategia de control înainte și folosea algoritmul RETE, versiunea 5.1 avea implementată programarea procedurală şi programarea orientată pe obiecte, respectiv o interfaţă cu trei medii de operare PC: X-windows, MS-DOS şi Macintosh. Versiunea 6.0 conţine cinci modificări majore faţă de versiunea 5.1 [6]:

- Poate fi aplicat un algoritm de pattern-matching părţilor stânga a regulilor instanţelor claselor definite de utilizator în COOL (CLIPS Object Oriented Language);
- Conţine un considerabil suport pentru cunoștințe bazate pe un sistem software ingineresc. Sistemul este folosit pentru construirea de sisteme modulare şi foarte multe facilităţi ale sistemului CRSV sunt implementate în CLIPS;
- Pot fi folosite fapte şi predicate multi - variabilă;
- Este posibilă crearea de noi elemente folosind două sau mai multe elemente condiţionale existente în BC. Acest avantaj ajută la implementarea sistemelor cu auto – învăţare în CLIPS;
- Interfaţa Windows 3.1 a mediului CLIPS este disponibilă pentru calculatoarele compatibile PC.

Datorită răspândirii şi accesibilităţii limbajului de programare C, mediul CLIPS este din ce în ce mai utilizat în defavoarea altor medii consacrate pentru dezvoltarea SE precum PROLOG și LISP.

Mediul CLIPS utilizează pentru reprezentarea cunoștințelor sistemele de producţie, care sunt formate din reguli de producţie, iar Baza de Cunoștințe cuprinde cele două componente: baza de reguli şi baza de fapte. Regulile se implementează prin intermediul unei sintaxe de forma:

(defrule regula_1
(premisa_1)
(premisa_2)
…….
=>
(concluzia_1/acţiunea_1)
(concluzia_2/acţiunea_2)
……..)

iar, un fapt poate avea forma:

(deftemplate (nod (slot nume) (slot tip) (slot U) (slot Umin) (slot Umax)))

O regulă este aplicabilă dacă în BC există fapte care satisfac toate premisele. În exemplul următor se poate observa o regulă.

Exemplu

```
(defrule regula
  (nod (nume 5) (tip sursa) (U ?u1) (Umin ?u2) (Umax ?u3))
  (> ?u3 ?u1)
  (< ?u2 ?u1)
  =>
(printout t „Nodul nu are probleme de tensiune.” crlf)
  )
```

Totalitatea regulilor care sunt aplicabile la un moment dat formează agenda sistemului. Agenda este o structură de tip stivă, în care sunt memorate înregistrările de activare ale regulilor în funcţie de prioritate. Pentru selectarea regulilor din agendă se folosesc mai multe strategii. Strategia implicită constă în executarea regulilor în ordinea priorităţii lor. Pe lângă aceste strategii, Motorul de Inferenţă a lui CLIPS pune la dispoziţia utilizatorilor alte şapte strategii predefinite pentru selectarea regulilor [6], care sunt descrise în tabelul 13.5.

Tabelul 13.5. Strategiile pre-definite ale mediului CLIPS

Strategia	Caracteristici
Căutarea în adâncime	Selecţia regulilor se face în funcţie de prioritatea regulilor
Căutarea în lăţime	Regulile sunt aplicate în funcţie de priorităţi Prima dată sunt aplicate regulile mai importante și apoi cele mai puţin importante
Simplicity	Regulile sunt aplicate în ordinea simplităţii Simplitatea unei reguli se ia în funcţie de numărul de teste de pe slot-uri şi numărul de apeluri Cu cât aceste numere sunt mai mici, cu atât regula este mai simplă
Complexity	Regulile sunt aplicate în ordinea complexităţii
MEA	Selecţia se face în funcţie de proprietatea de vechime a unei reguli O regulă este mai recentă, cu cât faptul care intră în componenţa ei este mai recentă
LEX	Regulile sunt selectate asemănător strategiei MEA, diferenţa constă în faptul că sunt analizate toate condiţiile unei reguli
Randomize	Selecţia se face aleator

13.3. Alegerea instrumentului adecvat

Alegerea instrumentului corespunzător pentru construirea unui Sistem Expert poate să fie un proces complicat, în special dacă cel care este managerul de proiect a SE nu are cunoștințe de programare sau în dezvoltarea Sistemelor Expert. Aşa cum este descris în capitolul precedent, există două abordări în construcţia SE, și anume prin utilizarea unui mediu de programare dedicat care se bazează pe un limbaj de programare de nivel înalt, sau a unui sistem cadru. În consecinţă, prima etapă în alegerea instrumentului pentru construcţia SE constă în optarea pentru una dintre cele două posibilităţi. Această alegerea este influenţată în principal de cunoştinţele de programare ale personalului care va construi SE, respectiv de timpul disponibil. Astfel, dacă programatorii din echipa proiectului SE deţin cunoştinţe avansate de programare logică, pot să utilizeze cu uşurinţă medii de programare dedicate, contrar este de indicat să se opteze pentru

sisteme cadru. Exemplele din practică, și din literatura de specialitate au prezentat faptul că de cele mai multe ori se folosesc ambele opţiuni, în funcţie de perioadele de proiectare şi dezvoltare a SE. Aşa cum s-a mai specificat şi în cadrul cursurilor precedente, sistemele cadru se folosesc în special în etapele de machetă şi prototip a unui SE, când este nevoie de obţinerea unui răspuns rapid asupra profitabilităţii şi eficienţei utilizării unui SE în rezolvarea unei probleme; iar mediile de programare sunt folosite pentru crearea unor SE specifice, personalizate rezolvării unei anumite probleme.

A doua etapă în alegerea instrumentului constă în analiza caracteristicilor instrumentelor disponibile. Astfel, în cazul mediilor de programare trebuie să se aibă în vedere dacă, datele și informaţiile specifice problemei se pot reprezenta corespunzător printr-un anumit limbaj de programare. Un alt aspect ce trebuie urmărit o reprezintă facilităţile care le cuprinde mediul de programare, în special în construirea interfeţei grafice cu utilizatorul. În această direcţie, limbajele de programare precum PROLOG, prezintă un neajuns, deoarece inginerul de cunoștințe trebuie să proiecteze tot Modulul de Interfaţă cu Utilizatorul: ferestre, icoane și alte elemente grafice.

Dacă după prima etapă s-a ales construirea SE prin utilizarea sistemelor cadru, atunci în cadrul etapei doi, trebuie avute în vedere caracteristicile şi facilităţile sistemului cadru, respectiv tipul şi domeniu problemei de rezolvat. În analiza unui sistem cadru sunt avute în vedere modul în care se reprezintă cunoştinţele, strategia de control și facilităţile legate de crearea interfeţei grafice cu utilizatorul și a Modulului Explicativ. În continuare, sunt descrise aspectele care trebuie luate în considerare în alegerea Sistemelor Expert cadru, şi anume: Baza de Cunoştinţe, Motorul de Inferenţă , facilităţile grafice și explicative. În figura 13.3 este descris grafic etapele ce trebuie parcurse în alegerea instrumentului adecvat şi elementele unui sistem cadru.

Baza de Cunoștințe

Modul în care sunt reprezentate cunoștințele în cadrul Bazei de Cunoștințe este principalul aspect care trebuie luat în considerare atunci când se analizează un sistemul cadru. Astfel, vor fi selectate SE cadru care cuprind un mod de reprezentare a cunoştinţelor conform cerinţelor proiectului SE. În rândul sistemelor cadru, cele mai populare sunt cele bazate pe reguli de producţie, cadre, exemple şi cazuri. Există sisteme cadru mai noi care oferă mai multe variante în reprezentarea cunoștințelor [2].

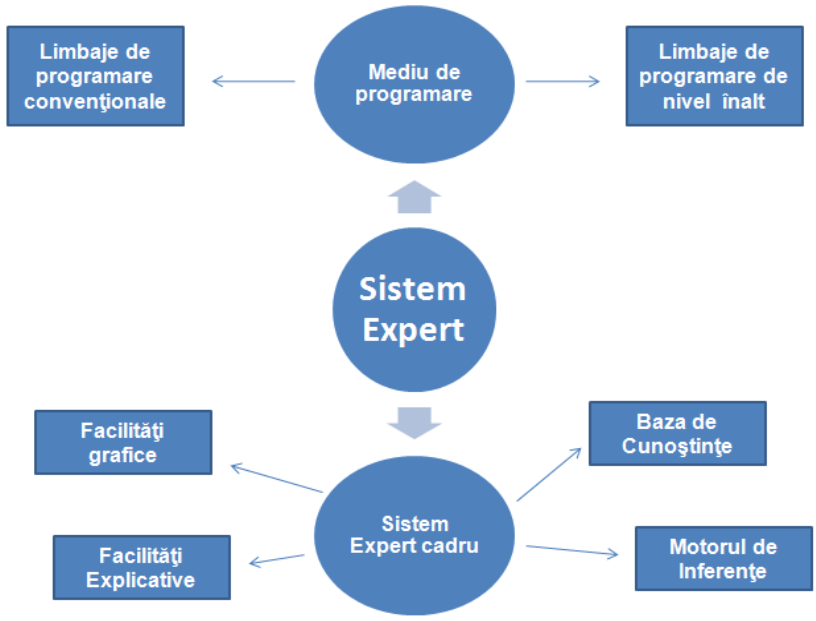


Fig. 13.3 Alegerea metodei de construire a Sistemelor Expert

În dezvoltarea Sistemelor Expert, este uneori necesar să se reprezinte date inexacte şi incomplete, în consecinţă, trebuie avute în vedere sistemele cadru care oferă posibilitatea introducerii a unor factori de pondere, care să indice implicarea unei cunoștințe în rezolvarea problemei. Aceste facilităţi sunt oferite în special de sistemele de dimensiuni mici, cele mari lăsând la latitudinea inginerului de cunoştinţe să îşi definească aceste aspecte.

Unii ingineri de cunoştinţe caută la sistemele cadru să ofere posibilitatea introducerii unor cunoștințe procedurale, sub forma unor expresii text. În concluzie, în alegerea unui sistem cadru din punctul de vedere al BC, trebuie avute în vedere următoarele: modul de reprezentare a cunoştinţelor, şi facilităţile ce la au sistemele în această direcţie.

Motorul de Inferenţă

La Sistemele Expert cadru, modul în care cunoștințele din Baza de Cunoştinţe este impus. În consecinţă, trebuie avut în vedere strategia de control și algoritmii de căutare utilizaţi, deoarece aceştia influenţează rapiditatea cu care este găsită soluţia. Din practică, se observă că sistemele cadru mici au tendinţa să aibă facilităţi limitate oferite utilizatorilor pentru gestiunii cunoștințelor, comparativ cu cele mari care acordă mai mare importanţă acestei ramuri a SE şi pun la dispoziţia inginerilor de cunoștințe mai multe variante de parcurgere a BC.

La sistemele cadru care utilizează reprezentarea cunoștințelor prin sisteme de producţie, cele mai utilizate strategii de control sunt: strategia de control înainte și strategia de control înapoi. Unele sisteme permit programatorilor să aleagă una sau chiar ambele metode, ba mai mult să poată schimba strategia de control în timpul unei sesiuni de rezolvare a unei probleme, astfel că BC va fi parcursă în ambele direcţii [2]. Selectarea SE cadru în funcţie de strategiei de control depinde de tipul problemei de rezolvat, deoarece strategia de control înainte este recomandată pentru SE de planificare și proiectare, iar strategia de control înapoi în cazul SE de clasificare.

În ceea ce priveşte controlul activităţilor unui SE, unele sisteme cadru oferă facilităţi programatorilor pentru [2]:

- Planificarea activităţilor SE prin intermediul unor agende. Această caracteristică a unor sisteme cadru este foarte importantă, în special pentru aplicaţiile de dimensiuni mari;
- Meta – reguli – permit programatorilor să descrie modul în care vor fi utilizate cunoștințele. Această facilitate este foarte comodă în cazul aplicaţiilor la care nu se poate folosi doar strategia de control înainte sau înapoi, iar programatorul direcţionează căutarea spre anumite cunoştinţe;
- Raţionamentul ne-monoton – acest tip de raţionament permite schimbarea raţionamentului de parcurgere a cunoștințelor, în funcţie de schimbarea stării problemei. Strategie de control ne-monoton a datelor permite retragerea unui sau mai multe fapte, în funcţie de noua stare a problemei. Această caracteristică a sistemelor cadru este foarte valoroasă pentru dezvoltarea SE de proiectare, planificare și programare, unde noi informaţii pot să schimbe starea problemei în mod decisiv.

Facilităţile grafice și explicative

Modulul de Interfaţa cu Utilizatorul (MIU) a unui SE este cel cu care utilizatorul intră prima dată în contact, astfel că importanţa lui este primordială, mai ales că încă de la începuturile SE inginerii de cunoștințe au observat că acceptarea precoce a unui SE este în strânsă legătură cu interfaţa sa grafică. Aşa cum s-a descris în cursul corespunzător, proiectarea MIU depinde de utilizatori și doleanţele acestora, astfel că în alegerea sistemului cadru trebuie să se aibă în vedere facilităţile ce le oferă în proiectarea şi dezvoltarea interfeţei grafice cu utilizatorul. În consecinţă, la un sistem cadru, trebuie căutate următoarele facilităţi [2]:

- Tipul afișării – această facilitate se referă la modul în care este afișat utilizatorilor toate datele, care poate fi de tip text sau grafic. Primul tip este caracteristic sistemelor cadru vechi, dar majoritatea sistemelor cadru noi prezintă facilităţi grafice, prin intermediul cărora inginerul de cunoștințe poate să creeze interfeţe grafice bazate pe ferestre și icoane;
- Introducerea datelor – există mai multe variante pentru introducerea datelor de către utilizatori, şi anume butoane, icoane, elemente pentru introducerea textului etc. Unele sisteme cadru au facilităţi limitate pentru introducerea informaţiilor, iar altele dau posibilitatea programatorilor să creeze interfeţe avansate centrate pe utilizatori;
- Afişarea informaţilor – Sistemul Expert comunică informaţii utilizatorului prin intermediul interfeţei cu utilizatorul. Datele afişate pot să fie valori intermediare ale problemei sau concluzia finală. Cele mai uzuale metode de afişare a informaţiilor sunt: graficele, ferestrele, tabelele, casete de text etc.;
- Controlul – pentru ca SE să fie este acceptat, trebuie ca utilizatorul să simtă mereu că deţine controlul. Această caracteristică a interfeţei grafice presupune începerea și încetarea unei sesiuni într-un mod simplu. Unele sisteme cadru deţin metode predefinite pentru administrarea acestei caracteristici, în timp ce altele lasă inginerul de cunoștințe să dezvolte aceste facilităţi ale interfeţei cu utilizatorii. În ceea ce priveşte controlul, unii utilizatori au pretenţii pentru a putea salva anumite informaţii intermediare etc.

Modulul Explicativ este unul dintre componentele principale ale unui SE, el intrând în arhitectura de bază. În ceea ce priveşte importanţa acestuia, pentru unii utilizatori practica a dovedit că este mai valoros decât soluţia a sine, deoarece dacă utilizatorul nu va înţelege modul în care s-a rezolvat o problemă, probabilitatea cea mai mare este că el va respinge acea soluţia, ba chiar întregul SE. În dezvoltarea SE, sistemele cadru nu oferă pentru construirea explicaţiilor, iar puţinele care prezintă astfel de caracteristici, cuprind doar un algoritm rudimentar prin care afișează regulile care au fost aplicate în soluţionarea problemei. Această facilitate nu este de folos utilizatorilor, pe care de multe ori îi induce în eroare, ei neavând cunoştinţe de programare. Pe inginerii de cunoștințe, pe de altă parte îi ajută îi procesul de verificare a SE dezvoltat, deoarece prin intermediul acestei facilităţi aceştia pot să determine rapid dacă anumite reguli au fost introduse greşit.

Înainte de alegerea sistemului cadru, managerul de proiect trebuie să ştie clar care sunt necesităţile și cerinţele utilizatorilor, respectiv nivelul de cunoștințe a inginerilor de cunoștințe .

13.4. Instrumente dedicate dezvoltării SE

Precum s-a prezentat în cadrul capitolului 12, SE cadru se pot împărţi în două categorii, în funcţie de modalitatea de reprezentare a cunoștințelor, respectiv de domeniul de aplicare a SE. În alegerea sistemelor cadru, recomandarea specialiştilor este să se aleagă sisteme instrumente care au fost aplicate cu succes în rezolvarea unor probleme asemănătoare cu problema studiată. În sub – capitolele următoare sunt descrise caracteristicile celor mai populare sistemele cadru.

SE de diagnoză

Sistemele Expert de clasificare sunt cele mai răspândite, iar sistemele cadru asociate de semenea cele mai variate. În aceeaşi categorie intră cele specializate în diagnoză, care realizează o clasificare a posibilelor cauzelor unui defect.

D-IAL, Diagnosis Intelligent Automation Language, asistă programatorii în dezvoltarea SE dedicate mentenaţei instalaţiilor de facbricaţie. Acest sistem foloseşte algoritmul „cel mai bun este primul” şi se bazează pe o strategie de control care foloseşte probabilitatea Dempster-Shafer. Acest mecanism de raţionament propagă probabilitatea de defect într-un arbore de căutare pentru a menţine consistenţa probabilităţilor de defectare. Mediu suport este scris în limbajul de programare C, astfel că un SE dezvoltat prin intermediul lui poate să lucreze cu uşurinţă pe calculatoarele personale. Sistemul a fost dezvoltat în Japonia.

EKO este un sistem cadru orientat pe obiecte, creat în Rusia, specializat în dezvoltarea SE de diagnoză. Acest sistem a fost iniţial proiectat şi creat pentru a asista în evaluarea stării structurilor hidraulice. Acest SE lucra cu date primite de la senzori sau dispozitive de memorare a datelor. Ulterior, pe baza SE originar a fost dezvoltat sistemul suport.

IDEA este un instrument pentru dezvoltarea SE bazate pe modele. Acest sistem suport este specializat în construirea SE care să asiste tehnicienii în identificarea componentelor defecte a dispozitivelor electromecanice. IDEA utilizează o strategie de control particulară sistemelor bazate pe modele pentru construirea și lansarea în execuţie a aplicaţiilor. Raţionamentul folosit are la bază

următorul conceptul: dacă utilizatorii ştiu cum un dispozitiv funcţionează, în contrast cu starea de defect a acestuia, aceştia pot să construiască o aplicaţie pentru diagnoză.

SE de simulare și proiectare

În domeniul Sistemelor Expert dedicate problemelor de simulare și proiectare există multe medii suport care pot să asiste și să ajute utilizatorii în activităţile lor. Cele mai populare sisteme cadru specializate creării SE de simulare și proiectare sunt descrise în paragrafele următoare.

DESIGNER este un sistem cadru care asistă dezvoltarea SE dedicate proiectării diferitelor procese. Principala caracteristică în proiectarea sistemelor tehnologice este complexitatea: proiectantul are sarcina de a specifica caracteristicile sistemului prin introducerea unei liste de obiective funcţionale cerute care vor trebui să fie îndeplinite. Având aceste date, sistemul cadru creat de specialiștii britanici, produce o schiţă a sistemului [2].

GOES, Graphics-Oriented Expert Shell, creat în Canada, este un sistem suport care a fost construit în interiorul mediului standard de proiectare CAD. Acest sistem asociază procedurile inginereşti de proiectare logică cu reprezentările grafice, ceea ce dă avantajul de a administra și gestiona activităţi de proiectare automată de mare anvergură. Particularitatea acestui sistem suport este că lucrează ca un instrument inteligent de selecţie şi elimină sarcini ce trebuie realizate de programator precum: identificarea, asamblarea și parametrizarea sub-rutinelor SE.

XpertRule Configurator este un sistem cadru care asistă inginerii de cunoștințe în dezvoltarea SE bazate pe reguli de producţie dedicate proiectării de produse. Sistemul are scopul de a construi sisteme care folosesc sisteme de producţie pentru selecarea corectă şi configurarea componentelor unui produs. Utilizatorul poate genera un tabel ierahic a componentelor fizice prin intermediul unui editor grafic de configurare de tip arbore şi a unui set de reguli asociate diferitelor sarcini. Acest sistem a fost proiectat și creat la Harvard.

ORBIS, Object-oriented Rule Base Interactive System, este un Sistem Expert cadru dedicat construirii de SE de simulare. Acest sistem poate fi folosit în asociere cu mai multe medii de simulare, care presupun simulări independente interactive, simulări în timp real etc. O simulare ORBIS cuprinde două părţi: suportul și aplicaţia particulară. Sistemul cadru cuprinde un motor de simulare, un editor de reguli, un editor de explicaţii, un editor de obiecte şi un editor de meniu, care împreună asigură elementele de bază pentru realizarea unei simulări.

O aplicaţie conţine obiecte, date, algoritmi şi un set de reguli caracteristice simulării, care au rol în generarea modului în care să se realizeze simularea.

SE de control

În dezvoltarea Sistemelor Expert specializate pe control cele mai populare sisteme cadru conform [2] sunt: ASIA, FAIN și G2.

ASIA este un sistem cadru care asistă și ajută programatorii în construirea de Sisteme Expert pentru controlul în timp real. Sistemul este capabil să gestioneze date externe în timp real prin procesarea simbolică. Acesta a fost dezvoltat la Institutul Mihailo Pupin, Belgrad, Serbia.

FAIN, Fast AI shell of Nippon Steel, a fost dezvoltat de către Nippon Steel pentru a ajuta în crearea de SE de control. Prin utilizarea acestui sistem cadru, un SE este dezvoltat prin adaugarea sau revizuirea unor caracteristici de proiectare într-o manieră secvenţială. Astfel, caracteristicile SE sunt conţinute într-un document de proiectare pe baza căruia sistemul cadru va genera automat aplicaţia particulară.

G2 este un mediu grafic orientat pe obiecte dedicat construcţiei programelor inteligent de management a proceselor cu scopul rezolvării unor probleme particulare. Sistemul prezintă un editor încorporat prin care utilizatorul poate introduce reguli de producţie, modele și proceduri care descriu operaţiile ce se realizează în timp real. G2 prezintă mai multe unelte adiţionale pentru programarea sistemelor, cu logica fuzzy, pachete de diagnostic, algoritmi genetici şi reţele neuronale artificiale. Aplicaţiile tipice construite prin intermediul G2 include componente pentru optimizarea proceselor, managementul calităţii în timp real, controlul de supervizare şi controlul avansat cu ajutorul logicii fuzzy și a reţelelor neuronale artificiale. Acest sistem cadru a fost dezvoltat în S.U.A.

În domeniu sistemelor suport, tendinţele sunt de a crea medii de programare care să asiste utilizatori uzuali în dezvoltarea SE.

13.5. Întrebări şi exerciţii

1. Aplicațiile dedicate dezvoltării SE sunt clasificate în funcție de tipul sistemelor expert.
Adevărat / Fals

2. Sistemele expert cadru sunt varianta cea mai uzuală în dezvoltarea SE.

Adevărat / Fals

3. Programele de nivel înalt dedicate dezvoltării SE sunt:
PROLOG / Pascal / LISP

4. Limbajele de programare uzuale precum C++ sunt folosite pentru dezvoltarea SE.
Adevărat / Fals

14

Aplicații ale sistemelor expert în energetică

Prezentul capitol prezintă o comparaţie între expertiza artificială obţinută prin intermediul Sistemelor Expert și cea umană dată de experţii umani, respectiv domeniile din electroenergetică în care s-au aplicat cu succes, şi se întrebuinţează Sistemele Expert.

14.1. Aspecte generale

Sistemele Expert sunt programe informatice dezvoltate special pentru a copia modelul raţional şi metoda de rezolvare a problemelor de către experţii umani.

Expertul uman se bazează pe memoria de termen lung în care a înmagazinat informaţii legate de un domeniu restrâns de activitate, care le-a obţinut de-a lungul a mulţi ani de muncă şi experienţă, în care s-a confruntat cu multe situaţii şi cazuri particulare, şi pe raţionamentul specific uman și intuiţie pentru a rezolva o problemă particulară din domeniul său de activitate.

Asemenea expertului uman, un Sistem Expert deţine o Bază de Cunoștințe în care are stocate toate informaţiile legate de problema de rezolvat (din domeniul de activitate al expertului uman), pe care le utilizează şi gestionează cu ajutorul unei strategii de control și a mai multor algoritmi de căutare, care formează Motorul de Inferenţă, pentru a determina o soluţie la problema studiată. Specific Sistemelor Expert este faptul că acestea conţin multe componente care asigură asistenţa în rezolvarea problemei asemenea unui expert uman: Modul Explicativ, Modul de Interfaţă cu Utilizatorul, Modul de Interfaţă cu Inginerul de Cunoștințe, Modul de Achiziţie a Cunoștințelor, Modul Dinamic și Baze de date externe. De altfel, diferenţa dintre un program convenţional de gestiune a datelor și un Sistem Expert constă în existenţa unui Modul Explicativ prin care programul poate să

ofere utilizatorului explicaţii legate de cunoştinţele utilizate, modul de determinare a soluţie etc.

În sub-capitolele următoare se face o comparaţie între utilizarea SE şi a experţilor umani prin descrierea avantajelor și dezavantajelor celor două părţi; de asemenea se prezintă pe larg aplicaţiile SE în energetică.

14.2. Expertiza artificială vs. Expertiza umană

Utilizarea Sistemelor Expert comparativ cu folosirea unui expert uman oferă multe avantaje pe de o parte, dar și dezavantaje, care sunt cauzate de natura programului.

Un prim argument este performanţa expertizei SE faţă de expertul uman, care nu este influenţată de factori externi. Astfel, expertul uman poate să ofere soluţii diferite la o aceeaşi problemă în funcţie de condiţia fizică și starea psihologică. Studiile au arătat că în condiţii de stres, randamentul și eficienţa experţilor umani scade simţitor, ceea ce nu se întâmplă în cazul SE care nu sunt afectate de astfel de factori. Tot legat de performanţa expertizei este și faptul că un SE va oferi date mai consistente şi uşor de reprodus cu privire la modul în care s-a rezolvat o problemă, ceea ce nu se întâlneşte la experţii umani, care de multe ori prezintă dificultăţi în a expune în termeni nespecifici domeniului modul în care au soluţionat problema [1].

Un alt argument este uşurinţa cu care sunt transmise informaţiile de la un SE la altul, respectiv la utilizatorii umani. Se ştie că transferul informaţiilor de la o persoană la alta (expertul uman, simplu utilizator) poate să dureze o perioadă lungă de timp, să implice complicaţii de natură subiectivă sau obiectivă. Comparativ, transferul de informaţii între calculatoare se face foarte uşor, la fel și de la SE la utilizator, mai ales dacă SE are un Modul Explicativ eficient (a fost dezvoltat prin chestionarea utilizatorilor și implicarea acestora).

Costul este un alt avantaj în ceea ce priveşte abordarea SE în rezolvarea unor probleme, deoarece achiziţia unui SE este mult mai mică comparativ cu angajarea unui expert uman. În plus, utilizarea SE nu înseamnă decât costuri de mentenanţă și achiziţia iniţială.

În figura 14.1 se prezintă avantajele utilizării expertizei SE, respectiv a expertului uman.

Precum se observă din figura anterioară, balanţa avantajelor utilizării SE şi a expertului uman este echilibrată, cu observaţia că sunt probleme la care este mai avantajos financiar să se utilizeze programul inteligent în defavoarea expertului

uman; însă sunt domenii de activitate la care necesitatea și utilitatea expertului uman sunt de necontestat.

Fig. 14.1 Comparaţie între avantajele expertizei SE și cea umană

Dezavantajele utilizării expertizei umane şi a celei artificiale sunt enumerate în figura 14.2.

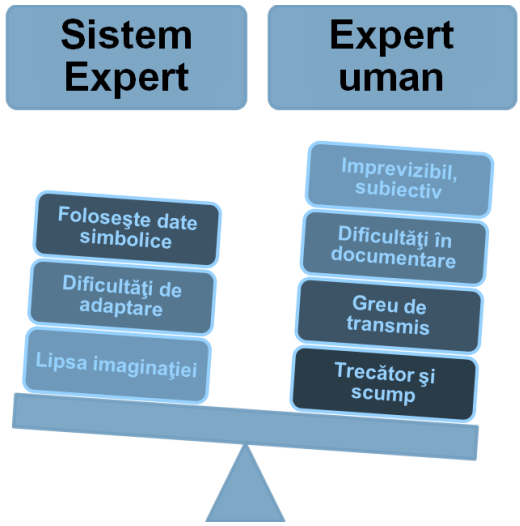


Fig. 14.2 Comparaţie între dezavantajele expertizei SE și cea umană

În prezent, în electroenergetică și multe alte domenii Sistemele Expert sunt utilizate şi este recomandat să fie utilizate doar ca asistenţi în activitatea umană.

14.3. Sistemele Expert în literatura de specialitate

În prezent aplicaţiile Sistemelor Expert ocupă o gamă largă de domenii, unele fiind specifice economiei, medicinei, ingineriei, culturii, iar altele au fost dezvoltate pentru a servi în cercetare. În electroenergetică, pe plan mondial, Sistemele Expert sunt folosite cu succes în transport, distribuţie, generare şi exploatarea în general a sistemelor electroenergetice.

În transport şi distribuţie, probleme precum cele prezentate în tabelul 1 au fost rezolvate cu succes cu ajutorul acestor programe inteligente de gestiune a datelor.

Tabelul 14.1. Aplicaţiile SE în transportul și distribuţia energiei electrice [1]

Transport	Distribuţie
Exploatare	
Evaluarea securităţii Controlul U-Q Tratarea alarmelor Localizarea și diagnoza defectelor Restaurarea sistemlului Estimarea stării Prognoza sarcinii pe termen scurt Interpretarea rezultatelor din programe complexe	Supraveghere și diagnoză Sprijinul exploatării Restaurarea sistemului Supravegherea şi controlul staţiilor electrice Controlul funcţionării sarcinilor
Planificarea exploatării	
Prognoza sarcinii pe termen scurt	Asistarea în planificarea exploatării
Planificarea dezvoltării	
Proiectarea liniilor electrice Proiectarea şi coordonarea sistemelor de protecţie Prognoza sarcinii pe termen lung	Programarea întreţinerii Proiectarea distribuţiei subterane din oraşe Acţiuni de corectare Management

O listă a SE aplicate în electroenergetică, prezentate în literatura de specialitate este enumerată în continuare:

- Expert system aid for design of overhead line distribution networks (Sistem Expert dedicate pentru proiectarea liniilor electrice aeriene din reţelele de distribuţie) [16].
- Application of expert system to power system restoration in local control center (Aplicaţie a Sistemelor Expert pentru restaurarea sistemului electroenergetic la un dispecerat local) [17].
- An expert system for reactive power control of a distribution system. Part 2: system implementation (Sistem Expert dedicat controlului puterii reactive din reţelele electrice de distribuţie) [18].
- A rule-based expert system for steady-state stability analysis (Sistem Expert bazat pe reguli de producţie pentru analiza stabilităţii regimului permanent în sistemele electroenergetice) [19].
- Short term load forecasting of Taiwan power system using a knowledge-base expert system (Prognoza pe termen scurt a sistemului electroenergetic din Taivan utilizând un sistem expert) [20].
- An expert system for locating distribution system faults (Sistem Expert dedicat localizării defectelor în sistemele de distribuţie) [21].
- An expert system for security trend analysis of a stability-limited power system (Sistem Expert pentru analiza securităţii unui sistem electroenergetic cu o stabilitate dinamică) [22].
- An expert system for voltage and reactive power control of a power system (Sistem Expert pentru controlul tensiune și putere reactivă a unui sistem electroenergetic) [23].
- SESA an expert system for auxiliary power services design (SESA un Sistem Expert dedicat proiectării / planificării servicilor auxiliare din sistemele electroenergetice) [24].
- An object-oriented expert system for power system alarm processing and fault identification (Sistem Expert bazat pe programarea orientată pe obiecte dedicat procesării alarmelor și identificarea defectelor într-un sistem electroenergetic) [25].
- Object-oriented synergetic expert system for fault diagnosis (Sistem Expert sinergetic orientat pe obiecte pentru diagnoza defectelor în sistemele electroenergetice) [26].

- Expert system for analysis of electric power system harmonics (Sistem Expert dedicat analizei armonicilor din sistemele electroenergetice) [27].
- Development of the expert system for operation planning of power system (Dezvoltarea unui sistem expert pentru planificarea operării unui sistem expert) [28].
- Power system fault diagnosis expert system using PROLOG (Sistem expert de diagnosticare a defecțiunilor sistemului de alimentare folosind PROLOG) [29].
- Enhancement of power system security and voltage control by an expert system using pattern recognition techniques (Îmbunătățirea securității sistemului de alimentare și a controlului tensiunii de către un sistem expert folosind tehnici de recunoaștere a modelelor) [30].
- Power system restoration by joint usage of expert system and mathematical programming approach (Restaurarea sistemului de alimentare prin utilizarea comună a unui sistem expert și a unei abordări de programare matematică) [31].

Bibliografie

[1]. Schank R.C., *What is Artificial Inteligence, Anyway ?*, AI Magazine, Vol. 8, No. 4, 1987.

[2]. Mircea Eremia et.al, T*ehnici de Inteligenţă Artificială. Concepte şi aplicaţii în sistemele electroenergetice, Cap. 1*, Ed. AGIR, Bucureşti, 2001.

[3]. Mihai Gavrilaş, *Inteligenţa Artificială şi Aplicaţii în Energetică, Cap. 1*, Vol I, Ed. Gh. Asachi, Iaşi, 2002.

[4]. Şerban Gabriela, Pop H.F., *Tehnici de Inteligenţă Artificială. Abordări bazate pe agenţi inteligenţi, Cap. 1*, Ed. Mediamira, Cluj-Napoca, 2004.

[5]. Chindriş M., Cziker A., *Utilizarea logicii fuzzy în energetică, Cap. 1*, Ed. Casa Cărţii de Ştiinţă, Cluj-Napoca, 2004.

[6]. Sheble G.B., *L'intelligence artificielle une solution aux problemes de conduite des reseaux*, IEEE, Vol. PWR-2, Novembre, 1987.

[7]. Dillon T.S., *Expert systems: potential and limitation in the application to power systems*, Symposium on Expert Systems Application to Power Systems, Helsiki, 1988.

[8]. Tănase Gh., *Sisteme Expert*, Universul Ingineresc, Asociaţia Generală a Inginerilor din România, http://www.agir.ro/univers-ingineresc/numar-22-2005/sisteme-expert_1249.html, Ultima accesare februarie 2013.

[9]. Cârstoiu D., *Sisteme Expert*, Cap. 1.3, Universitatea Bucureşti, Ed. ALL, Bucureşti, 1994.

[10]. Mircea Eremia et.al, T*ehnici de inteligenţă artificială. Concepte şi aplicaţii în sistemele electroenergetice, Cap. 1*, Ed. AGIR, Bucureşti, 2001.

[11]. Mihai Gavrilaş, *Inteligenţa Artificială şi Aplicaţii în Energetică, Cap. 1*, Vol. I, Ed. Gh. Asachi, Iaşi, 2002.

[12]. Cârstoiu D., *Sisteme Expert*, Cap. 1.3, Universitatea Bucureşti, Ed. ALL, Bucureşti, 1994.

[13]. http://year12ipt.ash.com/untitled-6.html, Ultima accesare, may 2013.

[14]. http://www.ida.liu.se/~TDDB66/slides/Lecture1/tsld017.htm, Ultima accesare may 2013.

[15]. Liebowitz Jay, The Handbook of Applied Expert Systems, *Publisher: CRC Press LLC, 1997.*

[16]. K.K. Kuan, K. Warwick, Expert system aid for design of overload line distribution networks, International Journal of Electrical Power & Energy Systems, Volume 13, Issue 5, 1991, Pages 277-286, ISSN 0142-0615, https://doi.org/10.1016/0142-0615(91)90051-V.

[17]. Y-M. Park, K-H. Lee, Application of expert system to power system restoration in local control center, International Journal of Electrical Power & Energy Systems, Volume 17, Issue 6, 1995, Pages 407-415, ISSN 0142-0615, https://doi.org/10.1016/0142-0615(94)00011-5.

[18]. J. R. P.-R. Laframboise, G. Ferland, A. Y. Chikhani and M. M. A. Salama, "An expert system for reactive power control of a distribution system. Part 2: system implementation," in *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1433-1441, Aug. 1995, doi: 10.1109/59.466507.

[19]. Yuan-Yin Hsu and Chung-Ching Su, "A rule-based expert system for steady-state stability analysis (of power systems)," in *IEEE Transactions on Power Systems*, vol. 6, no. 2, pp. 771-777, May 1991, doi: 10.1109/59.76724.

[20]. Ku-Long Ho *et al*., "Short term load forecasting of Taiwan power system using a knowledge-based expert system," in *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1214-1221, Nov. 1990, doi: 10.1109/59.99372.

[21]. Yuan-Yih Hsu *et al*., "An expert system for locating distribution system faults," in *IEEE Transactions on Power Delivery*, vol. 6, no. 1, pp. 366-372, Jan. 1991, doi: 10.1109/61.103760.

[22]. A. A. Fouad, S. Vekataraman and J. A. Davis, "An expert system for security trend analysis of a stability-limited power system," in *IEEE Transactions on Power Systems*, vol. 6, no. 3, pp. 1077-1084, Aug. 1991, doi: 10.1109/59.119249.

[23]. S. J. Cheng, O. P. Malik and G. S. Hope, "An expert system for voltage and reactive power control of a power system," in *IEEE Transactions on Power Systems*, vol. 3, no. 4, pp. 1449-1455, Nov. 1988, doi: 10.1109/59.192952.

[24]. J. . -M. Pelletier and A. Boyer, "SESA an expert system for auxiliary power services design," *Proceedings of TENCON '93. IEEE Region 10 International Conference on Computers, Communications and Automation*, Beijing, China, 1993, pp. 375-378 vol.5, doi: 10.1109/TENCON.1993.320661.

[25]. K. Hasan, B. Ramsay, S. Ranade and C. S. Ozveren, "An object-oriented expert system for power system alarm processing and fault identification," *Proceedings of MELECON '94. Mediterranean Electrotechnical Conference*, Antalya, Turkey, 1994, pp. 909-912 vol.3, doi: 10.1109/MELCON.1994.380954.

[26]. Changsheng Xu, Sixing Liu, Zhaoying Zhou and Yaoqing Zhang, "Object-oriented synergetic expert system for fault diagnosis," *Conference Proceedings. 10th Anniversary. IMTC/94. Advanced Technologies in I & M. 1994 IEEE Instrumentation and Measurement Technolgy Conference (Cat. No.94CH3424-9)*, Hamamatsu, Japan, 1994, pp. 398-401 vol.1, doi: 10.1109/IMTC.1994.352040.

[27]. D. D. Shipp, W. Vilcheck, M. E. Swartz and N. H. Woodley, "Expert system for analysis of electric power system harmonics," in *IEEE Industry Applications Magazine*, vol. 1, no. 2, pp. 34-39, March-April 1995, doi: 10.1109/2943.384625.

[28]. S. Osaka, Y. Kono, R. Fujiwara and A. Yamanishi, "Development of the expert system for operation planning of power system," *Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications*, Hitachi City, Japan, 1988, pp. 545-550, doi: 10.1109/AIIA.1988.13345.

[29]. S. B. Jadid, B. Jeyasurya and S. A. Khaparde, "Power system fault diagnosis expert system using PROLOG," *Fourth IEEE Region 10 International Conference TENCON*, Bombay, India, 1989, pp. 778-781, doi: 10.1109/TENCON.1989.177053.

[30]. C. S. Chang and Zhang Yao, "Enhancement of power system security and voltage control by an expert system using pattern recognition techniques," *1991 International Conference on Advances in Power System Control, Operation and Management, APSCOM-91.*, Hong Kong, 1991, pp. 170-174 vol.1.

[31]. T. Nagata, H. Sasaki and R. Yokoyama, "Power system restoration by joint usage of expert system and mathematical programming approach," in *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1473-1479, Aug. 1995, doi: 10.1109/59.466501.

Anexa 1

Sistemul expert WPPES

Sistemul expert WPPES (Wind Power Plant Expert System) este prezentat în articolul științific:

Duer, R.; Duer, S.; Zajkowski, K.; Woźniak, M.; Bernatowicz, D.; Paś, J.; Stawowy, M.; Iqbal, A.; Harničárová, M. Wind Power Plant Expert System Diagnostic Knowledge Base Creation. Energies 2025, 18, 1843. https://doi.org/10.3390/en18071843.

Articolul se focalizează pe procesul de construire a unei baze de cunoștințe despre o centrală eoliană, care este folosită de un sistem expert, și anume WPPES.

Baza de cunoștințe cuprinde informații despre echipamentele centralelor eoliene, mai exact modelele funcționale și de diagnostic a acestor echipamente. Modelele funcționale de diagnostic ale obiectelor au fost folosite ca baza pentru obținerea de informații de diagnostic despre obiectul studiat (setul de fapte). În plus, au fost caracterizate și descrise condițiile de funcționare ale echipamentelor centralei eoliene și ale împrejurimilor acestora, și au fost determinate condițiile admisibile și limită pentru funcționarea obiectului tehnic testat. În final, aceste informațiile legate de echipamente au fost utilizate ca seturi de date de diagnostic pentru construirea bazelor de fapte și reguli.

În prima etapă a studiului, autorii au ales structura anasamblului de centrale eoliene care stă la baza cercetării. Acesta este prezentată schematic în figura A1.1, unde se pot observa cinci centrale eoliene, WPP – Wind Power Plant, de câte 2 MW fiecare. Aceste centrale sunt conectate la două linii de 2 kV, care apoi se conectează într-un nod de MPP, Main Power Point, la o linie de transport de 110 kV.

În a doua etapă a studiului au fost culese informații despre echipamentele și dispozitivele care intră în componența centralelor eoliene și a generatoarelor eoliene (WTG), iar autorii au explicat și problemele cu care s-au confruntat în culegerea datelor și transformarea lor în informații utile pentru construirea bazei de cunoștințe.

Întrucât, în cercetare modelele funcțional-diagnostice ale echipamentelor tehnice au servit drept bază pentru cercetarea diagnostică, au fost creat modele

funcțional-diagnostic ale echipamentelor tehnice. În figura A1.2. este ilustrat modelul pentru o centrală eoliană, WPP.

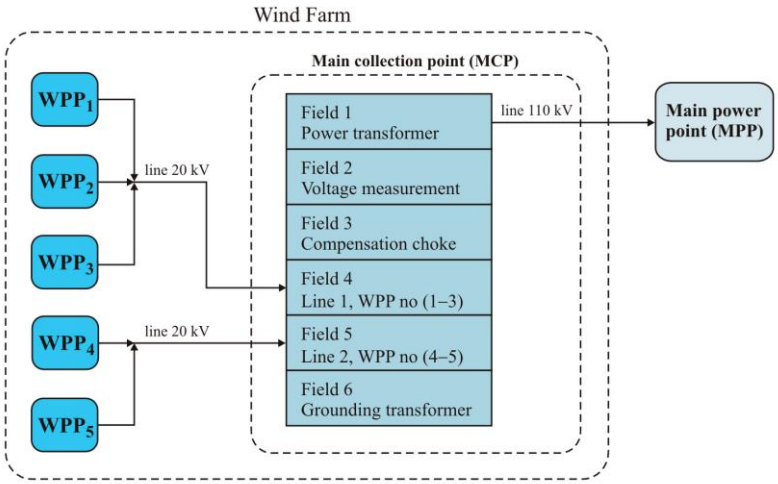


Fig. A1.1. Schema funcțională și de diagnosticare a echipamentelor centralei eoliene [A1]

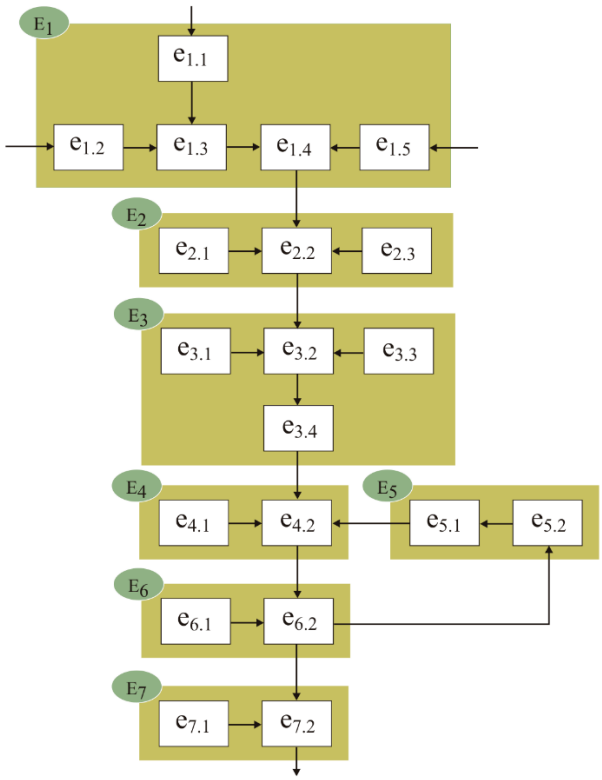


Fig. A1.2. Diagrama funcțională a unei turbine eoliene [A1]

În figura A1.2. notațiile E și e sunt componente esențiale ale unui generator eolian, precum E_1 sistemul de acționare a generatorului, E_2 sistemul generatorului sincron, $e_{1,1}...e_{1,5}$ sunt generatoarele turbinelor eoliene, $e_{2,1}$ este cablu electric de medie tensiune etc. În legătură cu aceste simboluri, autorii menționează faptul că în modelul centralei eoliene, au fost determinate șapte seturi *i*-funcționale ca urmare a evaluării funcțional-diagnostice. În fiecare *i*-set, a fost determinat un subset de *j*-elemente fundamentale (funcționale). Structura internă a modelului centralei eoliene este determinată de setul de elemente fundamentale $e_{i,j}$.

Modelul funcțional-diagnostic dezvoltat al centralei eoliene, prezentat în diagrama din figura A1.2, a servit autorilor bază pentru determinarea atât a unui set de semnale de diagnostic, cât și a semnalelor de referință aferente, care au fost notate cu $X(e_{i,j})$.

În următoarea etapă a cercetării, aceste modele au fost implementate în programul DIAG 2 și apoi folosite pentru a construi un sistem expert de supraveghere și siguranță pentru centrale eoliene. Figurile A1.3 și A1.4 ilustează capturi din programul menționat care arată implementarea unui model funcțional.

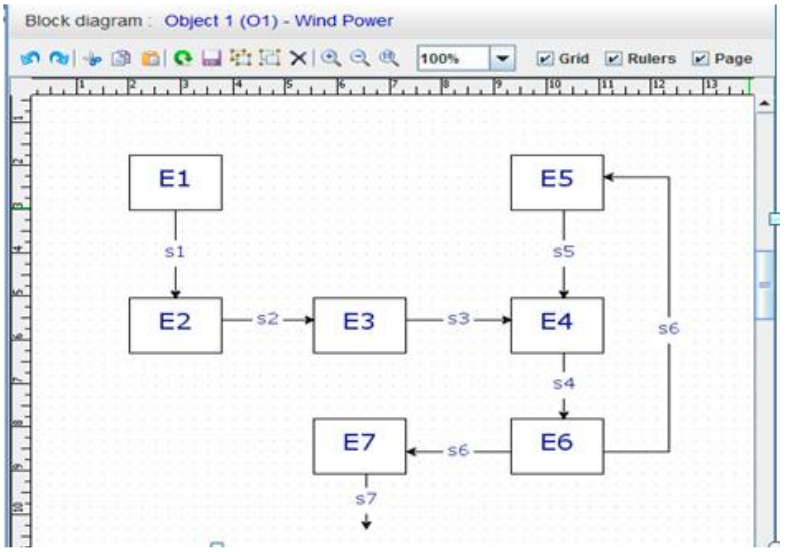


Fig. A1.3. Captură ecran Diag 2 cu modelul unei centrale eoliene [A1]

Analiza funcțională și diagnostică a modelului parcului eolian (figurile A1.3 și A1.4) au fost utilizate de autori pentru a produce semnale de măsurare și de referință, cum ar fi $\{X(e_{i,j})\}$ la ieșirile elementului funcțional $_j$.

În următoarea etapă a cercetării, autorii au construit baza de reguli a sistemului expert. Autorii au folosit datele (parametrii și unitățile de lucru) despre

centrala eoliană care au fost determinate prin analiza componentelor fundamentale ale acesteia (figura A1.5). Acest lucru a dus la identificarea a două unități funcționale-structurale fundamentale: un generator de turbină eoliană și o substație (figura A1.5). Apoi, pentru fiecare unitate funcțională-structurală, au fost determinate blocuri de bază care cuprind parametrii echipamentelor, sistemului și rețelei.

Diagnostic signals : Object 1 (O1) - Wind Power

Table of signals (Diagnostic) ☑ Sign Ø ☑ Color Ø

Unit	e1	e2	e3	e4	e5
E1	18	240	239	57	12,5
E2	0,0038	746	28	Ø	Ø
E3	38	1 487	1 487	1 487	Ø
E4	38	690	Ø	Ø	Ø
E5	3,35	1,38	Ø	Ø	Ø
E6	1,67	678	Ø	Ø	Ø
E7	37	1,75	Ø	Ø	Ø

Fig. A1.4. Captură ecran Diag 2 -introducerea datelor corespunzătoare modelului [A1]

În diagrama din figura A1.5. notațiile au următoarele semnificații: A – factori de mediu și înconjurători, B – rețeaua electrică, C – sistemul de conectare etc.

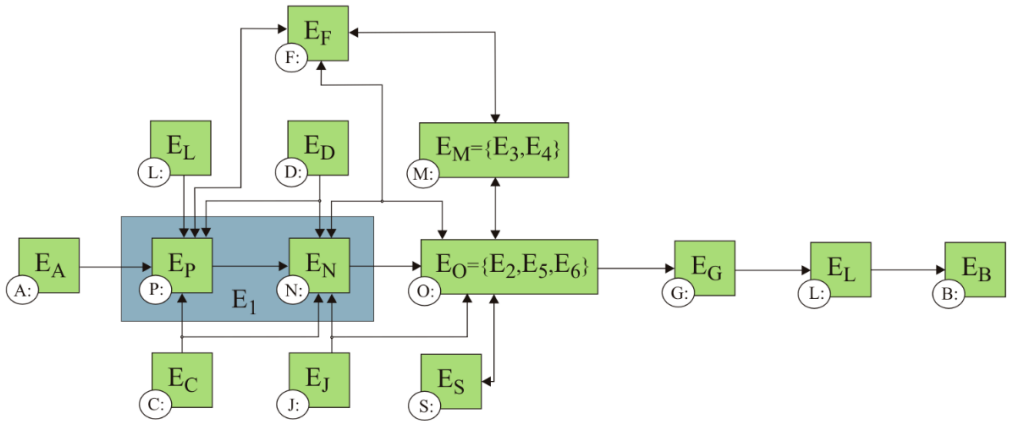


Fig. A1.5. Diagrama funcțională de diagnostic a unui echipament [A1]

La final, dezvoltarea parametrilor pentru întregul parc eolian, adică generatoarele turbinelor eoliene, rețelele electrice de medie și înaltă tensiune și stația electrică de înaltă tensiune a centralei, a permis definirea faptelor din care au fost create 109 reguli pentru sistemul de supraveghere și siguranță planificat al parcului eolian. În plu, regulile au fost create pentru toate blocurile funcționale dezvoltate, combinând și faptele din diferite blocuri, ceea ce a facilitat o rețea de relații reciproce. Exemple de reguli în legatură cu parametru A sunt ilustrate în figura A1.6.

RA001 := PA001 <3.5
RA002 := PA005 >= 20
RA003 := PA001 <3.5 or PA005>=20
RA004 := PA003 >= 12 and
RA005 := PA003 < 20 and
RA006 := PB003 = 2000
RA007 := PA003 >=12 and PA005 <20 and PB003 = 2000 and PI001 >= 85 and PI002 >= 85 and PI003 >=85
RA008:= PA004 >=3.5 and PA004 <12; and
RA009 := PB003 <2000
RA010:= PA004 >=3.5 and PA004 <12 and PB003<2000
RA011:= (RA001 = N); and
RA012 := PB013 > -120
RA013:= PA001>=3.5 and PB013> -120
RA014:= PA002 < 3.5; and
RA015 := (RA003 = N); and
RA016 := (RA006 = N); and
RA017:= PA002 < 3.5 and (RA013 = N) and (RA006 = N)

Fig. A1.6. Reguli asociate parametrului A [A1]

Baza de reguli și fapte au fost salvate în fișiere separate, care pot să fie folosite prin aplicații dezvoltate cu CLIPS.

În concluzie, se poate afirma despre baza de cunoștințe a sistemului expert WPPES următoarele:

- Stă la baza unui sistem expert decizional, deci a unui sistem expert de clasificare.
- Folosește relațiile dintre elemente, dar într-o formă simplificată.
- A fost implementat și poate fi folosit în aplicațiile CLIPS.

Anexa 2

Sistemul expert ESS4EE

Sistemul expert ESS4EE (Expert System Shell for Energy Efficiency) este prezentat în articolul științific:

[A2] Ioshchikhes, B.; Zink, R.; Ozen, O.; Weigold, M. A Holistic Framework for Developing Expert Systems to Improve Energy Efficiency in Manufacturing. Energies 2025, 18, 1406. https://doi.org/10.3390/en18061406.

Cercetarea prezentată în articol are drept scop dezvoltarea unui cadru holistic pentru dezvoltarea sistematică a sistemelor expert, precum și o structură de sistem expert care servește drept șablon software (sistem expert cadru). Cadrul este demonstrat și evaluat prin aplicarea sa într-un lanț de procese de prelucrare a metalelor.

Metodologia propusă de autori cuprinde trei faze: conceptualizare, dezvoltarea instrumentului și aplicare și validare. În faza de concepție, se selectează factorii de influență, iar regulile de calcul sunt definite și rafinate iterativ. Aceasta este urmată de implementarea computațională în faza de dezvoltare a instrumentului. În cele din urmă, SE este validat atât calitativ, cât și cantitativ.

Cadrul conceptual propus se bazează pe paradigma Design Science Research (DSR, care își propune să extindă capacitățile indivizilor și organizațiilor prin crearea de artefacte inovatoare. Aceste artefacte sunt entități structurate care variază de la software, logică formală și modele matematice până la descrieri informale în limbaj natural. În plus, DSR oferă perspective asupra modului în care artefactele pot fi proiectate în mod intenționat prin acțiunea umană pentru a atinge obiective specifice.), care este prezentată în literatura de specialitate.

Metodologia propusă în articol (figura A2.1) începe cu identificarea și prioritizarea echipamentele electrice relevante care contribuie la procesele analizate din cadrul companiei producătoare. Metode precum analiza Pareto sau portofoliul energetic pot fi aplicate de către managerul energetic, utilizând măsurători istorice sau date de sarcină nominală pentru a determina echipamentele electrice cu cea mai mare cerere de energie. Ulterior, se identifică informațiile legate de consumul de energie pentru a evalua performanța energetică. Regulile sunt de obicei structurate ca instrucțiuni IF-THEN, unde partea IF reprezintă cauzele (antecedentul), iar partea THEN efectele

(consecințele). Această etapă beneficiază, de asemenea, de o consultare cu operatorul echipamentelor pentru fezabilitatea tehnică. Pentru a automatiza extragerea informațiilor legate de energie, datele de măsurare trebuie colectate sistematic pentru dezvoltarea de modele. Managerul de energie este responsabil pentru planificarea experimentelor și coordonarea celor două persoane, adică managerul energetic și operatorul echipamentului. Procesul de dezvoltare a modelelor și algoritmilor bazați pe date poate urma modelul CRoss-IndustryStandard Process pentru dezvoltarea de aplicații de învățare automată cu metodologia de asigurare a calității (CRISP-ML(Q)). Etapele precedente produc algoritmi, modele bazate pe date și o bază de reguli, care sunt integrate în ES complet împreună cu cunoștințele expertului de către inginerul de cunoștințe în etapa finală. În timpul validării, toate instrumentele software dezvoltate, inclusiv sistemul expert general, sunt validate prin studii de caz cantitative și interviuri calitative cu utilizatorii. Se mai poate observa în figura A2.1 că în diferite etape ale dezvoltării SE pot fi implicate mai multe persoane (operatorul echipamentului electric, managerul energetic, inginerul de cunoștiințe și specialistul în prelucrarea datelor).

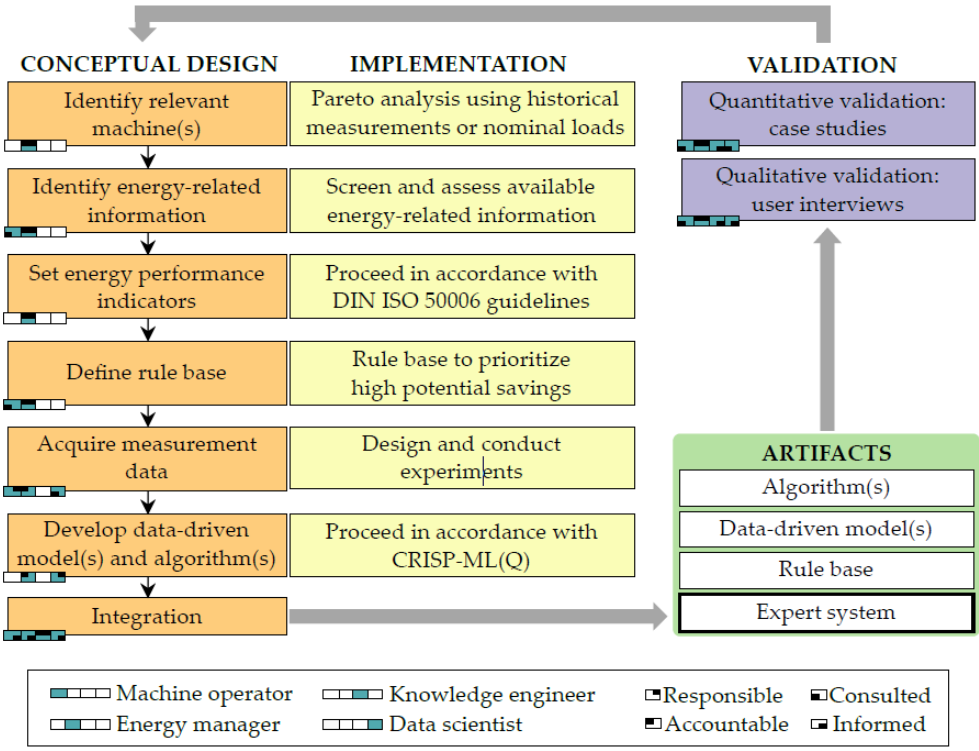


Fig. A2.1. Cadrul propus în [A2]

În articol se menționează folosirea sistemul expert cadru - Expert System Shell for Energy Efficiency (ESS4EE), care este disponibil pe GitHub întrucât acesta acceptă metodologia propusă de autori în lucrare. Sistemul expert cadru a fost implementat în Python 3.11 în Jupyter Notebook 7.3.2, și utilizează o arhitectură software modulară, așa cum este prezentat în figura A2.2.

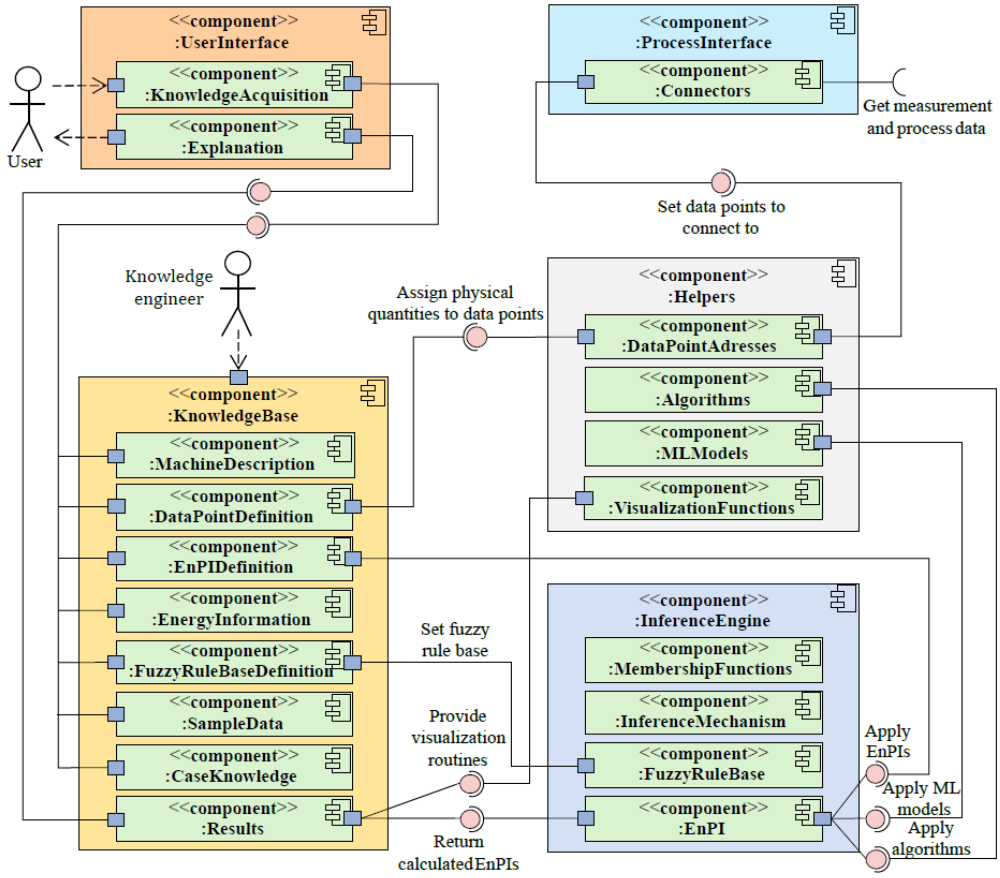


Fig. A2.2. Diagrama componentelor a sistemului ESS4EE [A2]

Așa cum se observă în figura de mai sus, arhitectura sistemului cuprinde module cheie pentru gestionarea cunoștințelor, manipularea datelor și luarea deciziilor. Baza de cunoștințe stochează cunoștințe de specialitate, inclusiv descrieri ale echipamentelor electrice și informații energetice. Aceste cunoștințe pot fi îmbogățite sau modificate de utilizatori prin intermediul interfeței utilizator. ProcessInterface permite accesul la date istorice și în timp real de la contoare de energie și controlere logice programabile, în timp ce modulul Helpers acceptă integrarea algoritmilor, organizarea adreselor și vizualizarea datelor. În

esență, InferenceEngine aplică logica fuzzy folosind funcții de apartenență, un mecanism de inferență și o bază de reguli. În cele din urmă, rezultatele ES sunt prezentate cu vizualizări, iar componenta explicativă îmbunătățește interpretabilitatea, asigurând informații utile pentru utilizatori.

În articol se prezintă o parte a bazei de reguli care este ilustrată în figura A2.3. Abrevierile folosite în definirea regulile se referă la caracteristicile echipamentelor electrice cu care se realizează procesul tehnologic considerat.

	Premise (IF)	Consequent (THEN)
1.1	$NPEF_m$ is high AND $NPTF_m$ is high	$P_{\text{npe},m}$ is very high
1.2	$NPEF_m$ is medium AND $NPTF_m$ is high	$P_{\text{npe},m}$ is high
1.3	$NPEF_m$ is low AND $NPTF_m$ is high	$P_{\text{npe},m}$ is medium
	⋮	
1.9	$NPEF_m$ is low AND $NPTF_m$ is low	$P_{\text{npe},m}$ is very low
2.1	$NPTF_m$ is high AND $NPTR_m$ is low	$P_{\text{npt},m}$ is very high
2.2	$NPTF_m$ is medium AND $NPTR_m$ is low	$P_{\text{npt},m}$ is high
2.3	$NPTF_m$ is low AND $NPTR_m$ is low	$P_{\text{npt},m}$ is medium
	⋮	
2.9	$NPTF_m$ is low AND $NPTR_m$ is high	$P_{\text{npt},m}$ is very low
3.1	$\overline{E}_{i,p}$ is high AND n_i is high AND s_i^2 is high	$P_{\text{pe},p}$ is very high
3.2	$\overline{E}_{i,p}$ is medium AND n_i is high AND s_i^2 is high	$P_{\text{pe},p}$ is very high
3.3	$\overline{E}_{i,p}$ is low AND n_i is high AND s_i^2 is high	$P_{\text{pe},p}$ is high
3.4	$\overline{E}_{i,p}$ is high AND n_i is medium AND s_i^2 is high	$P_{\text{pe},p}$ is high
	⋮	
3.27	$\overline{E}_{i,p}$ is low AND n_i is low AND s_i^2 is low	$P_{\text{pe},p}$ is very low

Fig. A2.3. Baza de reguli a sistemului expert dezvoltat [A2]

Ulterior, sistemul expert dezvoltat a fost validat și testat folosind date reale oferite de compania care a suportat cercetarea.

În concluzie, despre rezultatele cercetării prezentate în articolul prezentat succint în această anexă se pot afirma următoarele:

- S-a folosit un sistem expert cadru ESS4EE, care are la bază logica fuzzy pentru definirea cunoștințelor și funcționarea motorului de inferență.
- Sistemul expert dezvoltat este un sistem de clasificare.
- Explică etapele dezvoltării unui sistem expert și rolul fiecărei persoane implicate în dezvoltarea lui.

- Sistemu expert dezvoltat a fost testat cu ajutorul unor date reale obținute de la echipamente electrice, care realizează un proces tehnologic de prelucrare a metalelor.

Anexa 3

Sistemul expert ES4TPCM

Sistemul expert ES4TPCM (Expert System for Thoughput Parts Cleaning Machine) este descris în articolul științific:

[A3] Ioshchikhes, B.; Frank, M.; Elserafi, G.; Magin, J.; Weigold, M. Developing Expert Systems for Improving Energy Efficiency in Manufacturing: A Case Study on Parts Cleaning. Energies 2024, 17, 3417. https://doi.org/10.3390/en17143417.

Articolul menționat descrie dezvoltarea sistematică a unui sistem expert, a cărui scop este de a ajuta la creșterea eficienței energetice a mașinilor de producție. Sistemul expert propus utilizează modele de regresie bazate pe date și o bază de reguli fuzzy pentru a identifica setările ineficiente ale parametrilor, a calcula economiile de energie realizabile și a prioritiza acțiunile. Măsurile propuse sunt aplicate mai întâi unui model de simulare analitică în timp real al unei mașini de producție pentru a verifica dacă sunt îndeplinite constrângerile necesare pentru calitatea specificată a produsului. Acest lucru oferă operatorului mașinii mijloacele experte pentru a aplica măsurile de eficiență energetică propuse entității fizice. La final, sistemul este testat pe o mașină reală.

Metodologia de dezvoltarea a sistemului expert, elementele sale componente și actorii care sunt implicați în construirea lui sunt prezentați în figura A3.1. În figura menționată se observă o abordare care conține patru etape, și anume, conceptualizarea, implementarea, aplicarea și validarea. În cursul procesului de construire a sistemului expert sunt dezvoltate mai multe rezultate, denumite de autori artefacte, precum modele bazate pe date, modele simulate, baza de reguli fuzzy și sistemul expert general.

Modelele bazate pe date sunt obținute experimental. Datele folosite sunt informații energetice și despre alți paramterii. Pentru dezvoltarea modelelor bazate pe date, este necesar un set de date în care ambele fluxuri de date sunt legate (date etichetate). În articol se menționează că dezvoltarea modelelor bazate pe date poate fi realizată conform modelului CRoss-Industry Standard Process pentru dezvoltarea aplicațiilor de Machine Learning cu metodologia de asigurare a calității (CRISP-ML(Q)).

Modelele de simulare sunt folosite pentru a reprezenta alte caracteristici fizice ale echipamentelor electrice. Întrucât, modele bazate pe date sunt folosite

pentru cuantificarea potențialelor de economisire a energiei și modele de simulare pentru verificarea acțiunilor recomandate.

Dezvoltarea modelelor de simulare este un proces cu mai multe faze. În primul rând, se definește obiectivul general al simulării. Pe baza acestuia, se formulează aspectele calitative și cantitative ale modelului. Aceasta include identificarea componentelor și structurilor necesare ale aplicației din lumea reală care trebuie simulate. Ulterior, relațiile dintre structurile identificate sunt izolate și descrise calitativ. Aici, pot fi aplicate diferite abordări pentru descrierea interacțiunii (greybox, whitebox, blackbox). În cercetarea prezentată în articol au a fost folosită reprezentarea cunoștințelor prin implicarea claselor, a subclaselor și a obiectelor.

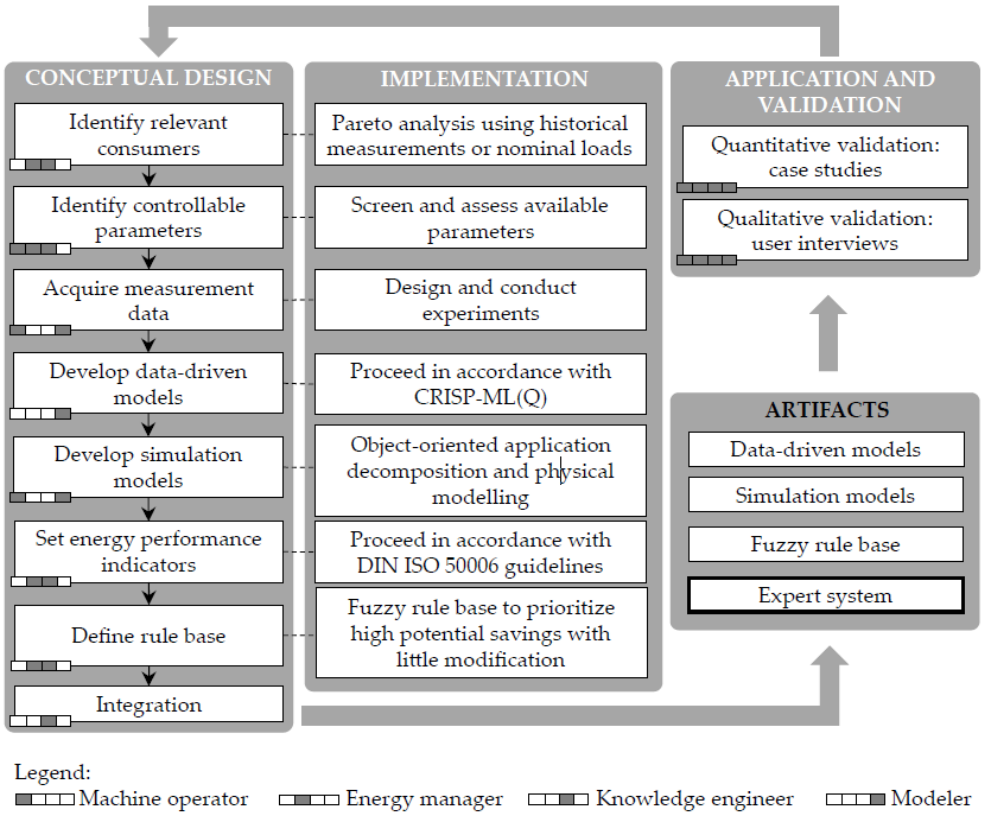


Fig. A3.1. Metodologia propusă pentru dezvoltarea SE [A3]

Baza de reguli fuzzy este dezvoltată prin folosirea indicatorilor de performanță energetică, care au fost obținuți cu ajutorul modelelor prezentate anterior.

În cele din urmă, cele trei artefacte prezentate sunt integrate într-un sistem expert general cuprinzător, care este, de asemenea, validat și eficientizat, dacă este necesar.

Sistemul expert are la bază un proces decizional, care este ilustrat]n figura A3.2. Astfel, sistemul expert poate ajuta operatorul echipamenului electric în a lua cea mai bună decizie pentru a crește eficiența energatică.

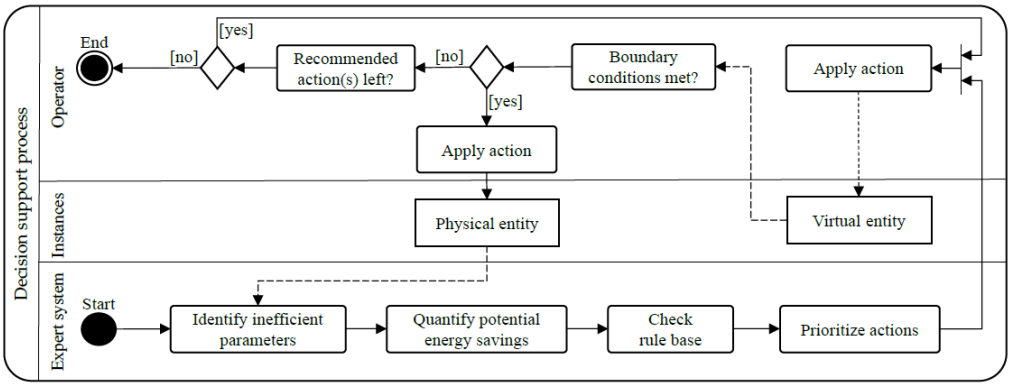


Fig. A3.2. Structura procesului decizional [A3]

Sistemul expert dezvoltat a fost testat printr-un studiu de caz. Studiul de caz analizează curățarea discurilor metalice de ghidare pentru cutiile de viteze din fabrica de cercetare ETA, care sunt contaminate cu ulei de tăiere. Piesele prezintă găuri de trecere și găuri înfundate pe suprafața superioară. Procesul are loc într-un echipament electric de curățare a pieselor cu bandă de transfer (TPCM). Schema de bază a unui astfel de echipament este ilustrată în figura A3.3.

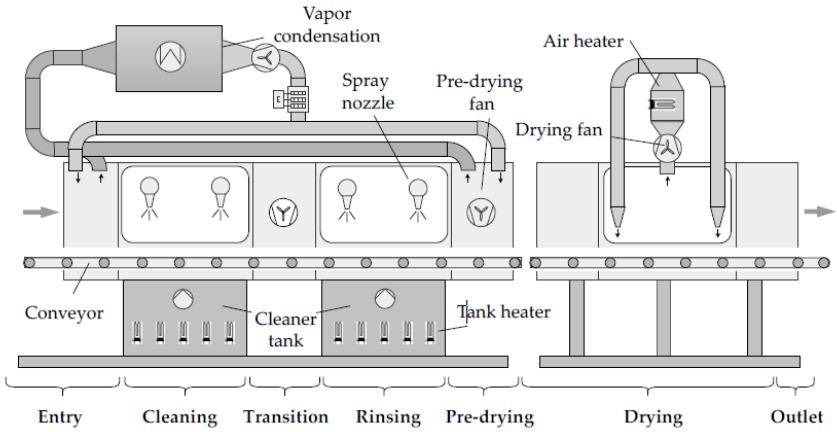


Fig. A3.3. Echipamenul electric de curățare

În studiul de caz prezentat de autori în articol, utilizarea sistemului expert a indicat un potențial considerabil de economisire a energiei, de până la 45,35% față de scenariul de referință.

În concluzie, despre sistemul expert prezentat în această anexă, se pot afirma următoarele:

- Este un sistem expert decizional, care ajută operatorii într-o întreprindere să ia cele mai bune decizii, cu scopul de a crește eficiența energetică.
- Reprezentarea cunoștințelor se face prin folosirea obiectelor, claselor și subclaselor.
- Se folosește modele bazate pe date și de simulare, precum și o bază de reguli fuzzy pentru construcția bazei de cunoștințe.

Anexa 4

Sistemul expert eBRBES

Sistemul expert eBRBES (explainable Belief Rule-Based Expert System) este prezentat în articolul științific:

[A4] Kabir, S.; Hossain, M.S.; Andersson, K. An Advanced Explainable Belief Rule-Based Framework to Predict the Energy Consumption of Buildings. Energies 2024, 17, 1797. https://doi.org/10.3390/en17081797.

Articolul menționat prezintă un sistem expert avansat, explicabil, bazat pe reguli de credință, cu explicații bazate pe cunoștințe de domeniu pentru predicția precisă a consumului de energie în clădiri.

Arhitectura sistemului expert propus este prezentat în articol este ilustrat în figura A4.1.

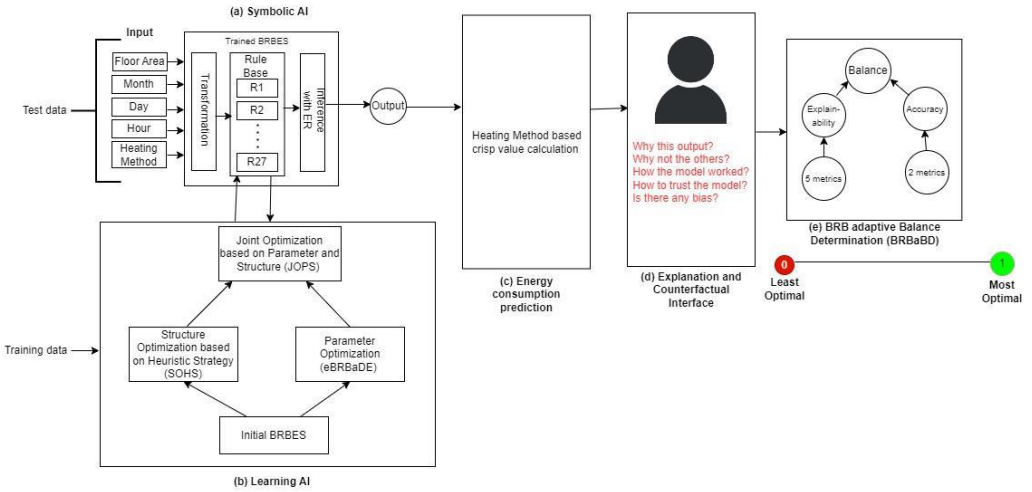


Fig. A4.1. Arhitectura sistemului eBRBES

În figură se observă cinci valori de intrare sunt introduse în BRBES:

- suprafața (metri pătrați),
- luna (ianuarie - decembrie),
- ziua (luni - duminică),
- ora (00:00 - 23:00),
- metoda de încălzire (centrală sau electrică).

Reprezentarea cunoștințelor s-a realizat prin reguli de credință. O regulă de credință constă din două părți: o parte antecedentă și o parte consecventă. Baza de reguli are trei atribute antecedente: suprafața clădirii, iluminarea din ziua considerată și nivelul de ocupare interioară a clădirii. Pentru determinarea nivelului de iluminare a unei zile (între 0 și 1), se face în funcție de lună și oră, pe baza orei de răsărit și apus. Autorii propun pentru calcularea gradului de ocupare interioară (între 0 și 1) pe baza zilei lucrătoare/weekend-ului, lunii și orei. Fiecare atribut antecedent are trei valori referențiale: ridicat (Î), mediu (M) și scăzut (L). Atributul consecvent „Consum de energie" are, de asemenea, aceleași trei valori referențiale. Considerând aspectele menționate anterior, autorii au dezvoltat 27 de reguli. Valorile numerice din partea consecventă a acestei baze de reguli reprezintă gradele de credință ale valorilor referențiale respective.

În figura A4.1. se observă IA simbolică (modulul notat cu a)), care este un algoritm rezolutiv cu rolul de realizare a raționamentului în patru etape, și anume transformarea intrării, calculul ponderii de activare a regulilor, actualizarea gradului de credință și agregarea regulilor.

Transformarea intrărilor - în această etapă, datele de intrare ale tuturor celor trei atribute antecedente ale bazei de reguli sunt distribuite în funcție de valorile lor referențialelor respective. Pentru suprafața locuinței, au fost setate valorile de utilitate pentru L, M și H la 10, 85 și respectiv 200. Pentru lumina naturală, valorile de utilitate pentru L, M și H sunt 0, 0,50 și respectiv 1. Pentru ocupare, valorile de utilitate pentru L, M și H sunt 0,10, 0,55 și respectiv 1.

În etapa de calcul al ponderii de activare a regulii, se calculează ponderiile de activare a fiecăreia dintre cele 27 de reguli ale bazei de reguli. În continuare, autorii iau în considerare gradul de potrivire al fiecărei reguli, ponderarea regulii, numărul total de atribute antecedente și ponderarea fiecărui atribut antecedent pentru a calcula ponderarea de activare (0 la 1) a fiecărei reguli în raport cu valorile de intrare. Ecuația matematică pentru calcularea ponderii de activare a fiecărei reguli este prezentată în articol.

Actualizarea gradului de credință se realizează astfel: dacă datele de intrare pentru oricare dintre atributele antecedente devin indisponibile din cauza incertitudinii datorate ignoranței, gradele de credință inițiale ale valorilor referențiale consecutive sunt actualizate după o relație definită.

În etapa următoare, autorii folosesc o abordare analitică bazată pe raționamentul evidențial, pentru a agrega toate regulile din baza de cunoștințe selectate. Apoi, se calculează gradul de credință agregat final pentru fiecare

dintre cele trei valori referențiale ale atributului consecutiv cu ecuația analitică a raționamentului evidențial. Gradul de credință agregat final pentru valorile referențiale H, M și L ale atributului consecutiv este 0,86, 0,14 și respectiv 0.

Modulul notat cu b) în figura A4.1. învățarea IA este folosit pentru optimizarea atât a parametrilor, cât și a structurii sistemului expert pentru o precizie mai mare. În ceea ce privește parametrii de învățare, autorii optimizează valorile de utilitate ale atributelor antecedente, ponderea regulii, ponderea atributului antecedent și gradele de credință ale atributelor consecutive cu ajutorul algoritmului îmbunătățit Belief Rule-Based Adaptive Differential Evolution (eBRBaDE). Autorii afirmă faptul că abordarea echilibrată a eBRBaDE între explorare și exploatare pentru a stabili valorile adecvate ale parametrilor de control (factori de încrucișare și mutație) este atribuită performanței sale mai bune decât cea a Evoluției Diferențiale (DE).

Predicția consumului de energie (figura A4.1c)) este codulul care transformă evaluarea multi-valorică a BRBES optimizată pentru JOPS într-o singură valoare numerică precisă, care reprezintă consumul de energie în kWh. Metoda de încălzire a apartamentelor este luată în considerare pentru a calcula valoarea precisă finală.

În interfața explicativă și contrafactuală din figura 4.1d), se explică rațiunea din spatele rezultatului predictiv. Această explicație se bazează pe regula cu cea mai mare pondere de activare, care în exemplul dat în articol este regula 16. Modelul explicativ este următorul:

„Lumina zilei este [e1] într-un [e2] [e3], rezultând o probabilitate [e4] ca oamenii să stea în interior într-un [e5] [e3]. Prin urmare, datorită suprafeței [e6], luminii zilei [e1], ocupării interioare [e4] și metodei de încălzire [e7], nivelul consumului de energie a fost prezis a fi în mare parte [e8].” Unde e1 = valoarea referențială a luminii zilei în conformitate cu cea mai mare pondere de activare; e2 = anotimpul anului. Iunie-august este sezonul de vară, septembrie-octombrie este toamna, noiembrie-martie este iarna, iar aprilie-mai este primăvara; e3 = ziua.

Autorii au folosit C++ (versiunea 20) și Python (versiunea 3.10) pentru a implementa framework-ul eBRBES propus în articol. Datele care se află în spatele sistemului expert sunt salvate în fișiere separate și apoi accesate în funcție de necesitate.

În final, după testarea sistemului de prognoză propus, autorii afirmă faptul că rezultatele experimentale ale sistemului arată că cadrul eBRBES propus de ei

are un echilibru optim mai mare între explicabilitate și acuratețe decât alte tehnici de învățare automată propuse în literatura de specialitate.

În concluzie, despre sistemul expert prezentat se pot afirma următoarele:

- Este un sistem expert de prognoză, folosit pentru a obține prognoza consumului de energie electrică în clădiri.
- Sistemul cuprinde mai mute module, inclusiv unul de învățare, deci are o structură hibridă, ci nu una clasică.

Anexa 5

Sistemul expert FES-EESHM

Sistemul expert FES-EESHM (Fuzzy Expert System for Energy Efficiency Home Management) este descris în articolul științific

[A5] Zhang, R.; V E, S.; Jackson Samuel, R.D. Fuzzy Efficient Energy Smart Home Management System for Renewable Energy Resources. Sustainability 2020, 12, 3115. https://doi.org/10.3390/su12083115.

Scopul sistemului expert hibrid este de a controla sistemul de management energetic pentru o locuință folosind avantajele combinație dintre logica fuzzy și un sistem expert de control. Rezultatul este un controler hibrid care este folosit pentru a crește eficiența energetică a unei locuințe care cuprinde receptoare electrice și generatoare bazate pe surse regenerabile.

Elementele componente ale sistemul expert fuzzy propus în articolul științific și modul în care a fost el realizat sunt ilustrate în figura A5.1.

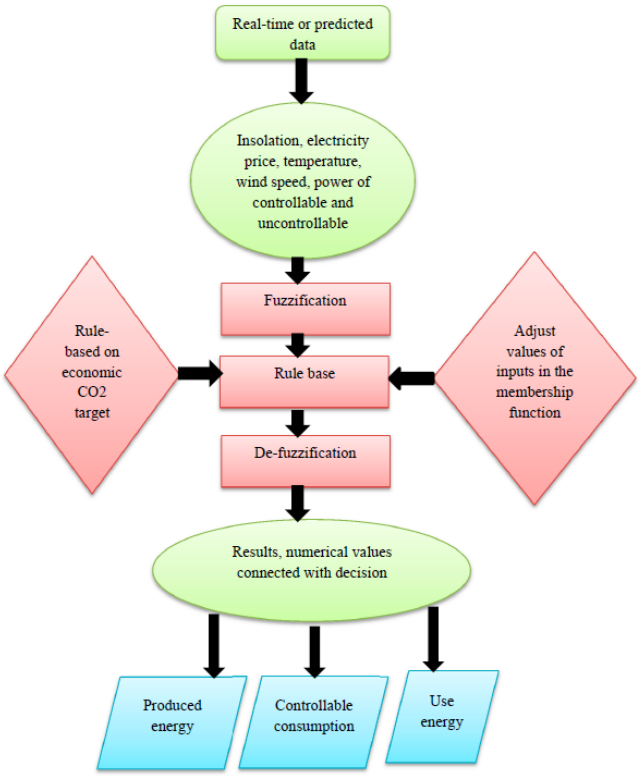


Fig. A5.1. Structura sistemului expert fuzzy

Trebuie menționat faptul că, sistemul folosește funcții de apartenență care au fost obținute din curbele de sarcină ale consumatorului considerat. Baza de cunoștințe ale sistemului expert este descrisă printr-o listă de reguli, care pot fi reformate pentru a maximiza profitul, a reduce costul energiei sau orice altceva este obiectivul utilizatorului. Autorii menționează faptul că, în cooperare cu companiile de utilități, se pot stabili reguli pentru reducerea emisiilor de CO2. Precum se observă în figura A5.1., defuzzificarea este următorul și ultimul pas. În această etapă se convertește inferența dispozitivului într-un semnal de ieșire.

Managementul energiei pentru o casă inteligentă (figura A5.2) constă dintr-un contor inteligent, un sistem de stocare a energiei, o unitate de control și monitorizare și electrocasnice programate. Contorul inteligent ajută la colectarea informațiilor privind stimulentele de preț, răspunsul la cerere (Demand Response) și stabilirea prețurilor în timp real din programul de management al energiei. Acesta joacă un rol important în rețelele inteligente, care facilitează comunicarea bidirecțională între clienți și locuințe. O varietate de tehnologii avansate, cum ar fi Wi-Fi și ZigBee, interacționează cu controlerul FES-EESHM. Receptoarele sunt clasificate în două module diferite: electrocasnice inteligente și electrocasnice tradiționale pe baza modelelor lor de consum de energie și a interacțiunii dintre acestea.

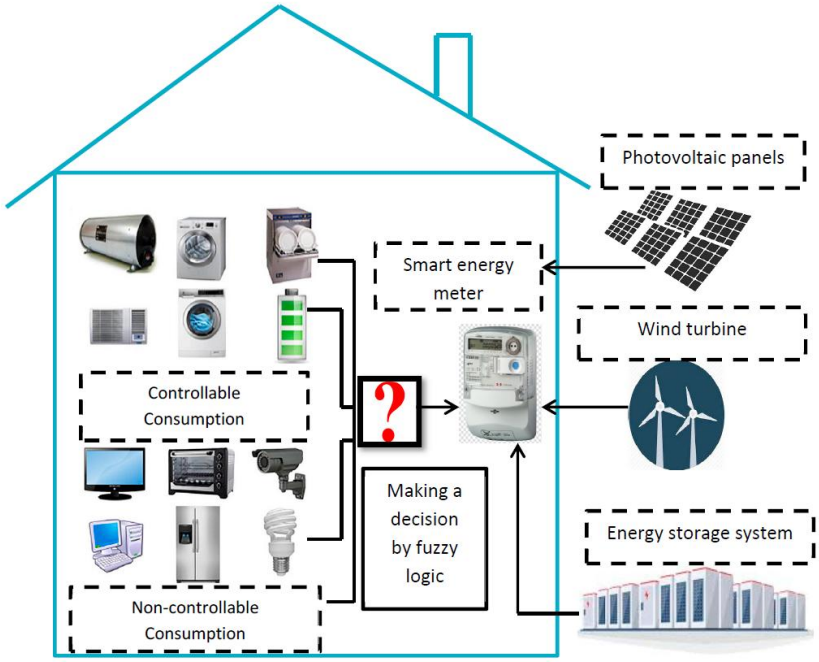


Fig. A5.2. Sistemul de management energetic al unei case inteligente

Consumul de energie al aparatelor inteligente este considerat în funcție de trei parametri, și anume consumul total de energie, consumul de energie și perioada de timp. Consumul de energie electrică depinde de puterile aparatelor precum răcitoarele de apă, frigiderele și aparatele de aer condiționat. Aceste puteri pot fi utilizate pentru a reduce la minimum consumul mare de energie, puterea de mare viteză până la puterea medie și costul energiei electrice.

Articolul descrie în detaliu baza matematică care guvernează sistemul de management. Aceasta, în esență susține algoritmul prin care se realizează alimentarea receptoarelor consumatorului luând în considerare energia produsă din sursele regenerabile, încărcarea sistemului de stocare și prețul energiei electrice.

Sistemul expert fuzzy a fost testat și evalut, iar rezultatele obținute comparate cu metodologii similare din alte studii prezentate în literatura de specialitate. Se observă din graficul din figura A5.3 faptul că, metoda propusă de autori în articol dă rezultatele cele mai bune.

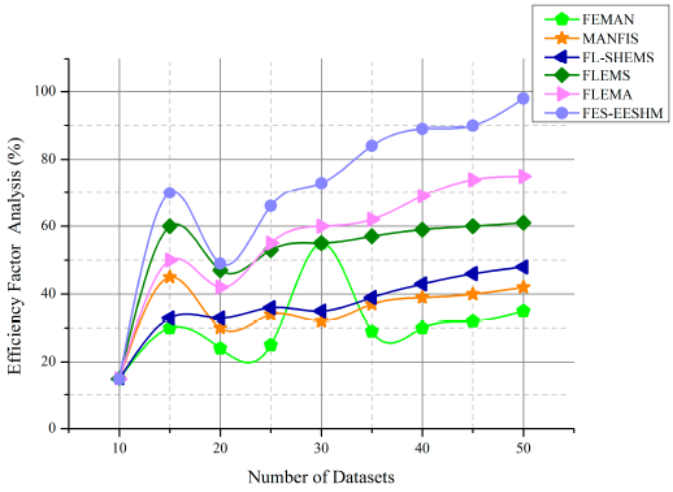


Fig. A5.3. Analiza factorului de eficiență [A5]

În concluzie, sistemul expert prezentat are următoarele caracteristici:

- Este un sistem expert fuzzy de control, care are drept scop realizarea managementului energetic într-o locuință, cu scopul de a crește eficiența energetică și prin folosirea surselor regenerabile de energie. Mai exact, sistemul FES-EESHM propus poate reduce eficient raportul dintre vârf și mediu și costul energiei electrice cu o valoare minimă a sarcinii de vârf.

El poate controla consumul de energie, stocarea și producția de energie, a programa tranzacționarea energiei și a gestiona sarcinile pentru a reduce fluxul de energie și costul energiei electrice.

- Reprezentarea cunoștințelor se realizează prin implicarea logicii fuzzy, adică mărimile fizice cu care lucrează sistemul expert sunt variabile și valori fuzzy.
- Motorul de inferență funcționează pe baza logicii fuzzy, deci lucrează cu numere și valori fuzzy.

Anexa 6

Sistemul expert ES4FD

Sistemul expert ES4FD (Expert System for Fault Diagnosis) este prezentat în articolul științific:

[A6] Zhou, Z.; Ma, Z.; Jiang, Y.; Peng, M. Fault Diagnosis Using Bond Graphs in an Expert System. Energies 2022, 15, 5703. https://doi.org/10.3390/en15155703.

Scopul sistemului expert este de a realiza diagnoza defecțiunilor apărute la sistemul de răcire a unui reactor nuclear. Caracteristica de bază a sistemului expert este reprezentarea cunoștințelor care se face prin implicarea grafurilor de legătură dintre cunoștințe.

Un graf de legătură este este un limbaj de modelare grafică utilizat în inginerie pentru a reprezenta și analiza fluxul de energie în cadrul sistemelor dinamice. Acesta oferă o reprezentare vizuală a modului în care diferite componente ale unui sistem interacționează și schimbă energie, permițând o înțelegere cuprinzătoare a comportamentului sistemului, în special în sistemele multi-domeniu. Tipurile de noduri folosite în studiu sunt prezentate în figura A6.1.

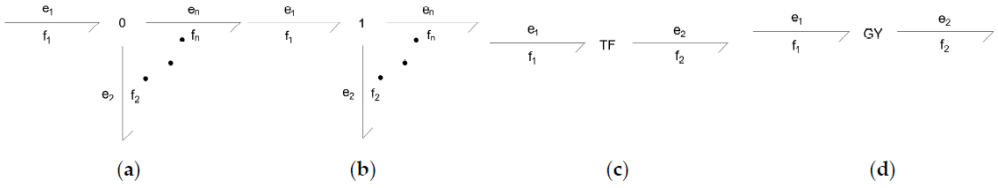


Fig. A6.1. Tipuri de noduri folosite în studiul prezentat în [A6] a) nod de tip 0, b) nod de tip 1, c) nod de tip transformator, d) nod de tip girator

Sistemul de răcire a unui reactor nuclear, care a fost folosit în studiu, și pentru care s-a dezvoltat un graf de legături este prezentat în figura A6.2. Așa cum se poate observa în figură, sistemul de răcire a reactorului este alcătuit din reactor, un presurizator, două generatoare de abur, patru pompe de răcire a reactorului și conducte și valve între componentele echipamentului.

Structura sistemului expert propus în articol pentru realizarea diagnozei defecțiunilor apărute în sistemul de răcire a reactorului nuclear propus este ilustrată în figura A6.3.

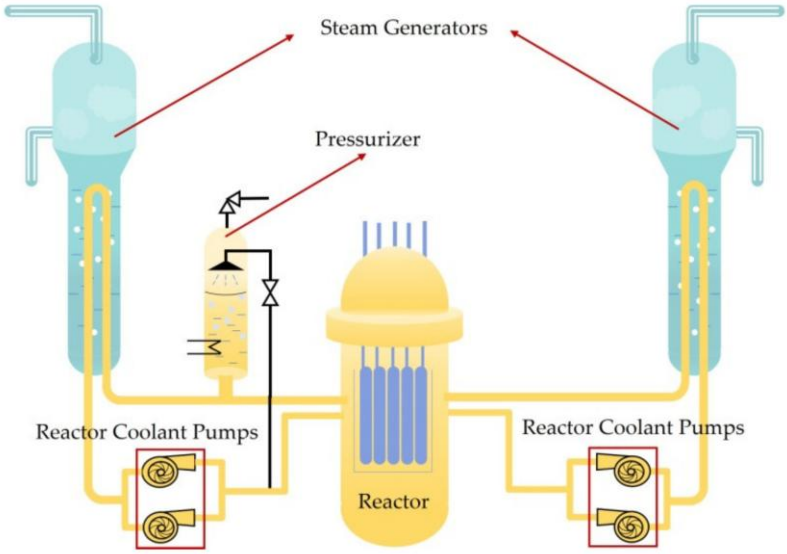


Fig. A6.2. Sistemul de răcire a unui reactor nuclear [A6]

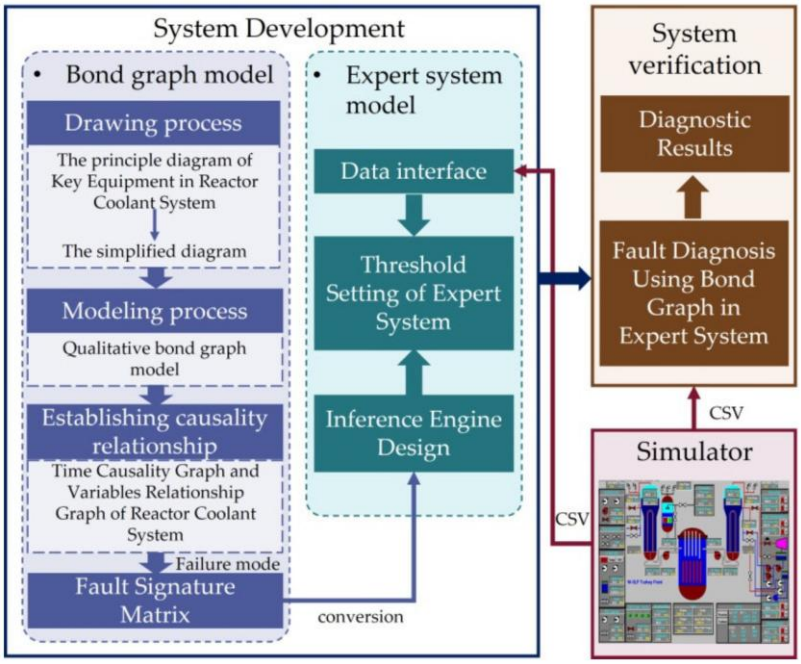


Fig. A6.3. Structura sistemului expert propus în articolul [A6]

Modul în care a fost obținut modelul graf a unei componente a sistemului de răcire este ilustrat în figura A6.4.

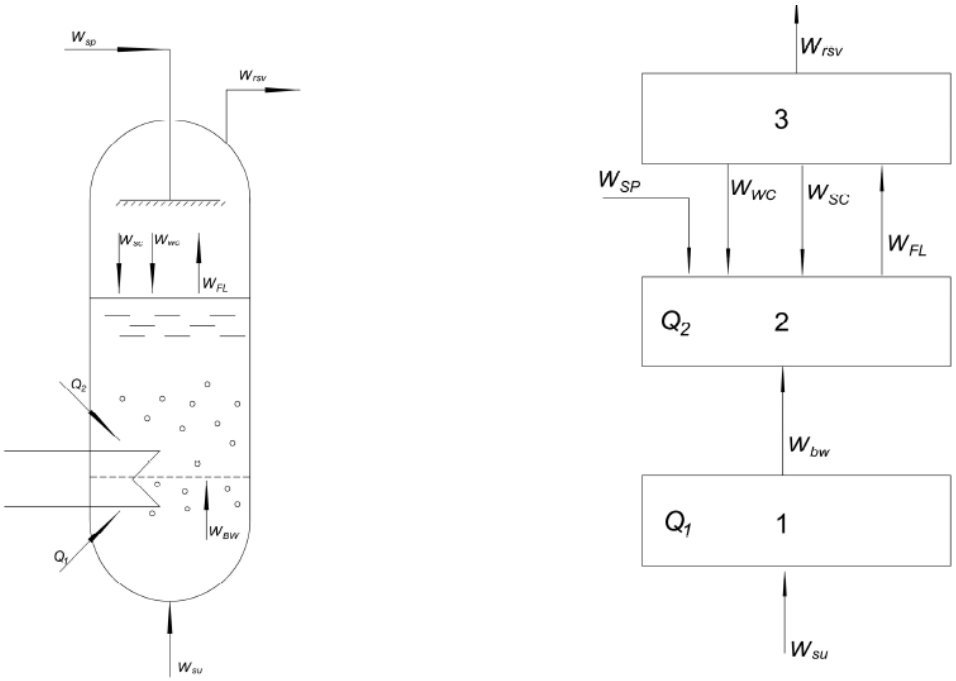


Fig. A6.4. Diagrama simplificată sub formă de graf a presurizatorului [A6]

În același mod se procesează pentru toate elementele sistemului de răcire, proces, care este prezentat în amănunt în articol. Modelele de grafuri de legături ale componentelor sistemului de răcire sunt folosite în continuare.

În studiu prezentat se trece apoi la grafuri de cazualitate temporală. Un graf al cauzalității temporale (TCG) este un graf orientat cu variabile de sistem ca noduri, care poate caracteriza relația dintre variabilele din sistem. Calea de la o variabilă la alta în TCG corespunde unei căi cauzale în graful de legături. Calea cauzală a grafului de legături se referă la un grup de chei în aceeași direcție prin care se poate obține matricea semnăturii defectului sistemului. De obicei, TCG este utilizat pentru a deriva ipoteza defectului stabilită în diagnosticarea calitativă a defectului. Graful de cazualități a presurizatorului este ilustrat în figura A6.5.

Autorii afirmă faptul că un graf al cauzalității temporale este o pretratare a unui graf de legături, iar graful relațiilor variabile se obține pe baza grafului cauzalității temporale. Deși graful cauzalității temporale poate reprezenta relația cauzală dintre variabile și este ușor să se extragă informații cheie din graful de legături, metoda sa de reprezentare nu este ușor de distins în procesul de modelare. Prin urmare, transformarea grafului cauzalității temporale, care nu este ușor de distins, în graful relațiilor variabile poate reflecta relația dintre variabile

și poate oferi o bază teoretică pentru variația coeficienților din matricea semnăturii defectelor. Prin intermediul grafului de legături, se determină intrarea și ieșirea fiecărui volum de control, și se trasează graful relațiilor variabile corespunzător. Prin intermediul TCG, se poate determina influența reciprocă a fiecărui parametru, iar apoi se poate trasa graful relațiilor variabile dintre parametri.

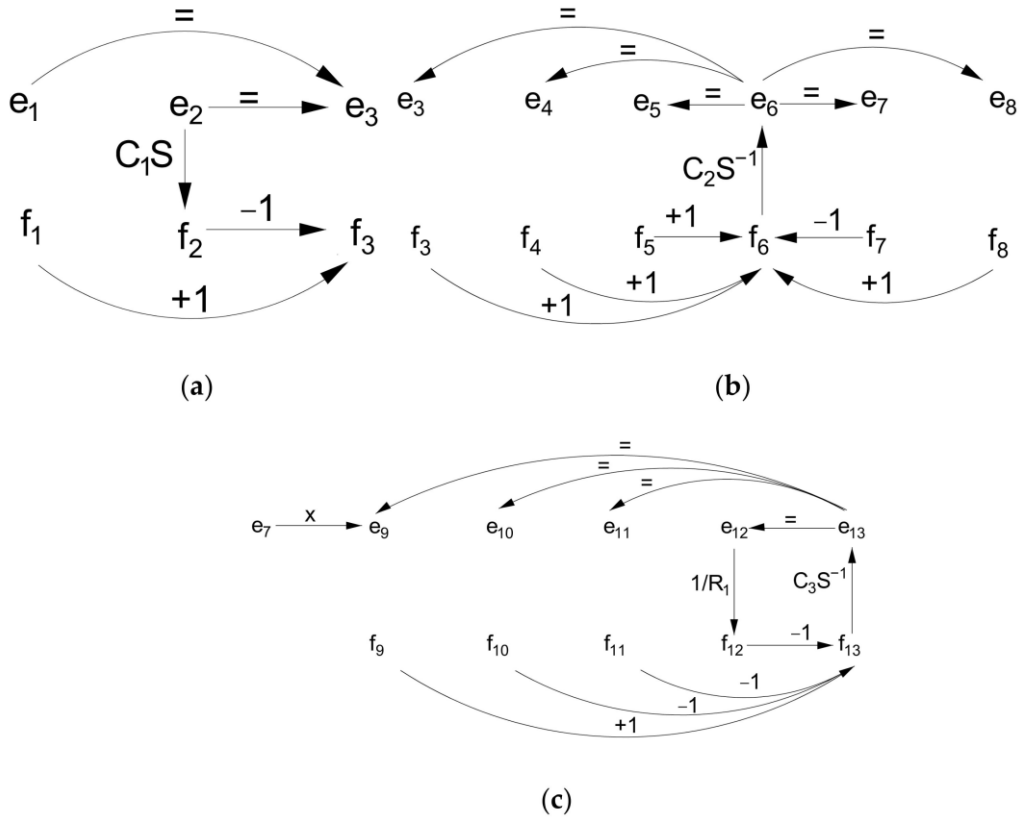


Fig. A6.5. Grafuri de cazualități corespunzătoare presurizatorului [A6]

Folosind aceste grafuri, autorii dezvoltă o matrice a defectelor sistemului de răcire și relațiile dintre acestea și elementele componente ale sistemului de răcire. Mai departe, această matrice a fost folosită pentru implementarea motorului de inferență a grafurilor de legătură pentru toate elementele sistemului de răcire a reactorului. În lucrare sunt descrise și explicate toate acest einformații.

În continuare, aceste legături sunt implementate; rezultatul obținut pentru presurizator este ilustrat în figura A6.6.

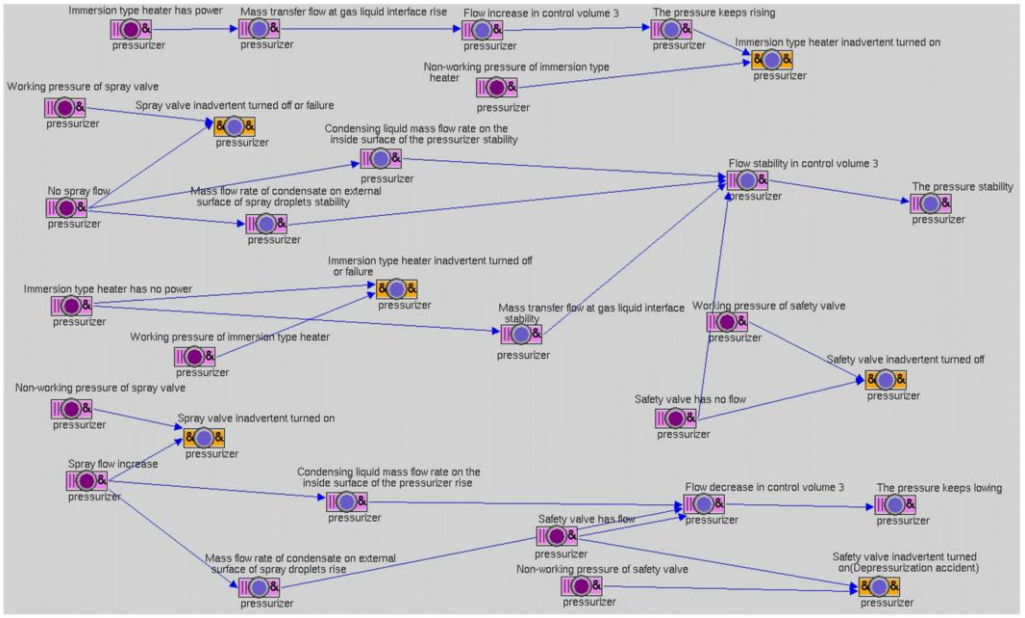


Fig. A6.6. Definirea motorului de inferență pentru presurizator [A6]

În concluzie, despre sistemul expert ES4FD prezentat în această anexă, se poat afirma următoarele:

- Este un sistem de clasificare, dedicat diagnozei defecțiunilor apărute la sistemul de răcire a unui reactor nuclear.
- Folosește grafuri de legătură pentru a realiza reprezentarea cunoștințelor printr-o abordare relațională.
- Motorul de inferență a SE cuprinde mai multe componente dedicate fiecărei ramuri a grafului de legături general.

Anexa 7

Sistemul expert ES4ECEC

Sistemul expert ES4ECEC (Expert System for Energy Consumption in Electric Cars) este prezentat în articolul științific

[A7] Deptuła, A.; Augustynowicz, A.; Stosiak, M.; Towarnicki, K.; Karpenko, M. The Concept of Using an Expert System and Multi-Valued Logic Trees to Assess the Energy Consumption of an Electric Car in Selected Driving Cycles. Energies 2022, 15, 4631. https://doi.org/10.3390/en15134631.

Scopul sistemului expert este de a susține procesul decizional privind consumul rațional de energie al unei mașini electrice. Factorii care sunt luați în considerare sunt stilul de condus (frânarea și accelerarea), viteza medie atinsă și temperatura ambientală. Sistemul propus se bazează pe metoda arborilor logici multi-valorici, ceea ce permite minimizarea funcției obiectiv și deci a consumului de energie a mașinii electrice la diferite temperaturi ambientale. Deciziile generate de SE sunt direcționate către sistemul de management al energiei, și pot fi procesate într-o mare varietate de situații. Sistemul poate da indicații soferului cu privire la modificarea stilului de condus pentru a minimiza consumul.

Sistemul expert a fost simulat împreună cu modelul mașinii electrice, a cărei diagrame bloc este ilustrată în figura A7.1. Modelul vehiculului electric a fost modelat luând în considerare că conține un motor, o baterie, un controler, convertoare și roți. Motorul a fost conectat la diferențialul roții.

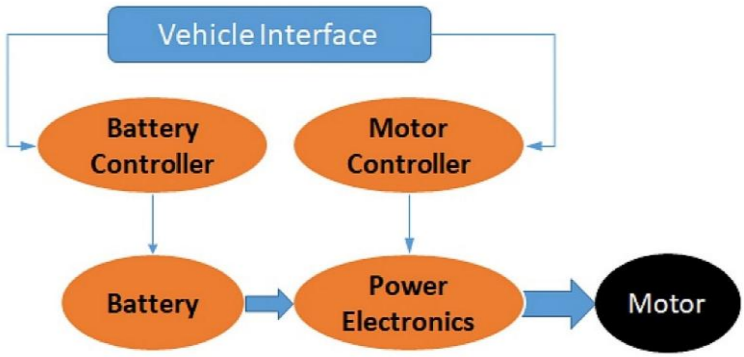


Fig. A7.1. Diagrama bloc a unui vehicul electric [A7]

Figura A7.2. prezintă schematic diagrama sistemului de acționare alectrică a vehiculului electric considerată în studiu descris în articol. În figură se observă

configurația în care un set de baterii care este utilizat pentru a furniza curent continuu circuitului, care este apoi conectat la circuitul invertorului. Roțile vehiculului sunt conectate printr-un diferențial, care asigură un echilibru între tracțiunea spate și tracțiunea față. O parte a diferențialului în modelul adoptat este un mecanism de angrenaje care permite arborelui de transmisie să se rotească la viteze diferite.

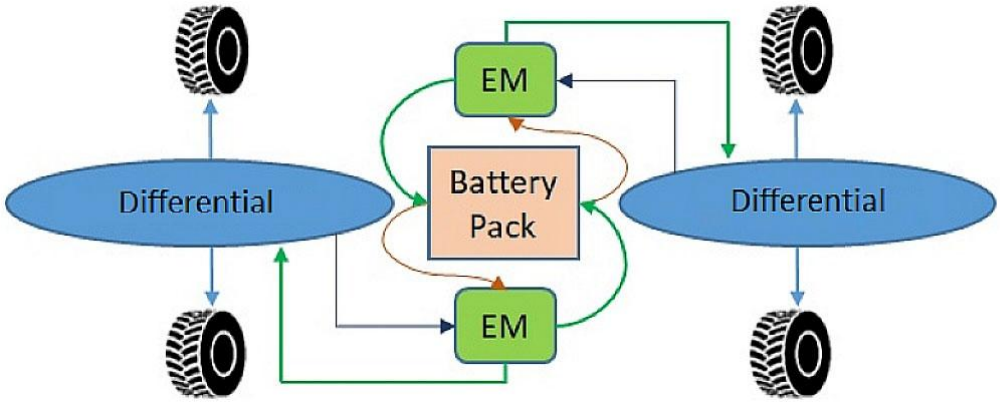


Fig. A7.2. Schema sistemului de acționare electrică a vehiculului [A7]

În implementarea modelului sistemului de acționare a vehiculului au fost luate în considerare modele energetice simplificate ale motorului și invertorului. Baza pentru dezvoltarea modelelor au fost hărțile simulate ale dependenței eficienței motorului și invertorului de viteza motorului și cuplul de sarcină. În fiecare etapă de calcul, informațiile privind eficiența motorului sau invertorului au fost determinate din viteza curentă și a cupluluia motorului. Modelul de simulare dezvoltat în studiu are capacitatea de a seta condițiile de conducere (viteză, direcție, altitudine) pe baza unei biblioteci de cicluri standard (normalizate) (de exemplu, ECE) sau a datelor proprii stocate în fișiere text. Scopul studiilor de simulare a fost de a analiza consumul de energie în timpul conducerii unei mașini în funcție de datele asumate și de a alege stilul optim de conducere utilizând sistemul expert integrat cu arbori de decizie multi-valorici. Datele de intrare pentru simulare au fost obținute dintr-o călătorie cu mașina în trafic mixt (presupus: 1200 s, viteză medie: 45 km/h, distanță: 20 km).

În studiul prezentat, pentru fiecare combinație de valori care a ajutat la determinarea unor grafice de consum a fost construit un arbore decizional multi-valoric care descrie sistemul decizional din cadru sistemului expert. Arborii decizionali multi-valorici au fost construiți pe baza unor tabele de valori a celor

trei factori menționați. Un exemplu de arbore decizional optimal este prezentat în figura A7.3. În exemplu se observă stratul de bază a arborelui multivaloric conține parametrii X1, X3, X5, X4 și X2, unde X1 este ciclul de conducere și poate lua valori de 0 și 1, X2 este viteza medie, X3 este eficiența, X4 este temperatura ambientală, iar X5 este energia consumată pe km, și pot lua valori de 0, 1 și 2.

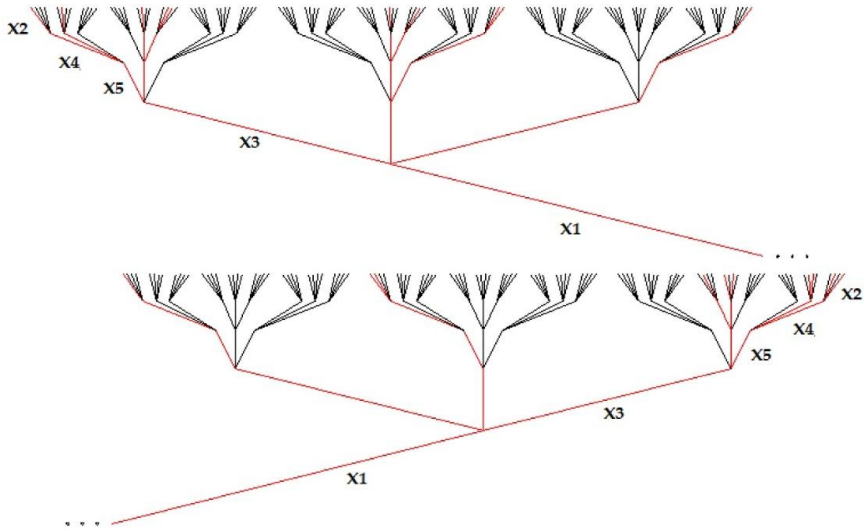


Fig. A7.3. Arbore decizional multi-valent [A7]

Rezultatele simulărilor au fost grafice de consum și viteză a vehiculului electric. Un exemplu de grafic este ilustrat în figura A7.4.

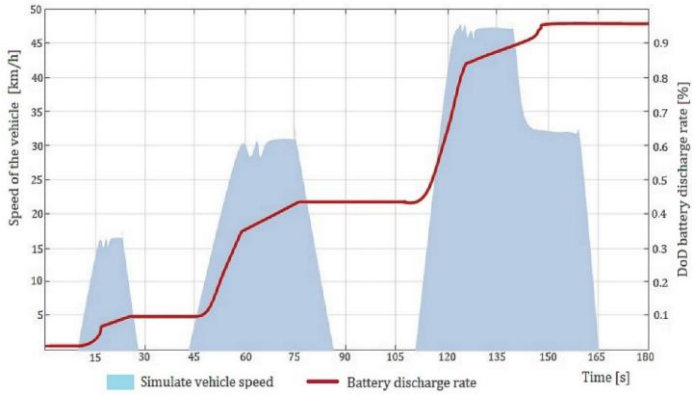


Fig. A7.4. Graficul ratei de descărcare a bateriei și al vitezei unei mașini electrice pentru un arbore decizional multivalent cu dispunerea la etaj a variabilelor de decizie: X1, X2, X3, X4, X5 [A7]

Dezavantajele arborilor logici multi-valorici includ necesitatea de a efectua discretizarea datelor. În acest fel, apar probleme în descrierea formală a caracteristicilor variabilelor stării sistemului folosind funcții continue și cvasi-continue în descrierea analizei mișcării mașinilor electrice în ciclurile de conducere selectate.

În concluzie, sistemul expert se caracterizează prin următoarele:

- Acesta este un sistem decizional, întrucât susține luarea deciziilor privind consumul eficiente de energie a unei mașini electrice.
- Reprezentarea cunoștințelor sub formă de arbori decizionali multi-valorici.
- Luarea deciziei în motorul de inferență este influențată de structura arborelui decizional.