



Universitatea
Transilvania
din Brașov

FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

PROIECT SISTEME MULTIAGENT

Conducător științific:

Conf. Dr. Ing. Dan ROSENBERG

Studenti:

Diana DUMITRU

Bogdan PATRANIA

Marina RADU

Gabriel TOMUȚA

BRAȘOV, 2024

1 TEMA PROIECTULUI

1.1 TEMA PROPUȘĂ

Tema de proiect A: (9)

Realizați o aplicație JADE care să creeze 5 agenți care să primească mesaje de la o interfață grafică. Exemplificați folosind agenți BDI.

1.2 TEMA ALEASĂ

Restaurant Inteligent

Agenți:

- ManagerAgent: Gestionează operațiunile generale ale restaurantului și se ocupă de solicitările utilizatorilor;
- ChefAgent: Agent cu aptitudinea de bucătar. Acesta se ocupă de prepararea alimentelor;
- WaiterAgent: Agent cu aptitudine de chelner: livrează mâncarea la masă, transportă comanda la bucătărie și curăță mesele;
- TableAgent: Gestionează înregistrările privind mesele din restaurant;
- CheckoutAgent: Calculează profitul restaurantului pentru ziua respectivă.

2 IMPLEMENTAREA SOLUȚIEI

Proiectul nostru utilizează JADE (Java Agent Development Framework) pentru a simula managementul unui restaurant printr-o arhitectură multi-agent. Scopul acestui proiect este de a demonstra cum agenții software pot interacționa pentru a îndeplini sarcini complexe într-un mediu dinamic. Sistemul nostru folosește agenți de tip BDI (Belief-Desire-Intention), un model care permite agenților să ia decizii bazate pe convingeri, dorințe și intenții, ceea ce îi face capabili să gestioneze situații complexe și imprevizibile.

Modelul BDI este inspirat din teoria acțiunii practice și permite agenților să funcționeze într-un mod similar cu cel al oamenilor. Agenții noștri BDI sunt programați să perceapă mediul înconjurător, să actualizeze starea internă a convingerilor lor (beliefs), să genereze dorințe (desires) pe baza obiectivelor de atins și să formuleze intenții (intentions) care ghidează acțiunile lor pentru a îndeplini aceste obiective. Proiectul simulează un mediu realist de operare a unui restaurant. Acest mediu este guvernat de cei cinci agenți, responsabili de diferite acțiuni create în baza unor scenarii definite apriori.

Agentul Manager are rolul principal de a interacționa direct cu clienții și de a coordona activitatea celorlalți agenți specializați. Acesta primește cererile de la clienți, le interpretează pentru a înțelege tipologia lor și le direcționează către agenții relevanți, cum ar fi Agentul Table pentru cereri de rezervare a mesei, Agentul Waiter pentru comenzi de mâncare sau Agentul Checkout pentru calcularea profitului. Agentul Manager asigură o colaborare eficientă între agenți și informează clienții despre starea cererilor lor.

Agentul Table gestionează informațiile despre statusul meselor în restaurant, inclusiv dacă sunt ocupate sau libere și starea clienților care se află la aceste mese. Acesta cunoaște, de asemenea, detaliile comenzilor de mâncare și prețurile acestora asociate fiecărei mese. Comunicarea agentului Table este esențială pentru colaborarea cu alți agenți, precum agentul Waiter pentru preluarea comenzilor, agentul Checkout pentru procesul de plată și agentul Manager pentru rezervări și igienizare.

Agentul Waiter acționează ca un intermediar între client și agentul Chef, preluând comenzile și transportându-le către bucătar. După ce preparatele sunt gata, Agentul Waiter le servește clienților și apoi curăță masa.

Agentul Chef are rolul simplu de a pregăti preparatele culinare, fără alte sarcini sau obiective adiționale.

Agentul Checkout îndeplinește funcția de casier în sistemul restaurantului. El stochează costurile interne ale restaurantului pentru diferitele tipuri de mâncare și prețurile plătite de clienți. La închiderea restaurantului, acesta calculează și returnează profitul obținut.

Fluxul de interacțiuni dintre acești agenți poate fi descris prin intermediul unor scenarii. Spre exemplu în cazul în care utilizatorul dorește să rezerve o masă, acesta trimite o solicitare către agentul Manager. Agentul Manager verifică disponibilitatea meselor prin schimbul de

informații cu Agentul Table. Dacă există mese disponibile, se alocă una dintre acestea clientului și confirmă rezervarea. În caz contrar, informează clientul că toate mesele sunt ocupate.

În cazul procesului de comandare a mâncării, clientul își exprimă dorința de a comanda un anumit preparat prin intermediul agentului Manager. Agentul Table înregistrează detaliile comenzii și actualizează suma totală de plată. Clientul își poate modifica comanda pe parcurs, informație actualizată în Agentul Table. Odată ce clientul finalizează plasarea comenzii, aceasta ajunge la bucătar pentru a putea fi preparată. După ce mâncarea este pregătită, agentul Waiter o livrează la masa clientului.

Când clientul dorește să achite nota de plată, acesta trimite o solicitare către agentul Manager. Acesta o redirecționează agenților capabili de a se ocupa de achitare. După ce aceasta a fost îndeplinită Agentul Checkout înregistrează profitul, iar agentul Waiter debarasează masa.

La finalul zilei, dacă sistemul consideră ca restaurantul este gol, afișează profitul total, reținut în Agentul Checkout.

Scenariile enumerate reprezintă bază de la care attributele BDI ale fiecărui agent au fost create. Întreaga funcționalitate a sistemului a fost construită de-a lungul unei logici de program derivată din scenariile prezentate. Utilizarea acestui sistem pentru managementul restaurantului oferă numeroase avantaje, printre care flexibilitatea în gestionarea resurselor și scalabilitatea sistemului. Fiecare agent poate funcționa independent, dar poate colabora eficient cu ceilalți agenți, ceea ce permite sistemului să se adapteze rapid la schimbări sau probleme neașteptate. În plus, distribuirea sarcinilor între agenții specializați reduce timpul de răspuns și crește eficiența operațională. Acest proiect evidențiază potențialul tehnologiei multi-agent și a modelului BDI în diverse domenii, contribuind la dezvoltarea soluțiilor inovative în gestionarea resurselor și proceselor

Contribuția mea la acest proiect a constat în implemetarea Agentului Chef. În cazul acestui agent elementele BDI prezente sunt:

- Convingeri: lista de feluri de mâncare pe care este capabil să le prepare;
- Cerințe: finalizarea comenzii;
- Intenții: prepararea felurilor de mâncare.

3 PROGRAMUL

```
import java.util.*;
import jade.core.behaviours.CyclicBehaviour;

import utils.RestaurantAgent;
import utils.Menu; // Belief

/**
 * The ChefAgent implement a BDI agent with the
 * purpose of cooking the food it gets.
 *
 * @author Dumitru, Radu.M, Patrania, Tomuta
 */
public class ChefAgent extends RestaurantAgent {
    // Desires
    LinkedList<String> desires = new LinkedList<>();

    @Override
    protected void setup() {
        init("Chef");

        addBehaviour(new CyclicBehaviour() {
            @Override
            public void action() {
                if (!desires.isEmpty()) {
                    String current_desire = desires.getFirst();
                    logInfo("Current desire: " + current_desire);

                    // Choose action
                    String aux;
                    if (current_desire.startsWith("cook")) {
                        // get value
                        aux = current_desire.split(":")[1];
                        if (cookFood(aux))
                            desires.removeFirst();
                    }
                    else if (current_desire.startsWith("finish_order")) {
                        // get value
                        aux = current_desire.split(":")[1];
                        if (finishOrder(Integer.parseInt(aux)))
                            desires.removeFirst();
                    }
                    else {
                        logError("Desire unknown: " + current_desire);
                        desires.removeFirst();
                    }
                }
                else {
                    block();
                }
            }
        });
    }
}
```

```

    }
    }
});
}

protected void interpretMessage(String msg) {
    /* Messages:
       - x-y;z : x - table num, y,z - food items
    */
    String[] split = msg.split("-");
    int table_num = Integer.parseInt(split[0]);
    for (String food_item : split[1].split(";")) {
        desires.add(String.format("cook:%s", food_item));
    }
    desires.addLast(String.format("finish_order:%d", table_num));
    this.doWake();
}

// Plan to cook
private boolean cookFood(String food_item) {
    logInfo("Executing plan: cook " + food_item);
    // Check if valid
    if (!Menu.checkItem(food_item)) {
        logError("Invalid food item: " + food_item);
    }
    else {
        // Simulate making food
        sleep(1000);
    }
    return true;
}

// Plan to finish the current order
private boolean finishOrder(int table_num) {
    logInfo("Executing plan: finish order for table " + table_num);
    sendMessage("Waiter", String.format("Order complete %d", table_num));
    return true;
}
}

```

BIBLIOGRAFIE

- [1] <https://download.actoron.com/docs/releases/jadex-3.0.0-RC51/jadex-mkdocs/BDI%20V3%20Tutorial/01%20Introduction/>
- [2] <https://chatgpt.com/>
- [3] Floroian, D. (2023). Sisteme multiagent. Alba Iulia: Editura Albastră.