

Universitatea Politehnică Timișoara
Facultatea de Automatică și Calculatoare

Sisteme expert

ChooseYourCompanyBot

Student:Iacov Cristian

Cuprins

INTRODUCERE.....	2
1.1 Istoric Sisteme Expert	2
1.2 Caracteristicile Sistemelor Expert.....	3
- Nivelul de experiență.....	3
- Reacția la timp.....	4
- Fiabilitate	4
- Mecanism eficient.....	4
- Manipulați problemele	4
- Componente	4
Motor de inferență	4
Concluzii.....	4
Dezvoltarea folosind Node.js	5
Proiectare si implementare	8
3.1 Prezentarea aplicației.....	8
Bibliografie	11

INTRODUCERE

1.1 Istoric Sisteme Expert

Conform uneia dintre cele mai răspândite definiții ale sistemului expert (SE), acesta este un program de calcul care încorporează cunoștințele unui expert uman și încearcă să simuleze raționamentele desfășurate de acesta din urmă în scopul rezolvării unei anumite probleme din domeniul său de expertiză. La rândul său, expertul este un specialist care stăpânește foarte bine un anumit domeniu. Atributele care deosebesc expertul de un începător au în mare parte un caracter simbolic, se sprijină pe cunoștințe dobândite în timp și decurg din raționamente care încearcă să pună de acord aceste cunoștințe cu faptele specifice problemei studiate.

Până în prezent, domeniile de interes în care s-au înregistrat cele mai numeroase implementări ale SE sunt : inginerie și producția de mărfuri (35%) ; afaceri (28%) ; medicină (11%) ; mediu și energie (9%) ; agricultură (5%) ; telecomunicații (4%) ; administrație (4%) ; legislație (3%) și transporturi (1%) [Liebowitz 97].

În domeniul electroenergeticii, principalele aplicații ale SE se referă la : (a) conducerea operativă a sistemului energetic; (b) analiza on / off – line a funcționării sistemului electroenergetic; (c) analiza și reconfigurarea postavarie a rețelelor electrice; (d) mentenanța echipamentelor din sistemul electroenergetic; (e) dezvoltarea rețelelor de distribuție ș.a.

Elementul central, în jurul căruia gravitează toate celelalte componente ale unui SE, îl reprezintă cunoștințele. De aceea, SE se mai numesc și Sisteme Bazate pe Cunoștințe. În cadrul unui SE, reprezentarea cunoștințelor se poate face pe mai multe căi, cum ar fi regulile de producție, cadrele și cazurile. Dintre aceste forme de reprezentare, cea mai răspândită este cea a regulilor de producție.

Cadrele (în engleză, frames) reprezintă obiecte complexe care sunt descrise de anumite proprietăți sau atribute și anumite proceduri sau metode. Structural, cadrele sunt foarte asemănătoare cu obiectele folosite în cadrul programării la nivel obiect. Astfel, una din proprietățile cele mai importante ale cadrelor este mecanismul de moștenire, care permite descrierea generică a unui obiect și crearea de instanțe ale acestuia care moștenesc toate atributele obiectului generic, la care se adaugă și atribute noi.

Reprezentarea cunoștințelor sub formă de cazuri are la bază premisa că, pentru a învăța și a rezolva probleme complexe, oamenii folosesc raționamentul analogic sau cel experimental. Cazurile folosite pentru desfășurarea analogiilor constau în informații despre situația analizată, soluția problemei în sine, rezultatele care se obțin prin adoptarea acelei soluții, anumite atribute care pot evidenția tipare specifice.(sursa[1])

1.2 Caracteristicile Sistemelor Expert

- Nivelul de experiență

Un sistem expert trebuie să ofere cel mai înalt nivel de expertiză. Oferă eficiență, precizie și rezolvare imaginativă a problemelor.

- Reacția la timp

Utilizatorul interacționează cu sistemul expert pentru o perioadă de timp destul de rezonabilă. Timpul acestei interacțiuni trebuie să fie mai mic decât timpul necesar unui expert pentru a obține cea mai precisă soluție pentru aceeași problemă.

- Fiabilitate

Sistemul expert trebuie să aibă o bună fiabilitate. Pentru a face acest lucru, nu trebuie să faceți nicio greșeală.

- Mecanism eficient

Sistemul expert trebuie să aibă un mecanism eficient pentru a gestiona compendiul de cunoștințe existent în el.

- Manipulați problemele

Un sistem expert trebuie să fie capabil să facă față problemelor provocatoare și să ia deciziile corecte pentru a oferi soluții.

- Componente

Bază de cunoștințe

Este o colecție organizată de date corespunzătoare sferei de experiență a sistemului.

Prin interviuri și observații cu experți umani, trebuie luate faptele care alcătuiesc baza de cunoștințe.

Motor de inferență

Interpretează și evaluează faptele din baza de cunoștințe prin reguli, pentru a oferi o recomandare sau o concluzie.

Această cunoaștere este reprezentată sub forma unor reguli de producție If-Then: „Dacă o condiție este adevărată, atunci se poate face următoarea deducere.”

Concluzii

Un factor de probabilitate este adesea atașat la concluzia fiecărei reguli de producție și la recomandarea finală, deoarece concluzia la care se ajunge nu este o certitudine absolută.

De exemplu, un sistem expert pentru diagnosticul bolilor oculare ar putea indica, pe baza informațiilor furnizate, că o persoană are glaucom cu o probabilitate de 90%.

De asemenea, se poate arăta succesiunea regulilor prin care s-a ajuns la concluzia. Urmărirea acestui lanț ajută la evaluarea credibilității recomandării și este utilă ca instrument de învățare.(sursa[2])

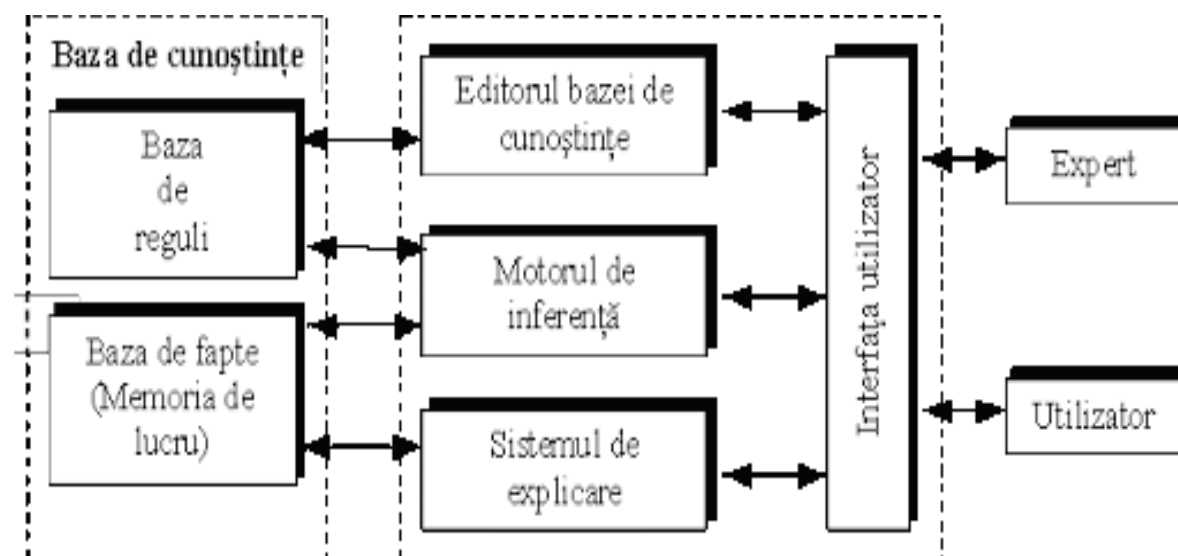


Figura 1.1 Structura unui sistem expert (sursa[3]).

Dezvoltarea folosind Node.js

Dahl created this language because he wasn't happy the way Apache Http server handles multiple concurrent connections. In fact, he criticized the way code was blocking the entire process or implying multiple execution stacks for simultaneous connections.

His urge compelled him to create a Node.js project that he demonstrated at the inaugural European JSConf on 8th November 2009. The smart utilization of Google's V8 JavaScript engine, event loop and low-level I/O API in the project managed to win millions of hearts.

In June 2011, Microsoft and Joyent together implemented a native Windows version of Node.js. The first version was released in July 2011.

In January 2012, Dahl stepped aside & promoted the co-worker and npm creator “Isaac Schlueter” to manage the project.

How Does it Work?

JavaScript and Node.js run on V8 JavaScript runtime engine. This V8 engine converts the JavaScript code into faster machine code.

Nodejs Architecture Includes:

Single Threaded: Node.js works on a single thread. This approach is worth considering as it avoids context switching.

Event Loop: The event loop allows node.js to perform non-blocking I/O operations despite the single-threaded feature. A transaction passing through Node traverses a cascade of callbacks. And using the ‘libuv’ library, it handles queuing & process the asynchronous events.

Non-Blocking I/O: This loop works on a single thread, but the runtime tasks are executed asynchronously on worker’s thread that returns the result via call back to the event loop thread. This is a great non-blocking way of handling code execution.

Front-End Development: This covers coding the UI interface, handling events and thereby interacting with back-end server to pull data at front end UI side. The front end is what is visible to the user, and he engages and performs activities.

Back-End Development: The back-end is the technical work that is not visible to the user. It deals with server-side scripting and database interaction so that the data can be requested by front end apps.

Nodejs Features:

Encourages Sharing: Robust Node Package Manager (NPM), encourages sharing. Having a repository of 50,000 packages, developers manage to build effective solutions. With inbuilt NPM, it becomes easy for developers to update, share or reuse codes.

Lightweight and Scalable: Nodejs development accelerates with the V8 JavaScript engine without compromising on quality or security terms. Additionally, the non-blockage of the thread makes the application lightweight, scalable behaving in a network-friendly manner.

Ideal for Real-Time Applications: The language has extraordinary features for creating real-time applications like chats and gaming apps. It is also an ideal fit for the programs that require an event-based server or non-blocking server.

Data Streaming: HTTP requests responses are considered as isolated events, but in reality, they are streaming data. You can leverage data streaming in Nodejs by incorporating features like processing files when uploaded. It drastically decreases the complete processing time.

High-Performance: The single threaded JavaScript runtime offers seamless network applications. Built upon Google Chrome's V8 runtime and coded in C++, Nodejs is specifically built for multiple operating systems. Both V8 and Node are updated on regular intervals with performance optimization & security patches while supporting JavaScript features.

Suitable for Microservices: All big names are utilizing Node.js for its microservices. Wondering why? Well, Node.js has the ability to optimize the performance of your application. Furthermore, eliminating the unwieldy modules make it simpler to operate microservices on Node.js applications.

Cross-Platform Development: Nodejs collaboration with Electron or NW.js allows you to build cross-platform applications.

Powerful Single Codebase: Nodejs has proved itself as a game-changing technology as it allows Node.js developers to code JavaScript server side and client side. This makes it easy to send & synchronize data between the points automatically, saving a lot of development time.(sursa[4])

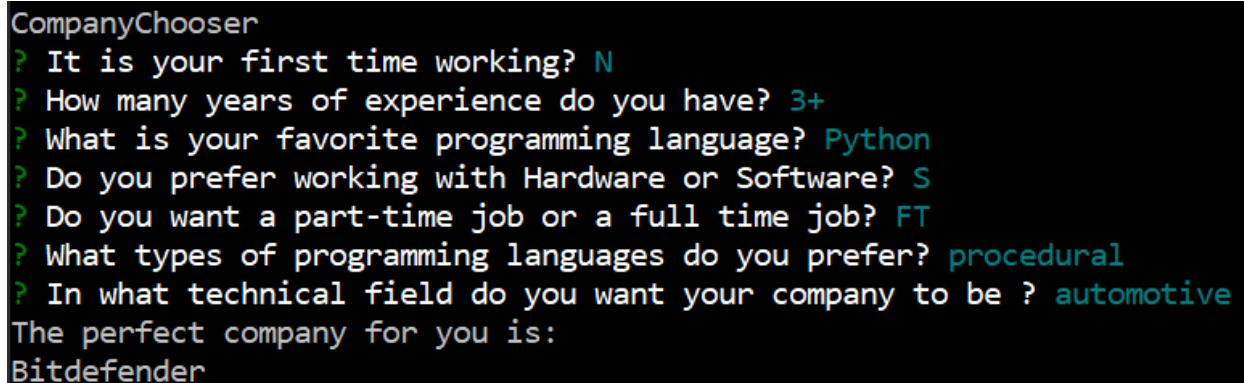
Proiectare si implementare

3.1 Prezentarea aplicatiei

Aplicatia ChooseYourCompanyBot foloseste limbajul javascript pentru a da utilizatorilor sugestii despre ce companii li s-ar potrivi acestora pe baza unor intrebari.

ChooseYourCompanyBot este structurat in 2 zone: masina de inferenta si baza de reguli. Motorul de inferenta strange raspunsurile utilizatorului, iar apoi apeleaza o functie de evaluare din baza de cunostiinte care ofera un raspuns pe baza stucturii de date primite ca parametru.

Aplicatia este dezvoltata folosind framework-ul Node.js si nu dispune de o interfata grafica bazata pe html, ci una bazata pe command prompt.



```
CompanyChooser
? It is your first time working? N
? How many years of experience do you have? 3+
? What is your favorite programming language? Python
? Do you prefer working with Hardware or Software? S
? Do you want a part-time job or a full time job? FT
? What types of programming languages do you prefer? procedural
? In what technical field do you want your company to be ? automotive
The perfect company for you is:
Bitdefender
```

Figura 3.1 Interfata grafica a aplicatiei

Baza de cunostiinte foloseste pachetul rools din npm din motive de simplitate si a oferi o interfata cat mai frumoasa si usor de inteles. Scopul principal a fost de a oferi o interfață frumoasă și de ultimă generație pentru JavaScript modern (ES6). Faptele sunt obiecte simple JavaScript sau JSON sau obiecte din clasele ES6 cu getters și setere. Regulile sunt specificate în JavaScript pur și nu într-un limbaj special, cu destinație specială, cum ar fi DSL.

Scopul secundar a fost de a oferi eficiență și optimizare asemănătoare RETE.


```

const { Rools, Rule } = require('rools');

const rules = [];

rules[0] = new Rule({
  name: "Continental",
  when: [
    facts => facts.firstTimeWorking === "Y" || facts.firstTimeWorking === "N",
    facts => facts.yearsOfExperience === "<1" || facts.yearsOfExperience === "1-3" || facts.yearsOfExperience === "3+",
    facts => facts.favoriteLanguages === "C" || facts.favoriteLanguages === "Java",
    facts => facts.hardwareOrSoftware === "H" || facts.hardwareOrSoftware === "S",
    facts => facts.parttimeOrfulltime === "FT" || facts.parttimeOrfulltime === "PT",
    facts => facts.programmingType === "procedural" || facts.programmingType === "oop",
    facts => facts.technicalField === "automotive" || facts.programmingType === "embedded",
  ],
  then: facts => {
    facts.companyName = "Continental";
  },
});

rules[1] = new Rule({
  name: "Hella",
  when: [
    facts => facts.firstTimeWorking === "Y" || facts.firstTimeWorking === "N",
    facts => facts.yearsOfExperience === "<1" || facts.yearsOfExperience === "1-3" || facts.yearsOfExperience === "3+",
    facts => facts.favoriteLanguages === "C" || facts.favoriteLanguages === "Java",
    facts => facts.hardwareOrSoftware === "H" || facts.hardwareOrSoftware === "S",
    facts => facts.parttimeOrfulltime === "FT" || facts.parttimeOrfulltime === "PT",
    facts => facts.programmingType === "procedural" || facts.programmingType === "oop",
    facts => facts.technicalField === "automotive",
  ],
  then: facts => {
    facts.companyName = "Hella";
  },
});

rules[2] = new Rule({
  name: "FEV",
  when: [
    facts => facts.firstTimeWorking === "Y" || facts.firstTimeWorking === "N",
    facts => facts.yearsOfExperience === "<1" || facts.yearsOfExperience === "1-3" || facts.yearsOfExperience === "3+",
    facts => facts.favoriteLanguages === "C",
    facts => facts.hardwareOrSoftware === "H" || facts.hardwareOrSoftware === "S",
    facts => facts.parttimeOrfulltime === "FT",
    facts => facts.programmingType === "procedural",
    facts => facts.technicalField === "automotive",
  ],
  then: facts => {
    facts.companyName = "FEV";
  },
});

```

Cod 3.1 Structura Knowledge base

In captura 3.1 putem observa structura oferita de pachetul rools.Regulile se scriu intr-o maniera when,then iar la final,se va returna o structura JSON.

```

});
async function diagnose(facts) {
  const rools = new Rools();
  await rools.register(rules);
  await rools.evaluate(facts);
  return facts;
};

module.exports = { diagnose };

```

Cod 3.2 Functia de evaluare

Prin functia de evaluare diagnose, toate regulile de mai sus vor fi înregistrate și evaluate, iar parametrul facts va conține JSON-ul returnat de functia evaluate cu numele companiei rezultat.

Masina de inferenta foloseste pachetul Inquirer.js care este o interfață de linie de comandă ușor încorporabilă și frumoasă pentru Node.js

```

const express = require('express');
const es = require('./es');
const app = express();
const inquirer = require('inquirer');

console.log('CompanyChooser');

var questions = [
  {
    type: 'list',
    name: 'firstTimeWorking',
    message: 'It is your first time working?',
    choices: ['Y', 'N'],
  },
  {
    type: 'list',
    name: 'yearsOfExperience',
    message: 'How many years of experience do you have?',
    choices: ['<1', '1-3', '3+'],
  },
];

```

Cod 3.3 Structura masina de inferenta

Masina de inferenta contine un array de obiecte ca in secventa de cod 3.3 care contine toate intrebarile pentru quiz.

Dupa ce utilizatorul raspunde tuturor intrebarilor, se apeleaza functia `prompt` care returneaza un JSON si apelam functia `diagnose` din knowledge base cu JSON-ul returnat, iar apoi afisam rezultatul.

```
inquirer.prompt(questions).then((answers) => {  
  console.log('The perfect company for you is: ')  
  es.diagnose(answers)  
  .then(result => console.log(result.companyName));  
});
```

Cod 3.4 Afisare rezultat

Bibliografie

- [1] <http://iota.ee.tuiasi.ro/~mgavril/Simpe/L2.htm>
- [2] <https://ro.warbletoncouncil.org/sistemas-expertos-11181#menu-7>
- [3] <http://iota.ee.tuiasi.ro/~mgavril/Simpe/L2.htm>
- [4] <https://www.yourteaminindia.com/blog/nodejs-development/>