**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**

**UNIVERSITE MOHAMED BOUDIAF - M'SILA**

**FACULT : MATHEMATIQUE ET INFORMATIQUE**

**DEPARTEMENT :INFORMATIQUE**

**N° :……..**


**DOMAINE :** Mathématiques et Informatique

**FILIERE : INFORMATIQUE**

**OPTION : TIC**

**Mémoire présenté pour l'obtention**
**Du diplôme de Master Académique**

**Par**: SACI HOUSSAM EDDINE

**Intitulé**

# DEVELOPPEMENT D'UN SYSTEME MULTI AGENTS POUR LA NEGOCIATION DES PRIX APPROCHE BASEE SUR LES AGENTS MOBILES

**Soutenu devant le jury composé de :**

| | | |
|---|---|---|
| …….. | Université de M'sila | Président |
| Mme. MELIOUH AMEL | Université de M'sila | Rapporteur |
| …… | Université de M'sila | Examinateur |

**Année universitaire : 2016 /2017**

**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**

**UNIVERSITE MOHAMED BOUDIAF - M'SILA**

**FACULT : MATHEMATIQUE ET INFORMATIQUE**

**DEPARTEMENT :INFORMATIQUE**

**N° :……..**


**DOMAINE :** Mathématiques et Informatique

**FILIERE : INFORMATIQUE**

**OPTION : TIC**

**Mémoire présenté pour l'obtention**
**Du diplôme de Master Académique**

**Par**: SACI HOUSSAM EDDINE

**Intitulé**

# DEVELOPPEMENT D'UN SYSTEME MULTI AGENTS POUR LA NEGOCIATION DES PRIX APPROCHE BASEE SUR LES AGENTS MOBILES

**Soutenu devant le jury composé de :**

| | | |
|---|---|---|
| …….. | Université de M'sila | Président |
| Mme. MELIOUH AMEL | Université de M'sila | Rapporteur |
| …… | Université de M'sila | Examinateur |

**Année universitaire : 2016 /2017**

# TABLE DES MATIERES :

# INTRODUCTION GENERALE

Au fil des années, le métier de commercial s'est profondément modifié en raison des pressions exercées par les acheteurs, particuliers comme professionnels. D'une part, les premiers, très avertis, disposent désormais des moyens nécessaires pour obtenir rapidement des renseignements, ils sont donc devenus très exigeants dans leurs achats.

D'autre part, les acheteurs professionnels, devenus très « pointus », ils recherchent des produits ou services à des prix très compétitifs. La négociation exige l plusieurs compétences Quelle que soit la forme de la négociation, si on espère en maximiser les chances de succès.

Aujourd'hui le commerce électronique est devenu une activité très importante, se réfère à des activités commerciales comme la vente et l'achat de produits et services effectués sur des systèmes électroniques comme l'Internet et les réseaux informatiques , Le plus grand et le plus important avantage du e-commerce, est qu'il permet à une entreprise ou à un particulier d'atteindre le marché mondial. Il a aussi répondu aux exigences du marché national et international, car les activités commerciales ne sont plus limitées par les frontières géographiques.

Dans ce cadre, Notre problématique est de créer une plateforme de négociation des prix entre les acheteurs et les vendeurs pour soumettre les produits a l'acheteur avec les meilleurs prix , Nous allons utiliser la plateforme de système multi agents JADE qui excelle par la facilité d'utilisation, aussi elle supporte la mobilité qui permets aux agents de faire transférer d'une station a l'autre , puis elle a la fonctionnalité de s'adapte à un environnement changeant. .

Donc, ce travail s'inscrit dans une démarche informatique visant à améliorer la relation existante déjà entre les acheteurs et les vendeurs par un Enchères .Le but d'utiliser la technologie des agents mobiles pour implémenter des agents vendeurs qui présentent leurs produit, et des agents chercheurs cherchent le meilleur prix. Les offres peuvent se trouver dans des machines séparées et les agents chercheurs doivent migrer pour obtenir les prix.

Ce manuscrit est organisé en deux chapitres, dans le premier nous présentons la technologie des agents mobiles, leurs caractéristiques, fonctionnement et utilisation, Le deuxième chapitre est consacré a présenter notre contribution qui est l'implémentation d'un système de négociation des prix avec les agents mobiles.

# CHAPITRE 01 :

# LES AGENTS MOBILES

# 1. Introduction :

La modernité dans la technologie est toujours associée à l'efficacité, la vitesse et la précision, et de réaliser tout cela nécessite plus d'efforts, plus de coopération, et de participer aux travaux.

Un système multi agents est l'une des solutions les plus importantes de travail divisé, Ce qui forme aujourd'hui un moyen pour développer et rendre des logiciels plus rapides et efficaces.

Ce qui rend un système plus utile, étant contenir des Agents mobiles qui se déplacent d'une machine à l'autre sur le réseau, sans perdre leurs codes ni leurs états.

Dans ce chapitre on va découvrir C'est quoi les agents, les agents mobiles et comment ils fonctionnent avec leurs plateforme de développement.

# 2. Les Agents :

## 2.1 Définitions d'agents :

Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre les objectifs dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

## 2.2 Caractéristiques d'agent :

Les caractéristiques d'un agent sont :

**Situé** l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement.

**Autonome** l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne.

**Proactif** l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment.

**Capable** de répondre à temps – l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis.

**Social** l'agent doit entre capable d'interagir avec des autres agents (logiciels ou humains) afin d'accomplir des taches ou aider ces agents à accomplir leurs taches. [1]

### 2.3 Systèmes multi-agents :

Un système multi-agents est un système distribué composé d'un ensemble d'agents.
Un SMA est caractérisé ainsi:

- chaque agent a des informations ou des capacités de résolution de problèmes limités (ainsi, chaque agent a un point de vue partiel);
- il n'y a aucun contrôle global du système multi-agents;
- les données sont décentralisées;
- le calcul est asynchrone. [2]

### 2.4 Typologie d'agents dans un SMA :

#### a) Les agents réactifs

Les agents réactifs sont les plus sommaires. Ils ont un comportement du type « stimulus – réponse ». L'agent réactif ne possède pas une représentation complète de son environnement et n'est pas capable de tenir compte de ses actions passées.

Les systèmes multi-agents constitués uniquement d'agents réactifs possèdent un grand nombre d'agents. La convergence du comportement de l'ensemble des agents vers un état décisionnel stable n'est pas forcément assurée, et si un état stable est atteint, il n'est pas sur qu'il s'agisse de la solution optimale.

**b) Les agents cognitifs**

Les agents cognitifs sont plus évolués. Ils sont les résultats directs des recherches menés dans le domaine de l'intelligence artificielle. Les agents cognitifs ont une représentation globale de leur environnement et des autres agents avec lesquels ils communiquent. Ils savent tenir compte de leur passé et s'organisent autour d'un mode social d'organisation.

Les systèmes multi-agents constitués uniquement d'agents cognitifs sont constitués d'un nombre d'agents assez faible. Ils réclament des ressources plus importantes que les systèmes d'agents réactifs. La convergence du système vers un état décisionnel stable n'est pas non plus assurée par l'utilisation de ce type d'agents, mais ils permettent de résoudre des problèmes plus complexe et nécessitant une plus grande abstraction. [3]

## 2.5 Langages de communications entre agents :

De nombreux langages de communications entre agents (ACL) se sont développés.

- **KQML (93, 97) Knowledge Query and Manipulation Language :**
  KQML a été conçu comme étant à la fois un format de message et un protocole de transfert de messages venant aider les agents intelligent au partage et échange de données de haut niveau tout en étant indépendant des machines.

**Quelques performatives de ce langage de communication :**

E : l'agent émetteur

R : l'agent récepteur

ask-one : E veut que seulement R réponde à sa question C

ask-if : E veut savoir si la réponse à la question précisée en C se trouve dans la BVC de R

tell : E affirme au R que C est dans la BVC de E

broadcast : E veut que R transmette à son tour la performative à toutes ses connexions

- **FIPA-ACL (97, 99, 2000) :**
  La FIPA (Foundation for Intelligent Physical Agents) est une organisation à but non lucratif fondée en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes. [4]

**Exemple *:***

L'agent A veut informer l'agent B du temps qu'il fera demain,selon ses prévisions :

(inform

:sender A
:receiver B
:content temps (demain, pleuvoir)
:language Prolog
)

## 3 - Les agents mobiles :

### 3.1 Définition :

Un agent mobile est une entité autonome qui se déplace d'une machine à l'autre sur le réseau, sans perdre son code ni son état. C'est l'environnement d'exécution qui se charge d'assurer cette fonctionnalité. Il permet la création et la migration d'un agent, la communication et l'échange de messages entre les agents mobiles et il assure la sécurité de l'agent et de son site d'accueil.

### 3.2 Caractéristiques d'un agent mobile :

- **Mobilité** : la capacité d'un agent de se déplacer dans un réseau informatique.
- **Autonomie**: les agents opèrent sans intervention directe d'un être humain ou autre, et ont un certain contrôle sur leurs actions et leur état interne.
- **Réactivité** : les agents perçoivent leurs environnements et répondent aux changements qui apparaissent.
- **Comportement intentionnel (la proactivité)** : les agents sont capables d'avoir un comportement dirigé vers un but et de prendre des initiatives.
- **Comportement social** : les agents interagissent avec d'autres agents (éventuellement humains) via une sorte de "langage de communication agent".
- **Rationalité** : la conjecture selon laquelle un agent agira de façon à atteindre ses objectifs au moins dans la limite de ses convictions.
- **Véracité** : la conjecture selon laquelle un agent ne communique pas de mauvaises informations sans le savoir. [5]

### 3.3 La structure d'un agent mobile :

Chaque agent mobile possède cinq attributs :

- **L'état** : considéré comme une image instantanée de son exécution . Quand un agent voyage, il transporte avec lui son état, ceci lui permet de reprendre son exécution quand il a arrivé à destination.
- **L'implémentation** : comme n'importe quel autre programme, l'agent mobile a besoin d'un code pour pouvoir s'exécuter.
- **L'interface** : un agent fournit une interface qui permet aux autres agents et autres systèmes d'interagir avec lui.
- **L'identifiant** : chaque agent possède un identifiant unique durant son cycle de vie, qui lui permet d'être identifié et localisé .
- **L'autorité** : une autorité est une entité dont l'identité peut être authentifie par n'importe quel système auquel elle essaye d'accéder. [5]

### 3.4 Pourquoi utiliser des agents mobiles :

L'utilisation des agents mobiles a des beaucoup mieux importances soit dans le domaine de réseau ou se trouve la latence et le débit

S'affranchir du réseau :

- ✓ La Latence, débit.
- ✓ Mode déconnecté.
- ✓ Permet d'encapsuler des protocoles.

Aussi concernant le fonctionnement d'asynchrone qui :

- ✓ Privilégie les interactions locales.
- ✓ Permet l'installation d'interface locale spécifique.
- ✓ Permet de s'affranchir de l'hétérogénéité.
- ✓ En ayant un modèle robuste.

### 3.5 Fonctionnement d'un agent mobile:

Un agent est composé de son code correspondant à un algorithme, ainsi que d'un contexte incluant des données. Ce contexte peut évoluer en cours d'exécution.

Le code et le contexte de l'agent sont déplacés avec l'agent lorsque celui-ci visite différents serveurs.

Lorsqu'un agent se déplace vers un serveur, il doit poursuivre son exécution sur le site destination. La plupart des systèmes à agents mobiles implantent une migration faible, c'est-à-dire une fonction de migration où l'agent redémarre son exécution depuis le début. En conséquence, le programmeur doit inclure dans le contexte de l'agent des informations sur l'état de l'exécution, et lorsque l'agent redémarre sur un site, le code de l'agent doit vérifier l'état de l'exécution et se brancher sur la partie de l'algorithme devant être exécutée sur ce site.

Un système à agents fournit en général des primitives de communication permettant aux agents d'interagir entre eux, mais aussi aux agents d'interagir avec les serveurs qu'ils visitent. Ces primitives de communication prennent la forme d'envois de messages ou d'appels de procédures ou de méthodes. [6]

### 3.6 Principe de fonctionnement :

- **Dans le coté système** :

1. Ne reste pas sur la station ou il a été créé.
2. Peut se transporter (migrer) au travers du réseau vers une autre station, disparaît du site de départ après migration.
3. Peut emporter un état lors de la migration.
4. Est capable de communiquer avec d'autres agents, avec les stations visitées.

- **Dans le coté d'utilisateur :**

1. Est autonome.
2. Peut raisonner et prendre des initiatives.
3. S'adapte à un environnement changeant.

### 3.7 Les plateformes de développement des agents mobiles :

Le meilleur moyen pour construire un système multi-agent (SMA) est d'utiliser une plate-forme multi-agents qui un ensemble d'outils nécessaires à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test du SMA ainsi créé.

Ces outils peuvent être sous la forme d'environnement de programmation (API) et d'applications permettant d'aider le développeur.

Nous allons étudier dans cette partie la plate-forme JADE (Java Agent Development framework).

### 3.8 La migration :

Est un mouvement d'une entité en cours d'exécution sur une machine distante , la migration de processus utilisé pour la répartition de charge, la tolérance aux pannes, le partage d'information , ce mécanisme est puissant mais complexe à mettre en œuvre . [7]

**Migration d'un agent :**

La migration permet le transfert d'un agent en cours d'exécution d'un site à un autre à travers le réseau. Nous allons prendre le cas d'un agent qui désire migrer entre deux machines avec la possibilité de recevoir des messages pendant son déplacement, Cette migration comporte les étapes suivantes :

1. la sérialisation du contexte de l'agent et la production d'un message incluant le contexte sérialisé de l'agent et son code ;

2. le processus de l'agent étant suspendu sur sa machine d'origine, toutes les communications avec d'autres agents sont donc suspendues et l'agent est détruit ;

3. l'envoi du contexte et du code de l'agent vers la machine de destination ;

4. l'état de l'agent migrant est restauré sur la machine de destination, un nouveau processus léger est créé pour la poursuite de son exécution ;

5. les communications en cours sont redirigées vers la machine de destination, l'agent n'étant pas encore activé, ces messages seront traités après sa réactivation ;

6. réactivation de l'agent sur la machine de destination, dès que la partie de son contexte nécessaire à son exécution est reçue ; la machine de destination s'occupe des liens dynamiques entre l'agent et son code. À partir de ce moment, la migration prend fin. Le processus sur la machine d'origine peut être définitivement effacé. [7]

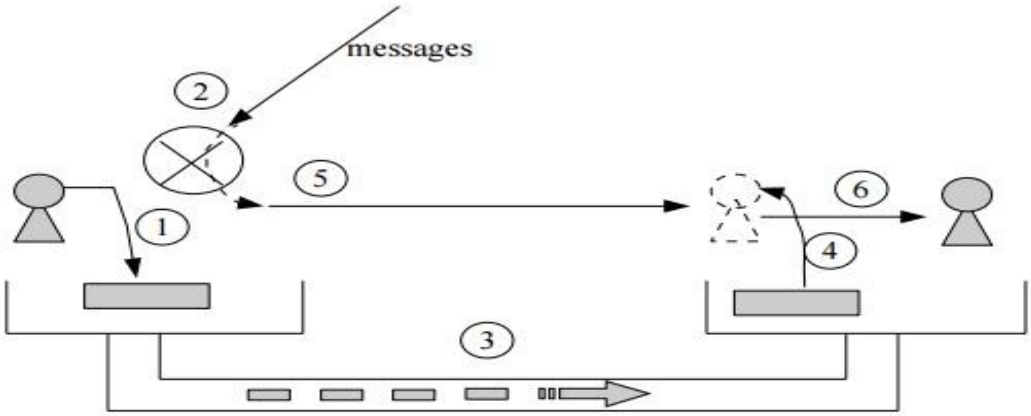



*Figure 1.4 : Migration d'un agent mobile*

Un schéma qui représente la migration d'un agent mobile d'une station a une autre.

1. **La migration forte** :

Permet à un agent de se déplacer quelque soit l'état d'exécution dans lequel il se trouve. Dans ce type de migration l'agent se déplace avec son code, son contexte d'exécution et ses données. Dans ce cas, l'agent reprend son exécution après la migration exactement là où elle était avant son déplacement. La migration forte nécessite un mécanisme de capture instantanée de l'état d'exécution de l'agent. Elle peut être proactive ou réactive. Dans la migration proactive la destination de l'agent est déterminée par l'agent lui-même. Dans la migration réactive la migration de l'agent est dictée par une partie ayant une relation avec l'agent mobile

2. **La migration faible :**

Ne fait que transférer avec l'agent son code et ses données. Sur le site de destination, l'agent redémarre son exécution depuis le début en appelant la méthode qui représente le point d'entrée de l'exécution de l'agent, et le contexte d'exécution de l'agent est réinitialisé. Pour que l'agent se branche sur une instruction particulière de son code après sa migration, le programmeur doit inclure dans l'état de l'agent des moments privilégiés (explicites) dans le code de l'agent (point

d'arrêt) pour pouvoir le relancer. La migration forte, bien que beaucoup plus exigeante à implanter que la migration faible [Grasshopper98], n'en est pas moins indispensable pour toutes les applications pour lesquelles les notions de fiabilité et de tolérance aux pannes sont primordiales. [7]

### 3.9 Cycle de vie et contrôle de l'agent :

Un environnement d'exécution pour les agents mobiles doit offrir à l'utilisateur la possibilité de contrôler les activités de l'agent. Depuis sa création jusqu'à sa terminaison, un agent peut passer par plusieurs étapes (cycle de vie d'un agent). Un langage de programmation d'agents mobiles doit offrir au programmeur des points d'entrée qui vont lui permettre de contrôler l'activité de son agent. Un agent passe par une partie ou la totalité des étapes suivantes :

1. **Création et initialisation**. Lors de sa création, l'agent peut être initialisé avec des informations nécessaires à son exécution telles qu'un itinéraire, des préférences de son utilisateur, etc. Par ailleurs, un agent est initialisé par le système avec des informations nécessaires à son interaction avec son environnement, comme par exemple l'identité de son utilisateur.

2. **Migration**. L'agent se déplace entre les machines du système. Le but de cette migration est souvent motivé par une coopération en local avec des agents fixes ou d'autres agents mobiles s'exécutant sur le même site. Avant de reprendre ses activités, l'agent aura besoin des informations sur le nouveau site.

3. **Activation et désactivation**. Dans une application basée sur les agents mobiles, un agent se désactive en suspendant son exécution afin d'être sauvegardé sur un support non volatile. L'agent mobile est donc gelé. L'activation est l'opération inverse par laquelle l'agent est restauré afin de continuer son exécution.

4. **Terminaison**. Une fois que sa tâche est réalisée, l'agent se termine et son processus d'exécution est tué. Avant sa terminaison l'agent a besoin de livrer un bilan à son utilisateur.

Souvent, un environnement d'agent mobile offre aux développeurs des méthodes relatives au cycle de vie d'un agent. Ces méthodes sont appelées des « call-back ». Un agent doit être informé chaque fois qu'un événement du cycle de vie commence, réussi ou échoue . Cela a

pour but non seulement de réagir aux événements en question mais aussi d'être capable de refuser une création, une migration ou une réinitialisation après un déplacement. [8]

# 4 .La Plateforme JADE :

## 4.1 Définition (Java Agent DEvelopment Framework) :

JADE est une plateforme de programmation multi-agents implémentée en Java. Les agents qui tournent sous JADE communiquent via le langage *Agent Communication Language* ou ACL.[9]

## 4.2 Brève description de JADE

JADE est une plate-forme multi-agents créée par le laboratoire TILAB. JADE permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA.

JADE est composée de trois parties :

- DF « Directory Facilitator » fournit un service de « pages jaunes» à la plate-forme .
- ACC « Agent Communication Channel » gère la communication entre les agents .
- AMS « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système. [9]

## 4.3 Applications de la plateforme :

- Les applications (applications nomades, agent de voyage personnel, applications de diffusion audiovisuelles, gestion de réseaux, assistant personnel…)
- Les architectures abstraites, définissant d'une manière générale les architectures d'agents

- Les langages d'interaction (ACL), les langages de contenu (comme SL, CCL, KIF ou RDF) et les protocoles d'interaction
- La gestion des agents (nommage, cycle de vie, description, mobilité, configuration)
- Le transport des messages : représentation (textuelle, binaire ou XML) des messages ACL, transport (par IIOP, WAP ou HTTP) de ces message

### 4.4 Langage de communication de la plate-forme JADE :

Les langages de communication de la plateforme jade sont deux :

- Le langage de Communication de la plate-forme JADE est FIPA-ACL (Agent Communication language).
- La classe ACLMessage représente les messages qui peuvent être échangés par les agents.

### 4.5 Format du message :

La communication de messages se fait en mode asynchrone. Lorsqu'un agent souhaite envoyer un message, il doit créer un nouvel objet ACLMessage, compléter ces champs avec des valeurs appropriées et enfin appeler la méthode send().

Lorsqu'un agent souhaite recevoir un message, il doit employer la méthode receive() ou la méthode blockingReceive()

### 4.6 Les primitives de JADE :

Une table qui représente quelques primitives nécessaires de plateforme JADE :

| | |
|---|---|
| Performative : | type de l'acte de communication |
| Sender : | Dénote le nom de l'agent qui envoie le message |
| Receiver : | Dénote le nom de l'agent qui reçoit le message |
| reply-to : | Elément qui le thread de conversation, au lieu du nom de l'agent |
| content : | Dénote le contenu du message : de l'objet ou de l'action |
| language : | Dénote le langage dans lequel le contenu est exprimé |
| encoding : | Dénote l'encodage de l'expression du contenu |
| ontology : | Dénote l'ontologie qui donne du sens à l'expression du contenu |
| protocol : | Dénote le protocole utilisé pour envoyer les messages ACL |
| conversation-id : | Introduit l'identificateur de conversation (expression) qui est utilisé pour la suite d'actes de communication qui constitue la conversation |
| reply-with : | Introduit l'expression qui est utilisé par l'agent répondant pour identifier ce message |
| in-reply-to : | Dénote l'expression qui fait référencer à l'action précédente au quel ce message répond |
| reply-by : | Dénote une expression du temps et/ou de la date qui indique le temps limite auquel l'agent envoyant désire recevoir une réponse |

**Exemple :**

L'agent A veut informer l'agent B du temps qu'il fera demain, selon ses prévisions

: (inform

:sender A

:receiver B

:content temps (demain, pleuvoir)

:language Prolog )

## 5. AGLET:

### 5.1 Définition :

AGLET est une plate-forme ou une bibliothèque d'agents mobiles Java qui facilitent le développement d'applications basées sur les agents. L'AGLET est un agent Java capable de se déplacer de manière autonome et expérimentée d'un hôte à l'autre. [6]

### 5.2 La description d' AGLET :

- L'architecture Multi-Agent contenant des agents mobiles administrés par une interface.
- Les agents peuvent être autonomes.
- La réactivité, il répond au message qu'il reçoit.
- Technologie basée sur Java, utilisant la sérialisation des objets
- Version actuelle : IBM Aglet version 2.2b
- Normes respectées en 2003 : Java 2 (incluant la sécurité built-in), MASIF émulé sans CORBA, KQML [6]

### 5.3 Les opération d'AGELT :

- **Dispatch** - Transférer l'agent vers un nouvel hôte (URL). L'agent est sérialisé, envoyé sur le réseau, reçu, et l'exécution est continuée.
- **retractAglet** - Retirer l'agent vers l'hôte précédent - demandé à partir du contexte. Même effet que l'envoi, mais en sens inverse.
- **MobilityListener** fournit des événements pour les deux méthodes.
- **Deactivate / activate**- Tous les threads Aglet sont supprimés et le Aglet dort pendant une période de temps spécifiée ou jusqu'à ce que le système soit activé par une autre Aglet ou entité système. L'Aglet est sérialisé et déplacé vers le stockage secondaire pendant ce temps. Lorsque l'Aglet est activé, l'exécution démarre en mode exécution.
- **PersistencyListener** fournit des événements pour la désactivation et l'activation

## 6.Conclusion :

Dans ce chapitre , nous avons fourni un aperçu complet sur les SMA et les agents mobiles et comment fonctionnent , Il visait plus précisément à proposer une solution sur le problème lié à la mobilité Ainsi, nous avons présenté la problématiques de migration ( mobilité des agents ) qu'elle était le truc de ce fonctionnement , Cela permettra à la division du travail , l'accélérer et d'assurer des résultats corrects ,elle Considérée comme l'une des techniques qui ont été venu par le système répartis .

Dans le prochain chapitre on va essayer de modéliser ce système et mettre une architecture générale , et visionner quelques codes et leurs résultats pour permettre au lecteur de visualiser les processus a travers la modélisation et les interface graphiques .

# CHAPITRE 02 :

# CONTRIBUTION ET IMPLEMENTATION

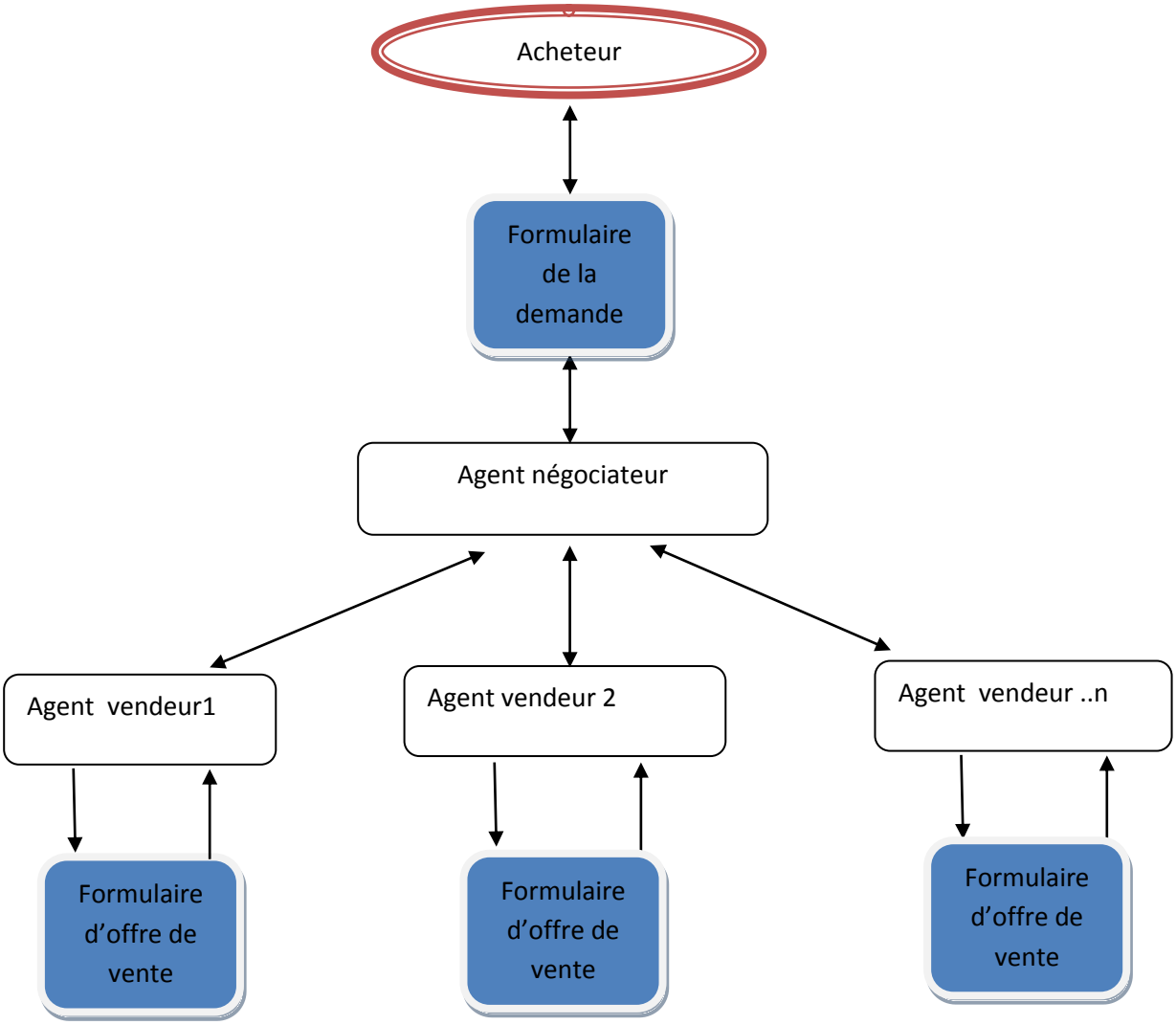
## 1. Introduction :

Dans ce chapitre nous voulons présenter la conception de notre système qui est nécessaire pour comprendre son mécanisme de fonctionnement, puisque elle donne une généralisation de ce travail, et établit un couplage explicite entre les concepts et les mécanismes internes.

Nous présentons ainsi les outils utilisés pour implémenter notre système et les résultats obtenus.

## 1. Architecture générale du système:

L'architecture générale de notre système est représentée dans la figure suivante :




Architecture générale du système

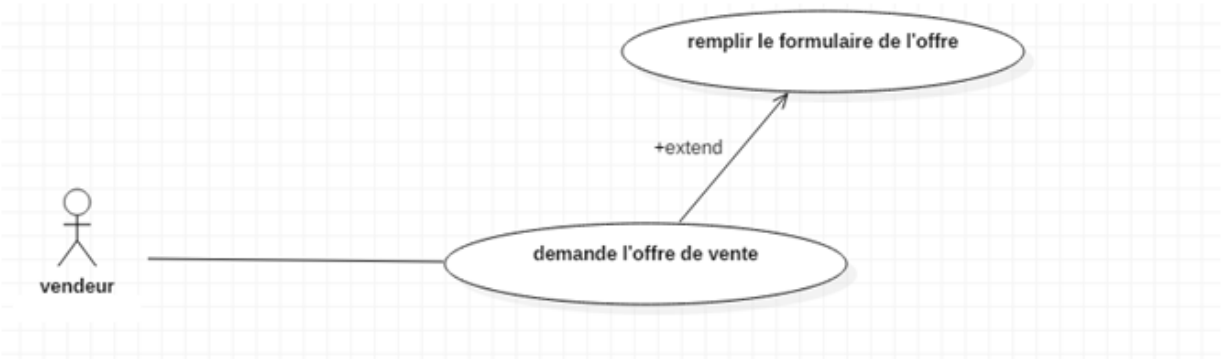
Un acheteur doit remplir un formulaire d'achat de livres, ensuite un agent négociateur se lance pour obtenir le titre du livre demandé par l'acheteur, ensuite cet agent lance une recherche sur les formulaires d'offre de vente fournis par les vendeurs pour détecter le meilleur prix.

## 2. La conception avec UML :

### 2.1 Diagramme de cas utilisation :

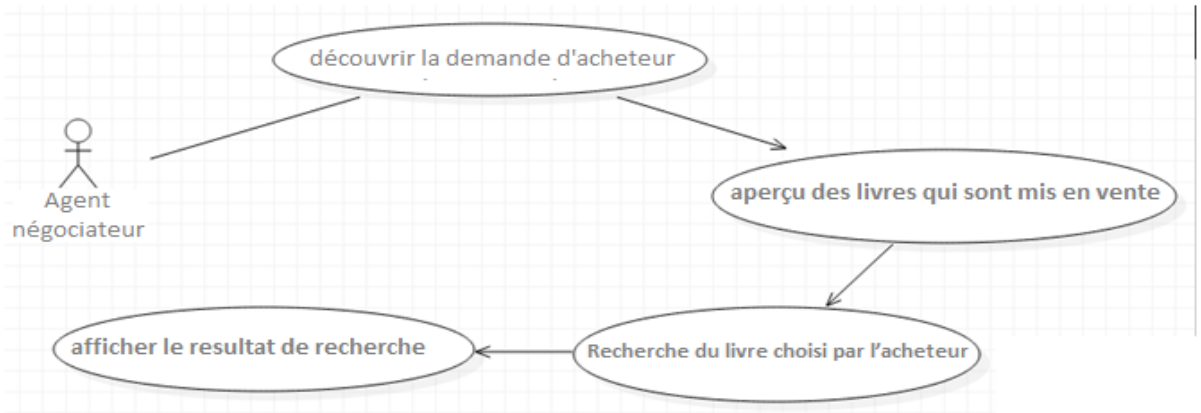
Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés.

#### 1.1.1. Diagramme de cas utilisation : (opération d'offre) :




Ce diagramme présente les étapes du processus d'offre , qui commence par la demande de vendeur pour faire l'offre de vente , et se termine par le remplissage du formulaire contenant des informations sur les livres qui veut les vendre.

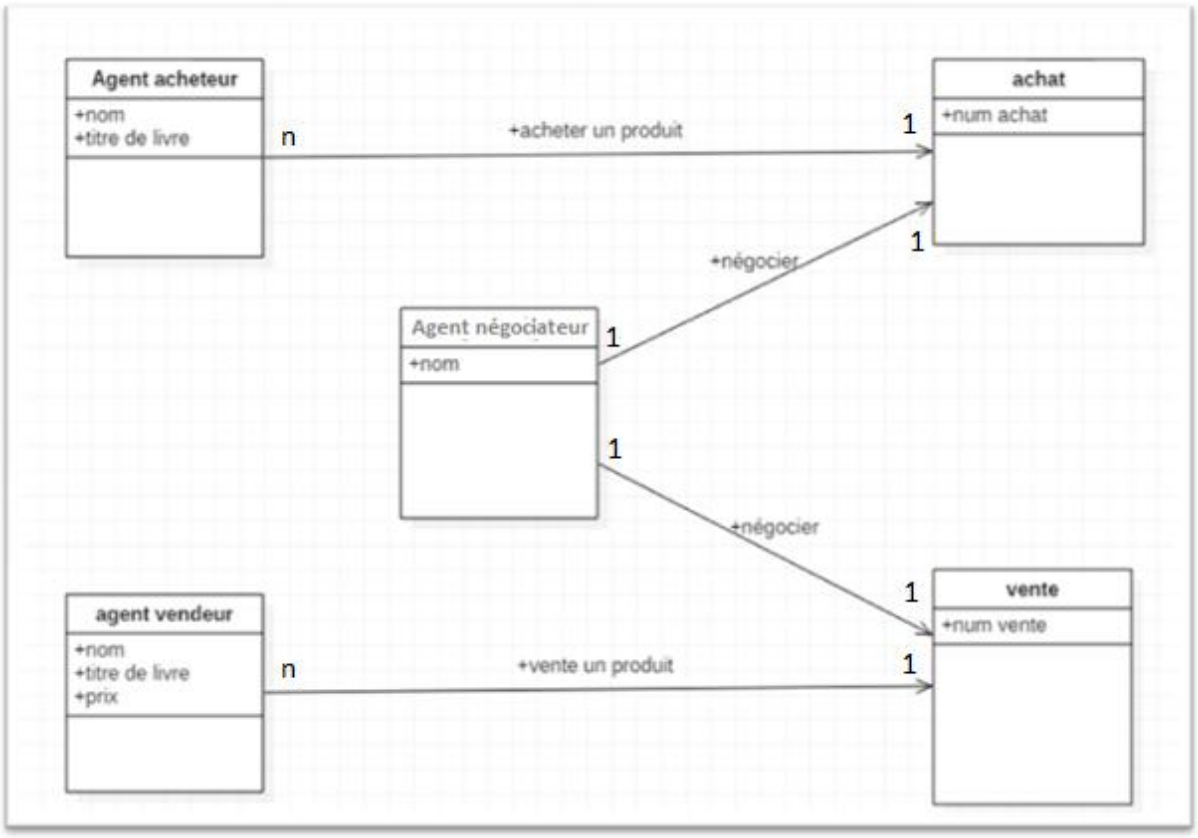
#### 1.1.2. Diagramme de cas utilisation : (opération de recherche) :




Ce diagramme présente le rôle d'agent négociateur qui est le responsable sur l'opération de connaitre le titre du livre , après il explorera les livres qui sont mis en vente et si il trouvera son choix il va ramener le livre de prix bas .

## 2.2 Diagramme de classe :

Le diagramme de classes est un schéma utilisé pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci .

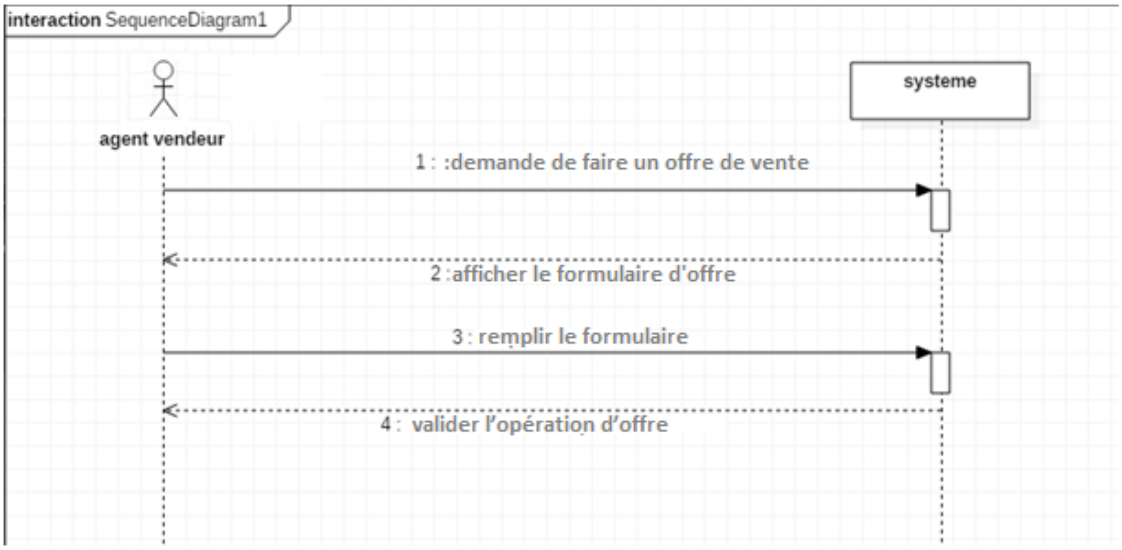



Le diagramme de classe contient 5 classes chacune a son rôle fondamentale, et toutes sont enchainées entre eux .

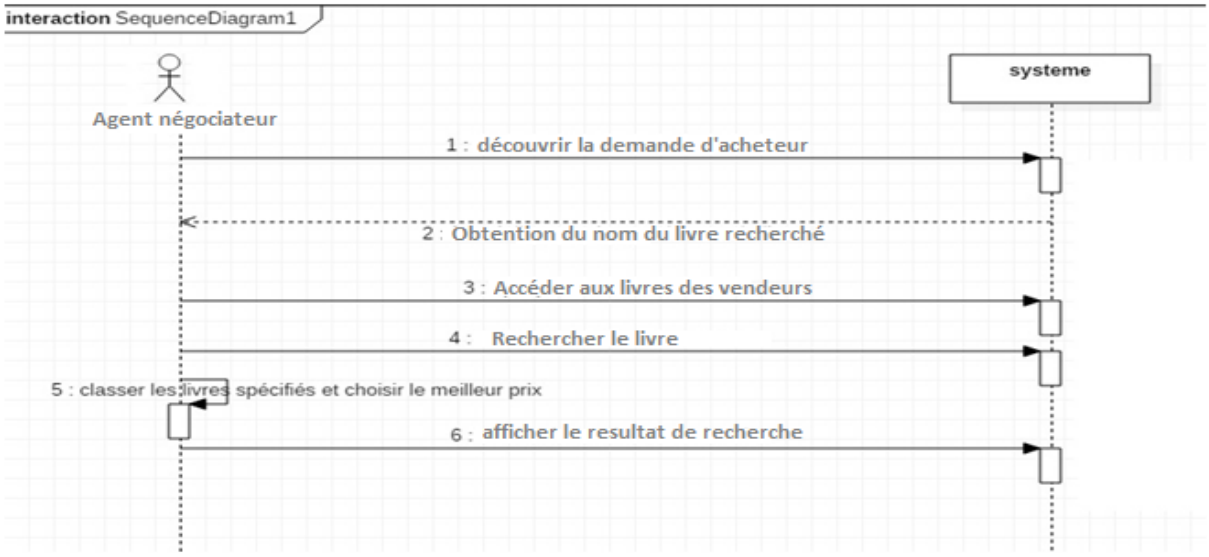
## 2.3 Les diagrammes de séquences :

Ces diagrammes sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML .

### 2.3.1 Diagramme de séquence (le processus du l'offre) :




### 2.3.2 Diagramme de séquence (processus de recherche du meilleur prix ) :



# 3. Les algorithmes utilisés :

## 3.1. Algorithme d'offre de vente :

**a. La recherche de meilleur prix.**

**Début:**

1. Formuler la demande d'un acheteur
2. Migration et lancement d'un agent mobile
3. traité la demande d'acheteur
4. accéder au livre des vendeurs
5. classer les livres spécifiés et choisir le meilleur prix
6. Obtenir le résultat
7. Affichage de résultat

**Fin**

**b. La migration**

**Début:**

1. Création des conteneurs
2. Création des agents
3. La migration des agents vers les conteneurs

**Fin**

# 4. Mise en œuvre du système :

## 4.1. Environnement de développement :

Pour implémenter notre système, nous avons utilisé le langage JAVA avec la plateforme JADE, qui permet de créer les systèmes multi-agents

### 4.1.1 Le langage JAVA :

Le langage Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

Java présentes certains avantages qui sont les suivants:

- Java a été conçu pour être facile à utiliser et est donc facile à écrire, à compiler, à déboguer et à apprendre que d'autres langages de programmation.
- Java est orienté objet. Cela nous permet de créer des programmes modulaires et des codes réutilisables.
- Java est indépendant de la plate-forme. [12]

### 4.1.2 La plateforme JADE :

La plateforme JADE est utilisée puisque elle permet de créer et de gérer les systèmes multi agents. Elle repose sur les principes suivants :

• Une plateforme est un ensemble de conteneurs actifs.

• La plateforme héberge un ensemble d'agents, identifiés de manière unique, pouvant communiquer de manière bidirectionnelle avec les autres agents.

• Chaque agent s'exécute dans un conteneur (container) qui lui fournit son environnement d'exécution ; il peut migrer à l'intérieur de la plateforme. [13]

• Toute plateforme doit avoir un conteneur principal qui enregistre les autres conteneurs.

### 4.1.3 Les contenants de plateforme JADE :

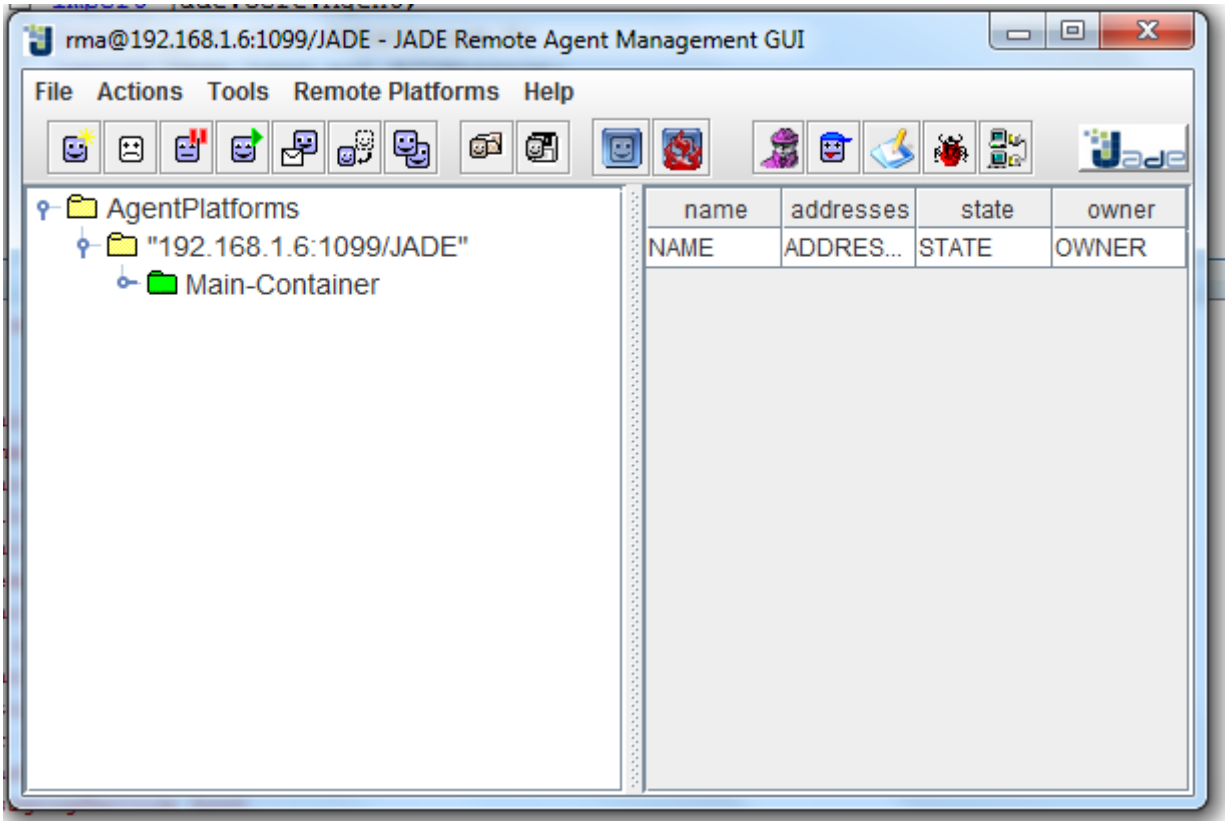
**1. Runtime Environment** : l'environnement où vivre les agent. Il doit être activé pour pouvoir lancer les agents.

**2. Une librairie de classes :** que les développeurs utilisent pour écrire leurs agents.

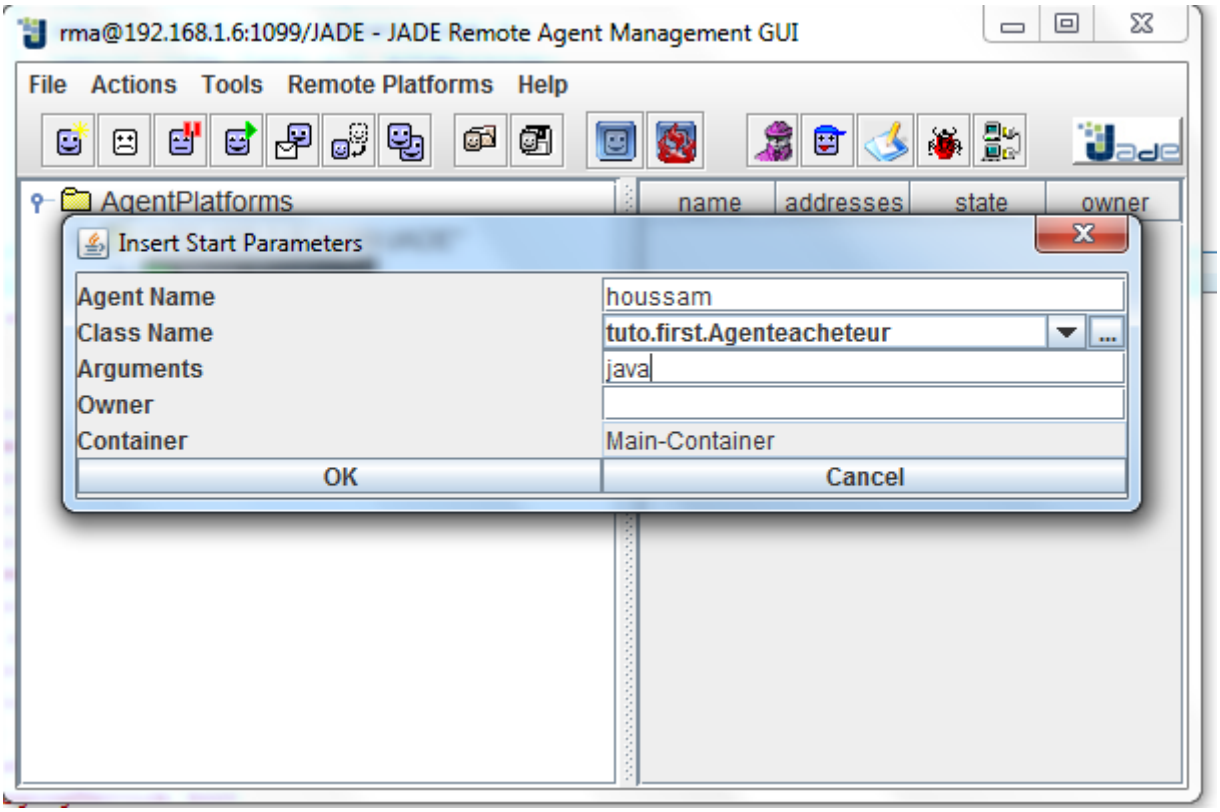
**3. Une suite d'outils graphiques:** qui facilitent le débogage, la gestion et la supervision de la plateforme des agents. [12]

## 5. Les résultats obtenus :

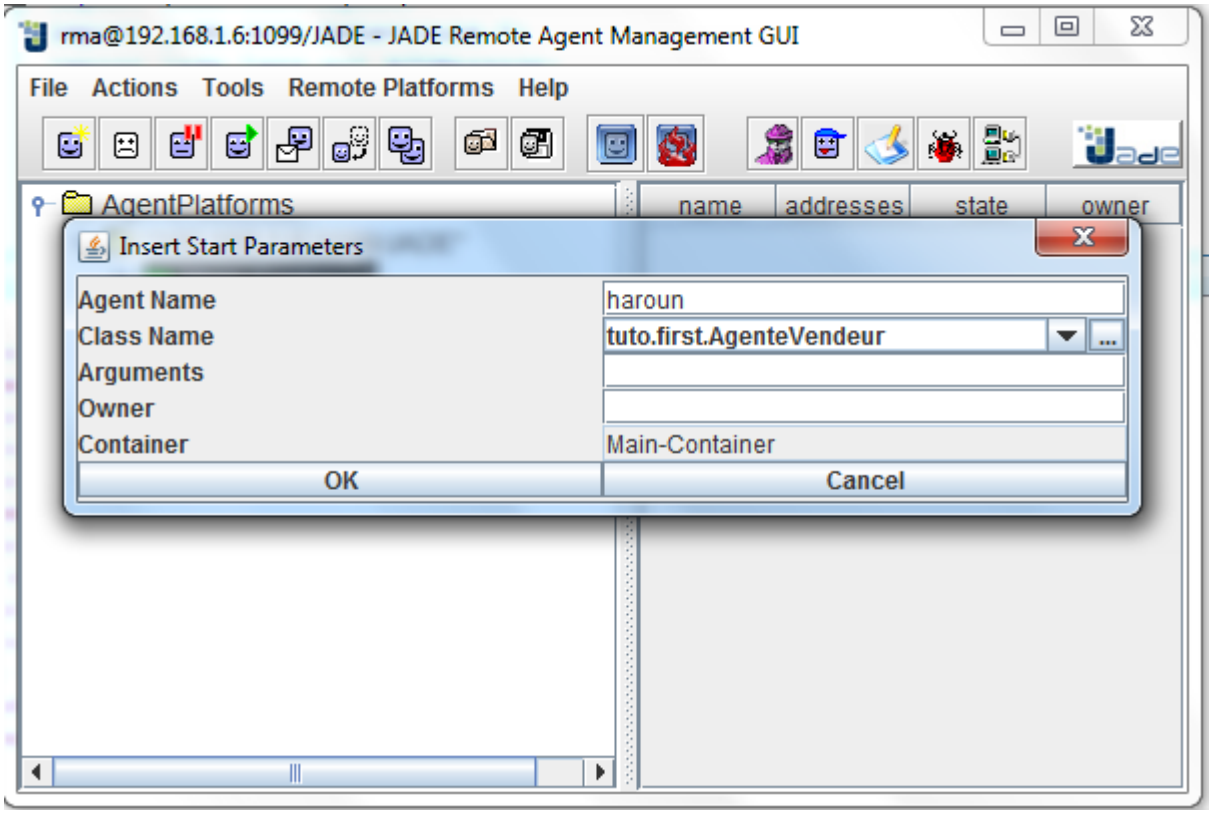
Les fenêtres suivantes représentent les résultats obtenus de l'implémentation de notre travail



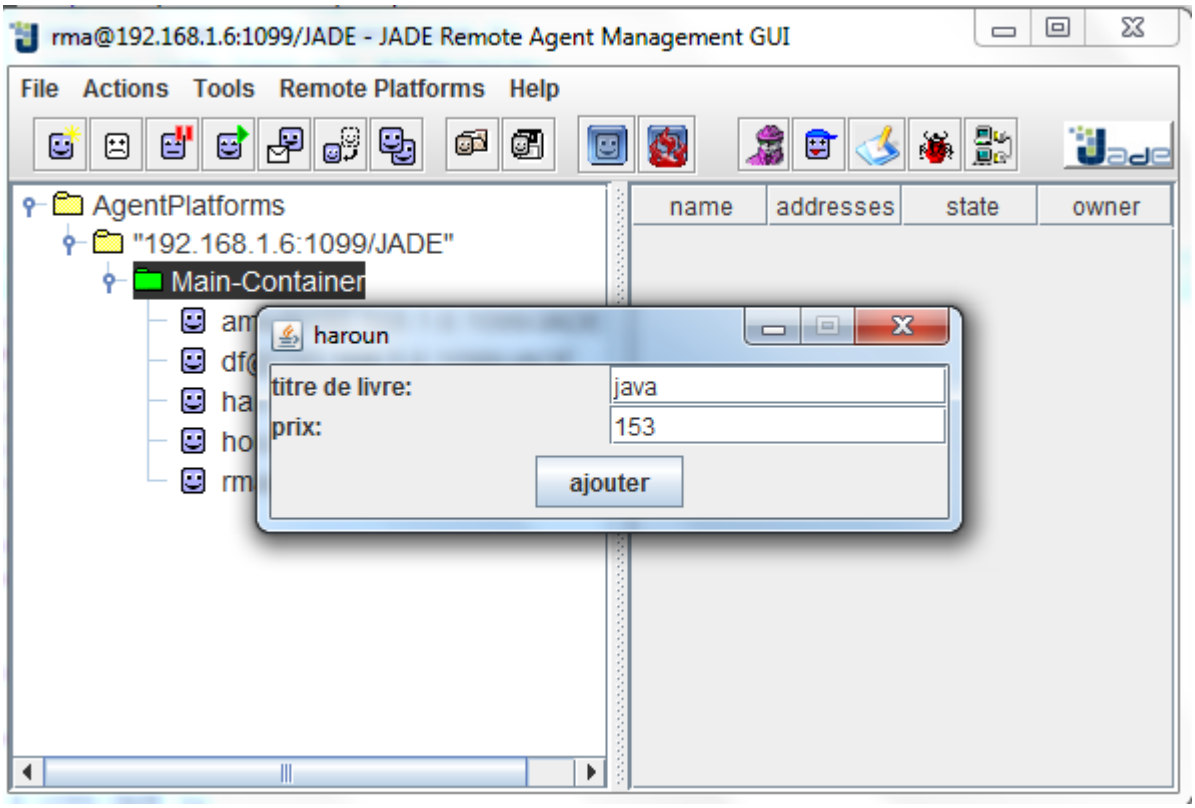

Le Démarrage de RMA JADE



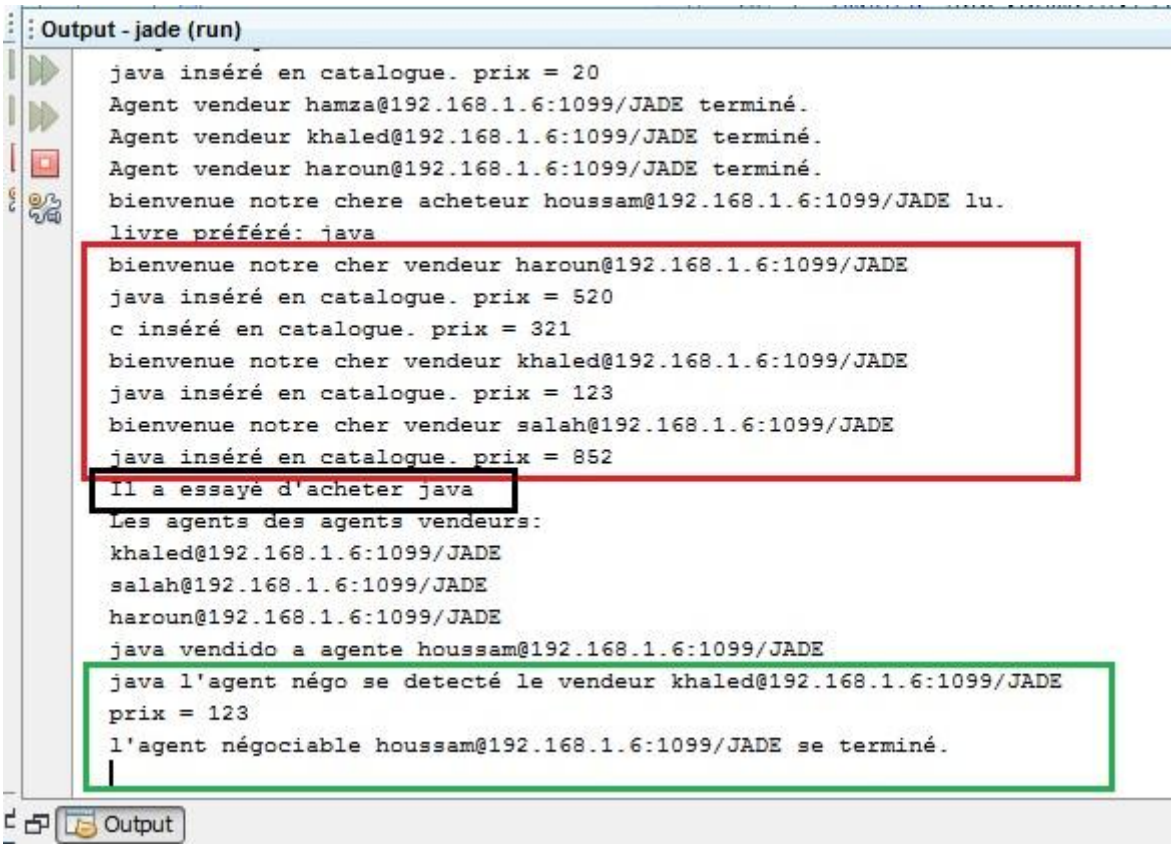

Le choix du livre à acheter




La déclaration de vendeur




Le formulaire de d'offre de vente d'un livre, contenant le titre et le prix.




Le cadre en rouge contient le  catalogue des livres à vendre.

Le cadre en noir contient la demande d'acheteur 'un livre JAVA'.

Le cadre en vert contient le résultat de négociation, l'agent négociateur ramène le livre java avec un prix plus bac '123'.

## Conclusion :

Dans ce chapitre, nous avons représenté l'architecture de notre système ainsi que son conception ,.qui nous a permis de bien comprendre la démarche à suivre pour la réalisation, Il semble très efficace de se tenir à utiliser UML car il prend en compte la plupart des problématiques de conception . Les outils utilisés ainsi que les résultats obtenus sont mis en fin

# Conclusion Générale :

Les négociations ont toujours été pour gagner la satisfaction d'acheteur et de vendeur que nous leur avons fait un point important de l'appliquer dans SMA qu'étant un système qui repose sur un certain nombre d'opérations effectuées en même temps entre le vendeur et l'acheteur et un médiateur entre eux, ce qui représente le rôle du système.

Le SMA a toujours été un médiateur qui facilite les processus de programmation, et surtout qu'il est conforme à l'utilisateur actuel qui est toujours voulu d'accélérer et de préciser les opérations sur tout avec la propriété de mobilité qui créer un espace virtuel commun pour permettre de migre a les autre machines sans déplacement réel.

Il n'y a jamais un travail complet donc il y a comme nous l'appelons , des perspectives d'avenir' qui sera lancé à partir de ce travail pour se rendre à d'autres développements dans une vision plus complète, on va travailler a l'ajout l'option de l'achat direct via Internet et de faire cette application connectée à Internet pour crée un espace commun de communication mutuelle, Afin de diffuser la culture de négociation organisée dans le monde entier.

# Bibliographie :


**Un ouvrage :**

[1] Nicole REGNIER, Une expérience de diffusion <intelligente > d'informations de tableau de bord qui généralise le concept de portail .

[2] C. SIBERTIN-BLANC, Introduction aux Systèmes Multi-Agents [2]

**Article :**

[3] Nicolas Dailly, Systèmes multi agents dans les EIAH

[4] Dr. Benmerzoug, Systèmes Multi agents, D. Département TLSI , université : Constantine .

**Mémoire :**

[5] Mr M. BENMOHAMED , Rapporteur : Mme Z. BOUFAÏDA, présenté en vue de l'obtention du diplôme de Magister en Informatique, Université Mentouri de Constantine.

[6] Leila Ismail, Agents mobiles et client/serveur : évaluation de performance et comparaison

[7] Systèmes Répartis – Agents Mobiles

[8] Salah El Falou, Programmation répartie, optimisation par agent mobile

[9] Chouchane Sahraoui Yacine Med Redha , Conception et réalisation d'un système multi-agents pour les enchères en ligne ,Université Larbi Ben M'Hidi Algérie

[10] philippe caillou, DÉVELOPPEMENT DE SMA

[11] Zina Mecibah , La génération des diagrammes AUML à partir d'un programme Jade Université Larbi Ben M'Hidi d'Oum El Bouaghi Algérie

[12] Jules GREGOIRE ,Mise en place d'un logiciel de gestion des missions : cas de la FCG (Fiducia Consulting Group) Sarl ,Université d'Abomey-Calavi au Bénin - Diplôme de technicien supérieur 2009

[13] Zina MECIBAH , La génération des diagrammes UML à partir de programme jade Université l'Arbi Ben Mhidi d'Oum el Bouaghi -Algérie- - master informatique -Systèmes distribués- 2012

**Résumé** :

Ce travail s'inscrit dans le domaine commercial où les négociations des prix jouent un rôle très important. Notre but est de réaliser un système multi-agents simulant un ensemble de vendeurs présentant leurs livres avec les prix et un ensemble d'acheteurs visant à trouver un meilleur prix pour un livre choisi. Les agents ont la caractéristique de migration inter-plateformes .Nous avons utilisé pour cela la plateforme JADE qui se base sur le langage de java .

**Abstract**:

This work is part of the commercial field ,where price negotiations play a very important role. Our goal is to realize a multi-agent system simulating a set of sellers presenting their books with prices, and a set of buyers aiming to find a better price for a chosen book. The agents have the characteristic of cross-platform migration. We used for this the platform JADE which is based on the language of java .

**الملخص** :

إن هذا العمل يندرج أساسا في المجال التجاري و المعاملاتي ، حيث تلعب المفاوضات دورا هاما للغاية في العملية التجارية . و هدفنا مجملا هو تحقيق نظام متعدد الوكلاء يعمل على خلق محاكاة لمجموعة من البائعين لعرض كتبهم مع الأسعار المخصصة لها ،و كذلك مجموعة من المشترين الذين يسعون للحصول على كتابهم المفضل بأحسن سعر ممكن .

لقد استعملنا خاصية JADE التي تقوم على لغة جافا .