Sistem expert pentru găsirea celui mai adecvat loc de muncă

1. Sisteme expert. Introducere.

Sistemul expert este un program inteligent care folosește cunoștințe și proceduri de inferenta pentru a rezolva probleme destul de dificile astfel încât să necesite expertiză semnificativă – Mitchell Feigenbaum.

Prin crearea unui sistem expert se încearcă replicarea părții logice a gândirii umane, așa cum un om se folosește de amintiri făcând legături pentru a ajunge la soluția unei probleme, un astfel de program folosește mecanisme de inferenta, o baza de cunoștințe și un input al utilizatorului având scopul de a da un răspuns cât mai corect problemei.

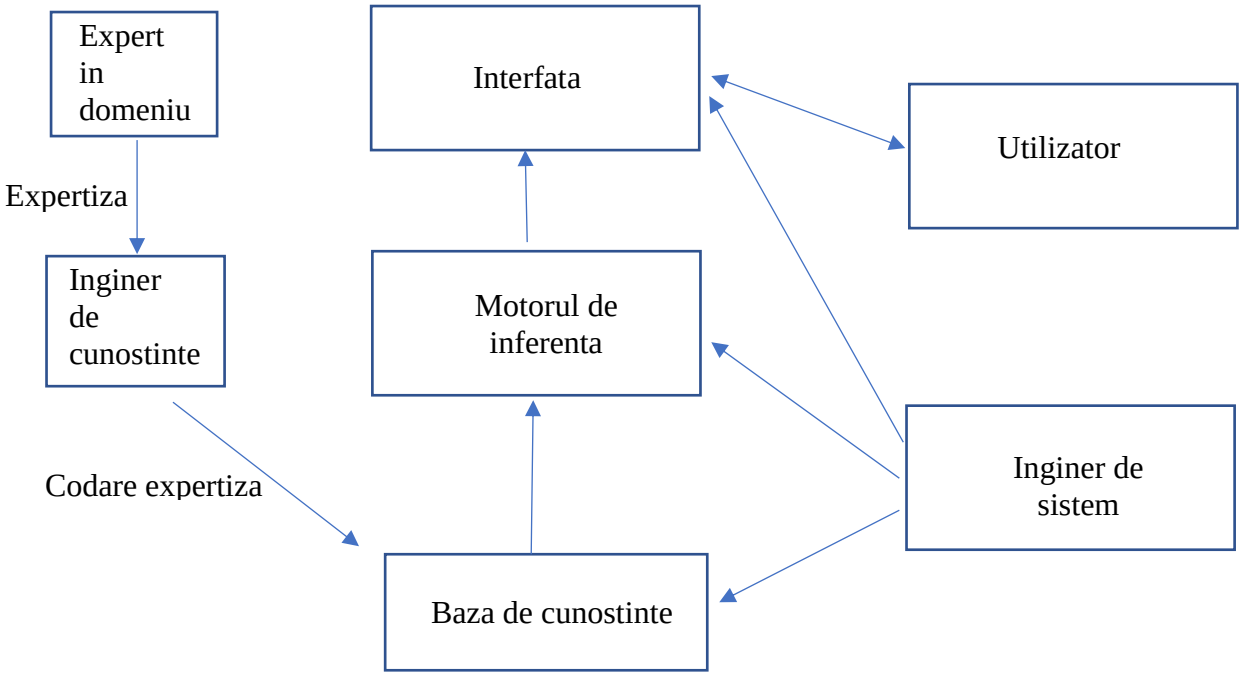
De exemplu, într-un spital un doctor verifică starea unui pacient. Prin conversație se stabilesc arii generale ale problemei, următoarele întrebări adresate vor avea legătură cu ce se știe deja ajungându-se într-un final la o problema specifică. Din răspunsurile aflate doctorul va diagnostica pacientul și va stabilii tratamentul. Un sistem expert poate fi creat pentru a afla diagnosticul și tratamentul bazat pe răspunsurile pacientului. Folosind un algoritm nu este nevoie de oprire sau pauze, nu este afectat de oboseală și nici nu este necesară învățarea într-un timp îndelungat a sistemului. Astfel un sistem expert este o soluție mai eficientă în găsirea soluțiilor pentru care este necesară o baza de cunoștințe.

O problema a sistemului este incertitudinea, este des intalnita in cazul sistemelor expert din domeniul medical atunci cand sistemul nu poate face legatura intre simptome si diagnostic, regulile sistemului putand fi vagi si utilizatorul neffiind sigur de raspunsul la intrebare. Intrebarile adresate utilizatorului trebuie sa fie simple pentru a nu crea confuzie utilizatorului, dar in acelasi timp sa se determine in urma raspunsului un atribut cheie in aflarea solutiei.

Asa cum se descrie in [2], principalele componente ale unui sistem expert sunt:

- Baza de cunostinte: o reprezentare declarativa a expertizei, de obicei sunt folosite reguli daca-atunci
- Motorul de inferenta: codul din nucleul sistemului, cod care deriveaza recomandari din baza de cunostinte
- Interfata cu utilizatorul: codul care controleaza dialogul dintre utilizator si sistem

Din aceste trei componente se disting trei roluri de utilizatori, prezentate in [2], care interactioneaza cu sitemul: expertul in domeniu, inginerul de cunostinte, inginerul de sistem si utilizatorul care foloseste sistemul expert fara a avea neaparat cunostinte despre cum functioneaza in spatele interfetei. In figura 1.1 se arata interactiunea dintre utilizatori si interactiunea dintre utilizatori si componente ale sistemului.



Reprezentarea cunoștințelor în sistemele expert.

In [1] se discuta despre urmatoare tipuri de cunostinte:

- cunostinte relationare simple
- cunostinte mostenite
- cunostinte inferentiale
- cunostinte procedural

Primul tip, cunostintele relationare simple, pot fi reprezentate ca o tabela in care fiecare coloana contine valorile care apartin unui aribut, valorile unui rand din aceasta tabela se pot caracteriza ca un set. Elementele acestui set sunt independente de orice alt set. Un exemplu de astfel de cunostinte din acest system expert este fisierul “descriere.txt” unde datele sunt reprezentate ca in urmatorul tabel:

Nume	Descriere	Limbaje
frontend_developer	‘Front-end web developm…’	[html, css, javascript,…]
backend_developer	‘A back-end developer…’	[html, css, javascript,…]
support_engineer	‘The Network Security…’	['c++',java]
software_tester	‘As a software tester, you…’	['sisteme de operare']

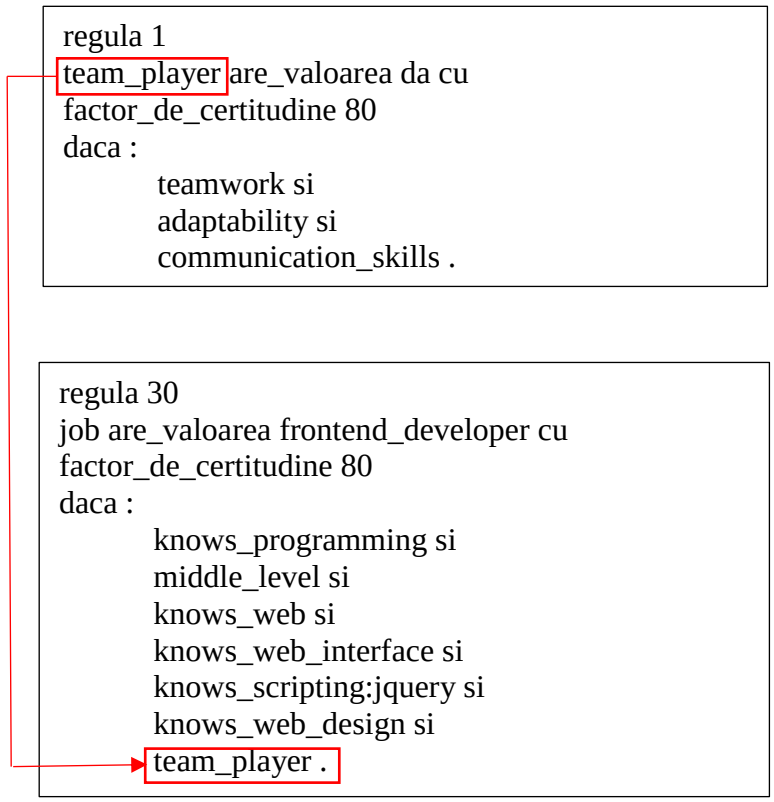
Informatiile din acest fisier (coloana limbaje) sunt citite din fisier si folosite ca input pentru mecanismul de inferenta. Din coloana descriere datele sunt folosite direct pentru a afisa o descriere a locului de munca, nu au un scop in deducerea altor informatii. Coloana “Nume” este folosita pentru a stii carui loc de munca apartin valorile din celelalte coloane, comportandu-se ca o cheie primara, dar fara a fi verificata unicitatea ei in fisier, datele fiind considerate corecte.

Pentru a elimina duplicatele se foloseste, in bazele de date care folosesc acest tip de reprezentare, o cheie primara obtinuta prin combinarea unor valori ale coloanelor sau creand una artificiala, de exemplu numerotarea randurilor.

Cunostintele mostenite. Daca exista o structura loc_de_munca si caracteristicile ei {nume, descriere, limbaje, experienta etc}, atunci creand obiectul “web developer” si spunand ca “web developer” is_a loc_de_munca atunci obiectul va avea proprietatile structurii loc_de_munca la care se pot adauga altele depinzand de nevoi. In sistemele complexe se foloseste mostenirea pentru a reduce (chiar elimina) structurile duplicat si pentru usura modificarea structurilor, modificarea facandu-se doar in structura din care se mosteneste. In acest sistem expert nu se foloseste mostenirea.

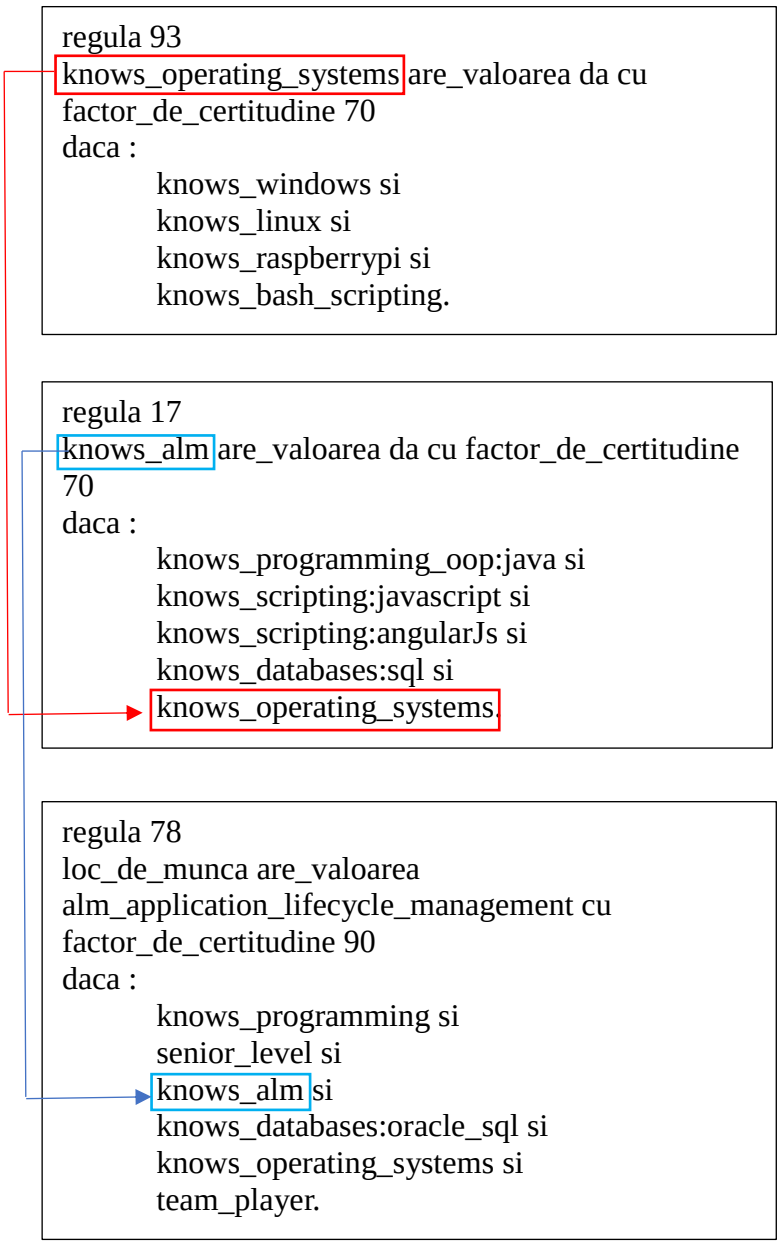
Daca toate datele necesare ar fi date ca input utilizatorul ar trebui sa cunoasca domeniul de expertiza foarte bine, asta ar duce chiar la incetarea folosirii sistemului, utilizatorul putand sa gaseasca solutia fara ajutor. A doua problema ar fi numarul mare de intrebari adresate utlizatorului, acest numar depinde de complexitatea domeniului de expertiza si rezolvarea problemei. Solutia acestor probleme este un mecanism de inferenta care sa deduca cat mai multe attribute (cu o probabilitate de adevar cat mai mare) din raspunsurile utilizatorului.

Un exeplu de atribut dedus din acest sistem expert:



In acest exemplu se deduce team_player ca adevarat cu, probabilitate de 80%, din atributele teawork, adaptability si communication_skills cu valori date de utilizator.

In urmatorul exemplu se arata un atribut care se deduce din atribute deduse si attribute cunoscute



Atributul knows_operating_systems este dedus din cele patru atribute (obtinute de la utilizator) din regula sa. knows_operating_systems este folosit pentru a deduce atributul knows_alm, iar knows_alm este folosit pentru a deduce loc_de_munca cu valoarea alm_application_lifecycle_management

Exista doua tipuri de inferenta:

- inlantuire inapoi (backward chaining)
- inlantuire inainte (forward chaining)

Folosind inlantuirea inapoi se pleaca de la concluzie, apoi se cauta fapte care sa sutina concluzia, atunci cand se gasesc toate faptele necesare, concluzia devine solutia. Prolog, asa cum se spune si in [2], foloseste inlantuirea inapoi, acest sistem expert foloseste de asemenea inlantuirea inapoi.

In cazul inlantuirii inainte se folosesc toate faptele cunoscute si se cauta concluzia care le sustin (nu neaparat toate).

2. Teoretic (Analiză limbajului natural)

Bibliografie:

1. Balcan F. M., Hristea F. (2005) Căutarea și reprezentarea cunoștințelor în inteligență artificială. Teorie și aplicații, București: Editura Universității din București

2. Merritt D. (2000) Building Expert Systems în Prolog, U.S.A.: Amzi! Inc.

3.