1. Introduction

« Un agent est un système informatique situé dans un environnement qu’il peut percevoir et sur lequel il peut agir. Il est capable d’actions flexibles et autonomes qu’il met en œuvre en vue d’atteindre ses objectifs de conception. » [1]

Dans l’optique d’étudier le fonctionnement des agents puis d’en implémenter, on commence par s’intéresser aux systèmes experts, qui font partie intégrante des agents. Ainsi, on s’attardera dans un premier temps sur des exemples types de systèmes experts existants, que nous tenterons de comprendre et d’analyser afin de partir sur de bonnes bases pour construire, à notre tour, des systèmes experts pour le problème qui nous ait posé (place de marché). Nous tâcherons dans un second temps, d’adapter ces systèmes-là pour mieux répondre à la problématique.

Plus concrètement, ce TP se subdivise en deux grandes parties intimement reliées : il s’agira d’abord de mettre en place deux systèmes experts « Acheteur » et « Vendeur », en constituant leurs bases de règles et de connaissances respectives. C’est ce que nous aborderons dans cette première partie. Ensuite, et c’est le but final recherché, nous construirons deux agents sur la base de ces systèmes experts, que nous ferons communiquer sur une place de marché virtuelle pour réaliser des opérations de vente et d’achat de voitures. Les deux parties étant liées, il est nécessaire de préparer la seconde en pensant à la première. Le processus que l’on a suivi pour ce faire, est détaillé dans ce rapport.

1. Étude de systèmes experts existants

En nous basant sur le *textbook* : « *Constructing Intelligent Agents with Java*», et dans le but de mieux comprendre les systèmes experts et d’aboutir à une certaine maitrise de leur fonctionnement, nous avons implémenté les classes d’un système expert dédié à la reconnaissance de différents types de véhicules. Ces classes se présentent comme suit :

* **Clause :** regroupe les données concernant une clause (la condition, les variables,…), ainsi que des méthodes permettant de tester sa véridicité, etc.
* **Condition :** permet de traduire la condition définie avec le type *String* en du code compilable.
* **Rule :** sert à définir une règle et contient des méthodes utilisées pour le processus d’inférence.
* **RuleApplet :** contient toutes les méthodes d’initialisation des différents systèmes, comprenant les instructions d’affichage de l’interface graphique, etc.
* **RuleBase :** contient la liste des *RuleVariables* et celle des *Rules* de la base, ainsi que les méthodes implémentant les moteurs d’inférence (chaînage avant et chaînage arrière).
* **RuleVarDialog :** génère la boite de dialogue des règles et des variables.
* **Variable :** permet d’identifier une variable de la base.
* **RuleVariable :** hérite des propriétés de la classe *Variable*, avec cependant, des spécifications supplémentaires pour pouvoir représenter des variables inférées.

[1] Prof. Drias, Habiba. Notes de cours : « La technologie des agents ». USTHB, Alger.

Avant de s’étaler davantage, rappelons brièvement les principes des moteurs d’inférence : il y a le chaînage avant qui procède à l’inférence des règles à partir de la base des faits, jusqu’à inférer le but. Il y a également le chaînage arrière qui entame l’inférence à commencer par le but et ne s’arrête que lorsque tous les éléments inférés appartiennent à la base des faits. Ces deux algorithmes sont déjà implémentés dans *RuleBase*.

L’étude de ces différentes classes nous permet de faire la transition vers la conception des systèmes experts dont nous aurons besoin pour établir notre place de marché.

1. Conception de nos systèmes experts

Maintenant que l’on est au fait des bonnes pratiques pour la conception de systèmes experts, après l’étude de ceux sur le *textbook*, on peut passer à la création des deux systèmes experts dont nous avons besoin dans ce projet.

1. **Simulation d’un concessionnaire automobile**

Notre système modélise un concessionnaire automobile virtuel. Ce dernier dispose dans ses magasins de plusieurs marques automobiles. Un stock commun alimente tous ces points de vente. Dans notre environnement, le concessionnaire est représenté par le « vendeur », et ses clients potentiels par les « acheteurs ».

Ainsi, nous définissons les acteurs suivants :

1. **L’acheteur**

Un acheteur est une entité disposant d’un budget bien défini et désirant acheter un véhicule dont il sélectionne les critères. Il cherchera donc soit à avoir le prix ou la disponibilité d’un certain véhicule. Il pourra aussi voir quels sont les choix qui s’offrent à lui selon son budget.

Dans notre cas, nous spécifions un profil d’acheteur bien défini : il a un budget lui permettant uniquement d’acheter une citadine ou une petite citadine, et ne s’intéressera donc qu’aux véhicules de cette catégorie. Notre acheteur est aussi « pressé » et voudrais avoir son véhicules dans les plus brefs délais ; ainsi, le délai de disponibilité sera un facteur clé dans l’achat ou non d’un véhicule qu’il aura choisi. Par ailleurs, cet acheteur a un caractère négociateur, qui le pousse à discuter les prix de vente dans la majorité des cas.

Puisqu’il n’y a pas encore de communication entre les agents sur la place de marché, l’acheteur doit donc disposer des informations sur les véhicules dans sa base de connaissances. Aussi, et pour la même raison, on simule une négociation entre l’acheteur et le vendeur sans que celle-ci n’ait réellement lieu, mais uniquement dans le but de préparer le terrain à la seconde partie.

1. **Le vendeur**

Le vendeur est une entité commerciale cherchant à maximiser son profit. Il peut consulter son stock, alimenter ce stock et vérifier la disponibilité et le prix d’un produit. Le vendeur établit une liste de profils des clients (donc des acheteurs) afin de leur offrir certains avantages visant à les fidéliser. Il aura aussi à son niveau les statistiques de ventes, et pourra annoncer des promotions sur la base de ces statistiques, entre autres. D’autres promotions incluent la participation à des salons de l’automobile, ou des promos ponctuelles (comme à l’occasion de l’anniversaire de la marque).

Le vendeur tâchera dans tous les cas, de pousser l’acheteur à faire la plus grande offre possible, en acceptant néanmoins les négociations, et en étant disposé à faire des concessions tant que le montant offert ne descend pas en dessous d’un seuil déterminé.

Les deux entités sus-décrites seront chacune modélisée par un système expert. Leur comportement est défini par les règles que nous établissons pour notre système.

1. **Variables et règles**

En s’inspirant du premier système expert vu dans le *textbook* (traitant des véhicules), on adapte les variables et de suite les règles pour mieux répondre à notre problème. Nous remarquons qu’il n’y a que la classe *RuleApplet* qui est amenée à changer, toutes les autres serviront de base immuable que l’on peut réutiliser à chaque fois que l’on voudra construire des systèmes experts.

//éventuellement ajouter phrase intro

1. **Le système expert « Acheteur »**

Les variables qu’emploie ce système se présentent comme suit :

* type de véhicule,
* produit : définit la marque et le modèle du véhicule,
* moteur,
* couleur,
* budget : somme dont dispose l’acheteur,
* négociation : détermine s’il y a ou non possibilité de négocier le prix,
* prix,
* quantité : nombre de véhicules commandés,
* commande : détermine si oui ou non l’acheteur désire lancer sa commande,
* changer\_vendeur,
* changer\_modèle,
* délai\_disponibilité : période d’attente avant l’acquisition du véhicule.

Connaissant à présent les variables manipulées, place à l’étude des règles qu’elles engendrent.

On définit plusieurs catégories de règles :

* Règles se rapportant au prix des différents modèles : elles infèrent le *prix* du véhicule présentant les caractéristiques données en prémisses, à savoir : le produit (*marque\_modèle*), le type de *moteur* et la *couleur*.

Exemple :

Rule R1 **=** **new** Rule**(**rb**,** "Prix Peugeot\_307 -1-"**,**

**new** Clause**(**produit**,**cEquals**,** "Peugeot\_307"**),**

**new** Clause**(**carburant**,**cEquals**,** "Essence"**),**

**new** Clause**(**color**,** cEquals**,** "Standard"**),**

**new** Clause**(**prix**,** cEquals**,** "120"**))** **;**

* Règles relatives aux actions de l’acheteur : les variables inférées dans cette catégorie de règles sont : *négociation*, *changer\_modèle*, *changer\_vendeur* et *commande*, en se basant sur les valeurs des variables : *délai\_dispo* et du *budget* selon le prix de base du modèle choisi.

Exemples :

Rule R37 **=** **new** Rule**(**rb**,** "Délai de disponibilité acceptable <60 -1-"**,**

**new** Clause**(**delai\_dispo**,**cLessThan**,** "60"**)** **,**

**new** Clause**(**produit**,**cEquals**,** "Peugeot\_307"**)** **,**

**new** Clause**(**budget**,**cMoreThan**,** "120"**)** **,**

**new** Clause**(**commande**,** cEquals**,** "oui"**))** **;**

Rule R38 **=** **new** Rule**(**rb**,** "Délai de disponibilité acceptable <60 -2-"**,**

**new** Clause**(**delai\_dispo**,**cLessThan**,** "60"**)** **,**

**new** Clause**(**produit**,**cEquals**,** "Peugeot\_307"**)** **,**

**new** Clause**(**budget**,**cLessThan**,** "120"**)** **,**

**new** Clause**(**budget**,**cMoreThan**,** "100"**)** **,**

**new** Clause**(**negociation**,** cEquals**,** "oui"**))** **;**

1. **Le système expert « Vendeur »**

Ce système expert possède des variables communes à celles du système précédent. Néanmoins, le besoin d’en déclarer de nouvelles s’est fait ressentir. Voici la liste exhaustive des différentes variables propres au vendeur :

* type de véhicule,
* produit : définit la marque et le modèle du véhicule,
* moteur,
* quantité : nombre de véhicules commandés,
* couleur,
* boite\_à\_vitesse : manuelle ou automatique,
* options : donne les différentes classes d’options disponibles (toutes options, de base…),
* stock : donne l’état du stock au niveau du concessionnaire,
* alimenter\_stock : permet d’alimenter le stock dès que celui-ci est à moitié vide,
* profil : définit le profil de l’acheteur (habitué, grossiste…) pour appliquer des avantages,
* avantage : privilèges offerts selon le profil de l’acheteur,
* licence\_moujahidine : attribuer une réduction de prix pour les anciens moujahidines,
* ancien\_moujahid,
* disponible : permet de connaitre la disponibilité ou non du véhicule recherché,
* vente,
* différence : permet de définir l’écart entre le budget de l’acheteur et le prix affiché du véhicule. Le vendeur ne négocie que si cet écart ne dépasse pas un certain seuil (déterminé),
* négociation : détermine si oui ou non, le vendeur est prêt à négocier le prix,
* promotion : définit les promotions qui pourraient être lancées (soldes, salon auto, …),
* date\_anniversaire : permet de savoir si oui ou non la date actuelle correspond à la date d’anniversaire de la marque,
* statistiques : donne une évaluation du taux de vente (bon, moyen, mauvais),
* prix,
* prix\_supplémentaire\_boiteVitesse : prix à ajouter en fonction du type de boite à vitesse choisi,
* prix\_supplémentaire\_options : prix à ajouter pour passer à une classe supérieure d’options.

Les règles qui permettent de traiter ces variables se divisent en plusieurs catégories :

* Règles attribuant des avantages selon les profils des acheteurs : ces règles permettent d’inférer la variable *avantage* en fonction de la valeur de la variable *profil*.

Exemple :

Rule VR1 **=** **new** Rule**(**rb**,** "Avantage selon profil -1-"**,**

**new** Clause**(**profil**,**cEquals**,** "Habitué"**),**

**new** Clause**(**avantage**,**cEquals**,** "vidange\_gratuite"**))** **;**

* Règles se rapportant à la gestion du stock : ces règles définissent la valeur de la variable *alimenter\_stock* selon l’état dans lequel se trouve le *stock.*

Exemple :

Rule VR3 **=** **new** Rule**(**rb**,** "Gestion de stock -1-"**,**

**new** Clause**(**stock**,**cEquals**,** "vide"**)** **,**

**new** Clause**(**alimenter\_stock**,**cEquals**,** "oui"**))** **;**

* Règles relatives aux actions entreprises (négociation/vente) par le vendeur : les prémisses sont les valeurs des variables : *différence* et *disponible*, inférant les variables *négociation* et *vente.*

Exemple :

Rule VR7 **=** **new** Rule**(**rb**,** "Déroulement selon une différence +seuil -2-"**,**

**new** Clause**(**difference**,**cEquals**,** "+seuil"**)** **,**

**new** Clause**(**disponible**,**cEquals**,** "oui"**)** **,**

**new** Clause**(**vente**,**cEquals**,** "oui"**))** **;**

Rule VR8 **=** **new** Rule**(**rb**,** "Déroulement selon une différence -seuil -1-"**,**

**new** Clause**(**difference**,**cEquals**,** "-seuil"**)** **,**

**new** Clause**(**negociation**,**cEquals**,** "non"**))** **;**

* Règles se rapportant au prix des différents modèles : définition identique à celle de l’acheteur (vue plus haut).
* Règles déterminant le prix supplémentaire selon la boite à vitesse choisie : en fonction de la valeur de la variable *boite\_à\_vitesse*, on infère la valeur de la variable *prix\_supplémentaire\_boiteVitesse*.

Exemple :

Rule V31 **=** **new** Rule**(**rb**,** "Prix supp selon type de Boite à vitesse -2-"**,**

**new** Clause**(**boiteV**,**cEquals**,** "automatique"**)** **,**

**new** Clause**(**prix\_suppV**,** cEquals**,** "+20"**))** **;**

* Règles déterminant le prix supplémentaire selon les options choisies : en fonction de la valeur de la variable *options*, on infère la valeur de la variable *prix\_supplémentaire\_options*.

Exemple :

Rule V33 **=** **new** Rule**(**rb**,** "Prix supp selon options choisies -2-"**,**

**new** Clause**(**options**,**cEquals**,** "toutes\_options"**)** **,**

**new** Clause**(**prix\_suppO**,** cEquals**,** "+20"**))** **;**

* Règles relatives aux promotions : différentes promotions peuvent-être lancées, on en choisit une en fonction de la valeur de la variable *statistiques*. En ce qui concerne la promotion *anniversaire\_marque*, elle est inférée en fonction de la valeur de la variable *date\_anniversaire*.

Exemple :

Rule V37 **=** **new** Rule**(**rb**,** "Promotion selon statistiques de vente -2-"**,**

**new** Clause**(**stats**,**cEquals**,** "moyennes"**)** **,**

**new** Clause**(**promo**,** cEquals**,** "salon\_auto"**))** **;**

* Règles relatives à l’attribution de la licence pour moujahidines : on infère la valeur de la variable *licence\_moujahidine*, en fonction de la valeur de la variable *ancien\_moujahid*.

Exemple :

Rule V40 **=** **new** Rule**(**rb**,** "Licence moujahidine -1-"**,**

**new** Clause**(**ancienM**,**cEquals**,** "oui"**)** **,**

**new** Clause**(**licence**,** cEquals**,** "oui"**))** **;**

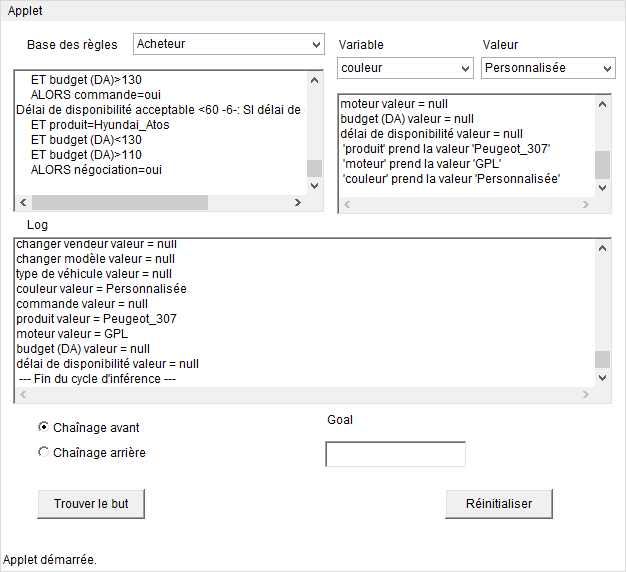
1. **Aperçu de l’interface développée**

Afin de pouvoir travailler avec les variables et les règles que l’on a énoncées plus haut, nous avons besoin de passer par une interface. Celle-ci est une adaptation de l’interface proposée dans le livre référence « *Constructing Intelligent Agents with Java*» [2].

Nous avons gardé le même agencement des zones de texte (cf. Figure 1 ci-dessous) :

* la base des règles est ainsi affichée dans le rectangle en haut à gauche,
* la sélection des variables et des valeurs leur correspondant se fait à travers des listes déroulantes en haut à droite, et la zone de texte juste en dessous permet d’afficher les changements de valeur d’une variable donnée,
* le processus d’inférence est décrit à travers un log dans la zone en dessous de celles sus-décrites,
* enfin, des boutons *radiobox* permettent de choisir le type de chaînage (avant/arrière) et une zone de texte permet d’entrer le but recherché.

Au-delà de cela, nous avons apporté quelques modifications à cette interface. Les plus notables étant la traduction de la totalité des textes affichés en Français (puisque le projet se fait en cette langue), et la suppression du bouton « Demo » qui ne paraissait pas nécessaire à notre application.



**Figure 1.** Capture d’écran de l’interface

[2] Bigus Joseph P., Bigus Jennifer (2001). Contructing Intelligent Agents with Java. Etats Unis : Wiley. (page 94)

1. Conclusion

Ce TP nous a permis d’en apprendre beaucoup sur les systèmes experts, et de voir le rôle primordial qu’ils jouent dans la construction d’un agent (qu’on fera en seconde partie). Nous avons vu que nous retrouvions toujours le même socle : le moteur d’inférence (les algorithmes de chaînage avant et arrière), l’interface… et qu’il suffisait d’adapter les variables (base de connaissances) ainsi que la base des règles à notre problème.

Par ailleurs, nous avons vu l’importance de définir très tôt l’environnement dans lequel allaient évoluer les agents, et préparer leur interaction. Les systèmes experts permettent de modéliser une situation du monde réel (ici celle de concessionnaires automobile) de manière concise, et donc de décrire le cadre dans lequel se trouvent les différentes entités (concurrence, collaboration…).

1. Perspectives

Comme explicité dans l’introduction, le but ultime à atteindre est de constituer des agents acheteurs et vendeurs et de les faire communiquer sur une place de marché. Au bout de cette première partie, nous avons maintenant les systèmes experts sur lesquels reposeront ces agents. Cependant, ceux-ci sont toujours indépendants l’un de l’autre et ne peuvent donc communiquer. Or, puisqu’il s’agit d’une place de marché, l’interaction sous toutes ses formes (communication entre acheteur et vendeur, concurrence entre les concessionnaires) est fondamentale. C’est pour cela que nous nous attellerons dans la deuxième partie à faire émerger ces interactions, en passant entre autres à la plateforme JADE (pour *Java* *Agent DEvelopment*) pour l’implémentation.