

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI
BOUMEDIENE



TECHNOLOGIE DES AGENTS

MINI-Projet
Tech-Agent

MOKEDDEM BILLEL, Groupe 3
Email: billemokeddem.ml@gmail.com

HEBBACHE IMAD EDDINE, Groupe 1
Email: imadeddine964@gmail.com

Mr.Smaili

September 19, 2020

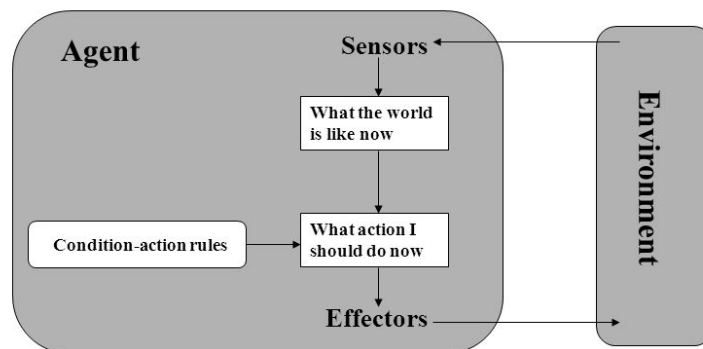
CHAPTER 1

INTRODUCTION À LA TECHNOLOGIE DES AGENTS

1.1 C'est quoi un Agent ?

Un agent est un système informatique situé dans un environnement qu'il peut percevoir et sur lequel il peut agir, il est aussi caractérisée par le fait qu'il est, au moins partiellement autonome.

A Simple Reflex Agent: Schema



1.2 C'est quoi un Système Multi-Agents ?

Un système multi-agent (SMA) est un système composé d'un ensemble d'agents situés dans un certain environnement et interagissant selon certaines relations.

1.3 Les Types Des Agents

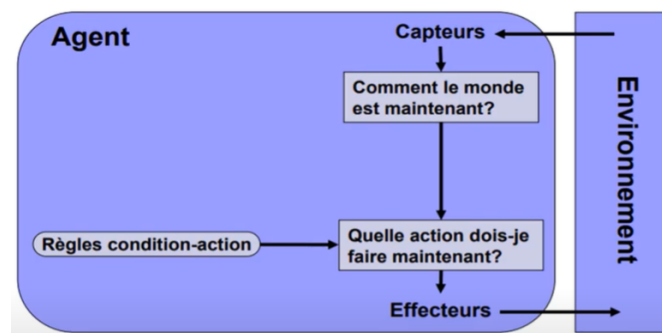
il existe cinq types d'agents:

- Agent réflexe simple (Simple Reflex Agent)
- Agent réflexe avec état interne (Model-Based Reflex Agent)
- Agent basé sur les buts (Goal-Based Agent)
- Agent basé sur l'utilité (Utility-Based Agent)
- Agent apprenant (Learning Agent)

Dans ce projet, nous utiliserons un agent simple réflexe

1.4 Agent réflexe simple (Simple Reflex Agent)

Un agent de type simple réflexe recherche dans sa base de règle une règle qui a pour condition qui vérifie la situation actuelle perçu de l'environnement et exécute l'action associées à cette règle à travers ces effecteur.



CHAPTER 2

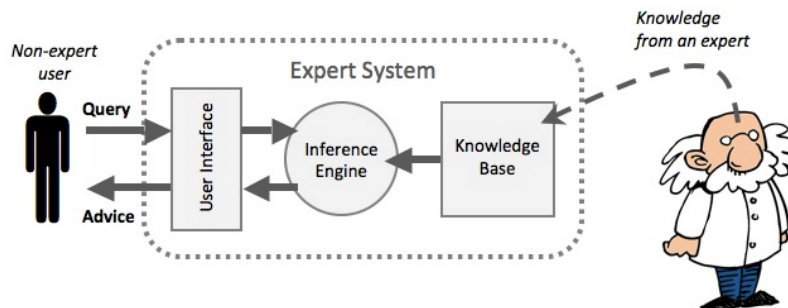
PARTIE 1: IMPLÉMENTATION D'UN SYSTÈME EXPERT

2.1 Définition

Un système expert est un système informatique qui émule la capacité de prise de décision d'un expert humain. Les systèmes experts sont conçus pour résoudre des problèmes complexes en raisonnant à travers des ensembles de connaissances, représentés principalement comme si - alors des règles plutôt que par un code de procédure conventionnel.

2.2 Les composants d'un système expert

Un système expert est composé principalement de deux composants principaux, la base des connaissances et le moteur d'inférence.



2.2.1 La Base des Connaissances (Knowledge Base)

Une base de connaissances est une technologie utilisée pour stocker des informations complexes structurées et non structurées utilisées par un système informatique. L'utilisation initiale du terme était en relation avec les systèmes experts qui étaient les premiers systèmes basés sur la connaissance.

2.2.2 Le Moteur d'inférence (Inference Engine)

Le moteur d'inférence est un composant du système qui applique des règles logiques à la base de connaissances pour en déduire de nouvelles informations. Les premiers moteurs d'inférence étaient des composants de systèmes experts.

2.3 Les types d'un Système Expert

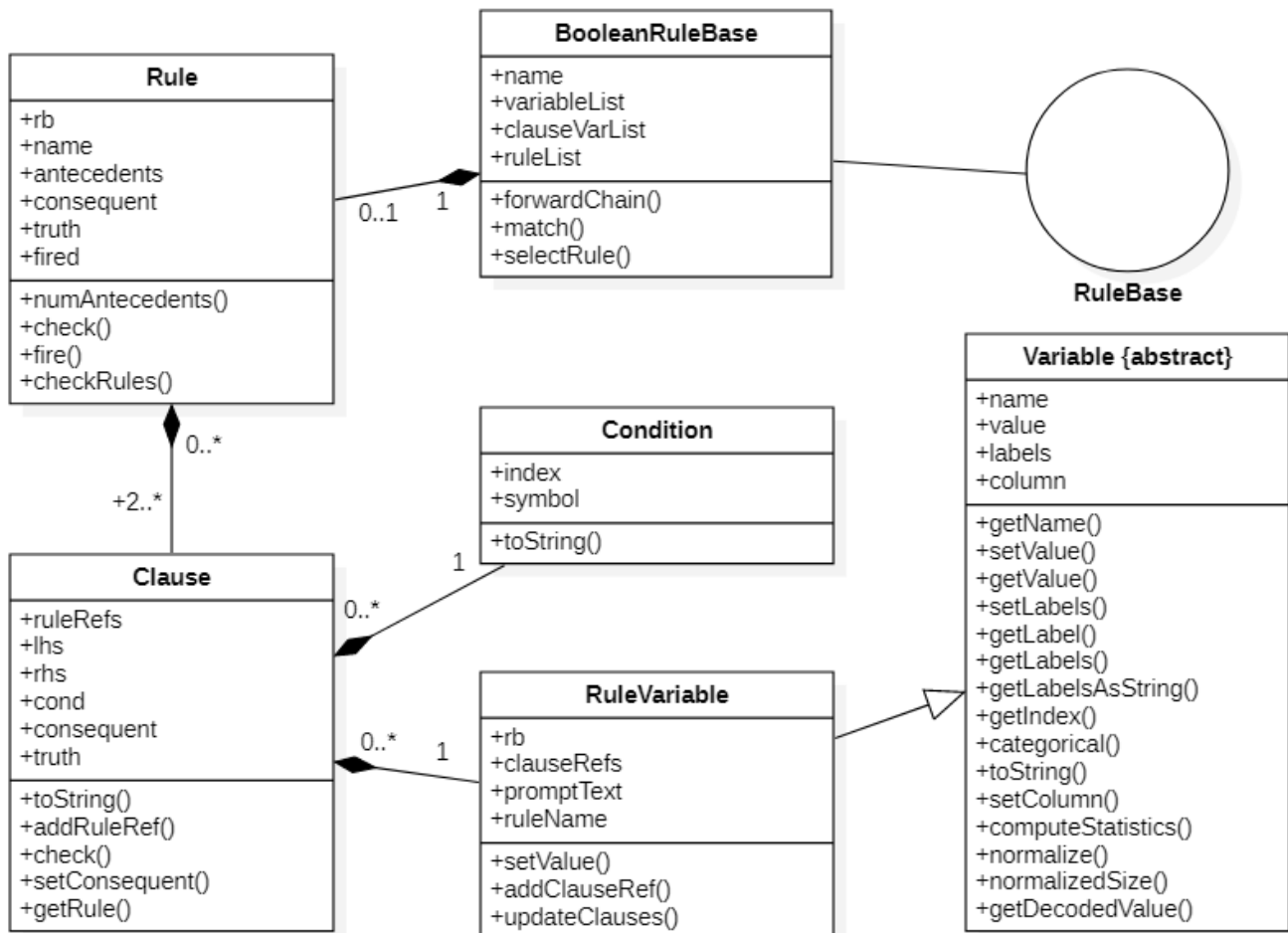
Il existe principalement cinq types de systèmes experts:

- un système expert basé sur des règles booléennes (boolean rule based expert system)
- un système expert basé sur une trame (frame based expert system)
- un système expert flou (fuzzy expert system)
- un système expert neuronal (neural expert system)
- un système expert neuro-flou (neuro-fuzzy expert system)

Dans cette implémentation, nous utiliserons un système expert basé sur des règles booléennes

2.4 La Conception D'un Système Expert

Cette Implémentation du système est composé du plusieurs classes qui forme la base de connaissances et le moteur d'inférence. ci dessous est le diagramme de classe du système expert :



2.4.1 Les classes du système expert

RuleBase et Variable

l'interface RuleBase et la classe abstraite Variable nous permettent d'utiliser cette conception pour construire d'autres types de systèmes experts ou d'autres plus compliqués.

BooleanRuleBase

Cette classe implémente l'interface RuleBase, et elle encapsule tous les attributs et méthodes nécessaires au système expert, elle contient des méthodes comme forwardChain et match qui sont le cœur du moteur d'inférence

Rule

Comme son nom l'indique, cette classe est utilisée pour représenter une règle, elle utilise des instances de la classe de clause pour représenter les clauses de la règle, elle contient également des méthodes importantes pour le moteur d'inférence comme fire pour utiliser une règle et check pour vérifier la validité d'une règle.

Clause

Cette classe représente simplement une Clause, il existe deux types d'objets instanciés à partir de cette classe, cela peut être une clause antécédent ou une clause conséquente, elle utilise des instances de la classe RuleVariable pour représenter la variable de la clause, elle contient également une méthodes check comme dans la classe Rule mais cette fois pour vérifier la validité d'une clause.

RuleVariable

Cette classe représente simplement une variable qui a un nom et une valeur.

Condition

Cette classe est très simple utilisée pour représenter une condition d'une clause.

2.4.2 les méthodes Importants du système expert

forwardChain

Cette méthode appartient à la classe BooleanRuleBase et son rôle est de crée une conflicRuleSet, cet ensemble de règles contient les règles qui sont valides pour les variables d'entrée et qui ne sont pas encore utilisées dans cette opération d'inférence, puis sélectionnez (utiliser) toutes les règles de cet ensemble.

match

Cette méthode appartient aussi à la classe BooleanRuleBase et elle est la méthode responsable de la création du conflictRuleSet.

fire

Cette méthode appartient à la classe Rule et elle est utilisée pour sélectionner ou on peut dire utiliser une règle.

check

Elle est utilisée par la méthode fire pour vérifier la validité d'une règle.

2.5 L'utilisation du système expert avec les produits fromagers

2.5.1 La description du cas d'utilisation

Nous avons recherché sur Internet le fromage, les types de fromage, les saveurs et plusieurs autres paramètres et nous avons choisi ces paramètres pour notre système (texture, saveur, méthode de préparation, source de lait, pays, nom et prix du fromage), puis nous avons rassemblé une petite base de données de règles, puis fait les quatre premiers attributs comme des entrées (texture, saveur, preparationMethod et milkSource), et le système expert détermine le pays, le nom et le prix du fromage, nous avons également établi les règles de manière à ce que le système expert fasse de multiples itérations d'inférence.

Pour enregistrer les règles et les variables que nous avons créées dans des fichiers (rules.txt et labels.txt), les règles et les étiquettes sont structurées comme ceci.

Structure du fichier de règles

(identifiant des règles, nombre d'antécédents, les clauses antécédent et la clause conséquente)

```
983675b4-2e6b-408f-a41e-deee101803b3,4,preparationMethod,=,unripened,texture,=,soft,flavor,=,mild,milkSource,=,cow,country,=,italy
75667485-10b3-95cd-34ba-37bfc1ff4196,4,preparationMethod,=,unripened,texture,=,soft,flavor,=,mild,milkSource,=,sheep,country,=,italy
94286415-10b3-34ba-c4fe-37bfad684146,4,preparationMethod,=,unripened,texture,=,soft,flavor,=,mild,milkSource,=,goat,country,=,italy
756dfdf0-bdef-47cb-36ef-eeeffaa65236,4,preparationMethod,=,unripened,texture,=,soft,flavor,=,mild,milkSource,=,buffalo,country,=,italy
61203548-65a1-9512-36ef-baba32165236,4,preparationMethod,=,unripened,texture,=,soft,flavor,=,lactic,milkSource,=,cow,country,=,italy
62845139-cefa-47cb-36ef-65412ecdf983,4,preparationMethod,=,bacteria-ripened,texture,=,hard,flavor,=,nutty,milkSource,=,cow,country,=,italy
8548c2c3-936d-4ca7-bea0-68bb19778fb5,2,country,=,italy,flavor,=,mild,cheese,=,ricotta
3e021474-bd26-43bb-8375-65b67b8c1c50,2,country,=,italy,flavor,=,lactic,cheese,=,mozzarella
```

Structure du fichier d'étiquettes

(le nom de la variable, les valeurs possible pour cette variable)

```
texture,soft,hard
flavor,mild,lactic,nutty,sweet,salty
preparationMethod,unripened,mold-ripened,bacteria-ripened
milkSource,cow,goat,sheep,buffalo
country,italy,united-kingdom,netherlands,switzerland,greek,french,america
cheese,mozzarella,ricotta,cheddar,gorgonzola,gouda,emmental,feta,roquefort,monterey-jack,parmesan
price,$3/kg,$4.3/kg,$8/kg,$7.3/kg,$10/kg,$2/kg,$9/kg,$6.8/kg,$6.5/kg,$3.6/kg
```

2.6 les choix de l'implémentation de l'application

Nous avons implémenté notre application en utilisant javafx avec css, nous l'avons fait dans deux onglets, un pour l'utilisateur et l'autre pour l'administrateur. voir les images ci-dessous:

Nous nous soucions beaucoup du Design de l'interface (UI) et nous avons essayé de la rendre plus propre et plus facile à utiliser

Interface Admin

Dans le panneau d'administration, il y a deux tableaux, celui de droite un tableau qui montre toutes les règles existantes afin que nous puissions choisir de supprimer la règle que nous voulons, et sur la gauche il y a un tableau qui montre les clauses quand l'admin veut ajouter une règle.

The Admin Panel interface features a sidebar on the left with two options: 'User Panel' and 'Admin Panel'. The main area contains a rule configuration form at the top and a table of existing rules below it.

Rule Configuration Form:

- Clause type:
- Variable:
- Operator:
- Value:
- Buttons: Add Clause, Add Rule

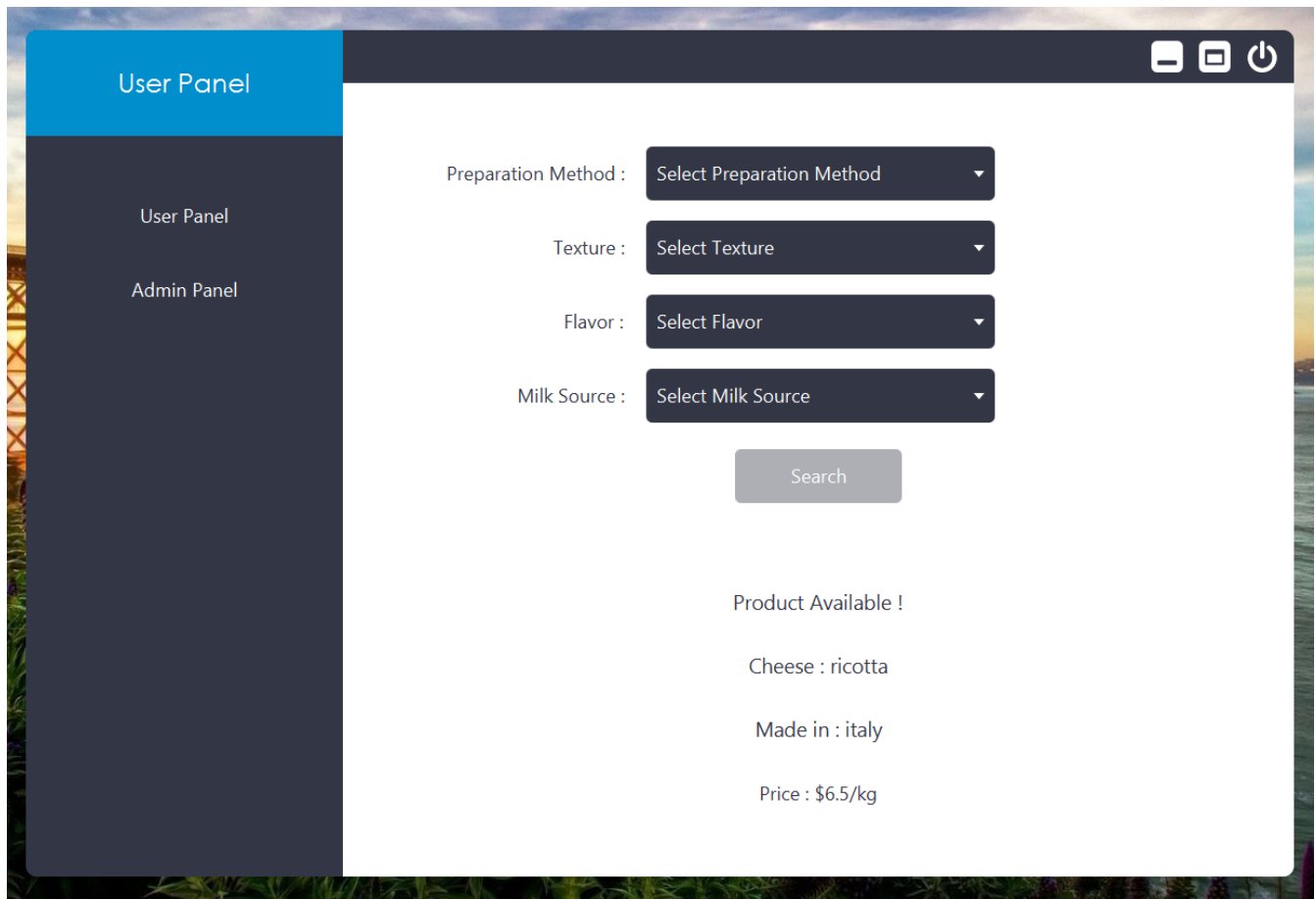
Table of Existing Rules:

Variable	Condition	Value	Consequent	Rule Name	antNbr	Consequent
				983675b4-2...	4	country=italy
				75667485-1...	4	country=italy
				94286415-1...	4	country=italy
				756dfdf0-b...	4	country=italy
				61203548-6...	4	country=italy
				62845139-c...	4	country=italy
				8548c2c3-9...	2	cheese=ricotta
				3e021474-b...	2	cheese=mozzarella
				fb376de4-b...	2	cheese=parmesan
				6ec496de-d...	4	country=united-kingdom
				282bf317-2...	2	cheese=cheddar
				1d21462e-a...	4	country=netherlands

Buttons: Delete Clause, Delete Rule

Interface Utilisateur

Dans le panneau de l'utilisateur, il y a quatre menus déroulants pour choisir les valeurs de l'entrée, puis le bouton de recherche pour effectuer l'opération d'inférence.



3.1 Introduction

Nous pouvons diviser cette partie en deux tâches principales, la première est d'adapter les interfaces utilisateur et admin pour pouvoir gérer différentes catégories de produits et de couse avec la création de deux autres classes wrapper ('Category' et 'Cats'), et la deuxième tâche consiste à implémenter les classes d'agents.

3.2 La première tâche

la première chose que nous avons faite dans cette tâche est de créer les deux classes wrapper.

3.2.1 La Classe Category

Cette classe contient tous les attributs nécessaires pour une catégorie de produits, voir l'image ci-dessous.

```
8      public class Category {
9          public String categoryName;
10         public String categoryRulePath;
11         public String categoryLabelPath;
12         public ArrayList<String> inputVariables;
13         public BooleanRuleBase rb;
14         public Map<String, RuleVariable> variables = new HashMap<>();
15         public Map<String, Condition> conditions = new HashMap<>();
16         public Map<String, Rule> rules = new HashMap<>();
```

Et bien sûr, il contient des méthodes pour lire, ajouter et supprimer les règles et d'autres méthodes.

3.2.2 La Classe Cats

Cats signifie catégories, c'est fondamentalement une classe pour instancier automatiquement toutes les catégories, il contient un attribut, une liste de catégories qui sera remplie plus tard avec des catégories en utilisant le constructeur.

Au début nous l'avons fait pour rendre l'instanciation automatique car nous avons l'idée de permettre la création et la suppression de nouvelles catégories de produits, et cela signifie instancier automatiquement de nouveaux agents. vous pouvez vérifier la classe Cats, vous trouverez une méthode nommée createCategory avec un fichier category.txt, pour stocker les catégories, voir les images ci-dessous.

```
33 public static void createCategory(String categoryName, ArrayList<String> variablesList, ArrayList<String> inputVariables) {
34     // Add the new category to the categories file
35     try {
36         BufferedWriter writer = new BufferedWriter(new FileWriter( fileName: ".\\src\\company\\categories.txt", append: true));
37         if (categories.size() > 0) {
38             writer.write( str: "\n");
39         }
40         writer.write(categoryName);
41         writer.flush();
42         for (String invar : inputVariables) {
43             writer.write( str: "," + invar);
44         }
45         writer.close();
46     } catch (Exception e){
47         System.out.println(e);
48     }
49     // Create new Labels file for this category and fill it with all the variables
50     try {
51         File labelsFile = new File( pathname: ".\\src\\company\\" + categoryName + "Labels.txt");
52         if (labelsFile.createNewFile()) {
53             System.out.println("File created: " + labelsFile.getName());
54         } else {
55             System.out.println("File already exists.");
56         }
57         BufferedWriter writer = new BufferedWriter(new FileWriter( fileName: ".\\src\\company\\" + categoryName + "Labels.txt", append: true));
```

le format du fichier categories.txt est (le nom de la catégorie et la liste des variables d'entrée pour cette catégorie)

```
1 cheese,preparationMethod,texture,flavor,milkSource
2 honey,flowerSource,taste,color,period
3 bigcat,weight,length,speed,lifeExpectancy
```

3.2.3 L'adaptation d'interface

L'interface d'utilisateur

The User Panel interface features a sidebar with 'User Panel' and 'Admin Panel' options. The main area contains five dropdown menus for filtering: 'Category of Product' (honey), 'Flower Source' (flowerSource), 'Taste' (taste), 'Color' (color), and 'Period' (period). A 'Search' button is located below these filters.

L'interface d'administrateur

The Admin Panel interface includes a sidebar with 'User Panel' and 'Admin Panel' options. The main area allows for rule configuration with dropdowns for 'Antecedent', 'Variable', and 'Value', and buttons for 'Add Clause' and 'Add Rule'. Below is a table of existing rules.

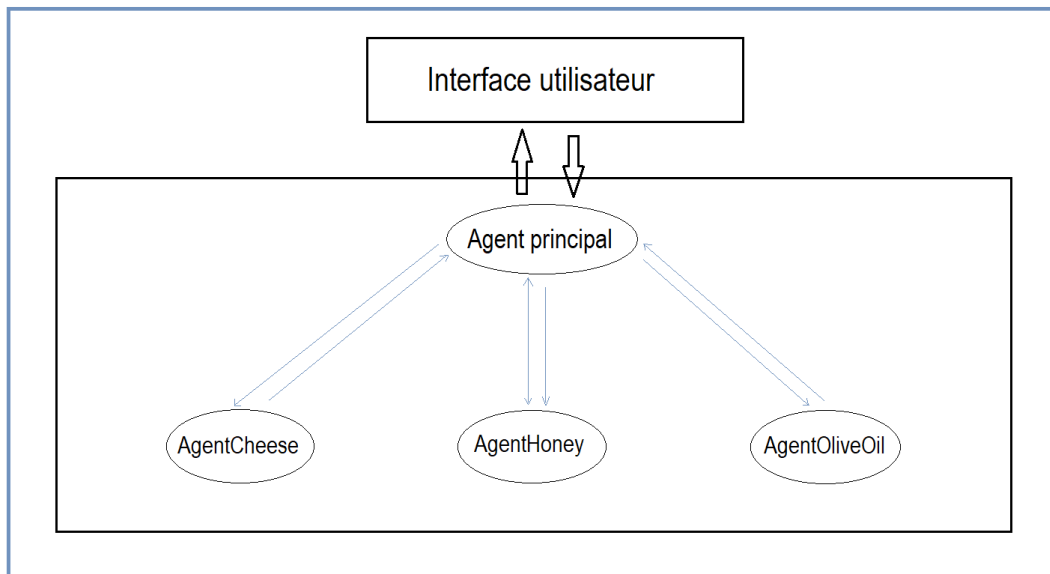
Variable	Condition	Value	Consequent	Rule Name	antNbr	Consequent
honey	=	whiteHoney	false			
period	=	spring	false			
taste	=	milder	true			

A message 'Aucun contenu dans la table' is displayed below the table. At the bottom, there are 'Delete Clause' and 'Delete Rule' buttons.

3.3 Deuxième tâche

Cette tâche est consacré a l'implementation du système multi-agent on utilisant la plateforme jade une plate-forme multi-agent créé par le laboratoire TILAB et décrite par Belli femine et al, Elle permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA. Elle est implémentée en JAVA, on a commencer par etablir la conception et l'architecture du système multi-agent.

3.3.1 Architecture du système multi-agents



Ensuite la creation des différentes classes d'agents, comme suite:

3.3.2 La classe MainAgent

Cette classe hérite de la classe Agent de jade.core, elle représente l'agent principal autrement dit l'agent policier de notre application, son rôle est de surveiller et guider la circulation des données entre l'interface utilisateur et les agents. Elle est composée de quatre paramètres consacrés à la communication entre l'interface et les agents, est deux comportements cycliques parallèles, un pour la communication avec l'interface et l'autre pour la communication avec l'agent chercheur. voir les images ci-dessous.

```

13 public class MainAgent extends Agent {
14     private ACLMessage envoye;
15     private ACLMessage reçu;
16     private String dataIn=null;
17     private String dataOut="";
18
19     protected void setup() {
20         UserController.setSender(this);
21         System.out.println("the agent "+ getLocalName()+" is created");
22         ParallelBehaviour parallel =new ParallelBehaviour();
23         parallel.addSubBehaviour(new CyclicBehaviour() {
24             @Override
25             public void action() {
26                 if(dataIn!=null){
27                     System.out.println(dataIn);
28                     System.out.println("the agent "+ getLocalName()+" get dataIn from the user");
29                     String[] labelsTMP = dataIn.split( regex: " ");
30                     if (labelsTMP[0].equals("cheese")) {
31                         envoye =new ACLMessage(ACLMessage.INFORM);
32                         envoye.addReceiver(new AID( name: "cheeseAgent", AID.ISLOCALNAME));
33                         envoye.setContent(dataIn);
34                         send(envoye);
35                     }
36                     else if(labelsTMP[0].equals("honey")){
37                         envoye =new ACLMessage(ACLMessage.INFORM);

```

```

36     else if(labelsTMP[0].equals("honey")){
37         envoye =new ACLMessage(ACLMessage.INFORM);
38         envoye.addReceiver(new AID( name: "honeyAgent", AID.ISLOCALNAME));
39         envoye.setContent(dataIn);
40         send(envoye);
41     }
42     else if(labelsTMP[0].equals("oliveOil")){
43         envoye =new ACLMessage(ACLMessage.INFORM);
44         envoye.addReceiver(new AID( name: "oliveOilAgent", AID.ISLOCALNAME));
45         envoye.setContent(dataIn);
46         send(envoye);
47     }
48     dataIn=null;
49     } }));
50 parallel.addSubBehaviour(new CyclicBehaviour() {
51     @Override
52     public void action() {
53         reçu=new ACLMessage(ACLMessage.INFORM);
54         reçu=receive();
55         if(reçu!=null){ System.out.println("the agent "+ getLocalName()+" set the dataOut to the user");
56             setDataOut(reçu.getContent());
57             System.out.println(dataOut);
58         }
59         else block(); }));
60 addBehaviour(parallel);

```

3.3.3 La classe AgentCats

Elle représente les agents chercheurs, ses agents interagissent avec l'agent principal 'MainAgent', ils reçoivent les données nécessaires pour la recherche du résultat, ils utilisent le système expert pour inférer la réponse à chaque catégorie précisée par l'utilisateur, puis l'envoyer à l'agent principal, ce dernier envoie la réponse à l'interface utilisateur pour qu'elle soit affichée.

Structure de données reçu par AgentCats

voir l'image ci-dessous.

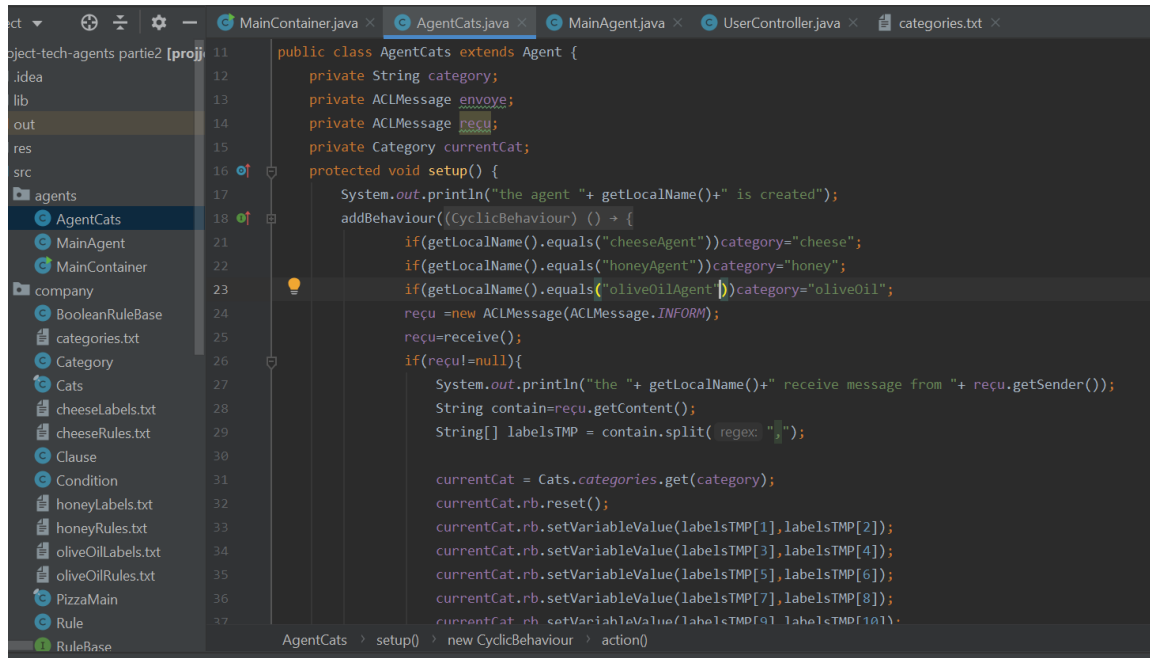
```
honey,flowerSource,sidrTrees,taste,milder,color,yellower,period,anytime,country,null,honey,null,price,null
```

Structure de reponse envoyer par AgentCats

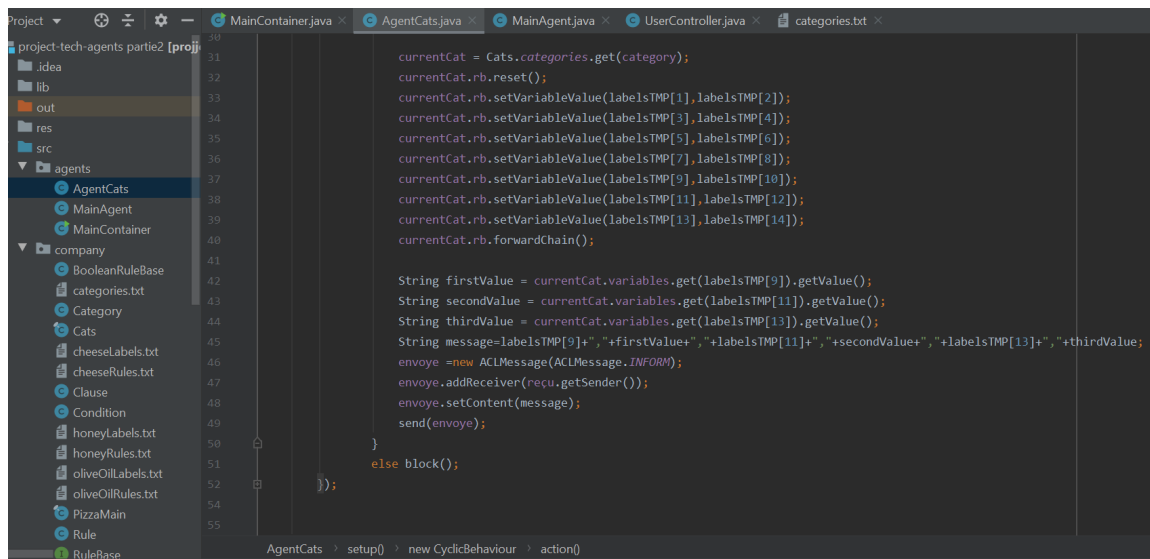
voir l'image ci-dessous.

```
country,yemen,honey,sidrHoney,price,$18
```

Structure classe AgentCats



```
public class AgentCats extends Agent {
    private String category;
    private ACLMessage envoye;
    private ACLMessage reçu;
    private Category currentCat;
    protected void setup() {
        System.out.println("the agent " + getLocalName() + " is created");
        addBehaviour(new CyclicBehaviour() {
            if(getLocalName().equals("cheeseAgent"))category="cheese";
            if(getLocalName().equals("honeyAgent"))category="honey";
            if(getLocalName().equals("oliveOilAgent"))category="oliveOil";
            reçu =new ACLMessage(ACLMessage.INFORM);
            reçu=receive();
            if(reçu!=null){
                System.out.println("the " + getLocalName()+" receive message from "+ reçu.getSender());
                String contain=reçu.getContent();
                String[] labelsTMP = contain.split( regex: " ");
                currentCat = Cats.categories.get(category);
                currentCat.rb.reset();
                currentCat.rb.setVariableValue(labelsTMP[1],labelsTMP[2]);
                currentCat.rb.setVariableValue(labelsTMP[3],labelsTMP[4]);
                currentCat.rb.setVariableValue(labelsTMP[5],labelsTMP[6]);
                currentCat.rb.setVariableValue(labelsTMP[7],labelsTMP[8]);
                currentCat.rb.setVariableValue(labelsTMP[9],labelsTMP[10]);
            }
        });
    }
}
```



```
currentCat = Cats.categories.get(category);
currentCat.rb.reset();
currentCat.rb.setVariableValue(labelsTMP[1],labelsTMP[2]);
currentCat.rb.setVariableValue(labelsTMP[3],labelsTMP[4]);
currentCat.rb.setVariableValue(labelsTMP[5],labelsTMP[6]);
currentCat.rb.setVariableValue(labelsTMP[7],labelsTMP[8]);
currentCat.rb.setVariableValue(labelsTMP[9],labelsTMP[10]);
currentCat.rb.setVariableValue(labelsTMP[11],labelsTMP[12]);
currentCat.rb.setVariableValue(labelsTMP[13],labelsTMP[14]);
currentCat.rb.forwardChain();

String firstValue = currentCat.variables.get(labelsTMP[9]).getValue();
String secondValue = currentCat.variables.get(labelsTMP[11]).getValue();
String thirdValue = currentCat.variables.get(labelsTMP[13]).getValue();
String message=labelsTMP[9]++,""+firstValue++,""+labelsTMP[11]++,""+secondValue++,""+labelsTMP[13]++,""+thirdValue;
envoye =new ACLMessage(ACLMessage.INFORM);
envoye.addReceiver(reçu.getSender());
envoye.setContent(message);
send(envoye);
}
else block();
});
});
```

3.3.4 La classe MainContainer

Cette classe représente la classe main, elle effectue l'activation du runtime environnement et l'instanciation des agents nécessaires pour le fonctionnement de notre système multi agent, et ainsi le lancement de l'interface graphique de l'application. voir l'image ci-dessous.

```

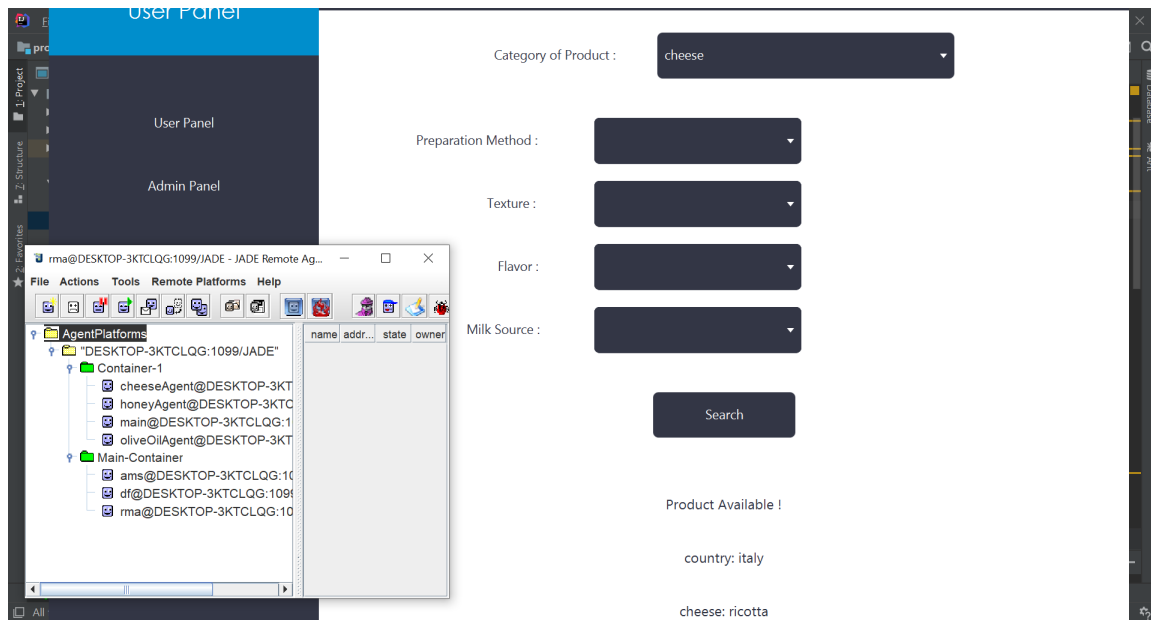
13 public class MainContainer {
14     public static void main(String[] args) {
15         Runtime runtime=Runtime.instance();
16         Profile profileImpl=new ProfileImpl();
17         Profile profileImpl=new ProfileImpl();
18         ContainerController mainContainer = runtime.createMainContainer(profileImpl);
19         ContainerController container = runtime.createAgentContainer(profileImpl);
20         try {
21             AgentController ag1=container.createNewAgent( nickname: "main",agents.MainAgent.class.getName(), args: null);
22             AgentController ag2=container.createNewAgent( nickname: "cheeseAgent", AgentCats.class.getName(), args: null);
23             AgentController ag3=container.createNewAgent( nickname: "honeyAgent", AgentCats.class.getName(), args: null);
24             AgentController ag4=container.createNewAgent( nickname: "oliveOilAgent", AgentCats.class.getName(), args: null);
25             AgentController ag=mainContainer.createNewAgent( nickname: "rma", className: "jade.tools.rma.rma", args: null);
26
27             ag.start();
28             ag1.start();
29             ag2.start();
30             ag3.start();
31             ag4.start();
32
33         } catch (ControllerException e) {
34             e.printStackTrace();
35         }
36         Launch.main(args);
37     }
38 }

```

La structure de notre système multi-agent

name	addresses	state	owner
NAME	ADDRES...	STATE	OWNER

Resultat final



3.4 Conclusion

Ce projet nous a permis d'apprendre beaucoup de choses sur les systèmes experts et les systèmes multi-agents. Nous avons réussi à concevoir un système multi-agent qui fonction et interagit avec le système expert. Nous avons pris conscience de l'importance des systèmes multi-agents.