REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED KHEIDER –BISKRA

FACULTÉ DES SCIENCES EXACTES ET DES SCIENCES DE LA NATURE ET DE LA VIE

DÉPARTEMENT D'INFORMATIQUE

N° d'ordre :……

Série :…….. MEMOIRE

Présenté pour l'obtention du diplôme
De Magister en informatique

Option : data Mining et Multi Media

Une approche pour la planification dans les systèmes multi-agents

Proposé par : Pr. Okba Kazar
Réalisé par : AL_Mutawakel abdallah

Soutenu le / / devant le jury composé de:

Pr. Okba Kazar	**Prof.**	**Université de Biskra**	**Rapporteur**
Dr. Terissa Labib Sadek	**M.C.A**	**Université de Biskra**	**Président**
Pr. Rida Laouar.	**Prof.**	**Université de Tébessa**	**Examinateur**
Dr. Bennoui Hammadi	**M.C.A**	**Université de Biskra**	**Examinateur**

Année Universitaire : 2014/2015

Remerciements

Tout d'abord, je tiens à remercier Dieu tout puissant qui m'a donné la volonté, la patience et la santé pour élaborer ce travail.

Ensuite, mes premiers remerciements iront à mon encadreur Pr. **Kazar Okba**, professeur à l'Université Mohamed Kheider de Biskra, pour m'avoir soutenu durant tout mon mémoire. J'aimerai lui adresser toute ma profonde gratitude et je le remercie du fond du cœur pour toutes ses remarques et suggestions techniques, académiques et professionnelles. Il a toujours su me consacrer des moments de son temps, me guider, me conseiller, et me témoigner son soutien et sa confiance. Je n'oublierai jamais sa gentillesse et sa bonne humeur avec moi.

Je remercie particulièrement le docteur **Terissa Labib Sadek**, maître de conférences **A** à l'université de Biskra, qui ma fait l'honneur de présider ce jury. Je remercie également très fort le docteur **Bennoui Hammadi** maître de conférences **A** à l'université de Biskra et le Professeur **Laouar Rida** de l'université de Tébessa, d'avoir accepté de juger ce travail.

Je voudrais témoigner au chef de département d'informatique Dr. **Mohamed Chaouki Babahenini**, ma profonde gratitude pour m'avoir faciliter toutes les procédures administratives tout au long des années d'études.

Finalement, je garde une place toute particulière à tous les membres de ma famille résidente au Yémen. Je leur exprime toute ma profonde gratitude parce qu'ils m'ont constamment aidé, malgré la distance pour la plupart, par leur soutien moral et leurs encouragements pour achever mon travail.

Dédicace

*Pour mes **parents** qui me sont les plus nobles et les plus chers du monde, pour leur soutien moral et encouragement dans toute ma vie*

*Pour mon grand frère **Abdulqader** qui a été toujours mon premier exemple, et pour ses fils*

Pour mes chères sœurs, mes oncles, mes tantes et leurs enfants

J'essaierai toujours d'être à la hauteur de votre confiance

A toute ma famille, tous mes amis sans exception et tous ceux que J'aime et qui m'aiment

Je dédie ce modeste travail …

Almutawakil Abdullah

Résumé

Dans ce mémoire, nous avons présenté les différentes technologies nécessaires pour proposer une approche pour la planification décentralisée dans les systèmes multi-agents. Notre travail a mis l'emphase sur le problème de satisfaction de contraintes distribué (DCSP). Résoudre un tel problème revient à coordonner des agents de façon distribuée en se basant sur la communication et la négociation. À cet effet, nous avons étudié le problème de « La gestion des réservations de voyageurs au saison du Hadj en l'Algérie » comme un exemple de DCSP. Le problème de départ a été formalisé sous la forme d'un problème de satisfaction de contraintes distribué (DCSP pour Distributed Constraints Satisfaction Problem), enfin cette approche a été implémentée et testée avec un système multi-agents.

Abstarct

In this thesis, we have presented the various technologies required to propose an approach for decentralized planning in multi-agent systems. Our work has focused on the distributed constraint satisfaction problem (DCSP). To solve such a problem comes down to how to coordinate distributed agents based on communication and negotiation. To this end, we studied the problem of "Management traveler bookings Haj season in Algeria" as an example of DCSP, the initial problem was formalized as a distributed constraint satisfaction problem (DCSP for distributed Constraints Satisfaction Problem), and finally this approach has been implemented and tested with a multi-agent system.

ملخص

في هذه المذكرة قمنا بعرض مختلف التقنيات الضرورية من لاجل وضع مقاربه للتخطيط الموزع في الانظمة متعدده الوكلاء في هذا مثل هذا النوع يقودنا إلى التنسيق بين مختلف الوكلاء باعتماد (DCSP) العمل ركزنا على مشكلة تلبية الشروط الموزعـــــه الاتصالات والمفاوضات . لهذا درسنا حالة تنظيم حجوزات المسافرين للحج في الجزائر . انطلقنا في البداية من مشكلة تلبية الشروط الموزعه واخيرا هذه المقاربة حققت باستعمال الانظمه متعددة الوكلاء.

Sommaire

Liste des figures

Liste des Tableaux

Introduction générale

De nombreuses applications distribuées nécessitent des mécanismes de distribution et de partage de données. Ce besoin de décentralisation du contrôle ou administration des données soulève de nombreuses problématiques scientifiques allant de la modélisation d'outils informatiques en passant par des aspects de formalisation. De nombreux chercheurs tentent de répondre à ces besoins selon différentes facettes, par exemple la protection des données privées, la gestion des bases de données hétérogènes, le passage à l'échelle, la confiance des données transmises et reçues, etc. Depuis de quelques années, des chercheurs du domaine des systèmes multi-agent (SMA) essaient de proposer des solutions.

Les systèmes multi-agents (SMA) représentent un autre paradigme de l'informatique distribuée basée sur de multiples agents en interaction qui sont capables d'un comportement intelligent, les systèmes multi-agents sont souvent utilisés pour résoudre des problèmes en utilisant une approche décentralisée où plusieurs agents contribuent à la solution en coopérant l'un avec l'autre. Les études en SMA s'intéressent principalement aux mécanismes liés aux problématiques de coordination et d'interaction entre différentes entités autonomes.

La coordination inter-agents peut s'effectuer de différentes manières. Nous considérons, dans le travail présenté ici, que la coordination peut être définie comme un processus de recherche basé sur la résolution de DCSP (Distributed Constraint Solving Problem) [Yokoo, 2001]. Les DCSP sont situés à l'intersection des domaines des SMA et des CSP (Constraint Solving Problem).

Dans de tels contextes, la coordination multi-agent peut être vue comme la décomposition d'un problème en sous-problèmes, la résolution des sous-problèmes, les mécanismes d'échanges de résultats partiels ou la diffusion des solutions des sous-problèmes aux agents, jusqu'à l'obtention de la (des) solution(s) globale(s). Les raisonnements intra-agent et inter-agents sont basés sur un ensemble de relations entre différentes variables. La résolution du problème s'effectue grâce aux agents qui interagissent afin d'obtenir une solution globale à partir des solutions locales.

De nombreux algorithmes permettant la résolution de DCSP existent dans la littérature telle que Multi-ABT [Hirayama et al. 2000, Hirayama et al. 2004], Multi-AWC [Yokoo et Hirayama, 1998] et AFC [Meisels et Zivan, 2007]. Ces algorithmes présentent plusieurs limites :

- les agents n'interagissent pas complètement de manière asynchrone.
- les agents peuvent créer des agents virtuels pour chacune de leurs variables locales, augmentant ainsi considérablement la complexité du problème si les agents virtuels ne sont pas correctement gérés.

- des créations de liens entre les agents apparaissent au cours de la recherche.
- des nogoods coûteux en calculs sont utilisés par les agents.

Le travail présenté ici porte sur la proposition d'un système complet pour la résolution de DCSP multi-variables par agent, nommé **DCSPH** (**Distributed Constraint Solving Problem Hadj**) don't l'objectif est de prendre en compte les limites présentées ci-dessus.

Le chapitre 1 présentera de manière générale le concept d'agent et les systèmes multi-agents, nous y expliquons ce qu'est un agent, un SMA ainsi que les interactions entre eux. Puis la planification d'un SMA est détaillé. Nous présentons différentes approches de planification comme, planification centralisée pour les plans distribués, planification distribuée pour les plans centralisés, planification distribuée pour les plans distribués et les techniques de planification système multi-agents.

Les différents types de planification décentralisée comme, Planification décentralisée avec fusion de plans, Planification décentralisée par la satisfaction de contraintes distribués (nous nous focalisons sur les DCSP, domaine intersectant celui des SMA et des CSP), Planification décentralisée par recherche dans l'espace d'états distribuée, Planification adversarielle et la théorie des jeux, sont illustrés dans le deuxième chapitre.

Le troisième chapitre sera consacré à la présentation de notre proposition nommée **DCSPH** « **Distributed Constraint Solving Problem Hadj** » permettant de résoudre des DCSP multi-variables par agent. Nous présenterons les hypothèses et notations de cette proposition. Notre solution au premier problème est de définir des agents, chaque agent jouant le rôle d'une agence pour rendre la coopération entre les agences plus facile. Le deuxième point peut être résolu par la définition d'un protocole de négociation et communication efficace entre les agents. Le troisième point oblige ce protocole de permettre la négociation les agents des différentes régions. Le quatrième point est résolu par l'utilisation d'un SMA de planification décentralisée pour les raisons mentionnées précédemment.

Le quatrième chapitre, nous poursuivons avec l'étude de cas afin de valider notre proposition, car elle est considérée comme un exemple typique pour un problème de DCSP. Nous précisons ensuite la plate-forme AGLET, différents composants de notre architecture, spécification d'agents, interaction d'agents, … etc. Enfin, nous terminons ce mémoire par une conclusion générale et un ensemble de perspectives envisagée.

Chapitre I

Systèmes Multi-Agents

I.1 Introduction

L'intelligence Artificielle Distribuée (IAD) est une discipline relativement nouvelle du champ de recherche Intelligence Artificielle (IA).

Il y a deux branches de l'IAD: résolution de problème distribué (DPS) et systèmes multi-agents systèmes (SMA).

En effet l'intelligence artificielle distribuée combine l'approche de l'I.A avec l'étude et la construction d'entités autonomes interagissant entre elles et avec leur environnement. Elle considère que la solution d'un problème est construite par la collaboration ou la compétition d'un groupe d'entités distribuées.

La technologie des agents fait partie du domaine de l'intelligence artificielle distribuée qui stipule que l'intelligence et l'expertise sont distribuées dans un groupe d'entités autonomes, en interactions, qui travaillent pour résoudre un problème, et en bénéficiant d'autres disciplines comme la sociologie, la psychologie sociale et les sciences cognitives.

L'apparition du paradigme agent donne une nouvelle vue pour le développement des systèmes de nature complexe, hétérogène, distribué et/ou autonome.

La planification et la coordination des plans d'agent sont deux des aspects les plus importants qui doivent être considérés par le concepteur d'un système multi-agent.

La coordination est un processus par lequel un ou plusieurs agent raisonnent sur leurs actions locales et sur les actions des autres (par anticipation) afin d'assurer la cohérence des actions communes, la planification en intelligence artificielle consiste à sélectionner et à ordonnancer des actions permettant d'atteindre un but donné à partir d'une base de connaissances sur les actions possibles. Cette dernière contient des préconditions (ce qui doit être vrai avant d'appliquer l'action) et des effets (ce qui arrive après), juste comme dans la planification classique, il n'est pas seul approprié algorithme de planification pour tous les systèmes d'agents.

Beaucoup de méthodes de planification sont décrites en littérature multi-agents, mais chacun d'eux est efficace pour des parties et inefficace pour les autres classes de systèmes d'agents ou des problèmes à résoudre.

I.2 Le concept de l'agent

I.2.1 Définition

Un agent est une entité réelle ou virtuelle, dont le comportement est autonome, évoluant dans un environnement, qu'il est capable de percevoir, sur lequel il est capable d'agir et d'interagir avec les autres agents [**1**].

Selon Jacques Ferber l'agent est : " une entité autonome physique ou abstraite qui est capable d'agir sur elle même et sur son environnement, qui 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**].

A partir de cette définition, on comprend que l'agent est une entité située dans un environnement comprenant des objets passifs, et d'autres agents où chacun cherche à satisfaire ses propres objectifs, pour cela il exécute un ensemble de tâches qu'il voit adéquates suivant sa situation courantes et suivant son répertoire d'action et le but qu'il désire atteindre.

Le processus de fonctionnement de l'agent comprend donc trois phases successives :

- Une phase de perception qui permet d'élaborer une idée sur l'état actuel de l'environnement via un ensemble de capteurs.
- Une phase de délibération qui permet de décider quelle action à exécuter suivant l'état de l'environnement et l'état interne de l'agent, c'est à ce niveau que le comportement de l'agent est décrit donc c'est la phase la plus importante.
- Et finalement la phase d'action c'est l'exécution de l'action par l'actionneur correspondant, l'exécution qui va apporter des modifications sur l'état de l'environnement et sur l'état de l'agent.

I.2.2 Caractéristique

Les différentes caractéristiques de l'entité agent peuvent être résumées dans la définition suivante : « un agent est un système informatique situé dans un environnement, et qui agit de façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu » [**3**].

- **L'autonomie**

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; en d'autre terme, les agents sont dits autonomes dans le sens où le créateur du système ne pilote pas leur comportement.

C'est l'agent qui décide de ses actions par rapport à un éventail de possibilités données.

- **La situation**

Les agents sont situés dans un environnement contenant également des entités passives, manipulées par les agents (par exemple, des ressources, des données, des objets physiques…) et communément appelées objets.

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

- **La réflexibilité**

L'agent est capable de réaliser des actions d'une façon autonome et réflexible afin d'atteindre les objectifs qui lui ont été fixés, la réflexibilité signifie dans ce cas:

- **La réactivité**

L'agent est capable de percevoir les changements dans son environnements, et doit élaborer une réponse dans les temps requis (changer son état interne, exécuter un tâche particulière…etc.).

- **La pro-activité**

L'agent doit exhiber un comportement proactif et opportuniste, c'est-à dire qu'il n'agit pas uniquement en réponse à son environnement mais, il est également capable de prendre l'initiative au "bon" moment;

- **La sociabilité**

L'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

I.2.3 Classification des agents

Les agents sont généralement classés selon deux courants de penser : l'école réactive et l'école cognitive.

- **Les agents réactifs**

L'école « réactive» prétend au contraire qu'il n'est pas nécessaire que les agents soient intelligents individuellement pour que le système ait un comportement global intelligent. Les agents réactifs sont des agents simples qualifiés de non intelligents, ils répondent d'une façon opportune aux modifications de leurs environnements résultants des stimuli externes (leurs actions sont provoquées et non pas choisies), ils agissent en fonction de ces dernières sans nécessité de compréhension de leurs univers ni de leurs buts.

Les sociétés d'agents réactifs sont caractérisées par le nombre important des agents qui sont capables ensemble de produire des actions évoluées qu'on appelle l'émergence de l'intelligence, mais dont les individus pris séparément ne possèdent qu'une représentation faible de leur environnement et des buts globaux [**4**].

Il y a deux modèles qui peuvent servir à la conception d'agent réactif:

✓ Agents à réflexes simples

Ce type d'agent agit uniquement en se basant sur ses perceptions courantes. Il utilise un ensemble de règles prédéfinies, du type **Si** condition **alors** action, pour choisir ses actions **[5]**.

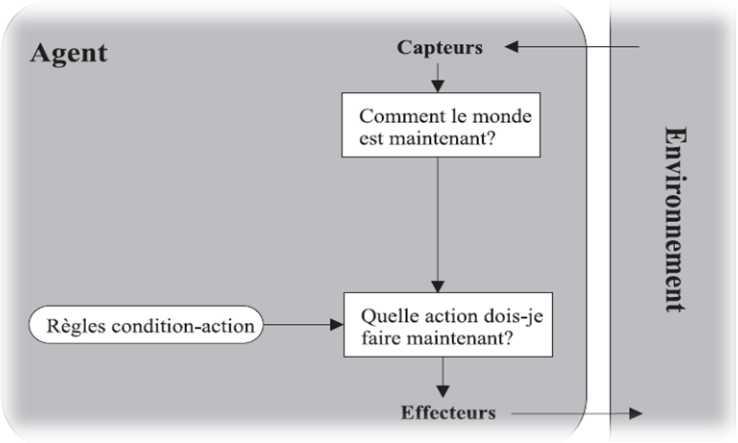


Figure I.1 : schéma d'un agent a reflex simple

✓ Agents conservant une trace du monde

Le type d'agent qui a été décrit précédemment, ne peut fonctionner que si un tel agent peut choisir ses actions en se basant uniquement sur sa perception actuelle [**5**].

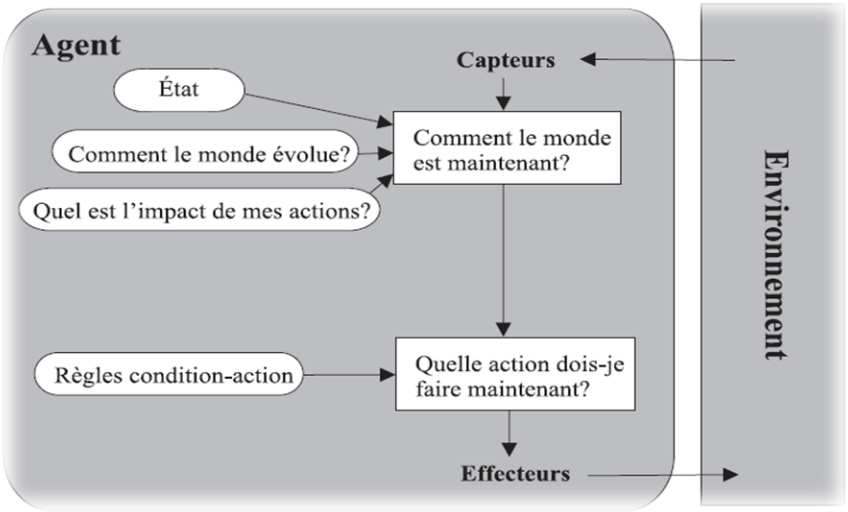


Figure I.2: schéma d'un agent conservant une trace du monde

- **Les agents cognitifs**

Ils sont parfois dits "intentionnels", leur caractéristique fondamentale est la volonté de communiquer et de coopérer et la mémorisation du passé. Ils possèdent des buts à atteindre à l'aide d'un plan explicite.

Les sociétés d'agents cognitifs contiennent communément un petit groupe d'individus qui sont des agents intelligents où chacun possèdera une base de connaissances comprenant l'ensemble des

informations et des savoir-faire nécessaires à la réalisation de sa tâche et à la gestion des interactions avec son environnement et les autres agents. Le système est régi par des règles sociales prédéfinies (c'est-à-dire lors de situations conflictuelles les agents seront amenés à négocier) [**4**].

Un agent cognitif raisonne sur ses croyances qui représentent sa compréhension du monde dans lequel il évolue. Il raisonne aussi sur ses désirs et intentions en relation avec ses croyances et capacités afin de prendre des décisions auxquelles sont associés les plans qu'il va accomplir pour agir dans le monde [**3**].

I.3 Système multi-agent

Plusieurs auteurs s'accordent généralement pour définir un système multi-agents (SMA) comme un système composé d'agents qui communiquent et collaborent pour achever des objectifs spécifiques personnels ou collectifs. En effet, les systèmes Multi-agents sont des systèmes distribués conçus idéalement comme un ensemble d'agents interagissant le plus souvent selon des modes de coopération, de coordination et de négociation et de coexistence [3] .

Une autre définition donne par Jacques Ferber [2] "les systèmes Multi-Agents comme des systèmes composés des éléments suivants:

- ✓ **Un environnement**

  C'est-à-dire un espace disposant généralement d'une métrique.

- ✓ **Un ensemble d'objets**

  Situés dans cet environnement cela signifie que pour tout objet il est possible à un moment donné d'associer une position, ces objets sont passifs, c'est-à-dire qu'ils peuvent être créés, détruits, manipulés et perçus par les agents.

- ✓ **Un ensemble d'agents**

  Ce sont des objets particuliers, ils représentent les entités actives du système.

- ✓ **Un ensemble de relations**

  Qui unissent les objets entre eux.

- ✓ **Un ensemble d'opérations**

  Ce sont les différents types de manipulation qu'appliquent les agents sur les objets du système et qui sont en générale: perception production, consommation, transformation…etc.

- ✓ **Un ensemble d'opérateurs**

Chargés de représenter l'application de ces opérations et la création du monde à cette tentative de modification, que l'on appellera les lois de l'univers. "

Les systèmes Multi-agents possèdent en plus des avantages traditionnels de la résolution distribuée des problèmes comme la modularité la rapidité et la fiabilité due à la redondance,

l'avantage de faire intervenir des schémas d'interaction sophistiqués qui incluent la coopération, la coordination et la négociation qui sont généralement assurés par une communication entre les différents participants.

I.3.1 Interaction dans un système Multi-Agents

Un système Multi-agents est un système cohérent et intelligent constitué de plusieurs agents qui se chargent de réaliser un but commun, mais cette cohérence et cette intelligence généralement ne vient pas de l'intelligence des agents qui le composent mais de leurs interactions.

L'interaction est définie comme une mise en relation dynamique entre deux ou plusieurs agents par le biais d'un ensemble de relations réciproques [**2**]. Les interactions ne sont pas seulement la conséquence des actions effectuées par plusieurs agents en même temps, mais aussi l'élément nécessaire à la constitution d'organisations sociales.

- ✓ **La coopération**

Chaque agent dispose des compétences qui lui permettent de résoudre certains problèmes ou effectuer certaines tâches, mais parfois ses capacités ne seront pas suffisantes devant des situations complexes, donc l'agent aura besoin de l'intervention des autres agents qui vont l'aider à faire évoluer le système vers ses objectifs. Donc la coopération est la participation de plusieurs agents pour satisfaire un but individuel ou commun.

- ✓ **La négociation**

Les buts des agents dans un système Multi-agents peuvent être incompatibles et leurs demandes sont parfois contradictoires, donc il faut trouver un moyen qui permet à chaque agent de poursuivre son travail, autrement dit les agents doivent négocier la solution.

On définit la négociation comme le processus d'améliorer les accords (en réduisant les inconsistances et l'incertitude) sur des points de vue communs ou des plans d'action grâce à l'échange structuré d'informations pertinentes [**6**].

- ✓ **La Coordination**

La coexistence des agents dans un environnement où les ressources sont limitées, et où les objectifs locaux des différents agents peuvent se contredire provoquant l'apparition de situations de conflits qui influencent sur le rendement global de tout le système, et ainsi diminuent les avantages de coopération. Pour vaincre ces situations de conflits et d'en sortir ou d'en éviter, les agents sont amenés à exécuter des actions supplémentaires (or que les actions productives), ces dernières sont dites actions coordinatrices. Le concept de coordination regroupe l'ensemble d'outils et méthodes qui peuvent être employés pour résoudre des conflits (dus aux ressources partagées limitées ou aux objectifs incompatibles) ou pour optimiser des comportements (éliminer des actions redondantes et inutiles) et plus généralement pour assurer un tout cohérent.

I.3.2 La Communication entre les agents

La communication est l'un des concepts pertinents dans les systèmes multi-agents. Comme chez les humains, la communication est à la base de l'interaction et de l'organisation sociale, elle permet aux agents de coopérer, négocier, échanger des informations et effectuer des tâches en commun. Sans communication, on distingue essentiellement deux modèles de communication :

- Communication par partage d'information.
- Communication par envoi de messages.

I.3.2.1 Communication par partage d'information

La communication entre les différents agents du système est réalisée par partage d'information lorsque ceux-ci disposent d'une zone de données commune dans laquelle ils rangent les conclusions qu'ils ont pu tirer. Outre ces résultats partiels, cette zone renferme les données du problème initial. Les agents peuvent ainsi y puiser les informations dont ils ont besoin pour résoudre une partie du problème globale. Ce type de communication correspond à ce que la littérature désigne communément sous le nom de modèle du blackboard (tableau noir).

I.3.2.2 Communication par envoi de messages

Les systèmes multi-agents fondés sur la communication par envoi de messages se caractérisent par le fait que chaque agent possède une représentation propre et locale de l'environnement qui l'entoure. Chaque agent va alors interroger les autres agents sur cet environnement ou leur envoyer des informations sur sa propre perception des choses. La communication se fait soit en mode point à point, soit en mode par diffusion [**7**].

I.3.3 Les avantages d'une approche multi-agent

Le système multi-agent présente les avantages suivants par rapport à un système d'un seul agent ou approche centralisée:

- Le système multi-agent distribue les ressources et les capacités de calcul à travers un réseau des agents interconnectés, considérant qu'un système centralisé peut être en proie à des limitations de ressources ou les pannes critiques,un système multi-agent est un système décentralisée et donc ne souffre pas d'une point unique de défaillance "single point of failure" problème associé à des systèmes centralisés.
- Un système multi-agent permet l'interconnexion et l'interopérabilité des multiples systèmes existants par construit un agent encapsuler.
- Un système multi-agent résoudre les problèmes en termes d'interaction autonomes des components des agents, qui s'avère être un moyen plus naturel de représenter l'allocation

des tâches, la planification d'équipe, les préférences des utilisateurs, les environnements ouverts, et ainsi de suite.

- Un système multi-agent récupère efficacement et filtrer des informations coordonnées à partir de sources qui sont spatialement distribués.
- Un système multi-agent propose des solutions dans des situations où l'expertise est spatialement et temporellement distribuée.
- Un système multi-agent améliore les performances globales du système, spécifiquement sur les dimensions de l'efficacité de calcul, la fiabilité, l'extensibilité, la robustesse, la maintenabilité, la réactivité, la flexibilité et la réutilisation.

I.4 Planification multi-agent

I.4.1 Définition

La planification est un domaine bien connu de l'intelligence artificielle, en général est une synthèse automatisée de séquences d'actions partiellement ordonnées, appelés plans, qui peuvent être exécutés dans des environnements donnés par un ou plusieurs agents, Un plan est appelé une solution pour un problème donné si son exécution à partir d'un état initial permet d'atteindre les objectifs finale de problème.

Planification multi-agent peut être vu comme une extension de la planification classique et dans [8] est définie comme «le problème de la planification par et pour un groupe d'agents", cette définition intentionnellement générale et comprend donc des nombreuses approches différentes.

La première distinction doit être établie entre les systèmes où il y a un seul planificateur pour tous les agents d'exécution et les systèmes où plusieurs agents sont autonomes, rationnel et ont les capacités de planification.

Généralement la première sont appelés planification centralisée tandis que ces derniers sont appelés planifications distribuée ou décentralisée. Planification multi-agent peut être appliquée à un grand nombre de problèmes.

Lorsqu'il existe des multiples acteurs qui opèrent dans environnement spécifie et ils ont besoin de décider la meilleure plan d'action à exécuter donc la planification multi-agent peut servir à trouver la solution.

Il est également à noter que la planification multi-agents n'est pas un nouveau domaine de recherche, mais de nombreuses contributions importantes à ce sujet sont tout à fait récente.

I.4.2 Problèmes de planification dans un système multi-agents

En général, un problème de planification multi-agent peut être défini comme un problème de planification par un groupe des agents. À l'exception des problèmes de planification centralisée, chaque agent dans un problème a effectivement un problème de planification individuelle et privée.

Un problème de planification individuelle d'un agent comprend un ensemble d'opérations qu'il peut effectuer, un ensemble d'objectifs et l'état actuel de cet agent. Alors la solution à un problème de planification multi-agents est un plan: une séquence partiellement ordonné d'actions qui, lorsqu'elles sont exécutées avec succès, conduit à un ensemble d'objectifs entendu par certains agents.

La plupart des techniques de la planification peuvent traiter les problèmes ou les actions et les objectifs des agents sont faiblement dépendants, où les agents sont coopératifs, et lorsque la communication est fiable. Mais en général les approches de la planification multi-agents peut rencontrent habituellement des différents situations.

I.4.3 Les approches de planification multi-agents

En général, la planification peut être considérée comme une tâche de résolution de problème par création un plan. La planification des tâches et les résultats de la planification peuvent constituer des sources de la distribution dans un système multi-agents.

Quand le processus de planification est répartir entre une multitude d'agents, on parle de constitution du plan distribué. Si le résultat du processus de planification (qui est un plan) est en outre distribué de l'exécution de plusieurs agents, on l'appelle un plan distribué.

Le terme de planification distribuée est utilisé pour décrire des situations où des agents participent en création du plan distribuer ou agir d'après d'un plan distribué, ou les deux situations.

Dans la partie suivante, nous allons discuter des techniques de planification distribuée.

I.4.3.1 Planification centralisée pour des plans distribués

La planification centralisée est utilisée quand il y a un seul agent qui devrait pour une raison quelconque (c'est le seul capable, ou avoir suffisamment d'informations) créer un plan. Ce plan doit être un ordre partiel afin de répartir entre des agents. ou bien l'agent qui a généré le plan « agent coordONATeur » est ensuite briser le plan en petits morceaux (threads) qui peut être éventuellement exécutées en parallèle et en ajoutant à chaque thread certaines actions de coordination en cas de besoin.

Les threads sont répartis entre les agents utilisant des techniques d'allocation des tâches et les agents agissent sur les sous-plans qu'ils reçoivent. En supposant que les agents suivent le plan strictement et ont une bonne connaissance des prédictions de l'entourage et leurs actions.

Evidemment, pour avoir efficacement travail de technique planification centralisée, il est plus important d'avoir une décomposition et une distribution des tâches plus efficace, alors il devient très difficile parce que la disponibilité des agents capables de réaliser un sous plan ne peut être garantie à tout moment, donc par conséquent plusieurs itérations de décomposition et distribution d'une tâche peut être nécessaire avant que les agents commencent réellement à suivre un plan distribué [**9**].

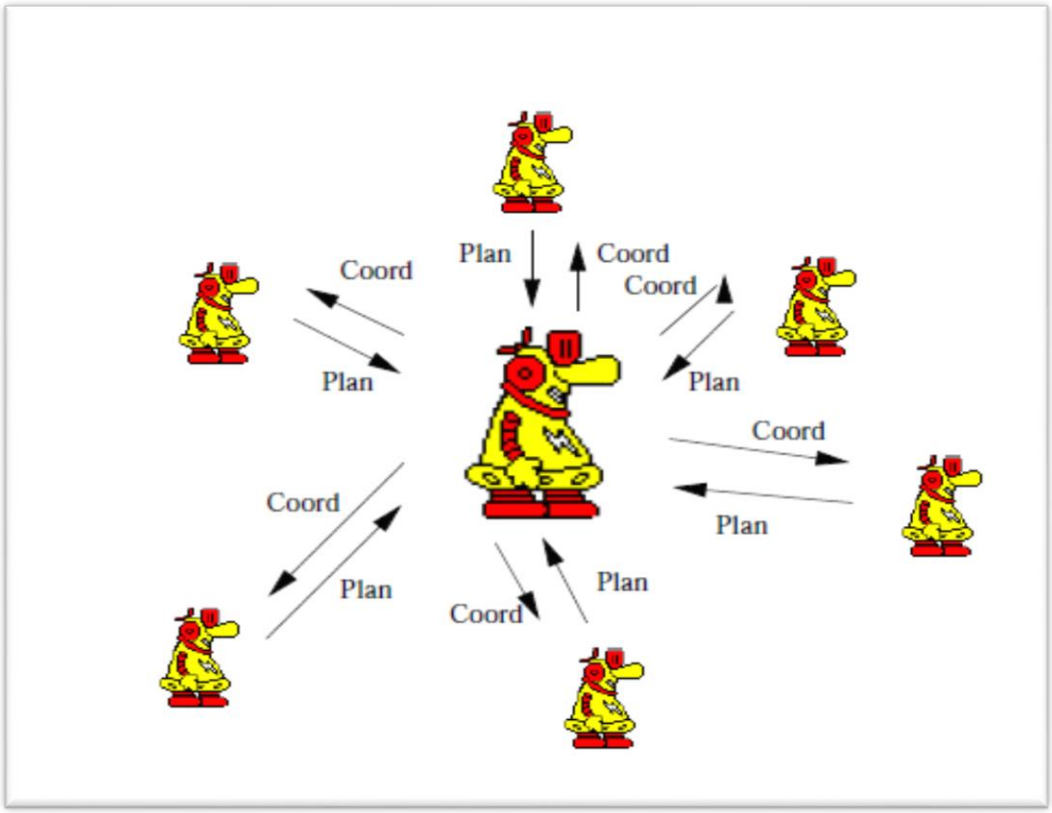


Figure I.3 : coordination centralisée pour plan distribuée [10].

I.4.3.2 Planification distribuée pour les Plans centralisés

La Planification distribuée peut avoir lieu même si le plan qui en découlera sera exécuté par un seul agent. La distribution peut être nécessaire en raison d'un manque de capacité ou de connaissances (spécialisation) de l'agent agissant ou pour des raisons d'efficacité.

C'est pourquoi la construction distribuée des plans centralisés peut être considéré comme une technique spéciale de résolution des problèmes distribués. Le partage des tâches et le partage des résultats peuvent être utilisés ici comme un moyen de coopération.

Au début, le problème (ou l'objectif) doit être décomposé et distribué entre les spécialistes agents participants de la planification en utilisant des méthodes de répartition des tâches. Alors la

coordination peut être réalisé soit par l'échange d'un spécifié planification «partiellement» ou par échange de résultats. Dans le premier cas, chaque planificateur essaie de modifier et d'étendre le plan partiel qu'il reçoit selon ses objectifs et il assure également le plan reste cohérent.

Si un planificateur ne peut pas d'augmenter le plan pour quelque raison ou il se rend compte que le plan est incompatible, une procédure de retour arrière est lancé pour permettre de trouver des plans alternatifs. Lorsque le partage de résultat est fait, les planificateurs essayé de générer des plans partiels en parallèle et ensuite les fusionner à former le plan controversé. Planification distribuée pour les plans centralisés est utilisé avec succès dans des domaines complexes tels que la fabrication, la logistique et la planification de mission pour les véhicules sans pilote [**11**].

I.4.3.3 Planification distribuée pour plans distribués

La configuration la plus complexe de la planification distribuée est celui où à la fois le processus de la planification et de l'exécution sont distribués. Le cas général de la planification dans un système multi-agents des agents de coopération, il peut apparaître qu'aucun des agents n'est conscient de tous les autres agents et leur comportement. Donc, il peut être impossible et non nécessaire pour tenter de générer un plan de multi-agent complet, contient toutes les actions de tous les agents du système.

D'un autre côté, il est préférable d'avoir aussi des conflits minimaux entre les agents des plans le plus possible et même de permettre la coopération par avoir des agents s'aident mutuellement dans leurs tâches (si il est possible et raisonnable). Donc, les agents doivent être conscients des plans distribués des autres et ils devraient être capables de résoudre les conflits et de trouver des inter-tâches relations [**9**]. Les techniques les plus populaires qui traitent de ces problèmes sont les suivants:

- ✓ Fusion du plan.
- ✓ Constitution du plan itératif.
- ✓ Négociation.

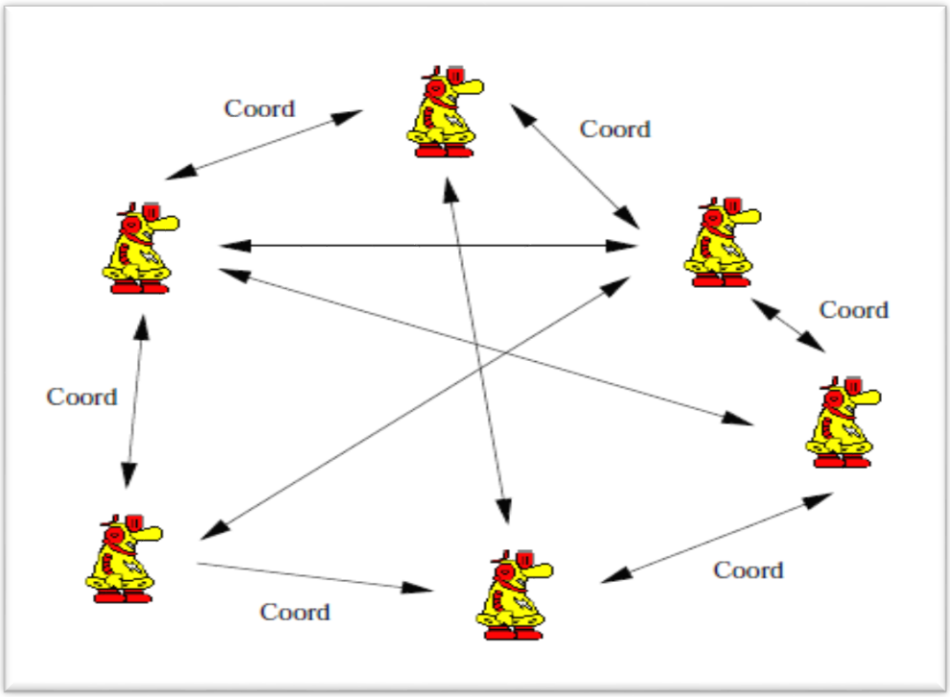


Figure I.4 : coordination distribuée pour plan distribuée [10].

I.4.4 Les techniques de planification multi-agents

Les techniques de planification multi-agents couvrent une gamme de solutions du problème aux différentes étapes. Dans cette section, nous structurons le travail existant qui utilisant des étapes dans le processus de résolution les problèmes dans la planification multi-agents. En général, les étapes suivantes peuvent être distinguées comme suit [**9**].

- **Etape 1**

  Affiner les objectifs ou les tâches globales même les sous-tâches peuvent être affectées aux agents individuels (tâche globale raffinement).

- **Etape 2**

  Allouer cet ensemble de sous-tâches pour les agents (allocation de tâche).

- **Etape 3**

  Définir des règles ou des contraintes pour les agents individuels pour les empêcher de produire des plans conflictuels (coordination avant la planification).

- **Etape 4**

  Pour chaque agent faites un plan pour réaliser ses objectifs (planification individuelle).

- **Etape 5**

  Coordonner les plans individuels des agents (coordination après la planification).

- **Etape 6**

  Exécuter les plans et synthétiser les résultats des sous-tâches (exécution du plan).

On peut dire que la planification est une combinaison des étapes 2 et 3, qui sont souvent entrelacées. Une de ces étapes pourrait être effectuée par un agent ou un sous ensemble, et pas toutes du ces étapes de ce processus de planification multi-agent doivent être intégrées. Par exemple, s'il n'y a pas communs ou globaux objectifs, il n'est pas nécessaire les étapes 1 et 2, et les conflits possibles peuvent être traitées directement (en étape 3) ou ultérieurement (en étape 5).

Aussi certaines approches se combinent différentes étapes, par exemple, les agents peuvent coordonner leurs plans lors de la construction leurs plans (combinaison de l'étape 4, et 5), ou de reporter la coordination jusqu'à la étape d'exécution (combinaison de l'étape 5 et 6), en général, tout entrelacement de ces six étapes peut être intéressant, selon le problème, qui indique une large variété de classes de problèmes possibles. Les sections suivantes décrivent certaines approches bien connues pour le traitement des enjeux qui se posent dans chacune des étapes.

I.4.4.1 Raffinement de Tâche globale

Dans la première étape, le global tâches ou objectifs sont raffinées de telle sorte que chaque tâche peut être faite par un seul agent. En dehors de techniques de planification d'un mono agent comme HTN (Ero 94), ou la planification non-linéaire (Pen 92), des techniques spécifiques ont été développés pour créer un plan multi-agent global. C'est ce qu'on appelle approches de planification multi-agent centralisée elle, utilise la structure de planification classique pour construire et exécuter des plans multi-agents.

I.4.4.2 Allocation des tâches

Les méthodes de planification multi-agents centralisés mentionnés avant prennent habituellement soin de l'affectation des tâches aux agents (étape 2). Il y a, cependant, de nombreuses autres méthodes pour établir une telle affectation de tâche de manière plus distribuée, donnant les agents un plus haut degré d'autonomie et de confidentialité,

Par exemple, les protocoles complexes d'allocation de tâches [**12**] ou des ventes aux enchères et les simulations du marché. Une enchère est une façon d'assurer que la tâche est attribuée à l'agent qui accorde la plus grande valeur pour lui [**13**]. L'enchère de Vickrey est un exemple d'un protocole de vente aux enchères qui est souvent utilisé dans les systèmes multi-agents.

Dans une enchère de Vickrey chaque agent peut faire une offre fermée, et la tâche est attribué au plus haut soumissionnaire pour le prix de la deuxième plus offrant. Ce protocole de vente aux enchères a une belle propriété que les agents soumissionnant devrait simplement soumissionner leurs vraies valeurs privées (par exemple, exactement ce qu'ils pensent que ça vaut pour eux).

Simulations du marché et de l'économie peuvent également être utilisées pour distribuer grandes quantités de ressources entre les agents. Ces méthodes non sont seulement utilisés pour affectation des tâches (étape 2), mais peuvent également être utilisés pour coordination les agents après la construction du plan (étape 5).

I.4.4.3 Coordination avant la planification

Dans l'étape 3, les agents sont coordonnés avant même de commencer la création de leurs plans. Ceci peut être réalisé, par exemple, en introduisant un droit social. Un droit social est en général une convention acceptée qui doit être suivre par chaque agent. Il peut être utilisés pour réduire les coûts de la communication et la planification et le temps de coordination. Exemples typiques de droit social dans le monde réel sont les règles de circulation: Parce que tout le monde conduit sur le côté droit de la route, pratiquement pas de coordination avec des voitures venant en sens inverse est nécessaire. En général, les solutions trouvées en utilisant le droit social ne sont pas optimales, mais ils peuvent être trouvés relativement rapides. Le droit social peut être créé dans la étape de conception d'un système multi-agents est étudiée par Shoham [**14**].

Briggs a proposé droit plus souple, où les agents essaient d'abord de planifier l'aide de la droit plus stricte, mais quand une solution ne peut être trouvée les agents sont autorisés à détendre un peu les droit [**15**].

Une autre façon de coordonner les agents est de comprendre les interdépendances exactes entre leurs tâches au préalable, contraintes préalables peuvent être traitées de manière centralisée via la technologie de planification existants (tels que la planification d'ordre partiel en regardant ces tâches comme des tâches mono-agent. Plus récemment, une approche a été proposée pour traiter des interférences (comme les ressources partagées) entre les objectifs d'un agent. [**16**]

La coordination avant la planification peut également être utilisée pour coordonner les agents compétitifs qui insistent sur leur autonomie de planification. Le problème est que nous avons un ensemble (sous) objectifs interdépendants qui doivent être atteint par un ensemble d'agents de planification, qui ne veulent pas être interféré au cours de leurs travaux de planification.

Autrement dit, chacun des agents requiert l'autonomie de la planification complète, mais en même temps, nous devons être sûr que tout ce (sous) plan qu'ils vont construire pour résoudre leur partie du problème, ces sous plans peuvent être coordonnées parfaitement sans nécessiter ré-planification.

Les problèmes de planification de ce genre se produisent souvent dans problèmes de transport multimodal, plusieurs parties doivent s'assurer que ces paquets sont transportés à partir de leur emplacement d'origine à leur destination, les agents de planification sont prêts à faire leur part du travail s'il peut être garanti qu'ils ne soient pas gênés par les activités des autres agents. Il est clair

que la plupart de ces problèmes de planification ne peut être décomposé en sous-problèmes indépendant sans changer le problème de la planification initiale.

I.4.4.4 La planification individuelle

La quatrième étape compose d'une planification individuelle de chacun des agents. Le principe, n'importe quelle technique de planification peut être utilisée ici, et aussi les différents agents peuvent même utiliser d'autres techniques.

Il y a une certaine approche qui intègre la planification (étape 4) et la coordination des plans (de l'étape 3 et 5). Dans la planification partielle chaque agent a une conception partielle des plans et des autres agents utilisant représentation de plan spécialisé [**17**]. Dans cette méthode, la coordination est réalisée comme suit, si un agent A informé un autre agent B d'une partie du son propre plan, B fusionne ces informations dans son propre plan partiel.

L'agent B peut ensuite essayer d'améliorer le plan global, par exemple, en éliminant les redondances qu'il observe. Un tel plan amélioré est affiché pour les autres agents, qui pourraient accepter, rejeter ou modifier.

Decker a utilisé une structure pour l'analyse des tâches, environnement de modélisation, et simulation pour modéliser un environnement multi-agents [**18**]. Un aperçu des approches de PGP est donné en [**19**]. Mais Clément et Barrett ont amélioré ce contexte de PGP en séparant l'algorithme de planification du la coordination des actions, en utilisant une approche plus modulaire appelé partagée activités (SHAC) [**20**].

I.4.4.5 Coordination après la planification

Un grand nombre de recherches ont porté sur ce qu'il faut faire après les plans ont été construits séparément (l'étape 5). Ces méthodes de fusion du plan visent à la construction d'un plan conjoint pour un ensemble d'agents donné sous plans de chacun des agents participant. Georgeff a été l'un des premiers à proposer un processus de plan de synchronisation à partir de plans individuels. [**21**]

Il a défini un modèle de processus pour formaliser les actions ouvertes à un agent. Parties de ce modèle sont les conditions de régularité, qui sont définis sur l'état du monde et doit être valide avant l'exécution du plan.

Deux agents peuvent s'aider mutuellement en modifiant l'état du monde de manière à ce que les conditions de régularité de l'autre agent deviennent satisfaites. Bien sûr, changer l'état du monde peut aider un agent, mais il peut également interférer avec les conditions de régularité d'un autre agent [**22**]. Stuart utilise une logique temporelle propositionnelle pour spécifier des contraintes

concernant les plans, de telle sorte qu'il est assuré que seuls les états possibles de l'environnement peuvent être atteints [**23**].

Ces contraintes sont données à un éprouveur de théorèmes pour générer des séquences d'actions de communication qui garantit aucun cas échouer. À la fois améliorer l'efficacité et résoudre les conflits, on peut introduire des restrictions concernant les plans individuels (en étape 3) pour assurer la fusion efficace. Ce produit est proposé par (Yang 92) et (Foul 92), et peut également être utilisé pour fusionner des plans de rechange pour atteindre le même objectif.

Une autre approche de la fusion d'un ensemble de plans dans un plan global traite des problèmes posés par les conflits et les actions redondantes à l'aide de la méthode de recherche et une heuristique smart basé sur les coûts, Ephrati et Rosenschein ont montré que, en divisant le travail de construction sous plans sur plusieurs agents,on peut réduire la complexité globale de l'algorithme de fusion [**24**].

Dans d'autres travaux sur le plan fusion, ils ont proposé un algorithme polynôme-temps distribués pour améliorer la protection sociale [**25**]. Grâce à un processus d'agrégation de la contrainte de groupe, les agents construisent progressivement un plan global amélioré par un vote sur actions conjointes. Ils proposent même des algorithmes pour traiter avec des agents non sincères et pour entrelacer la planification, la coordination et l'exécution.

Tsamardinos a réussi à développer un algorithme de fusion de plan qui traite à la fois les actions duratives et le temps [**26**]. Ils construisent un réseau temporel et conditionnel pour spécifier les conflits entre les plans, fondé sur cette description, une série de contraintes est obtenu et pouvant être résolu par un résolveur de la contrainte.

Cox et Durfee décrivent la façon de maintenir l'autonomie, tout en étant capable d'utiliser les résultats d'autres agents pour améliorer l'efficacité, fondamentalement, leur idée est d'ajouter ces dépendances conditionnellement au plan, si l'autre agent réussit, cette branche plus efficace du plan peut être exécuté; sinon le cadre normal de l'action peut encore être suivie [**27**].

I.5 Conclusion

La planification est un domaine bien connu et étudié de l'intelligence artificielle et un domaine de recherche ouvert.

Tandis que les propriétés théoriques des systèmes multi-agents sont bien étudiées dans la communauté de système multi-agents, la relation de la planification n'est pas un sujet bien étudié

et d'autres travaux de recherche peuvent améliorer la compréhension de ce problème de planification multi-agent.

On a présenté, dans ce chapitre, une vue détaillée sur un approche qui est actuellement un domaine de recherche très actif, l'approche de « la planification » et plus précisément « les planification multi-agent », la planification multi-agents concerne la construction des plans pour un groupe d'agents autonomes qui peuvent interagir. Nous avons vue ce que cette approche offre plus des avantages qui justifient le choix d'utilisation la planification des système multi-agent pour les différentes applications.

Dans le prochain chapitre nous allons décrire la théorie de la planification décentralisée.

Chapitre II

Planification décentralisée

II.1. Introduction

La planification (distribuée) peut être considérée simplement comme une spécialisation de la résolution distribuée de problèmes, où le problème est résolu par un plan distribué. Cependant, une ambigüité existe puisqu'on ne sait pas exactement ce qui est distribué. En effet, il est possible que le plan soit distribué entre plusieurs systèmes d'exécution ou alors, le processus de planification est distribué.

Dans la planification décentralisée avec fusion de plans, le problème d'avoir plusieurs agents élaborant individuellement des plans pour eux-mêmes, puis assurant que leurs différents plans puissent être exécutés sans conflit est un des défis de distribution résolu par la méthode de fusion des plans. Généralement, un agent central de coordination analyse l'ensemble des ces différents plans pour déterminer les séquences d'actions pouvant conduire à des conflits. Ensuite il modifie les plans en éliminant ces conflits. Cependant, il existe des systèmes où chaque agent vérifie que son plan local n'entre pas en conflit avec les plans locaux des autres agents.

Un autre moyen (formation itérative des plans) de planifier sans conflits entre les plans locaux est d'obliger les agents à la recherche dans un espace de plans plus vaste, plutôt que chacun propose un seul plan. Ainsi, chaque agent construit l'ensemble de tous les plans possibles pour l'accomplissement de ses objectifs propres. Le processus de planification distribuée consiste alors à une recherche d'un sous-ensemble de plans pouvant s'ajuster ensemble.

Dans la négociation, nous avons introduit le problème des conflits qui peuvent exister entre les plans des agents et les méthodes pour éviter ces conflits. Parfois, le processus permettant de déterminer l'agent qui doit changer son plan ou attendre un autre agent est assez aléatoire et arbitraire. D'autres fois, ce choix doit réviser les plans locaux en tenant compte des possibilités ouvertes à l'agent.

II.2. Type de Planification décentralisée

Dans la littérature il y a plusieurs modes de planification décentralisée.

II.2.1. Planification décentralisée avec fusion de plans

Dans le système multi-agents, les agents effectuent certaines tâches par exécution des plans, par conséquent, le problème de trouver un plan compte tenu d'un certain objectif, a donné beaucoup d'attention dans la littérature.

Au lieu de se concentrer sur ce problème, la mise au point de cette partie du chapitre est sur la coopération entre les agents qui ont déjà construit des plans pour leurs objectifs, en coopérant, les agents pourraient réduire le nombre d'actions qu'ils ont à accomplir pour atteindre leurs objectifs.

L'idée principale est que la mise en œuvre d'un plan ou un agent qui donne éventuellement produits secondaires qui peuvent être utilisées en tant que ressources par d'autres agents, en conséquence, un autre agent peut ignorer certaines de ses actions programmées.

Ce processus d'échange des produits, appelés fusion du plan, utilisés dans les plans distribués dans lesquels les agents deviennent dépendants les uns des autres, mais qui sont capables d'atteindre leurs objectifs plus efficacement. Les agents indépendants de planification opérant dans un environnement partagé peuvent être capables d'exploiter les actions redondantes entre leurs plans d'augmenter leur efficacité de l'exécution, ces actions redondantes peuvent être "fusionnées".

Le problème de trouver l'ensemble optimal d'actions, ou les étapes du plan, de fusionner entre les différents plans est appelé le problème de fusion de plan. Habituellement, les plans d'un seul agent ne peuvent être réduits beaucoup sans une aide extérieure. Donc, il est plus intéressant de voir si les plans multi-agents peuvent être réduits.

Les plans de plusieurs agents peuvent être considérés comme un seul plan global constitué de la composition des plans d'agents individuels, à la suite de la fusion du plan, les agents deviennent dépendants d'autres agents [**28**].

II.2.1.1. Le problème de la fusion de plan d'un seul agent

Beaucoup de chercheurs ont porté sur le problème fusion de plan, Qiang Yang a été l'un des premiers chercheurs à développer une théorie de calcul sur le problème de la fusion de plan.Yang décrit le problème xde fusion de plan comme un ensemble de plans P, où chaque plan Pi est un tuple <G, S, L, O>, S est l'ensemble des étapes dans le plan, L est l'ensemble des liens de causalité sur des marches en S, O est l'ensemble des contraintes sur S, et G est l'ensemble des conditions de

l'objectif atteint par le plan. Chaque plan Pi doit être acyclique (ce qui signifie qu'il y a ordres légitimes du plan), chaque étape est d'un certain type, et a un certain coût, les étapes sont classées à la complexité du problème, et les étapes peuvent fusionner si elles sont du même type.

Une solution optimale pour le problème de fusion de plan est un plan d'ordre partiel fusionné de la même forme comme un sous plan Pi dans le problème d'origine dont la somme des coûts de l'union des étapes est minimale, et le réseau de plan est toujours acyclique [29].

II.2.1.2. Le problème de la fusion de plan d'un système multi-agent

Dans le contexte multi-agent, cependant, nous n'avons généralement pas un seul plan représentant les plans de tous les agents, parce que les agents préfèrent garder l'information pour eux-mêmes, chaque agent a son propre plan.

Le problème de la fusion de plan dans le système multi-agents peut être formalisé de la même manière que le problème d'un seul agent (où chaque plan Pi appartient maintenant à un agent Ai) et il est en principe possible d'utiliser l'algorithme de Qiang Yang pour le résoudre. Mais, contrairement aux hypothèses de Qiang Yang, les problèmes plus «dures» dans le cas multi-agents sont généralement des problèmes impliquant plusieurs agents. Par conséquent, l'hypothèse de Qiang Yang que le nombre de plans (ou agents) reste constante est déraisonnable dans le cas multi-agents, car il ne permet aucune mise en échelle du problème.

Nous estimons que, dans les problèmes multi-agents dans laquelle la fusion du plan a un sens, une meilleure hypothèse est que les agents autonomes ont tendance à être faiblement couplés, ce qui signifie que la plupart des actions d'un agent de plans ne peut pas fusionner avec ceux d'un autre agent. Cette forme de l'amélioration des plans multi-agents est la fusion du plan [**29**].

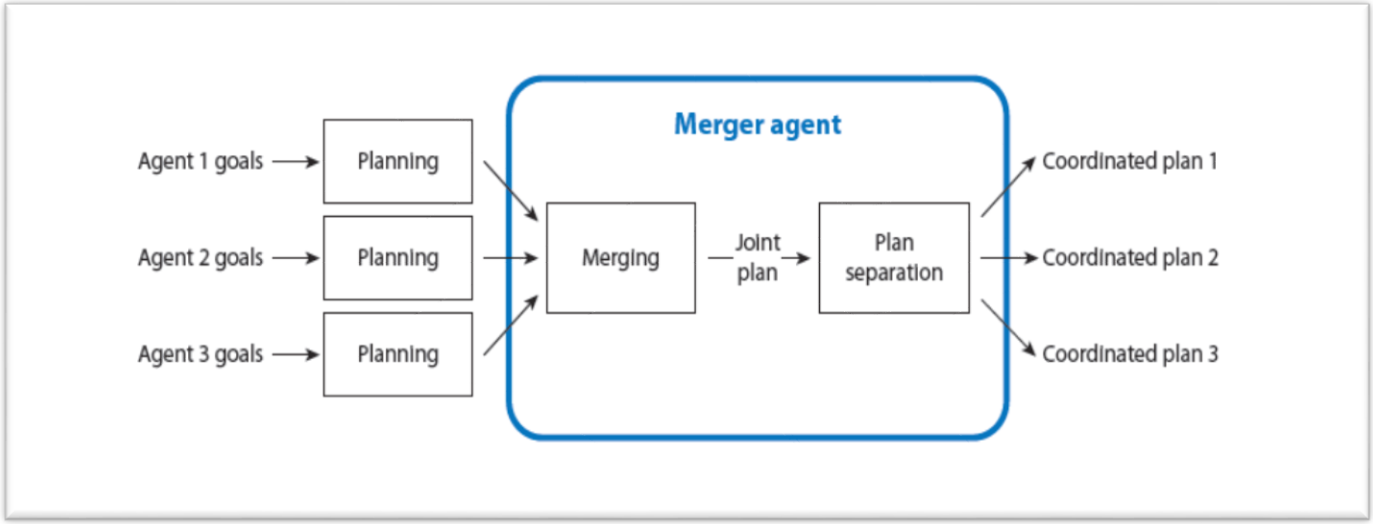


Figure II.1: Circulation de l'information dans un système de fusion de plan simple [30].

II.2.1.3. Les Algorithmes pour la fusion des plans

L'algorithme de la fusion du plan est une adaptation de l'algorithme de la planification centralisé, la méthode la plus simple de résoudre un problème de planification multi-agents est de mettre en commun toutes les connaissances des objectifs, contraintes de l'état, les tâches et les méthodes et passer le tout à un agent simple de la planification, car les agents de l'étape de la planification créent des plans sans tenir compte des objectifs les uns des autres, les connaissances, les capacités et ainsi de suite, donc il y a de fortes chances que leurs solutions individuelles va contenir des tâches redondantes ou contradictoires.

Les plans sont fusionnés en utilisant une variante de l'algorithme de la planification centralisé, le serveur crée une tâche commune initiale à partir des solutions individuelles téléchargées par les agents la planification. Un agent de fusion télécharge la tâche et utilise l'algorithme de planification centralisée à créer un plan final de solution au problème de la planification complète, la solution finale est ensuite téléchargé sur le serveur [**30**].

II.2.1.4. Les inconvénients de la fusion de plan

La fusion de plan est une approche populaire pour planification et coordination, toutefois, elle présente deux inconvénients majeurs:

- Le succès du processus entier est dépendent de la capacité de l'agent de fusion pour fusionner les plans individuels dans un plan commun coordonnée, cela peut être impossible en raison de conflits inter-plan que la fusion ne peut pas résoudre, ou à cause d'un manque de temps, l'information ou la capacité de la mémoire, donc si la phase de fusion échoue, le temps passé de planification sera perdu.
- les agents individuels ne peuvent pas se donner mutuellement assistance lors de la planification parce qu'ils planifient de façon isolée, ce qui signifie que, par exemple, un agent avec des contrôles exclusifs sur une ressource ne peut pas aider les autres agents dans leur planification de tâches connexes.
- Plusieurs problèmes potentiels peuvent survenir au cours de la fusion des deux plans. Les plans peuvent entrer en conflit parce qu'ils exigent la même ressource pour l'exécution correcte, ou l'ordre prescrit des actions peut entraîner impasse, par exemple, les deux doivent attendre que l'autre pour terminer une certaine action. Afin de modéliser de façon réaliste les situations de la vie réelle, nous devons suivre les différents attributs de ressources - tels que le carburant, le temps, l'espace et la capacité d'une manière numérique. En outre, comme un agent peut prendre en charge les uns des autres pour satisfaire une (sous) objectif, il doit y avoir une certaine forme de modèle de coût et de profit [**30**].

II.2.2. Planification décentralisée par la satisfaction de contraintes distribuées

Un grand nombre de problèmes en IA et d'autres domaines de la science informatique peut être considéré comme des cas particuliers du problème de satisfaction de contraintes. Quelques exemples sont la vision par machine, la maintenance de croyance, la planification, le raisonnement temporel, les problèmes de graphes, la planification des expériences génétiques, et satisfiabilité.

Constraint Satisfaction Problem (CSP) ont trois composantes de base: les variables, les valeurs et les contraintes. L'objectif est de trouver une affectation de valeurs à des variables, à partir de leurs domaines séparés, de telle sorte que toutes les contraintes sont satisfaites. De nombreux algorithmes séquentiels pour la résolution de CSP ont été développés, mais dans le monde réel, nous pourrions avoir à faire face à CSP distribués. Il y a deux raisons principales pour traiter les CSP distribués.

Premièrement CSP eux-mêmes peuvent être logiquement ou géographiquement distribués. Ces problèmes peuvent être mieux résolus par une plate-forme multiprocesseur.

Deuxièmement les ordinateurs parallèles ou distribués peuvent fournir plus de puissance de calcul s'ils sont utilisés efficacement et ce qui est important en considérant le montant des calculs nécessaire pour résoudre les CSP.

Le problème de satisfaction de contraintes distribuées (Distributed Constraint Satisfaction Problem) est défini comme un CSP dans lequel plusieurs agents (processus logiciels) sont impliqués. DCSP est sous-problème important de l'intelligence artificielle distribuée (IAD). Tel que décrit par Makoto Yokoo, différents problèmes de DAI peuvent être formalisés comme DCSP, À son tour DCSP peut fournir un cadre formel pour l'étude de différentes méthodes de DAI [**31**].

II.2.2.1. Constraint Satisfaction Problem (CSP)

Un problème de satisfaction de contraintes (CSP) consiste à trouver l'assignation consistante de valeurs à des variables prenant leur valeur dans des domaines discrets et finis. Il est à noter que la solution d'un CSP n'est pas forcément la solution optimale. Traditionnellement, l'optimisation est faite en sélectionnant la meilleure de ces solutions, et ce après que toutes les solutions possibles aient été propagées (Yung et Yang, 1999).

Différentes recherches (Meisels et Schaerf, 2001 ; Burke et Petrovic, 2002) notent que les contraintes peuvent être découpées en deux : d'une part les contraintes fortes qui doivent absolument être respectées et d'autre part, les contraintes faibles que l'on va essayer de respecter au mieux. Si l'on ne tient compte que des contraintes fortes, le problème consiste donc à trouver

une solution (parmi d'autres) qui satisfasse aux contraintes. Si on y ajoute en plus les contraintes faibles, une fonction d'évaluation ou de pénalité mesurant le degré de respect des contraintes faibles est ajoutée. Il s'agit alors de trouver dans l'espace des solutions respectant les contraintes fortes la solution qui maximise (ou, selon les cas, qui minimise) cette fonction.

Le problème de satisfaction de contraintes ressemble alors à un problème d'optimisation où l'algorithme cherchera à violer d'abord les contraintes faibles définies comme non-prioritaires, c'est à dire les contraintes faibles ayant un petit poids dans la fonction d'évaluation, avant de violer les contraintes faibles prioritaires. Deux étapes sont donc nécessaires : une détermination de l'espace des solutions respectant les contraintes fortes, puis une recherche dans cet espace de la solution minimisant ou maximisant la fonction d'évaluation. Ce problème ressemble encore plus à un problème d'optimisation lorsque les contraintes fortes et faibles sont mesurées toutes les deux par cette fonction. En effet, la satisfaction de contraintes consiste alors uniquement à optimiser cette fonction : il n'y a plus d'étape de recherche des solutions respectant les contraintes fortes, seule demeure l'étape de minimisation ou de maximisation de la fonction d'évaluation. Par exemple, Kanoh et ses collègues (Kanoh et al., 2001) construisent une fonction de péna-lité en donnant un poids très élevé aux contraintes fortes et un poids très faible aux contraintes faibles.

Formellement, un CSP consiste en n variables x1, x2, ..., xn dont les valeurs sont prises respectivement dans des domaines discrets finis D1, D2, ... , Dn et en un ensemble de contraintes sur ces valeurs, chaque contrainte étant définie par un prédicat. Plus précisément, la contrainte pk(xk1, ... , xkj) est un prédicat défini sur le produit cartésien Dk1 × ... × Dkj. Ce prédicat est vrai ssi l'assignation de valeur de ces variables satisfait cette contrainte. Résoudre un CSP revient à trouver une assignation de valeur à toutes les variables, de manière à ce que toutes les contraintes soient satisfaites. Comme de façon générale la satisfaction de contraintes est NP-complète, une exploration par essai et erreur des alternatives est inévitable (Yokoo et Ishida, 1999). Les algorithmes connus qui sont mis en oeuvre lors de la résolution d'un CSP se divisent en deux groupes (Yokoo et Hirayama, 2000) :

- les algorithmes de consistence, c'est à dire des algorithmes de prétraitement qui améliorent le travail des algorithmes de recherche.
- les algorithmes de recherche, parmi lesquels on distingue :
  - les algorithmes à retour arrière (backtracking algorithms).
  - les algorithmes d'amélioration itérative (iterative improvement algorithms).

Une autre classification des techniques de résolution d'un CSP est proposée par Yung et Yang (Yung et Yang, 1999). Les trois catégories de cette classification sont : (i) la réduction du problème qui consiste à retirer les valeurs et les étiquettes de valeur redondantes, (ii) la synthèse de solution

en construisant incré-mentalement un treillis appelé « graphe du problème minimal » qui représente le problème minimal et (iii) la recherche qui inclut les algorithmes à retour arrière (backtracking algorithms), les algorithmes de vérification prévisionnelle (forward checking algorithms), les systèmes de maintenance de la vérité et la propa-gation de contraintes [**32**].

❖ **Amélioration Itérative**

GSAT [Selman et al., 1992] tente de trouver une solution en changeant itérativement la valeur d'une seule variable à la fois, en minimisant le nombre de contraintes violées. Le nombre global d'itérations est limité. Breakout [Morris, 1993] pondère chaque contrainte. On part d'un assignement (aléatoire), si on ne se trouve pas dans un minimum local (affectation complète avec aucune possibilité d'amélioration), on change la valeur d'une variable qui diminue le coût total (des contraintes violées), à défaut on augmente le poids des contraintes violées. Ces deux algorithmes utilisent l'amélioration itérative et le nombre d'itérations autorisées est limité, remarquons également qu'aucun d'eux n'est complet. De plus, à tout moment, ils fournissent un assignement qui est une affectation complète mais qui n'est pas forcément une solution pour le problème [**33**].

- **Backtracking (BT)**

L'algorithme Backtracking fournit une recherche systématique. Le principe est de partir d'une affectation aléatoire et de construire une solution. On a à tout moment une solution partielle ; pour chaque variable on essaie de trouver une valeur consistante avec la solution partielle construite. Si on trouve une valeur consistante, on passe à la variable suivante, à défaut on revient à la variable précédente, on recherche une nouvelle valeur pour celle-ci et on garde en tant que nouvelle contrainte la solution partielle trouvée. L'heuristique Min-Conflicts [Minton et al.,1994] peut être combinée au Backtracking afin d'augmenter son efficacité, Cette heuristique permet de choisir les variables à affecter en évaluant les conséquences de celle-ci par rapport à l'ensemble des contraintes. Cet algorithme est complet, mais a, dans le pire des cas, un temps d'exécution exponentiel, ce qui n'est pas acceptable pour de larges problèmes [**33**].

- **Weak-Commitment Search (WCS)**

[Yokoo, 2001] présente l'algorithme Weak-Commitment Search (WCS) qui s'appuie sur le Backtracking avec Min-Conflicts. La seule différence réside dans le fait qu'au lieu de revenir à la dernière variable assignée lorsque l'on ne peut pas trouver de solution, on va chercher à réassigner toutes les variables, en gardant la solution partielle comme nouvelle contrainte. Cette petite modification par rapport au Backtracking le rend plus efficace. Lorsqu'une mauvaise solution partielle est construite, le Backtracking peine à en sortir puisqu'obligé de mener une recherche

exhaustive, ce qui n'est pas le cas ici ; toute la solution étant abandonnée. Il existe d'autres méthodes métaheuristiques permettant la résolution de CSP et l'optimisation combinatoire, que nous ne détaillerons pas [**33**].

II.2.2.2. DCSP – Distributed CSP

Il existe différentes variantes du CSP original. Par exemple, dans les ICSP (Interval CSP) l'objectif n'est plus de trouver l'assignation des variables qui satisfait aux contraintes, mais les intervalles de ces variables où les contraintes sont respectées (Yung et Yang, 1999). Les systèmes multi-agents étant des systèmes distribués, nous nous intéressons plus particulièrement aux travaux effectués dans un autre sous-domaine des CSP : les CSP distribués ou DCSP.

Un DCSP est un CSP dans lequel les variables et les contraintes sont distribuées entre des agents. Ces variables se divisent en deux ensembles disjoints : les contraintes intra-agents et les contraintes inter-agents. Une contrainte intra-agent n'est connue que par un agent. Habituellement, on considère qu'une contrainte inter-agent est connue par tous les agents ayant une variable dans cette contrainte. Comme dans le cas centralisé, résoudre un DCSP consiste à trouver une assignation de valeur aux variables en ne violant aucune contrainte (bien que la littérature des DCSP se concentre principalement à la résolution des contraintes in-ter-agents). Trouver une assignation de valeur aux variables inter-agents peut être vu comme réaliser la cohérence ou la consistance d'un système multi-agent (Bessière et al., 2001 ; Yokoo et al., 1998).

Les heuristiques de résolution d'un DCSP se classent en deux catégories. D'une part, la programmation orientée-marché repose sur les mécanismes d'enchères. D'autre part, des agents peuvent être en compétition pour utiliser des ressources, ces ressources pouvant calculer leur demande aggrégée (Durfee, 1999). En outre, les algorithmes de résolution des CSP semblent similaires à des méthodes de traitement parallèle/distribué pour résoudre des CSP, quoique les motivations de recherche soient fondamentalement différentes (Yokoo et Hirayama, 2000) [**32**] .

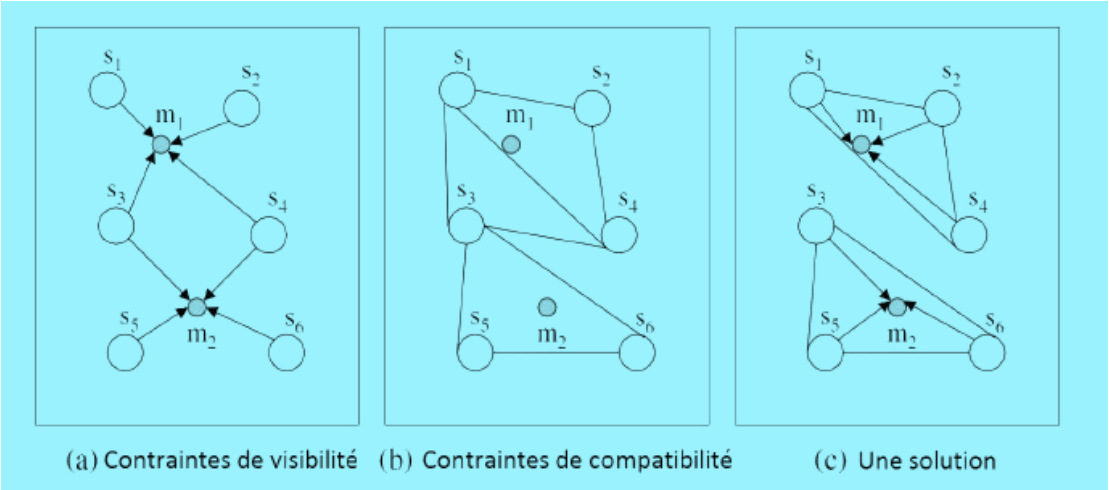
II.2.2.2.1. Exemple d'application des DCSP

Il y a beaucoup d'exemples pour DCSP: le problème des n-reines distribué, le problème m-échiquier n-pièces distribuées et le problème de capteur-Mobile distribué.

- **Le Sensor DCSP**

Les DCSP permettent de modéliser de manière simple et efficace de nombreux problèmes naturellement distribués. Parmi eux, nous pouvons citer le Sensor DCSP (pour Distributed Sensor-Mobile problem [Fernandez et al. 2002]). Il s'agit d'un problème contenant un ensemble de détecteurs et un ensemble de mobiles. Les détecteurs doivent suivre les mobiles. Le but de ce

problème est de faire en sorte que chaque mobile soit en permanence suivi par un ensemble de détecteurs (en général trois pour ce problème). Chaque détecteur ne peut suivre qu'un seul mobile. Afin de résoudre ce Sensor DCSP, une solution globale doit être trouvée où chaque mobile est suivi par trois détecteurs distincts. Les contraintes de visibilité et les contraintes de compatibilité doivent être satisfaites [**34**].



FigureII.2 : Un problème de type Sensor DCSP [34].

II.2.3. Planification décentralisée par recherche dans l'espace d'états distribuée

Ici nous nous demandons comment les agents peuvent optimiser une fonction objective globale dans un mode distribué. Plus précisément, nous considérons quatre familles de techniques et exemples des problèmes associés. Ils sont, dans l'ordre:

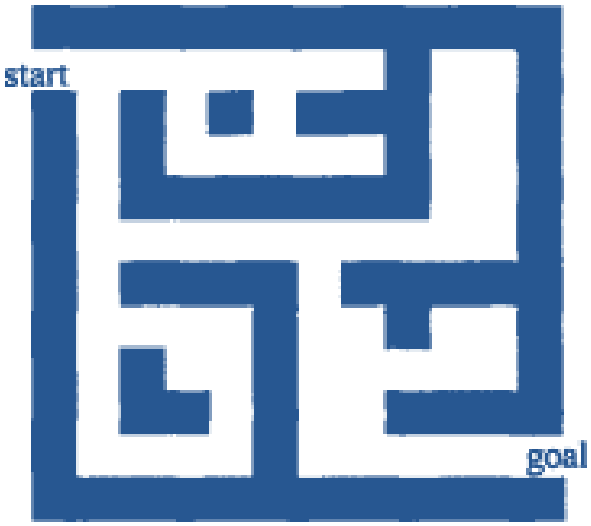
- programmation dynamique distribuée (Appliquée à des path-planning problèmes).
- solutions distribuées de la décision de Markov problèmes (MDP).
- Les algorithmes d'optimisation avec une saveur économique (appliquée à la correspondance et d'ordonnancement des problèmes).
- Coordination par les lois sociales et les conventions, et l'exemple des règles de circulation [**35**].

II.2.3.1. Programmation dynamique distribuée (des path-planning problèmes).

Comme la coloration de graphes, planification de chemin constitue le cadre abstrait commun pour résolution des problèmes. Un problème de planification de chemin est constitué de les éléments suivants: un ensemble de nœuds N, chacune représentant un état, et un ensemble de liens dirigés L, chacun représentant un opérateur disponible à un agent de résolution de problèmes.

Nous supposons qu'il existe un nœud unique s appelé le nœud de départ, ce qui représente l'état initial, en outre, il existe un ensemble de nœuds G, dont chacune représente un état but. Pour chaque liaison, le poids de la liaison est définie, ce qui représente le coût de l'appliquer l'opérateur, nous appelons le poids du lien entre deux nœuds la distance entre les nœuds.

Le problème 8-puzzle peut être formalisé comme un problème de chemin exploratoire en représentant les arrangements possibles de carreaux en tant que nœuds, et autorisés déplace comme liens. Les arrangements qui peuvent être atteints en la glissant une tuile sont les voisins de l'entente initiale. Dans ce problème, les poids de tous les liens sont 1, et pour chaque liaison, il existe un lien dans le sens inverse.



FigureII.3 : Le problème 8-puzzle [36].

Dans ce qui suit, nous introduisons la programmation dynamique asynchrone en tant que la base d'autres algorithmes. Ensuite, nous présentons l'algorithme d'apprentissage en temps réel A* [**36**].

II.2.3.1.1. La programmation dynamique asynchrone

Dans une planification de problème de chemin, le principe de l'optimalité tient. En résumé, le principe de l'optimalité déclare qu'un chemin est optimale si et seulement si tous les segments qui les imposent sont optimaux.

Par exemple, s'il existe un chemin optimal (le plus court) à partir du nœud de départ à un nœud de but et il existe un nœud x intermédiaire sur le chemin, le segment à partir du nœud de départ au nœud x est en fait le chemin optimal de nœud de départ au nœud x. De même, le segment de nœud x au but est aussi le chemin optimal de nœud x à l'état final.

Nous représentons la plus courte distance entre un nœud i au but t comme h*(i). Ainsi, la distance la plus courte de i à t via un nœud j voisin de i est donnée par $f*(i, j) = w(i, j) + h*(j)$,

et h∗(i) = minj f∗ (i, j). Basé sur ces données, l'algorithme asynchrone à chaque nœud effectué de manière répétée la procédure suivante. Dans cette procédure, donné dans la figure suivant, chaque nœud i maintient une variable h (i), qui est une estimation de h*(i) [**36**].

procedure ASYNCHDP (node i)
if i is a goal node then
h(i) ← 0
else
initialize h(i) arbitrarily (e.g., to ∞ or 0)
repeat
forall neighbors j do
f(j) ← w(i, j) + h(j)
h(i) ← minj f(j)

II.2.3.1.2. Apprentissage en temps réel A *

Quand un seul agent est chargé de résoudre un problème de chemin, il n'est pas toujours possible d'effectuer des calculs locaux pour tous les nœuds. Par exemple, les robots autonomes peuvent ne pas avoir assez de temps pour la planification et doivent entrelacer planification et l'exécution. Donc, l'agent doit exécuter les calculs de façon sélective pour certains nœuds. Compte tenu de cette exigence, quel nœud doit l'agent choisir? Une façon intuitive naturelle est de choisir le nœud actuel où l'agent est situé. Il est facile d'imaginer que la zone de détection d'un robot autonome est toujours limitée. Tout d'abord, l'agent met à jour la valeur de h du nœud actuel, puis se déplace vers le nœud le plus voisin. Cette procédure est répétée jusqu'à ce que l'agent atteint un état but. Cette méthode est appelée algorithme d'apprentissage en temps réel A* [**35**].

procedure LRTA*
i ← s // the start node
while i is not a goal node do
foreach neighbor j do
f(j) ← w(i, j) + h(j)
i' ← argminj f(j) // breaking ties at random
h(i) ← max(h(i), f(i'))
i ← i

II.2.3.2. Solutions distribuées de la décision de Markov problèmes (MDP)

Un processus de décision de Markov (MDP) est un cadre mathématique pour des problèmes de prise de décision séquentiels dans les domaines stochastiques. MDP fournit donc sous-jacente sémantique à la tâche de planification sous incertitude.

Un processus de décision de Markov (MDP) M est défini comme un 4-tuple M = (X, A, R, P)

où:

X est un ensemble fini de N états;

A est un ensemble fini d'actions;

R est une fonction de récompense

R: X *A —› R , tel que R (x, a) représente la récompense obtenue par l'agent dans l'état x après l'action a, et P est un modèle de transition de Markov où P (x´ ׀x: a) représente la probabilité de passer de l'état x à l'état x´ après l'action a. Nous supposons que les récompenses sont bornées, c'est, il existe Rmax tels que Rmax ≥ ׀r (x: a)׀; ∨x; a. [**37**].

❖ **Résolution MDP**

Il existe plusieurs algorithmes pour calculer la politique optimale dans un MDP. Les trois plus couramment utilisés sont la programmation linéaire, itération de la valeur, et itération de la politique. Un élément clé dans les trois algorithmes est le calcul des fonctions de valeur, Rappelons qu'une fonction de valeur définit une valeur pour chaque état x dans l'espace d'état. Avec une représentation explicite des fonctions de valeur comme un vecteur de valeurs pour les différents états, les algorithmes de résolution peuvent appliquées comme plusieurs étapes algébriques simples. Une fois la fonction de valeur optimale V* est calculée, la politique optimale Π* est tout simplement la politique cupide à l'égard de V* [**37**].

➢ **La programmation linéaire**

La programmation linéaire fournit une méthode simple et efficace pour trouver la fonction de valeur optimale pour un MDP. Dans la première formulation proposée, les variables de LP sont V (x) pour chaque état x, où V (x) représente la valeur de départ à l'état x

Le LP est donnée par

Les variables V (x); ∨x;
Minimiser Σ x α(x) V (x);
Sujet à V (x) ≥ R(x, a) + γ Σ x´ P (x´׀ x, a) V (x´); ∨x ϵ X; a ϵ A;

➢ **Itération de la valeur**

Itération de la valeur est une approche alternative couramment utilisée pour résoudre les MDP. L'algorithme ci-dessous commence à partir d'une estimation initiale V(0) de la fonction de valeur. Cette estimation est itérative améliorée grâce à des applications répétées de l'opérateur de Bellman [**37**].

VALUEITERATION $(P, R, \gamma, \mathcal{V}^{(0)}, \varepsilon, t_{max})$
// P – transition model.
// R – reward function.
// γ – discount factor.
// $\mathcal{V}^{(0)}$ – any initial estimate of the value function.
// ε – Bellman error precision.
// t_{max} – maximum number of iterations.
// Return near-optimal value function.
LET $t = 0$.
REPEAT
LET:

$$\mathcal{V}^{(t+1)}(\mathrm{x}) = \mathcal{T}^*\mathcal{V}^{(t)}(\mathrm{x}) = \max_a \left[R(\mathrm{x}, a) + \gamma \sum_{\mathrm{x}'} P(\mathrm{x}' \mid \mathrm{x}, a)\mathcal{V}(\mathrm{x}') \right], \quad \forall \mathrm{x} .$$

LET $t = t + 1$.
UNTIL BellmanErr$(\mathcal{V}^{(t+1)}) \leq \varepsilon$ OR $t \geq t_{max}$.
RETURN $\mathcal{V}^{(t+1)}$.

- **Politique itération**

Politique itération est un algorithme très efficace pour résoudre les MDP. L'algorithme itère sur les politiques, produisant une amélioration de la politique à chaque itération. En commençant par une certaine politique initiale Π˚, chaque itération se compose de deux phases. Détermination de la valeur calcule pour une politique Π^x, la fonction de valeur VII (x). L'étape de l'amélioration de la politique définit la prochaine politique comme Π(x+ 1) = Greedy [VΠ(t)].

POLICYITERATION $(P, R, \gamma, \mathcal{V}^{(0)}, \varepsilon, t_{max})$
// P – transition model.
// R – reward function.
// γ – discount factor.
// $\pi^{(0)}$ – any initial policy.
// ε – Bellman error precision.
// t_{max} – maximum number of iterations.
// Return (near-)optimal policy.
LET $t = 0$.
REPEAT
// Value determination step.
COMPUTE VALUE OF POLICY $\pi^{(t)}$ BY A SOLVING LINEAR SYSTEM OF EQUATIONS:

$$\mathcal{V}_{\pi^{(t)}}(\mathrm{x}) = R_{\pi^{(t)}}(\mathrm{x}) + \gamma \sum_{\mathrm{x}'} P_{\pi^{(t)}}(\mathrm{x}' \mid \mathrm{x})\mathcal{V}_{\pi^{(t)}}(\mathrm{x}), \quad \forall \mathrm{x} .$$

// Policy improvement step.
LET $\pi^{(t+1)} = \text{GREEDY}[\mathcal{V}_{\pi^{(t)}}]$.
LET $t = t + 1$.
UNTIL $\pi^{(t)} = \pi^{(t+1)}$ OR BellmanErr$(\mathcal{V}^{(t+1)}) \leq \varepsilon$ OR $t \geq t_{max}$.
RETURN $\pi^{(t+1)}$.

II.2.3.3. Les lois sociales et les conventions

Les lois sociales (ou systèmes normatifs) sont apparues comme un paradigme naturel et puissant pour la coordination des systèmes multi-agents. Les lois sociales paradigme exposent l'ensemble du spectre entre les mécanismes de coordination entièrement centralisés et entièrement décentralisés. La loi sociale est, de manière intuitive, une contrainte sur le comportement des

agents, qui assure que leurs comportements individuels sont compatibles, en règle générale, une loi sociale est imposée hors ligne, minimiser les risques de conflit en ligne ou la nécessité de négocier.

Les lois sociales paradigme est fondé sur l'utilisation de la logique de calcul de raisonner sur systèmes multi-agents, mais souvent rend également l'utilisation de la théorie des jeux. En loi sociale nous devons discuter les questions telles que: comment une loi sociale assurant un certain comportement global soit automatiquement construit? Si deux lois sociales atteindre le même objectif, que l'on doit-on utiliser? Comment pouvons-nous construire une loi sociale qui fonctionne même si certains agents ne sont pas conformes? Quels sont les agents les plus importants pour une loi sociale pour atteindre son objectif? La coordination porte sur le comportement d'un système dans son ensemble, avec les propriétés globales du système.

Dans de nombreux cas, il se peut que certains agents choisissent de ne pas se conformer à une loi sociale donnée. Il existe de nombreuses causes possibles de non-conformité; il pourrait être délibéré, car l'agent ne considère pas qu'il est dans son intérêt de se conformer, ou il se pourrait que d'un composant dans le système échoue. Doit ensuite discuté comment analyser les propriétés d'une loi sociale sous non-respect éventuel. En particulier, la robustesse de la loi sociale, et tenter d'identifier les agents qui sont les plus importantes pour le bon fonctionnement du système. La loi sociale est robuste si l'objectif est atteint si seul un petit nombre des agents choisissent de ne pas se conformer.

Les principaux problèmes: quels agents sont nécessaires, dans le sens que l'objectif ne tient pas si elles sont conformes? Existe-il une loi sociale qui est robuste possible dans le sens que le respect d'un groupe donné (ou plusieurs) des agents est suffisante pour assurer l'objectif? Considérons la tâche d'un responsable des transports de la ville qui souhaite optimiser les flux de trafic dans la ville. Alors qu'il ne peut pas redessiner voitures ou de créer de nouvelles routes, il peut imposer des règles de la circulation. Une règle de trafic est une forme de loi sociale: loi sociale une restriction sur les stratégies données des agents. Une règle de trafic typique interdit les gens de conduire sur le côté gauche de la route ou les feux rouges. Pour un agent donné, une loi sociale présente un compromis; il souffre de la perte de la liberté, mais peut bénéficier du fait que d'autres en perdent la liberté. Une bonne loi sociale est conçue pour bénéficier à tous les agents [**35**].

II.2.4. Planification adversarielle et la théorie des jeux

La théorie des jeux est un ensemble d'outils pour analyser les situations dans lesquelles l'action optimale pour un agent dépend des anticipations qu'il forme sur la décision d'un autre agent. Cet agent peut être aussi bien une personne physique, une entreprise ou un animal. L'objectif de la

théorie des jeux est de modéliser ces situations, de déterminer une stratégie optimale pour chacun des agents, de prédire l'équilibre du jeu et de trouver comment aboutir à une situation optimale. La théorie des jeux est très souvent utilisée en économie, en sciences politiques, en biologie ou encore en philosophie [**38**].

Un jeu est défini par des éléments suivants :

(i) Un ensemble de joueurs quels sont les agents qui interagissent?,

(ii) Les règles du jeu, quelles sont les actions qu'un individu peut entreprendre ? Quelles informations les agents disposent-ils lorsqu'ils prennent leurs décisions ?

(iii) Les résultats, quels sont-ils à l'issue du jeu ?

(iv) Les paiements, est---ce que les agents peuvent gagner ou perdre?

L'informa/on dont disposent les joueurs est une donnée fondamentale de toute interaction stratégique. On oppose informa/on parfaite et informa/on imparfaite d'une part, informa/on complète et informa/on incomplète d'autre part [**39**].

II.2.4.1. Taxonomie du jeu

Dépendamment du contexte d'étude, un jeu peut être classé selon plusieurs dimensions qui permettent de décrire l'environnement dans lequel évoluent les joueurs ainsi que les règles prédéterminées qui régissent leurs interactions. Les dimensions les plus importantes dans la littérature sont les suivantes [**40**]:

- Jeux à information complète ou à information incomplète (Rasmusen, 1989; Friedman, 1990).
- Jeux à information parfaite ou à information imparfaite (Rasmusen, 1989; Friedman, 1990).
- Jeux coopératifs ou non-coopératifs (Friedman, 1990).
- Jeux à 2-joueurs ou Jeux à n-joueurs (Rapoport, 1970 &1969).

Généralement, les jeux se distinguent selon que l'information est complète ou incomplète. On parle de jeu à information complète si les joueurs ont une connaissance totale du jeu et de sa structure. Dans ce cas, chaque joueur connaît l'ensemble des choix qui s'offrent à lui et aux autres joueurs, leurs conséquences, ainsi que les profits qui y sont associés. Dans le cas contraire, le jeu est dit à information incomplète. Ainsi, par exemple, le jeu d'échec est à information complète, alors que le Bridge est un jeu à information incomplète.

Toujours selon la dimension informationnelle, on distingue les jeux selon que l'information est parfaite ou imparfaite. Un jeu est défini comme étant à information parfaite lorsque chaque joueur

a une connaissance parfaite de l'ensemble des décisions prises antérieurement par les autres participants. On parle de jeu à information imparfaite si un ou plusieurs joueurs ne connaissent pas, lors de leurs prises de décisions, ce que les autres joueurs ont choisi. Ce cas se manifeste, par exemple, lorsque les joueurs doivent miser simultanément sans savoir ce que misent les autres.

Par ailleurs, la théorie des jeux est répartie en deux principales catégories : jeu non-coopératif et jeu coopératif. Dans le premier cas, les joueurs sont en conflit et ne collaborent pas entre eux. Leurs raisonnements se basent uniquement sur les concepts de la rationalité économique. Chaque joueur cherche à prendre la meilleure décision qui maximise son gain ou concrétise ses objectifs individuels.

Dans le deuxième cas, les joueurs ont la possibilité de communiquer entre eux. Leurs objectifs sont d'établir des ententes, de coordonner leurs stratégies afin d'optimiser leur bien-être collectif, et de trouver des mécanismes de répartition de la richesse collective basés sur des propriétés telle que l'équité, l'altruisme, etc. Dans ce cadre de jeu, les joueurs peuvent former des coalitions entre eux. Les coalitions génèrent des profits qui résultent de la coopération de leurs membres. En plus, on distingue deux classes de coalitions, à savoir : la coalition avec utilité transférable et la coalition avec utilité non transférable. Pour une coalition avec utilité transférable, il est permis d'additionner les utilités des joueurs et de les redistribuer à ses membres. Chaque joueur reçoit une part proportionnelle au rôle qu'il joue dans la création de la valeur de la coalition. Cela suppose l'existence d'une unité monétaire ou d'une utilité commune à tous les participants avec laquelle on peut effectuer des transferts de gains.

Les jeux peuvent être classés aussi selon le nombre de joueurs participant au jeu. On distingue le jeu à 2-personnes du jeu à n-personnes. Même si le jeu à 2-personnes peut être considéré comme étant un cas particulier du jeu à plusieurs personnes (n=2), les méthodes de résolution appliquées à ces deux cas sont différentes. Par exemple, dans le cas des jeux coopératifs à n-personnes, une des solutions possibles consiste en une élaboration de coalitions entre deux ou plusieurs joueurs. Cette technique ne s'applique pas pour le cas d'un jeu à 2-personnes.

Dans le présent travail, nous nous intéressons surtout aux jeux coopératifs à n-personnes avec utilité transférable. Dans ce qui suit, nous définissons plus formellement ce type de jeux, de même que le concept d'équilibre que nous utiliserons.

II.2.4.2. Les typologies

Les jeux stratégiques peuvent être typés selon quelques critères comme le comportement, l'information du jeu et la décision [**41**].

❖ **Jeux coopératifs / compétitifs**

Les jeux sont typés selon le comportement du joueur par rapport aux autre joueurs, pour un joueur soit il est en coopération/compétition avec les autres joueurs.

- **Jeux coopératifs**

Un jeu est coopératif lorsque des joueurs peuvent passer entre eux des accords qui les lient de manière contraignante (par exemple, sous la forme d'un contrat qui prévoit une sanction légale dans le cas du non respect de l'accord). On dit alors qu'ils forment une coalition dont les membres agissent de concert.

- **Jeux compétitifs**

Par définition, dans un jeu compétitif nous spécifions toutes les options stratégiques offertes aux joueurs, alors que les contrats qui sous-tendent les coalitions dans un jeu coopératif ne sont pas décrits. Chaque joueur cherche à avoir ses biens sans tenir compte aux autres joueurs.

❖ **Jeux avec décisions simultanées / séquentielles**

Les jeux sont typés selon l'ordre de décision des joueurs, les décisions des joueurs sont prises soit simultanées ou séquentielles.

- **Jeux avec décisions simultanées**

Dans ce type de jeux, les joueurs prennent leur décisions simultanément, sans connaitre la décision des autre joueurs, nous pouvons citer quelque exemple : Dilemme des prisonniers, pierre-feuille-ciseau.

- **Jeux avec décisions séquentielles**

Ici les décisions de joueurs se font séquentiellement (ne se font pas en même temps), autrement dit les décisions des joueurs sont pris avec un décalage temporelle). La décision du joueur est influée par les décisions déjà prises des autre joueurs, quelque exemples des jeux à décision séquentielle, le plus populaire c'est : le jeu d'échec.

❖ **Jeux à information parfaite/imparfaite**

Dans ce typage les jeux sont typés selon l'information sur les autres joueurs, autrement dis c'est ce que le joueur connaissent au moment où il prend sa décision. Jeux à information parfaite est un jeux ou les actions effectués auparavant par les joueur influent sur la décision du joueur désirant prendre une décision, car le joueur possède des information sur les actions déjà effectuées par les

autres joueurs et il faut que les décisions des joueurs se font séquentiellement ,hors que dans le jeu à information imparfaite le joueur ne possède pas toutes les informations sur les autres joueurs ou bien aux moins deux décisions sont déjà effectuée simultanément ,ce qui rend la décision plus délicate que dans le jeu a information parfaite.

❖ **Jeux à information complète/incomplète**

Ce typage se base sur l'information des joueurs par rapport aux autres joueurs.

- **Jeux à information complète**

Tous éléments de jeux est une connaissance commune entre les joueurs, plus précisément chaque joueur connait l'ensemble des comportements possible pour tous les autres joueurs et il connait tous le paiement.

- **Jeux à information incomplète**

C'est un jeu ou chaque joueur ne connaît pas tous les comportements des autres joueurs.

II.2.4.3. Complexité de la théorie des jeux

Dans une situation d'interaction stratégique, l'objectif est de trouver un équilibre qui maximise le gain des joueurs, tel que l'équilibre de Nash. Cependant, la recherche de cette solution est un problème combinatoire. Julien L'aumônier a étudié la complexité des algorithmes de calcul d'équilibres dans la théorie des jeux [2], il a donné les résultats suivants :

- Équilibre de Nash en stratégie pure: Complexité exponentielle en nombre d'agents (joueurs).
- Jeux à somme nulle : Programmation linéaire donc complexité polynomial.
- Jeux à somme non nulle : Est un problème combinatoire de la classe NP-Difficile.
- Existence d'un équilibre où chaque joueur à un gain d'au moins k : est problème d'existence NP-Complet.
- Compter le nombre d'équilibre de Nash : Est un problème combinatoire de la classe NPDifficile.
- Existence d'un équilibre de Nash en jeux stochastiques : Complexité de la classe PSPACEDifficile.

Dans un jeu à n joueurs, où chaque joueur a m stratégies possibles. L'espace de recherche est exponentiel en nombre de joueur (nombre de combinaisons possibles est mn). Pour trouver un équilibre de Nash, il faut vérifier la règle (1) de complexité (m-1) pour chaque joueur (la complexité de la vérification de la règle (1) pour tous les joueurs devient : n*(m-1)). Donc, la complexité de ce problème est : n*(m-1)*mn. Si le nombre de joueurs est grand, la taille de l'espace de recherche devient très grande. Il est pratiquement impossible d'énumérer toutes les

combinaisons possibles. Donc nous avons besoin d'appliquer les méthodes de résolution approchées des problèmes d'optimisation combinatoire (métaheuristiques), pour trouver une bonne solution en temps raisonnable, comme les algorithmes génétiques, l'optimisation par colonie de fourmis, le recuit simulé…etc [**42**].

II.3. Conclusion

La planification distribuée est lorsque le processus de planification et ses résultats sont distribués. Dans ce cas, il ne serait peut-être pas nécessaire d'avoir un plan multi-agents représenté entièrement dans le système, mais pourtant, la répartition des morceaux de ce plan peut être plus compatible. Cela signifie que les agents ne devraient pas être en conflit les uns avec les autres lors de l'exécution des plans partiels, et devraient s'entraider les uns avec les autres pour réaliser leurs plan.

Dans le chapitre suivant nous allons décrire la conception du système proposé et les détails de ses composants, et aussi les objectifs et les motivations de notre proposition.

Chapitre III

Conception de l'approche

III.1. Introduction

Les techniques d'intelligence artificielle ont été appliquées avec succès pour résoudre des problèmes de satisfaction de contraintes. Le système multi-agent (SMA) s'intéresse aux mécanismes liés aux problématiques de coordinations et d'interactions entre différentes entités. Nous considérons, dans ce cadre que la coordination peut être définie comme un processus de recherche dans des contextes de résolution de DCSP (*Distributed Constraint Solving Problem*).

Les DCSP peuvent être utilisés, par exemple pour résoudre de nombreux problèmes naturellement distribués tels que les problèmes d'affectation des réservations aux hadjs, d'emploi du temps, de trafic routier ou d'exploration multi-robots. Dans de tels contextes, la coordination peut être vue comme la décomposition d'un problème en sous problèmes, la résolution des sous problèmes, les mécanismes d'échanges de résultats partiels ou la diffusion des solutions des sous problèmes aux agents, jusqu'à l'obtention de la (des) solution(s) globale(s).

Les raisonnements intra-agent et inter-agents sont basés sur un ensemble de relations entre différentes variables. La résolution du problème s'effectue grâce aux agents qui interagissent afin d'obtenir une solution globale à partir des solutions locales. Les DCSP sont parfaitement adaptés pour résoudre des problèmes utilisant des données physiquement réparties et ne pouvant être résolus de manière centralisée.

Dans ce chapitre nous allons présenter :

- La conception de l'approche de planification décentralisée.
- Adapter cette approche à un système réelle, le modèle adapté est appelé **DCSPH** (**Distributed Constraint Solving Problem for Hadj**).
- Nous allons discutons par la suite les détails de ses composants.

Nous avons évoqué dans les chapitres précédents les raisons pour lesquelles nous avons eu recours au concept d'agent pour implémenter les fonctionnalités attendues. Ce système informatique est une structure qui utilise l'agent pour la coordination et la communication afin d'obtenir une solution globale à partir des solutions locales. Il offre la décomposition d'un problème en sous problèmes et en collaboration entre les agents afin de réaliser le but commun.

III.2. Objectifs et motivations

Dans notre étude du cas de « la gestion des réservations de voyageur au saison du Hadj en l'Algérie», on trouve les problèmes de planification décentralisée suivants:

1- la coopération entre les agences de tourisme n'est pas utilisée d'une façon efficace, à cause de l'absence un système de coordination entre eux (ex : ONAT, TVA).

2- problème de temps de réponse à cause des moyens de communication utilisé dans la négociation (téléphone, fax, ….).

3-dans le cas où un voyageur ne trouve pas un vol satisfaisant dans sa région, il doit se déplacer pour réserver d'une autre région.

4- Le système doit posséder un mécanisme pour coordonner ces activités.

Notre solution au premier problème est de définir des agents, chaque agent jouant le rôle d'une agence pour rendre la coopération entre les agences plus facile. Le deuxième point peut être résolu par la définition d'un protocole de négociation et communication efficace entre les agents. Le troisième point oblige ce protocole de permettre la négociation les agents des différentes régions. Le qutrieme point est résolu par l'utilisation d'un SMA de planification décentralisée pour les raisons mentionnées précédemment.

III.3. Architecture générale

L'architecture générale du notre système, illustrée à la figure 4.1, s'articule autour de quatre principaux niveaux en interaction :

III.3.1. Niveau initial

Ce niveau contient le problème global qui est grand et complexe. Il faut étudier ce problème et déterminer le but et comment décomposer en sous problèmes.

III.3.2. Niveau décomposition du problème global

Dans ce niveau, on décompose le problème global en sous problèmes pour faciliter la solution et gagner le temps et l'effort, on considère que chaque sous problème est associé à un agent qui possède un plan claire et déterminé. Chaque plan contient un ensemble de tâches et contraintes

Exemple :

L'agent1 possède le plan (Tâche1, Tâche2,…, Tâche_n + Contrainte1, Contrainte2,…Contrainte_m).

La Contrainte soit une expression mathématique, ou une expression logique.

III.3.3. Niveau d'exécution des plans

Les agents peuvent exécuter leurs plans, communiquer entre eux et échanger les tâches. On conséquence les plans peuvent être modifié. Le problème demeure de trouver une assignation qui satisfait les contraintes. Par contrainte, chaque agent décide de la tâche avec une certaine autonomie, et les agents le font en parallèle.

Chaque agent n'a pas une vue globale des assignations, mais il peut communiquer avec les autres agents (selon le graphe de contrainte) pour connaître leurs tâches. Un système consistent à avoir les agents qui communiquent entre eux, chacun mettant à jour son plan, de sorte que le processus converge éventuellement vers une assignation complète et satisfaisante.

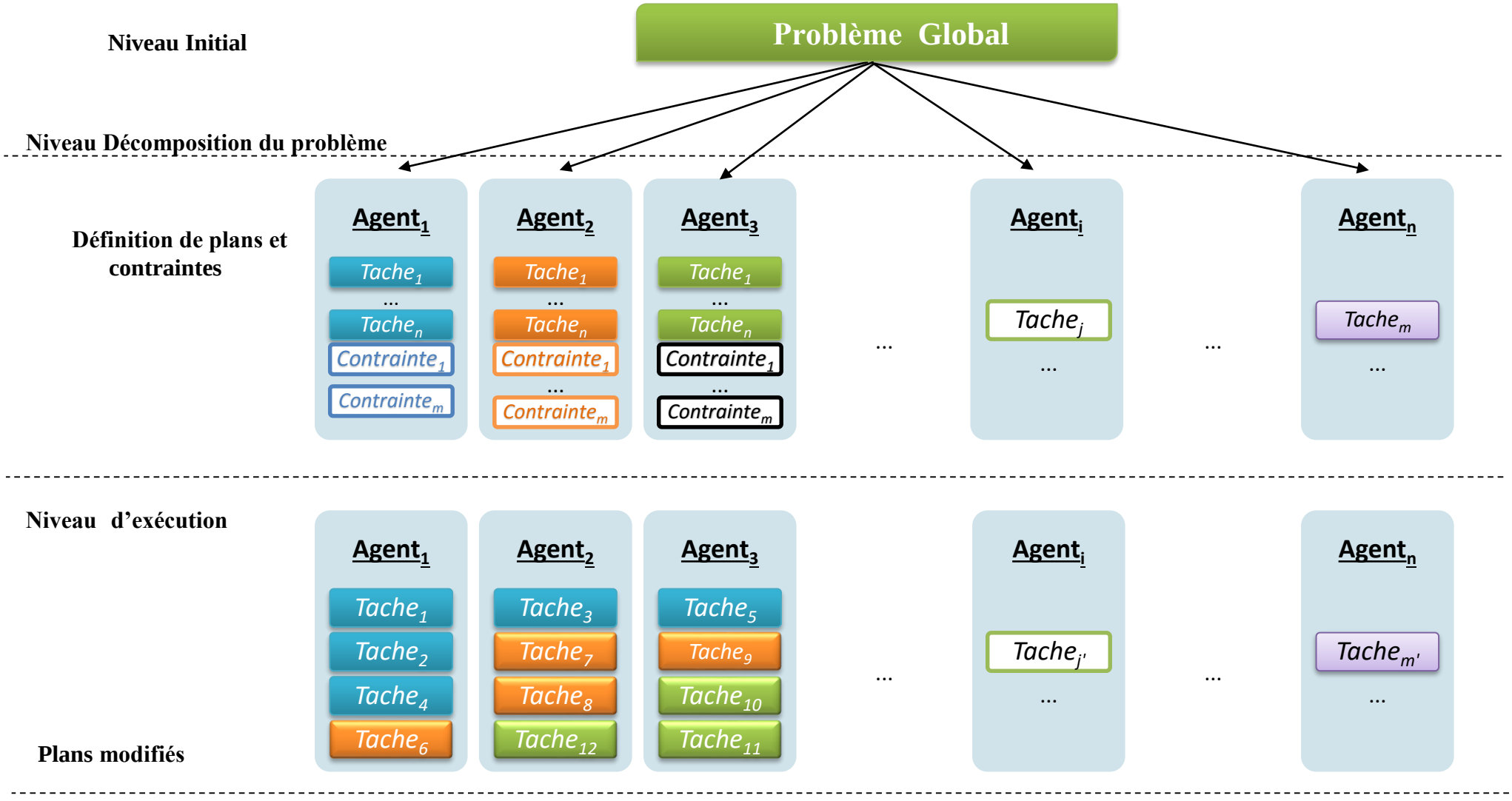


Figure : III.1 Architecture globale du système

III.4 Conception globale

La figure (**Figure III.2**) représente l'architecture générale de notre système, qui se compose de trois types d'Unités: Unité Agence(**UA**), Unité Aéroport (**U**Aero) et Unité Région(**UR**).

Le principe de fonctionnement du système est le suivant :

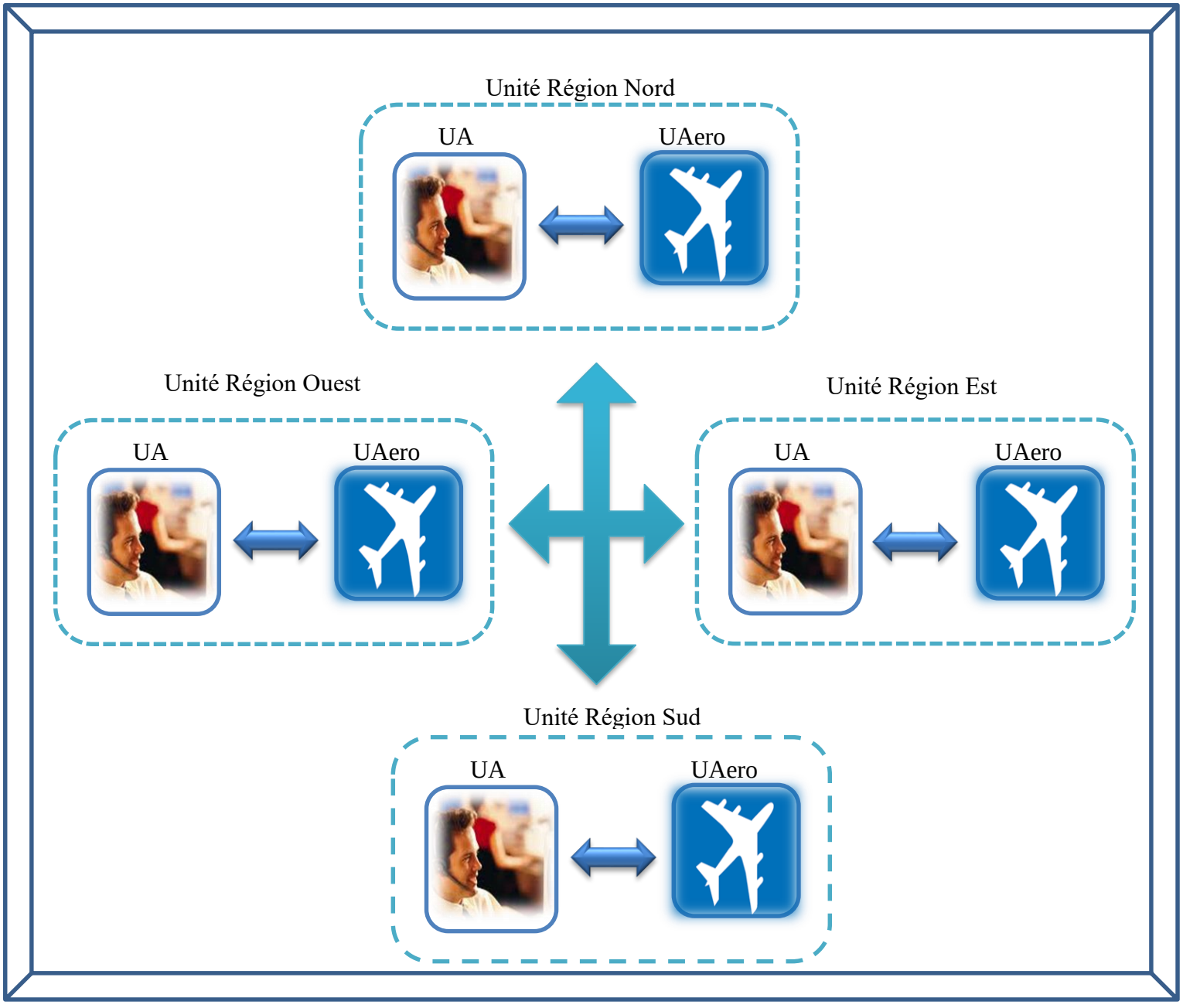


Figure : III. 2 Architecture globale du système étudié

➢ ***Unité Région (UR):***

Le système est composé de quatre régions (Nord, Sud, Est, Ouest), on a pour chaque région un seul aéroport et une seule Unité Région.

Cette unité est chargée de coordonner avec les autres Unité Région, elle comprend deux unités élémentaires : Unité Aéroport et Unité Agence **(Figure III.3)**.

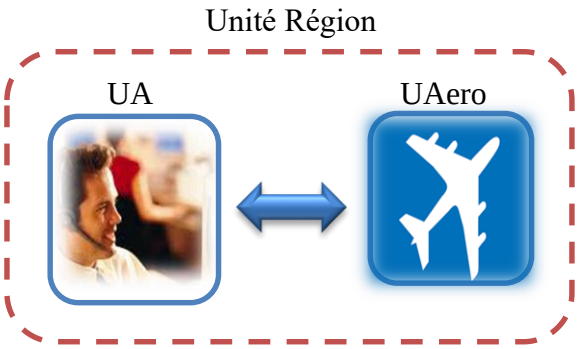


Figure III.3 Architecture Unité Région

➢ ***Unité Aéroport (UAero) :***

Cette unité envoi à l'Unité Agence les listes des vols disponibles dans son Aéroport. Ensuite, l'unité Agence envoie les listes des voyageurs après la confirmation de la réservation au niveau Unité Agence (**Fig.III.4).**

Cette unité contient un ensemble d'agents de type Agent Aéroport (***AAero***), chaque agent représente une compagne aérienne (comme : Aire Algérie, Tassili, …etc).

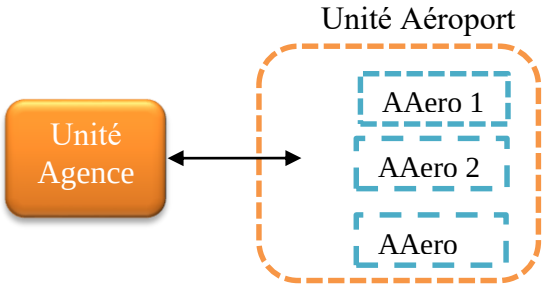


Figure III.4 Architecture Unité Aéroport

➢ ***Unité Agence (UA) :***

Cette unité est chargée de la communication et la négociation entre les agences de tourisme de sa région ou des autres régions.

Cette unité contient un ensemble d'agents de type Agents Agence (AA), chacun représente une agence de tourisme d'une ville de la région. **(Figure III.5)**.

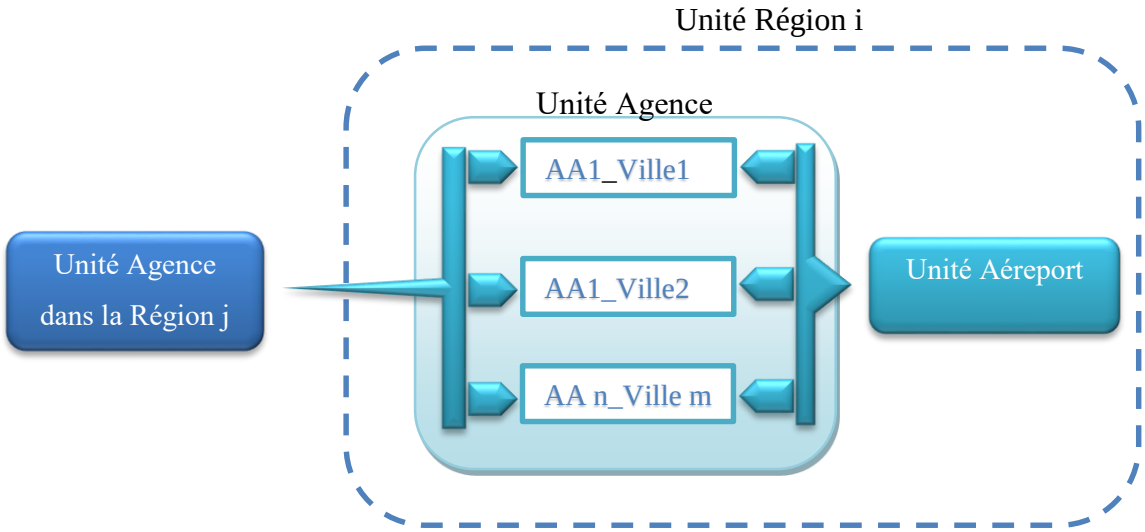


Figure III.5 Architecture Unité Agence

III.5 Conception détaillée

III.5.1. Formulation du DCSP

Nous formalisons ici le problème de ***« la gestion des réservations de voyageur à la saison du Hadj dans l'Algérie »*** par :

III.5.1.1. Un ensemble fini de variables

Variable	Définition	Domaine
R	Régions	$\{ R_1, \dots, R_r, \dots R_T \}$
NT	Nombre total de Hadj dans le système	Nombre Naturel
NTR	Nombre total de Hadj par région	$\{ NTR_1, \dots, NTR_r, \dots, NTR_T \}$
Comp	Compagne Aérienne	$\{ Comp_1, \dots, Comp_i, \dots, Comp_n \}$
Vol	Vol proposé par une Compagne Aérienne	$\{ Vol1_{1\,1}, \dots, Vol_{ij}, \dots, Vol_{nm} \}$
C	Capacité de Vol	$\{ C_{1\,1}, \dots, C_{ij}, \dots, C_{nm} \}$
A	Agence de tourisme	$\{ A_1, \dots, A_k, \dots, A_p \}$
N	nombre de postes affectés à une agence par vol	$\{ N_{1\,1\,1}, \dots, N_{ijk}, \dots, N_{nmp} \}$
N'	Nouvelle valeur du **N** après un changement avec une autre agence (même région) sur le nombre de postes dans un vol,	$\{ N'_{1\,1\,1}, \dots, N'_{ijk}, \dots, N'_{nmp'} \}$ p' : le nombre d'agences effectués le changements

Tableau III.1 : Ensemble de variables définies du problème

III.5.1.2 Un ensemble fini de contraintes sur les variables

✓ **Contrainte1** : Le nombre total de voyageurs (Hadj) est devisé sur quatre régions :

$$\sum_{r=1}^{T} NTR_r \leq NT \quad \text{.....(1)}$$

✓ **Contrainte2** : Dans une région, chaque compagne aérienne possède un nombre de Vols, et chaque Vol a une capacité déterminée :

$$\forall R_r : \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} \geq NTR_r \quad \text{..... (2)}$$

✓ **Contrainte3** : Dans une région, et une compagne, chaque Vol est devisé sur les agences de tourisme de cette région :

$$\forall Comp_i, \forall Vol_{ij} : \sum_{k=1}^{p} N_{ijk} \leq C_{ij} \quad \text{.....(3)}$$

✓ **Contrainte4** : en cas du changement entre deux agences sur le nombre de postes d'un Vol d'une compagne :

$$\forall Comp_i, \forall Vol_{ij} : \sum_{k=1}^{p'} N'_{ijk} + \sum_{k=p'+1}^{p} N_{ijk} \leq C_{ij} \text{ ..... (4)}$$

III.5.2. Classes d'agents nécessaires

➢ ***Agent humain***

(Le voyageur) : Qui saisit le formulaire de réservation.

➢ ***Agent Aéroport (AAero)***

Chaque Agent Aéroport représente une compagne aérienne (comme Air Algérie, Tassili Air lines, …), et il fournit des listes de vols prédéterminées aux déférents Agents Agence de tourisme (dans la même région).

A la fin de l'opération de la réservation il reçoit des listes des voyageurs de chaque Agent Agence de tourisme (dans sa région).

➢ ***Agent Agence (AU)***

Chaque Agent Agence représente une agence de tourisme d'une ville dans une région.

Au début, Il reçoit sa liste de vols (prédéterminée) de chaque Agent Aéroport (de sa région), aussi il joue le rôle d'un moyen ou un canal d'entrés des requêtes de réservation au système. Il fournit au voyageur, un formulaire pour lui permettre de faire une réservation.

Puis, il cherche le vol satisfaisant à la requête dans les listes fournies par les Agents Aéroport et affiche les informations du billet à la fin de la réservation.

Aussi, en cas il ne trouve pas le vol satisfaisant, il va négocier avec les autres Agents Agence de la même région puis ceux des autres régions pour trouver le vol satisfaisant.

III.5.3. Architectures internes des Agents

On considère qu'un agent est caractérisé par des savoirs (Connaissance sur son état, son environnement, etc...), des savoir-faire (des actions ou des interactions qu'ils peuvent mener), et peut avoir des modules (Module de communications, Module traitement, Module de mobilité,…)

Agent Aéroport (AAero)

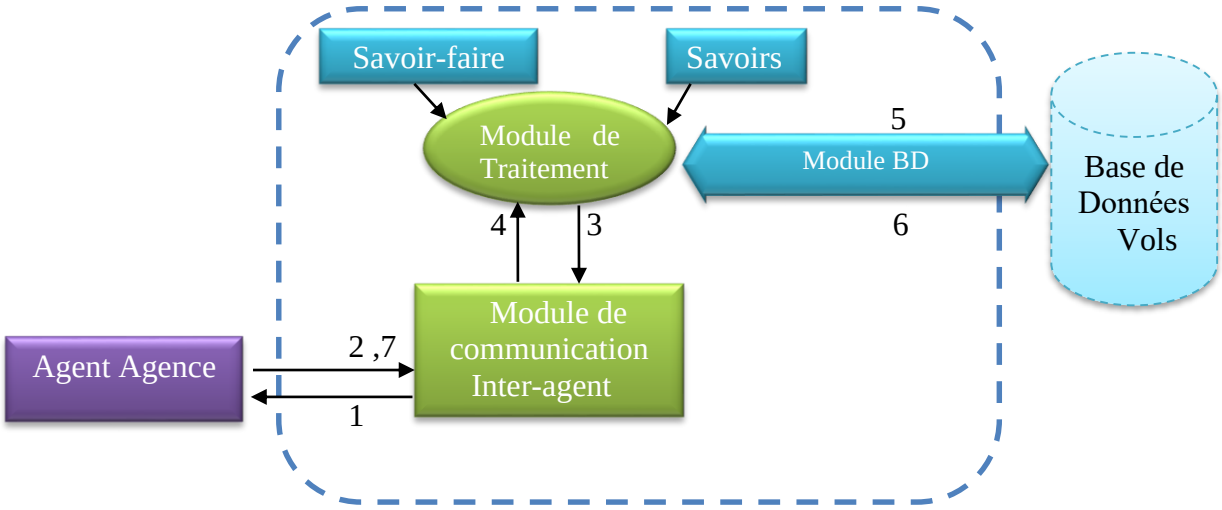


Figure III.6 : Architecture interne de l'Agent Aeroport

➢ ***Savoirs : il connait***

- La liste de voyageurs réservés.
- Tous les Agents Agence de sa région.

➢ ***Les savoir-faire***

- communiquer avec l'Agent Agence de sa région.

- ***Module de communication inter-agent***

Il envoie

- 1 : à l'Agent Agence de tourisme un message contient la liste de vols prédéterminée (dans la même région).

Il reçoit

- 2 : de l'Agent Agence un message contient la liste de voyageurs réservés.
- 7 : de l'Agent Agence un message contient une demande de MAJ sur sa liste de vols prédéterminée.

- ***Le module de traitement :***

- 3 : construire les messages (liste des vols) à envoyer vers les Agents Agence par le Module de communication Inter-agent.
- 4 : traitement des listes de voyageurs réservés.

- ***Module de base de données :***

- 5 : Gestion des listes de vols prédéterminées d'une Agence de tourisme.
- 6 : Enregistrement de la liste des voyageurs réservés.

Agent Agence (AA)

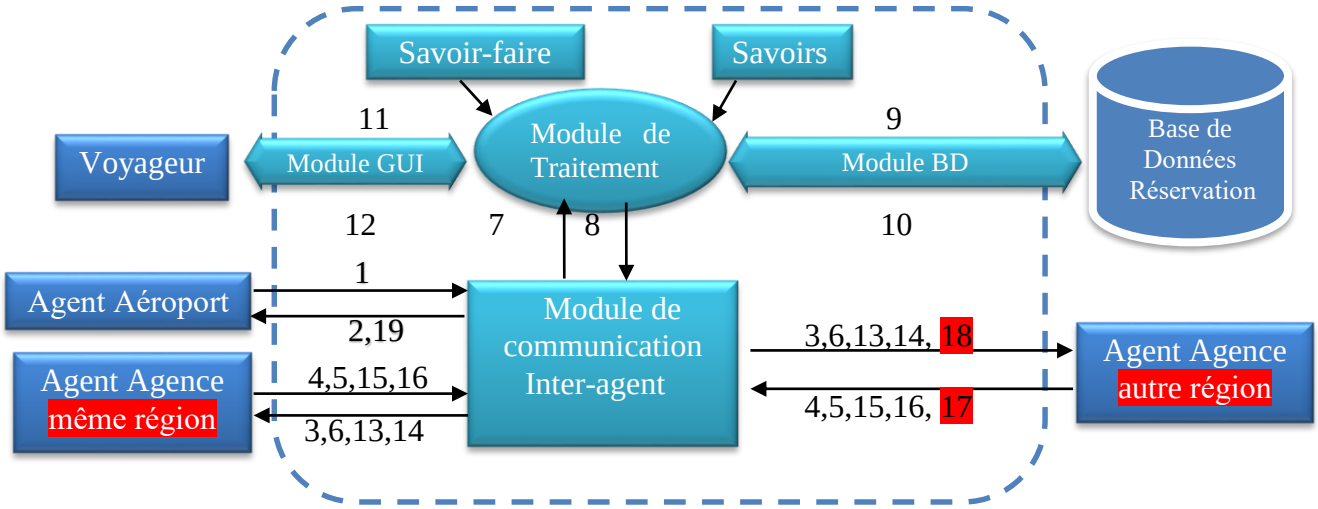


Figure III.7: Architecture interne de l'Agent Agence

- ***Savoirs : il connait***

- Sa liste de Vols prédéterminé.
- Agent Aéroport de sa région.
- Tous les Agents Agence de sa région.

- Tous les Agents Agence des autres régions.

➢ ***Les savoir-faire***

- Il interagit avec le voyageur.
- Communiquer avec l'Agent Aéroport de sa région.
- Négocier avec les Agents Agence de sa région.
- Négocier avec les Agents Agence des autres régions.

➢ ***Module de communication inter-agent:***

Il reçoit

- 1 : de l'Agent Aéroport de sa région un message contient la liste de vols prédéterminée.
- 4: d'un autre Agent Agence (même ou autre région) un message de résultat de négation pour une réservation (interne ou externe).
- 5: d'un autre Agent Agence (même ou autre région) un message de négation pour une réservation (interne ou externe).
- 15 : d'un autre Agent Agence (même ou autre région) un message pour confirmer la réservation et fin de la négociation (interne ou externe).
- 16 : d'un autre Agent Agence (même ou autre région) un message pour annulation de la réservation et fin de la négociation (interne ou externe).
- 17 : en cas de réservation hors région (externe), il reçoit d'un autre Agent Agence **d'une autre région** un message contient l'info de billets de réservation et fin de la négociation externe.

Il envoie

- 2 : à l'Agent Aéroport un message qui contient la liste de voyageurs réservés.
- 3 : en cas il ne trouve pas le vol demandé par le voyageur, il diffuse aux autres Agents Agence (même ou autre région) un message de négation pour une réservation (interne ou externe).
- 6 : à un autre Agent Agence (même ou autre région) un message contient le résultat de négociation (interne ou externe) pour une réservation.
- 13 : à un autre Agent Agence (même ou autre région) un message pour confirmer la réservation et fin de la négociation (interne ou externe).
- 14 : diffusion aux autres Agents Agence (même ou autre région) un message pour annulation de la réservation et fin de de la négociation (interne ou externe).

- 18 : en cas de réservation hors région (externe), il envoie d'un autre Agent Agence **d'une autre région** un message contient les infos de billets de réservation et fin de la négociation externe.
- 19 : à l'Agent Aéroport un message contient une demande de MAJ sur sa liste de Vols prédéterminée après la négociation avec autre Agent Agence de la même région.

➢ ***Le module de traitement***

- 7 : traitement de données des massages reçus (soit : listes des vols, résultat de négociation, demande de négociation, …).
- 8 : construire les messages à envoyer (soit : liste des voyageurs réservés, demande de négociation, résultat de négociation, …) vers les différents Agents par le Module de communication Inter-agent.

➢ ***Module de base de données***

- 9 : Enregistrement des listes des vols prédéterminés, demandes de vols de voyageurs et leurs Billets de voyages, demandes de négociation pour les réservations arrivée des Agent d'Agence et les résultats de négociations.
- 10 : Ramener les voyageurs réservés, résultat de recherche d'un vol demandé, les demandes non satisfaisais dans la BD locale.

➢ ***Module de Graphic User Interface (GUI)***

- 11 : il reçoit les données du formulaire de réservation et les transfère au module de traitement.
- 12 : recevoir les informations du billet de voyage de Module de traitement pour les montrer au voyageur.

III.5.4. Les bases de données utilisées dans le système

Le Système utilise les types des Bases de données suivants :

- **BD_Vol** : chaque compagne aérienne possède une Base de données pour les vols.
- **BD_Resevation** Base de données pour les réservations des voyageurs pour chaque agence de tourisme de chaque ville.

Dans ce que suit, nous présentons les structures logiques de ces différentes bases de données :

- **BD_Vol :** Elle contient quatre tables **:**
  - ***Table Vol*** : Cette table contient les informations des vols d'une Compagne aérienne dans une région ;

{

Num_Vol: entier ;

Ville_Départ: chaîne de caractère ; // ville de la région

Ville_d'arrivée: chaîne de caractère ;

Date_Aller : DateHeur ;

Date_Retour : DateHeur ;

Nbr_Total_postes : entier ;

}

Num_Vol	Ville_Départ	Ville_d'arrivée	Date_aller	Date_Retour	Nbr_Total_postes
12333	Costantine	Djeddah	02-10-2013 12 : 36 AM	22-10-2013 07 : 45 AM	150
	Costantine	Makka			
	Costantine	Riad		Date_aller	

Tableau III.2 : Table Vol

  - ***-Table Agence_Tourisme*** : Cette table contient les informations de toutes les Agences de tourisme dans la région ;

{

Num_Agence : entière auto incrément;

Raison_sociale : chaîne de caractère ;

Ville : chaîne de caractère ;

Adresse : chaîne de caractère ;

Site_Web : chaîne de caractère ;

}

N°Agence	Raison_sociale	Ville	Adresse	Site_Web
123	ONAT	Annaba	1, Rue Tarek Ibn Ziad	www.onat.dz

Tableau III.3 : Table Agence_Tourisme

➢ ***Table Liste_Predetermine*** : Cette table contient les nombres de postes prédéterminés (par l'agence de voyage aérienne) pour chaque Agence de tourisme dans la même région ;

{

Num_Agence : entière auto incrément;

Num_Vol : chaîne de caractère ;

Nbr_poste : chaîne de caractère ; // le nombre de poste déterminé pour une Agence

// $\sum Nbr\_Postes\ de\ Voli \leq Nbr\ Total\ de\ Potes\ Voli$

}

N°Agence	N°Vol	Nbr_postes
123	12333	40
201	12333	60

Tableau III.4 : Table Liste_Predetermine

➢ ***Table Voyageur_reserve :*** *Cette table contient les informations des voyageurs réservés et leurs billets de voyage.*

{

Ref_Billet : entière; //

Nom : chaîne de caractère ;

Prénom : chaîne de caractère ;

Num_vol : chaîne de caractère ;

Num_Agence : chaîne de caractère ;// N° d'agence de tourisme

}

Ref_Billet	Nom	Prénom	Num_vol	N°_Agence
2424 123 6789004	Moutwakil	Abdallah	12333	123

Tableau III.5 : Table Voyageur_reserve

- **BD_Réservation :** Elle contient quatre tables **:**
  - ***Table Vol_Interne*** : Cette table contient les informations des vols de toutes les Compagnes aérienne s dans une région ;

{

Compagne_arrienne : chaîne de caractère ;

Num_Vol_Int: entier ;

Ville_Départ: chaîne de caractère ;

Ville_d'arrivée: chaîne de caractère ;

Date_Aller : DateHeur ;

Date_Retour : DateHeur ;

Nbr_postes : entier ;

}

N°_Vol	Ville_Départ	Ville_d'arrivée	Date_aller	Date_retour	Nbr_postes	Compagne_arrienne
12333	Costantine	Djeddah	02-10-2013 12 : 36 AM	22-10-2013 07 : 45 AM	40	Air Algérie

Tableau III.6 : Table Vol_Interne

  - ***Table Vol_Externe*** : Cette table contient les informations des vols de toutes les Compagnes aérienne s hors la région (cas de réservation externe);

{

Num_Vol_Ext: entier ;

Ville_Départ: chaîne de caractère ;

Ville_d'arrivée: chaîne de caractère ;

Date_Aller : DateHeur ;

Date_Retour : DateHeur ;

Région : chaîne de caractère;

Compagne_arrienne : chaîne de caractère ;

}

N°_Vol	Ville_Départ	Ville_d'arrivée	Date_aller	Date_retour	Région	Compagne_arrienne
1277	Alger	Djeddah	18-09-2013 12 : 36 AM	10-10-2013 07 : 45 AM	Centre	Air Algérie

Tableau III.7 : Table Vol_Externe

- ***Table Reservation_Inerne*** : Cette table contient les informations de Formulaires de demandes des vols et Bielles, où les vols sont dans la même région.// soit : le poste trouvé de cette agence ou d'une autre de même région,

{

Num_dem: entière; //

Nom : chaîne de caractère ;

Prénom : chaîne de caractère ;

Adresse : chaine de caractère ;

Date : Date;

Num_Vol_*Int* : entier **;**

Compagne_arrienne : chaîne de caractère ;

Ref_Biellet : entier ; // 11 chiffres

}

Num_dem	Nom	Prénom	Adresse	Com-arrienne	Date	N° Vol *Inerne*	Ref_Billet
56	Moutwakil	Abdallah	05, Rue Hamdi Ahmed, Annaba	Air Algérie	12-10-2013	12333	2424 123 6789004

Tableau III.8 : Table Reservation_Inerne

Table Reservation_Externe : Cette table contient les informations de Formulaires de demandes des vols et copies des Billets, où les vols sont hors la région.//

{ **Num_dem**: entière; //

Nom : chaîne de caractère ;

Prénom : chaîne de caractère ;

Adresse : chaine de caractère ;

Date : Date;

Num_Vol_Ext : entier **;**

Compagne_arrienne : chaîne de caractère ;

Ref_Biellet : entier ; // 11 chiffres

}

Num_dem	Nom	Prénom	Adresse	Date	Com-arrienne	N° Vol	Ref_Billet
56	Moutwakil	Abdallah	05, Rue Hamdi Ahmed, Annaba	12-10-2013	AIR Algérie	12333	2424 123 6789004

Tableau : III.9 Table Reservation_Externe

III.5.5. Interactions dans le système

Nous présentons ci-dessous le protocole de planification (Interaction) entre les différents agents du SMA. Un protocole planification définit une séquence d'actes (pour négociation, communication,…) qui seront échangés entre agents.

Nous commençons par présenter les actes utilisés pour la communication et négociation entre agents, ensuite nous modélisons les différentes interactions par des diagrammes de séquence AUML.

- ***Actes échangés entre les agents***

Nous définissons les actes suivants :

- ***Liste_Vol_Pre :*** listes de vols prédéterminées d'une agence de tourisme.
- ***Liste_Voygeur_Resrv :*** liste de voyageurs réservés.
- ***Negociation_Interne :*** négociation pour « demande une réservation » avec des agents Agence dans la même région.
- ***Rep_Negociation_Interne :*** Repense de la demande d'une négociation interne.
- ***Confirmer_Reservation_Interne :*** Confirmation de réservation interne.
- ***Annuler_Reservation_Interne :*** annuler la demande de réservation interne.
- ***MAJ_Liste_Vol_Pre :*** information de l'agent AAero pour une changement dans le nombre de poste dans une liste des postes prédéterminés d'une agence de même région.
- ***Negociation_Externe :*** négociation pour « demande une réservation » avec des agents Agence dans une autre région.
- ***Rep_Negociation_Externe :*** repense de la demande d'une négociation Externe.
- ***Confirmer_Reservation_Externe :*** Confirmation de réservation externe.
- ***Annuler_Reservation_Externe :*** annuler la demande de réservation externe.
- ***Copie_Billet :*** envoi d'une copie de billet réservé par un agent Agence d'une autre région.

Le tableau suivant (**Tableau : III.9**) présente pour chacun des actes : L'émetteur, le récepteur et son code (pour l'identifier sur le diagramme de séquence du protocole d'interaction).

N°	Code Acte	Sender	Receiver	Content
1	Liste_Vol_Pre	AAero	AA	liste de vols prédéterminée
2	Liste_Voygeur_Resrv	AA	AAero	N°Agence, Une liste de {ref_biellet, num_vol, nom, prenom, … }
3	Negociation_Interne	AA	AA	date
4	Rep_Negociation_Interne	AA	AA	«Disponible»{Info_vol} «NonDisponible»{ }
5	Confirmer_Reservation_Interne	AA	AA	« OK »
6	Annuler_Reservation_Interne	AA	AA	« Annuler »
7	MAJ_Liste_Vol_Pre	AA	AAero	{N°Agence, N° Vol, nbr_postes (+/-)}
8	Negociation_Externe	AA	AA	date
9	Rep_Negociation_Externe	AA	AA	«Disponible»{Info_vol} «NonDisponible»{ }
10	Confirmer_Reservation_Externe	AA	AA	« OK », Formulaire de Voyageur {nom, prénom,…}
11	Annuler_Reservation_Externe	AA	AA	« Annuler »
12	Copie_Billet	AA	AA	Ref_Billet, info_vol :{vile_dep, ville_arriv, date, …}

Tableau : III.10 les actes échangés entre les agents

A. Interactions ou niveau de chaque Unité Région :

Dans l'Unité Région, les Agents Agence peuvent interagir entre eux et aussi avec les Agents Aéroport dans deux Cas :**Réservation normale (sans négociation)**, **Réservation avec négociation Interne.**

- ***Réservation normale (sans négociation)***

1 : AAero envoi ***Liste_Vol_Pre*** à l'agent AA.

2 : AA reçoit ***Liste_Vol_Pre*** et enregistre les données dans la BD.

3 : Le Voyageur saisit le **formulaire et demande** la réservation du l'agent AA.

4 : AA reçoit la demande et recherche le vol satisfaisant dans sa base de données.

Si (le vol satisfaisant est trouvé) alors

6 : Imprime le billet du voyage et enregistre la réservation.

7 : Le voyageur reçoit le billet de voyage.

8 : AA envoie ***Liste_Voyageur_Resrv*** à AAero

9 : AAero reçoit ***Liste_Voyageur_Resrv*** et l'enregistre dans sa BD

Sinon

C'est le cas de **Réservation avec négociation Interne** (dans la section suivante)

FinSi

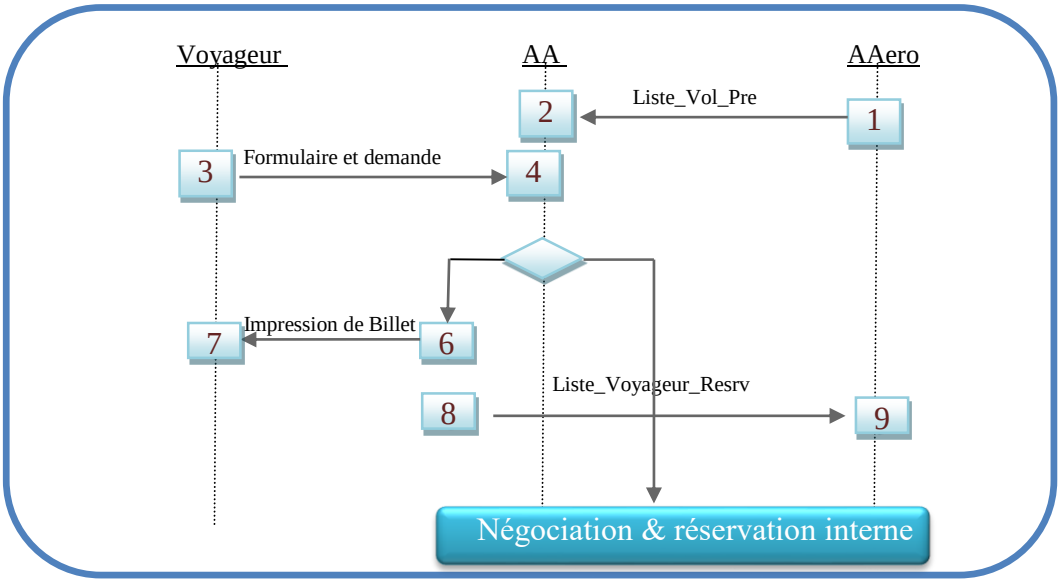


Figure III.8 Diagramme d'interaction (Réservation normale).

- **Réservation avec négociation Interne**

10 : AA **demandeur** envoie (Diffuse) ***Negociation_Interne*** aux autres agents AA dans sa région (même Unité région).

11: Un agents AA donneur d'une autre région reçoit ***Negociation_Interne*** de l'agent AA demandeur et recherche le vol satisfaisant dans sa base de données.

Si (le vol satisfaisant est trouvé) alors

12: l'agent AA donneur, envoie ***Rep_Negociation_Interne****(disponible)* ***à*** AA demandeur.

13: L'agent AA demandeur reçoit ***Rep_Negociation_Interne*** *(disponible)*.

Si (elle est la première repense reçue) alors

14: L'agent AA demandeur envoie ***Confirmer_Reservation_Interne*** *(OK)* à AA donneur.

:15agent AA donneur reçoit ***Confirmer_Reservation_Interne*** (OK) de AA demandeur.

16: L'agent AA demandeur envoie ***MAJ_Liste_Vol_Pre****(+)* à l'agent AAero.

17: L'agent AAero reçoit ***MAJ_Liste_Vol_Pre****(+)* de l'agent AA demandeur.

18 : L'agent AA donneur envoie ***MAJ_Liste_Vol_Pre****(-)* à l'agent AAero.

19: L'agent AAero reçoit ***MAJ_Liste_Vol_Pre****(-)* de l'agent AA donneur.

Sinon

20 : L'agent AA demandeur envoie ***Annuler_Reservation*** à AA donneur.

21: L'agent AA donneur reçoit ***Annuler_Reservation*** de AA demandeur.

FinSi

Sinon

22: l'agent AA donneur, envoie ***Rep_Negociation_Interne*** *(non disponible)* à AA demandeur.

23: l'agent AA demandeur de la demande reçoit ***Rep_Negociation_Interne*** *(non disponible)*.

C'est le cas de Réservation *avec* négociation *Externe*

FinSi

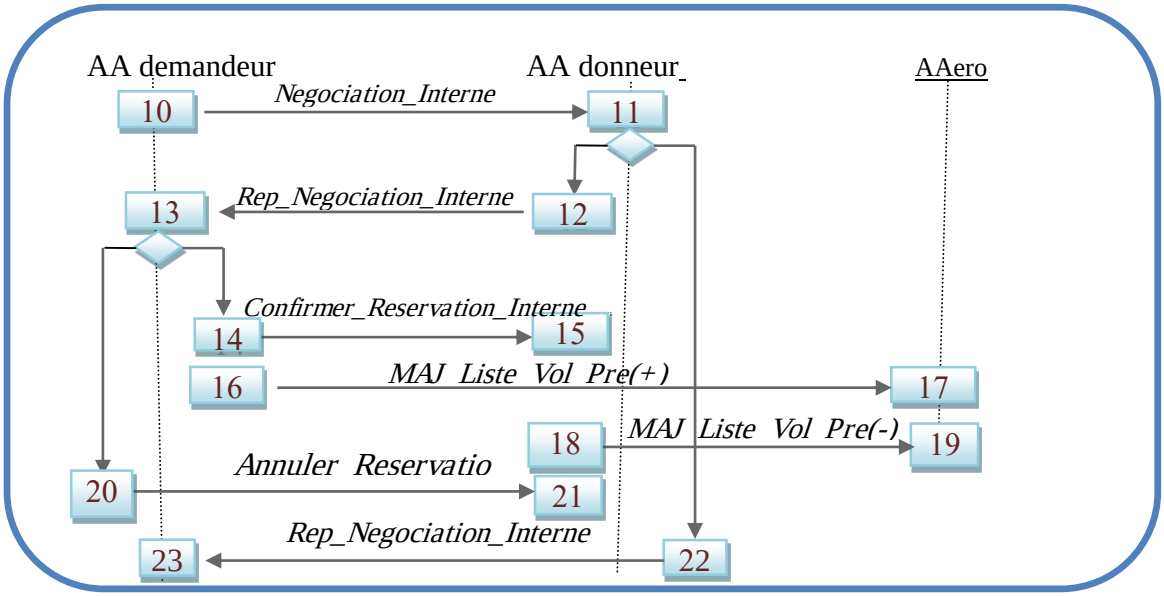


Figure III.9 Diagramme d'interaction (Réservation avec négociation Interne).

A. Interactions entre Unités Région :

Un Agent Agence d'une région peut interagir avec des Agents Agence d'une autre Unité région en cas de **Réservation avec négociation Externe**, s'il ne trouve pas le vol satisfaisant ni dans sa BD ni dans les BDs des autres Agents Agence dans sa région.

- **Réservation avec négociation externe**

24 : AA **demandeur** envoie (Diffuse) ***Negociation_Externe*** aux autres agents AA d'une autre Unité Régions

25 : Un agents AA donneur reçoit ***Negociation_Externe*** de l'agent AA demandeur et recherche le vol satisfaisant dans sa base de données.

Si (le vol satisfaisant est trouvé) alors

26: l'agent AA donneur, envoie ***Rep_Negociation_Externe****(disponible)* à AA demandeur de la demande.

27 : L'agent AA demandeur de la demande reçoit ***Rep_Negociation_Externe*** *(disponible)*.

Si (elle est la première repense reçue) alors

28: L'agent AA demandeur envoie ***Confirmer_Reservation_Externe*** *(OK)* à AA donneur.

29: L'agent AA donneur reçoit ***Confirmer_Reservation_Externe****(OK)* de AA demandeur.

30: L'agent AA donneur enregistre la demande, crée le billet et envoyer une ***Copie_Billet*** à AA demandeur.

31: L'agent AA demandeur reçoit ***Copie_Billet*** de AA donneur et l'enregistre dans la table « réservation externe » dans sa BD.

32 : L'agent AA demandeur Imprime la **copie du billet** du voyage.

33: Le voyageur reçoit le billet de voyage.

34 : AA donneur envoie ***Liste_Voygeur_Resrv*** au AAero de sa région.

35: AAero reçoit ***Liste_Voygeur_Resrv*** de AA donneur de sa région et l'enregistre dans sa BD.

Sinon

36 : L'agent AA demandeur envoie ***Annuler_Reservation_Externe*** à AA donneur.

37: L'agent AA donneur reçoit ***Annuler_Reservation_Externe*** de AA demandeur.

FinSi

Sinon

38: l'agent AA donneur, envoie ***Rep_Negociation_Externe*** *(non disponible)* à AA demandeur.

39: l'agent AA demandeur de la demande reçoit ***Rep_Negociation_Externe****(non disponible).*

40 : L'agent AA demandeur va essayer de négocier avec les autres agents dans une autre région.

FinSi

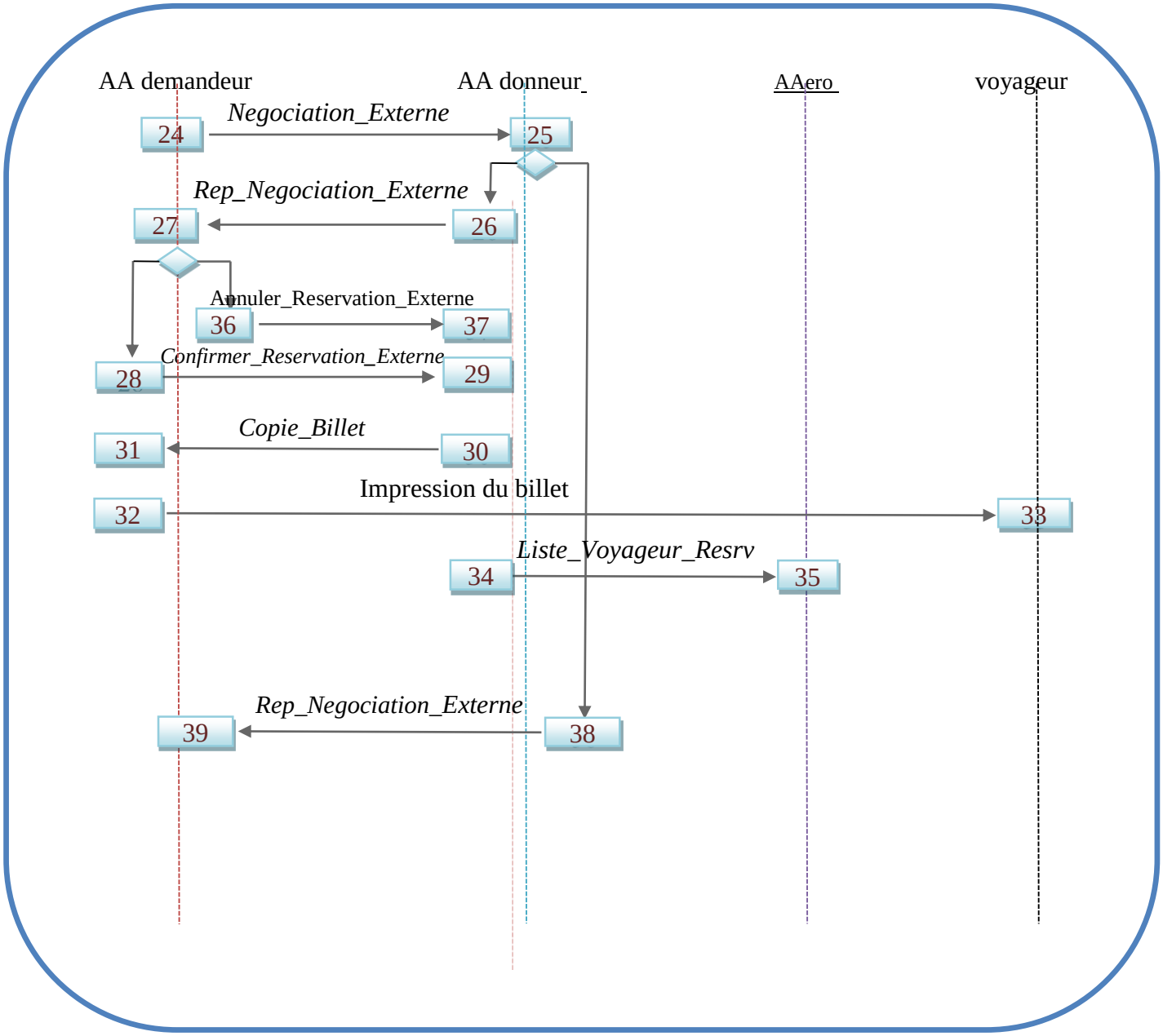


Figure III.10 Diagramme d'interaction (Réservation avec négociation externe).

➢ ***Interaction globale du système :***

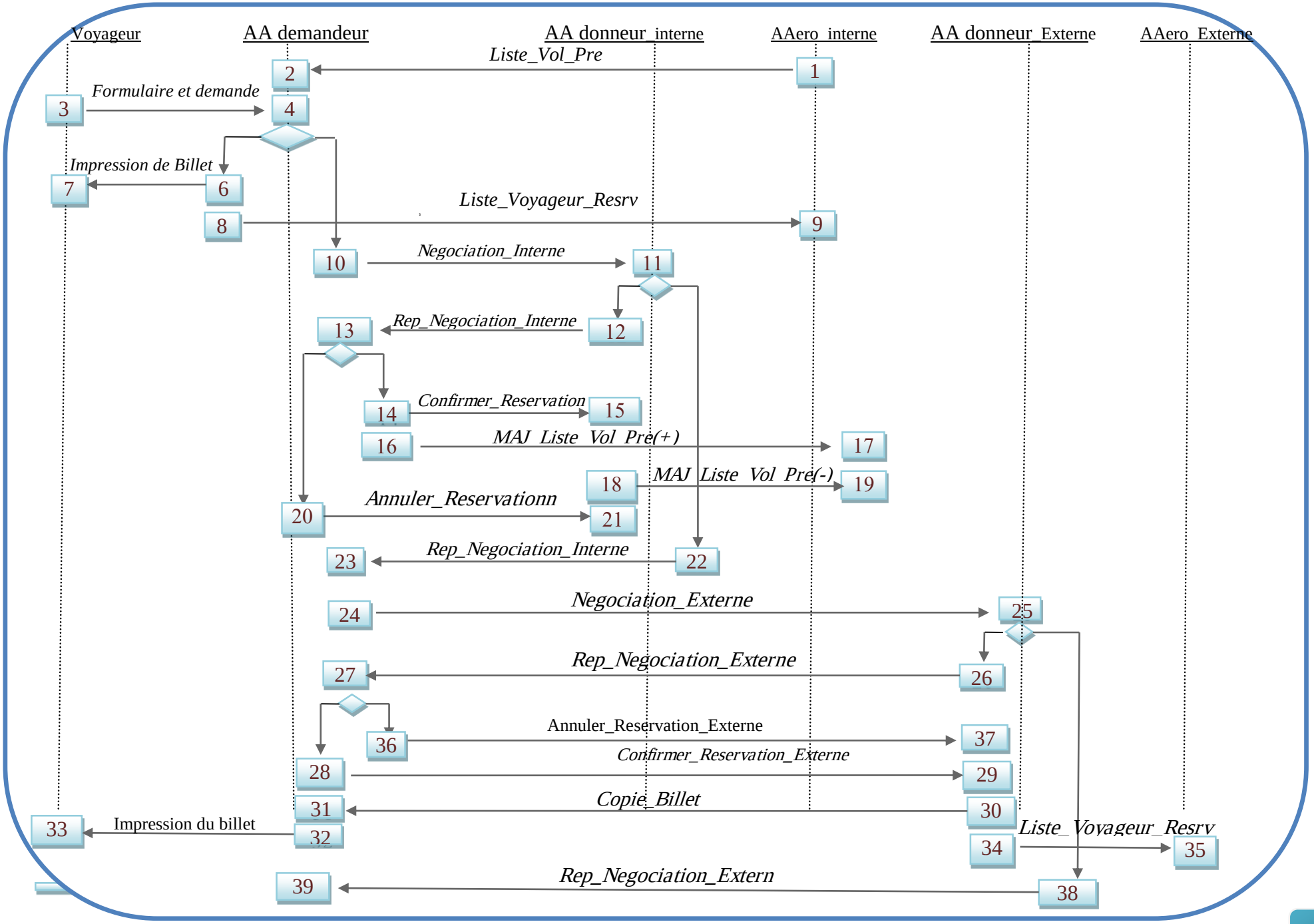


Figure : III.11 Diagramme d'interaction globale du système.

III.6 Conclusion

Dans ce chapitre, nous avons présenté la conception de notre système. Ce dernier a pour objectif de faciliter la recherche d'informations sur le web en utilisant l'architecture Système Multi-Agents. Les différents types d'agents utilisés et les communications entre eux ont été expliqués, en précisant le rôle de chaque agent pendant la recherche.

Dans le chapitre suivant nous allons décrire les outils utilisés pour réaliser notre système, avec quelques résultats obtenus.

Chapitre IV

Validation de l'approche

IV.1. Introduction

Dans ce chapitre, nous allons aborder l'aspect validation de notre approche Il s'agit ici d'expliquer l'environnement matériel et logiciel sur lequel notre système a été développé, en commençant par les outils et les langages de programmations utilisés, ensuite la plate-forme choisie, et comment nous avons l'exploité pour programmer les différents agents. Au cours de ce chapitre, nous allons montrer les techniques et les moyens avec les idées adoptées, et la manière dont nous avons implémenté chaque composant de ce système ainsi que ses algorithmes de fonctionnement.

IV.2. Outils de développement du système

IV.2.1. Outils matériels

On a utilisé les matériels suivants :

- Quatre PCs (**RAM** 2Go, **i3, HDD** 250 Go, Windows 7).
- UN Switch (**TP-LINK** 4 ports fast Ethernet switch).
- Les câbles de type **FTP** droits relient les PCs avec le Switch.

IV.2.2. Outils de programmation

Dans la réalisation de ce projet, nous avons employé beaucoup d'outils, à savoir : java, IDE Netbeans, API, … etc.

Dans les sections suivantes, nous allons présenter ces différentes techniques, en les organisant selon leurs catégories :

IV.2.2.1 Java

JAVA est un langage de programmation développé par Sun Microsystème (les premières versions datent de 1995). Il réussi à intéresser beaucoup de développeurs à travers le monde. Java assure les caractéristiques conceptuelles d'un agent ou bien d'un système d'agent.

➢ ***Caractéristiques de JAVA***

Java possède plusieurs caractéristiques importantes, en voici quelques-unes **[34] :**

Java est simple

C'est un langage simple à prendre en main, basé sur le langage C/C++ mais laisse de côté les sources de problèmes (pointeurs, structures, gestion de la mémoire, héritage multiple, macros etc.).

Java est orienté objet

Java est un langage purement orienté objet, tout est classe. Héritage simple.une librairie plus de classes est fournie.

Java est distribuée

Propose une API réseau standard. Cette dernière permet de manipuler, par exemple, les protocoles HTTP & FTP avec aisance.

Java est indépendante de l'architecture

Le bytecode généré n'est pas lié à un système d'exploitation en particulier. De ce fait, il peut être interprété très facilement sur n'importe quel environnement disposant d'une JVM.

Java est portable

Java est robuste

Pas de pointeurs.

Gestion de mémoire indépendante.

Mécanisme d'exceptions pour la gestion des erreurs.

Compilateur très contraignant.

Pas d'héritage multiple ni surcharge des opérateurs.

Java et sûr

4 niveaux de sécurité :

- Langage et son compilateur contraignant.

- Vérifier : vérifier le bytecode.

- Class Loader : le chargeur de classe.

- Security Manager : protection des fichiers et accès au réseau.

Java charge dynamiquement les classes suivant les besoins de l'application (pas d'édition de lien).

Java est multithreadé

Un Thread est un flot d'instruction s'exécutant en concurrence avec d'autres threads dans un même processus.

- ***Quelques notions de base JAVA***
  - *JDK* (*Java Development Kit) :* est le kit de développement basique que propose gratuitement la firme *Sun Microsystems*. Le Kit de développement comprend plusieurs outils, parmi lesquels :

- ***javac*:** le compilateur Java
- ***java*:** un interpréteur d'applications (machine virtuelle)
- ***applet viewer***: un interpréteur d'applets
- ***jdb***: un débogueur

- ***javap*:** un décompilateur, pour revenir du *bytecode* au code source
- ***javadoc***: un générateur de documentation
- ***jar* :** un compresseur de classes Java

- **Java Virtual machine (JVM)**

Est une machine virtuelle permettant d'interpréter et d'exécuter le bytecode Java. Cet outil est spécifique à chaque plate-forme ou couple (machine/système d'exploitation) et permet aux applications Java compilées en bytecode de produire les mêmes résultats quelle que soit la plate-forme.

La machine virtuelle la plus utilisée est celle de Sun Microsystems. Elle est gratuite, propriétaire jusqu'à la version 6 (stable) et libre à partir de la version 7.

- **API JAVA**

Cela signifie « Application Programming Interface ». Une API contient un ensemble de fonctions qui organise la programmation. L'API de java contient paquetages standard : java.awt, java.io, java.lang, java.net et java.util. **[43].**

IV.2.2.2 NetBeans IDE 7.1.2

Pour notre choix de l'environnement de développement Java, nous avons opté sur l'utilisation de NetBeans 7.1.2, est l'IDE (*Integrated Development Environment*) Open Source de Sun conçu pour concevoir, déployer et tester des applications développées sous différents langages (Java, PHP, Ruby, C/C++, etc.).

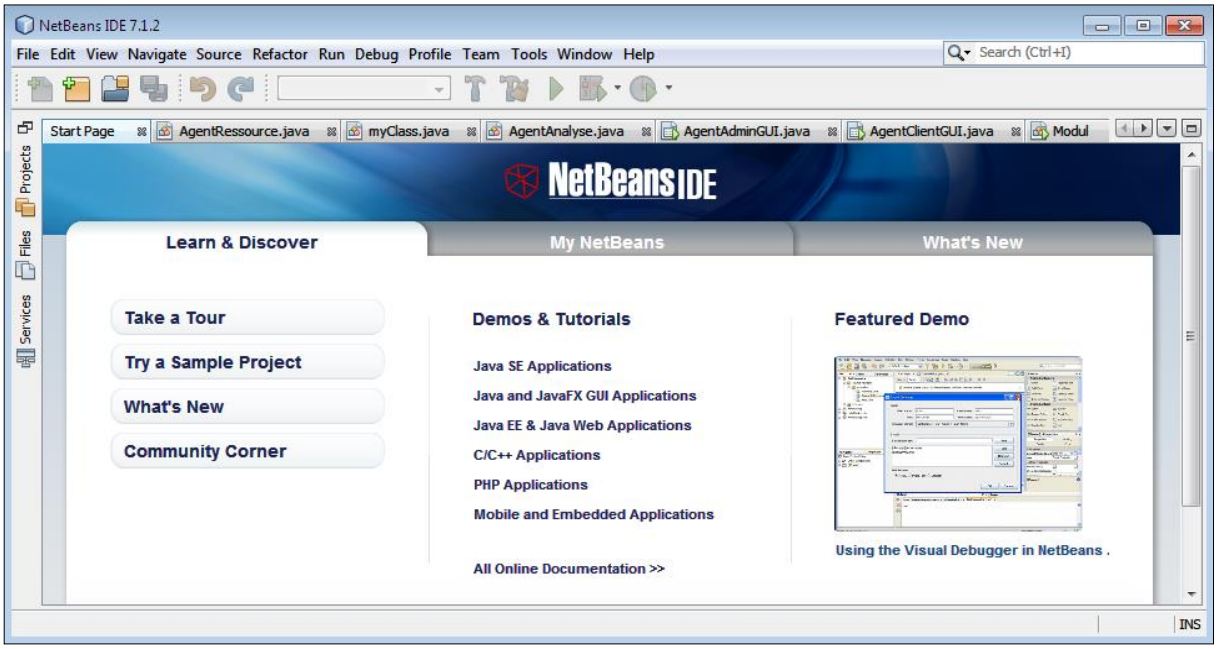


Figure IV.1 : l'IDE netbeanse7.1.2

IV.2.3. Outils de bases des données

Pour créer et gérer les bases de données nécessaires pour notre système, nous avons utilisé les outils suivants :

IV.2.3.1. MySql

MySql est un Système de Gestion de Base de Données Relationnelles. Avec plusieurs millions de serveurs installés en production dans le monde, MySql est devenu en quelques années le serveur de base de données libre le plus utilisé. Il est compatible avec les principaux standards SQL, MySql a su s'imposer comme la référence base de données des applications Internet, Intranet et Extranet.

A fin de créer une base de données en MySql on a utilisé l'outil **phpMyAdmin** qui offre une interface intuitive pour l'administration des bases de données MySql.

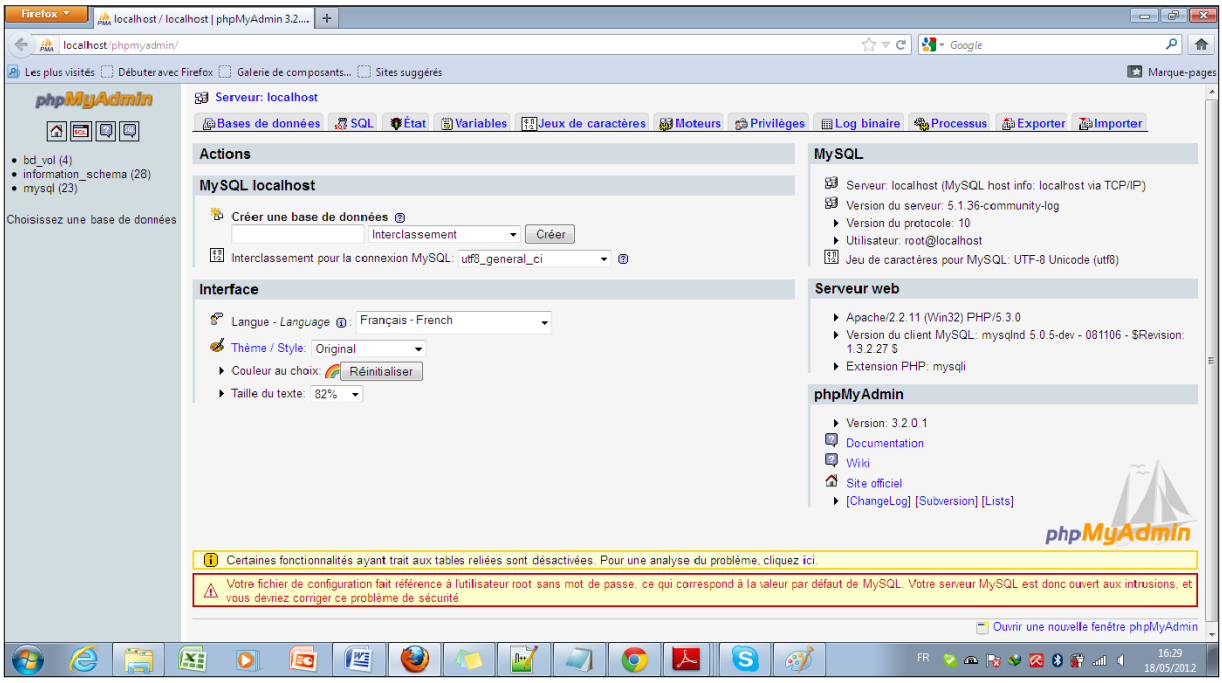


Figure IV.2 : L'outil phpMyAdmin

IV.2.3.2. JDBC (Java Data Base Connectivity)

JDBC est une API Java permettant d'accéder et manipuler des bases de données avec requêtes SQL. Cette API permet d'atteindre de manière quasi transparente les bases de données de type PostgreSQL, Oracle, MySql, … etc. (voir Figure IV.3).

```
import java.sql.DriverManager;
import java.util.Enumeration;
import java.util.Properties;

public class Connect {
  public static void main(String[] args) {

 //*******************Inclure API JDBC MySql
  Class.forName("com.MySql.jdbc.Drive");

 //*******************Creation paramettres de connexion
  String url = "jdbc:MySql://127.0.0.1//MyDB_name";
  String user = "useer";
  String passwd = "password";

 //*******************Connexion avec la BD dans MySql
  Connection conn = DriverManager.getConnection(url, user,
passwd);
  System.out.println("Connection Etablit !");
}
```

Figure IV.3 : Figure IV.3 : Pseudo code java JBDC pour Etablir une connexion avec serveur

IV.2.4. La plate-forme d'agent Aglet

Aglet est une API Java pour développer des agents. Elle a été développée par une équipe de chercheurs du laboratoire de recherche d'IBM à Tokyo au début 1995; son but est de fournir une plate-forme uniforme pour les agents mobiles dans un environnement hétérogène tel que celui de l'Internet **[44].**

IV.2.4.1 Définition d'un Aglet

Les aglets (**Ag**ents App**lets)** sont des objets Java qui peuvent se déplacer d'une machine à une autre. Ainsi, un aglet qui s'exécute sur un hôte peut stopper son exécution, se déporter vers un hôte distant et continuer cette exécution dans son nouvel environnement. [**44], [45**] :

IV.2.4.2. Architecture d'un Aglet

Les principaux éléments sont:

- **Aglet** :

Un aglet est un Objet Java autonome et réactif car il peut répondre (réagir) à des événements de son environnement.

- **Proxy** : Un proxy est un représentant d'un aglet. Il sert de bouclier à l'aglet contre l'accès direct à ses méthodes publiques. Le proxy fournit également la transparence à l'emplacement pour l'aglet. C'est-à-dire qu'il peut cacher le vrai emplacement de l'aglet.

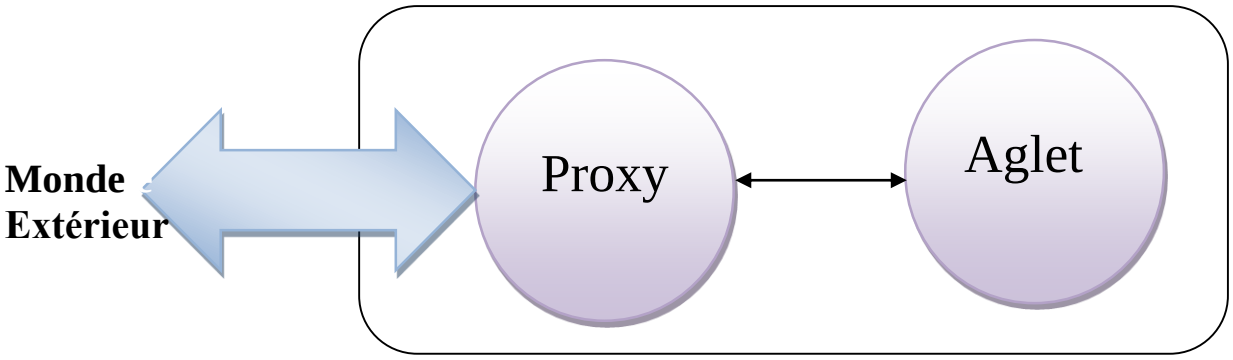


Figure IV.4: Relation entre un Aglet et son Proxy.

Contexte :

Le contexte est l'environnement d'exécution de l'aglet. Il fournit des moyens pour mettre à jour et contrôler des aglets dans un environnement uniforme d'exécution.

Hôte *:*

Un hôte est une machine capable d'héberger plusieurs contextes. L'hôte est généralement un nœud dans un réseau.

IV.2.4.3. Cycle de vie d'un Aglet

Les types de comportement des Aglets ont été implémentés de manière à répondre aux principaux besoins des agents. Les principales opérations affectant la vie d'un Aglet sont :

- **Création**

La création d'un Aglet a lieu dans un contexte. Le nouveau Aglet est assigné un identificateur, inséré dans le contexte, et initialisé. L'Aglet commence à exécuter dès qu'il sera avec succès initialisé.

- **Clonage**

Le clonage d'un Aglet produit une copie presque identique de l'Aglet initial dans le même contexte, la seule différence est que les Aglets sont relancés.

- **Disposition**: la destruction d'un Aglet, le stoppera dans son exécution actuelle et la retirera de son contexte actuel.

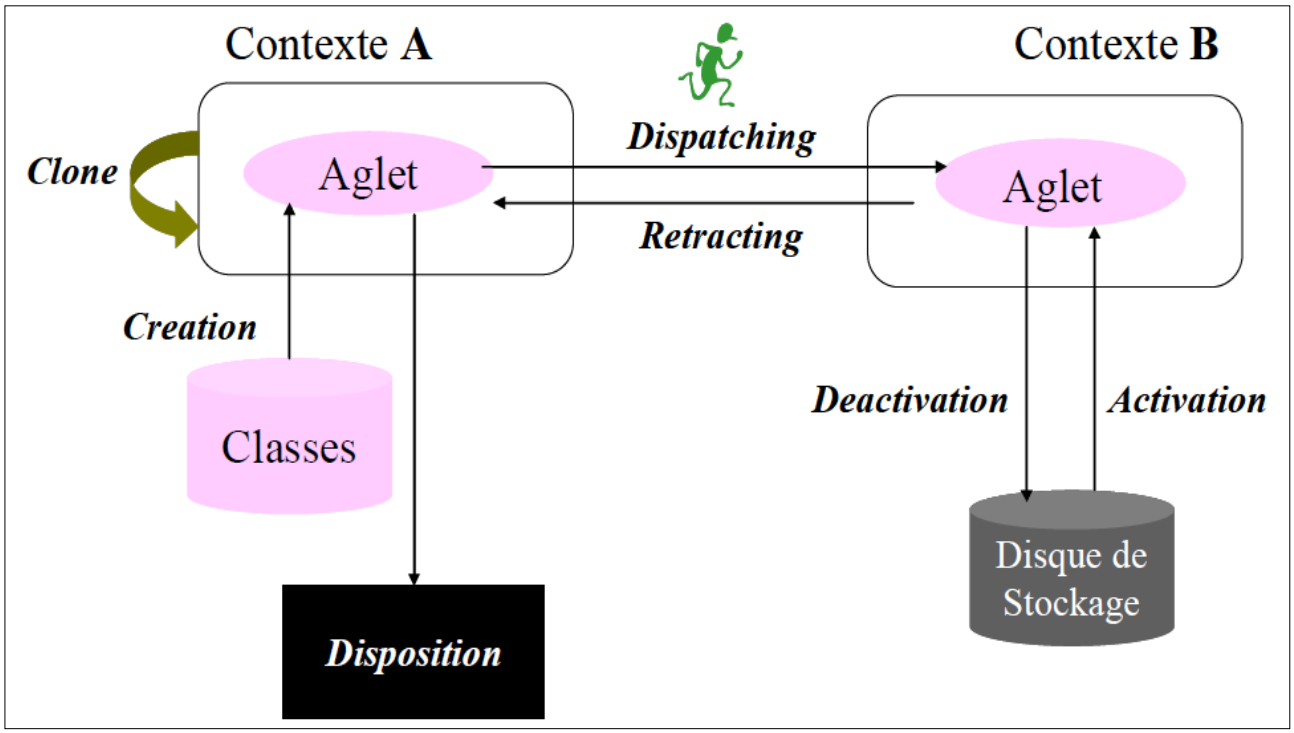


Figure IV.5 : Cycle de vie d'un Aglet

- **Dispatching**: la déportation d'Aglet d'un contexte vers un autre, le retirera de son contexte actuel et l'insérera dans le contexte du destinataire, où son exécution sera relancée.
- **Retracting**: la récupération d'un Aglet, le retirera de son contexte actuel et l'insérera dans le contexte où le retrait a été demandé.
- **Désactivation/Activation:** la désactivation d'un Aglet permet de l'enlever temporairement de son contexte actuel, et de l'enregistrer dans la mémoire secondaire. L'activation de l'Aglet va le restaurer en mémoire secondaire.

➢ **La classe Aglet [46]**

La classe Aglet est la classe principale dans l'API Aglets, c'est la classe de laquelle tous les aglets doivent hériter. Cette classe définie des méthodes pour commander son propre cycle de vie, à savoir, méthode pour le clonage, le déplacement, le traitement des messages arrivés et pour la destruction.

La table suivante (Table IV.1) présente quelques méthodes :

Méthode	Description
clone ()	La méthode clone () permet de cloner un aglet. Elle retourne le proxy associé au clone.
dispatch (URL)	Déplace l'aglet à l'url spécifié.
dispose ()	Detruit l'aglet et l'enlève du context en cour d'exécution, un appel réussi à cette méthode cause la destruction de tous les processus créés par cet aglet.
Deactivate (Long)	Stocker l'aglet dans une mémoire secondaire pendant une durée spécifiée.
activate ()	Active l'aglet
getAgletContext ()	Retourne le contexte dont lequel l'aglet s'exécute
getProxy ()	Retourne un proxy (représentant) de l'aglet.
handleMessage (Message)	traitement du message spécifié
getAgletInfo ()	Retourne toutes les informations sur l'aglet
Run ()	Le point d'entrée de l'aglet
onCreation (Object)	Initialise l'aglet, cette méthode est appelée seulement lors de la création de l'aglet.

Table IV.1 : Quelques méthodes de la classe Aglet

IV.2.4.4. Tahiti : un gestionnaire d'agents visuel

Tahiti [44] utilise une interface graphique unique pour suivre et contrôler l'exécution des Aglets. Il est possible en utilisant le glisser – déposer de faire communiquer deux agents ou de les faire migrer vers un

site particulier. Tahiti dispose d'un gestionnaire de sécurité paramétrable qui détecte toute opération non autorisée et empêche l'agent de la réaliser.

Le lancement de Tahiti se fait par l'exécution du fichier de commandes « agletsd.bat » (sous plate-forme Windows) (Figure IV.6). Ce dernier exécute le démon Tahiti (le serveur) ainsi qu'un Viewer d'Aglet qui porte le même nom. C'est grâce à ce dernier que nous pouvons faire des opérations précises sur les Aglets : les exécuter, les détruire, les envoyer sur un serveur distant, etc…

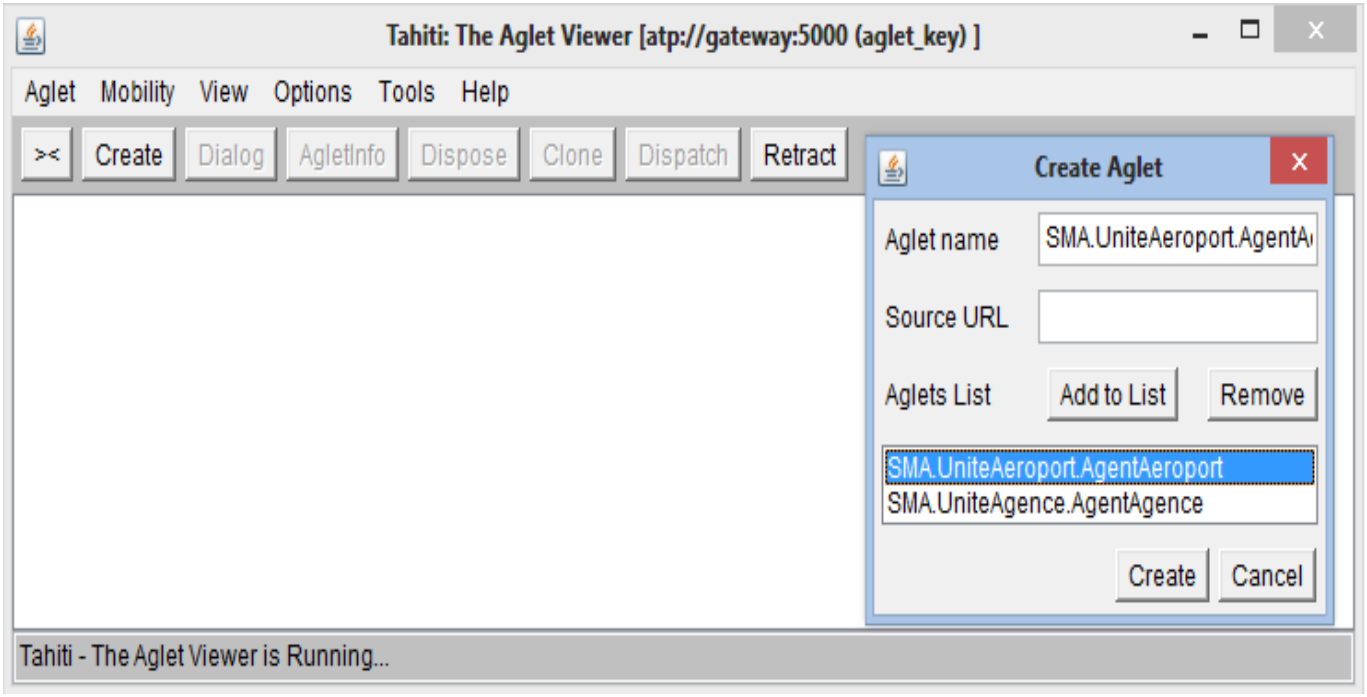


Figure IV.6 : Exemple d'un Tahiti

IV.3. Architecture logicielle du prototype

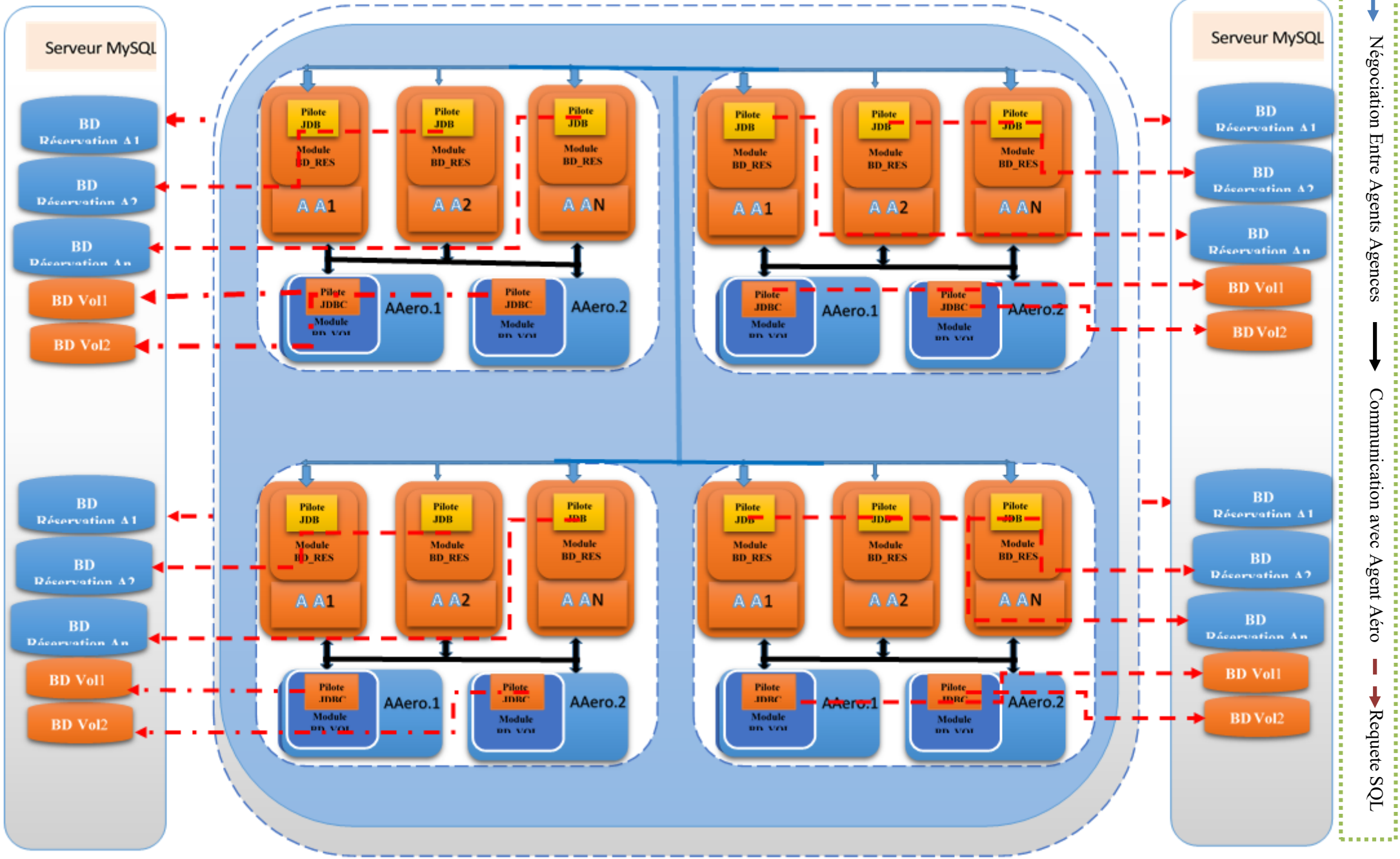


Figure IV.7 L'architecture logicielle du prototype

IV.4. L'implémentation des agents

Pour décrire l'implémentation de SMA, nous allons présenter les différentes classes agents du système sous forme d'un pseudo-code:

IV.4.1 Agent Agence

L'agent Client est un agent hybride. Il interagie avec le client (formulaire GUI), interagie avec la Base des sonnées (Module_Réservation) et communique avec les autres agents du système dans une autre fois (Panneau de contrôle).

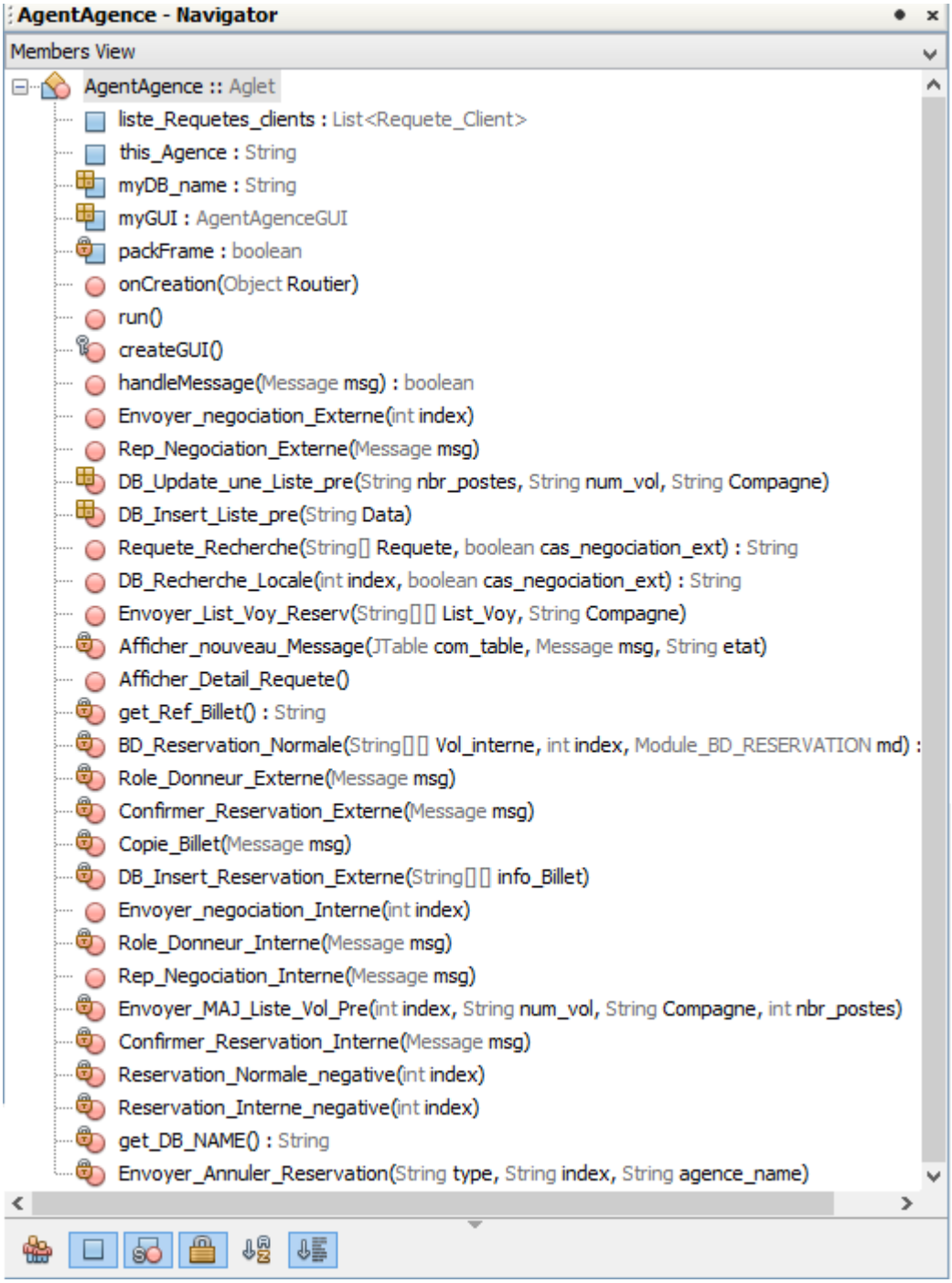


Figure IV.8: Pseudo code de l'agent Agence.

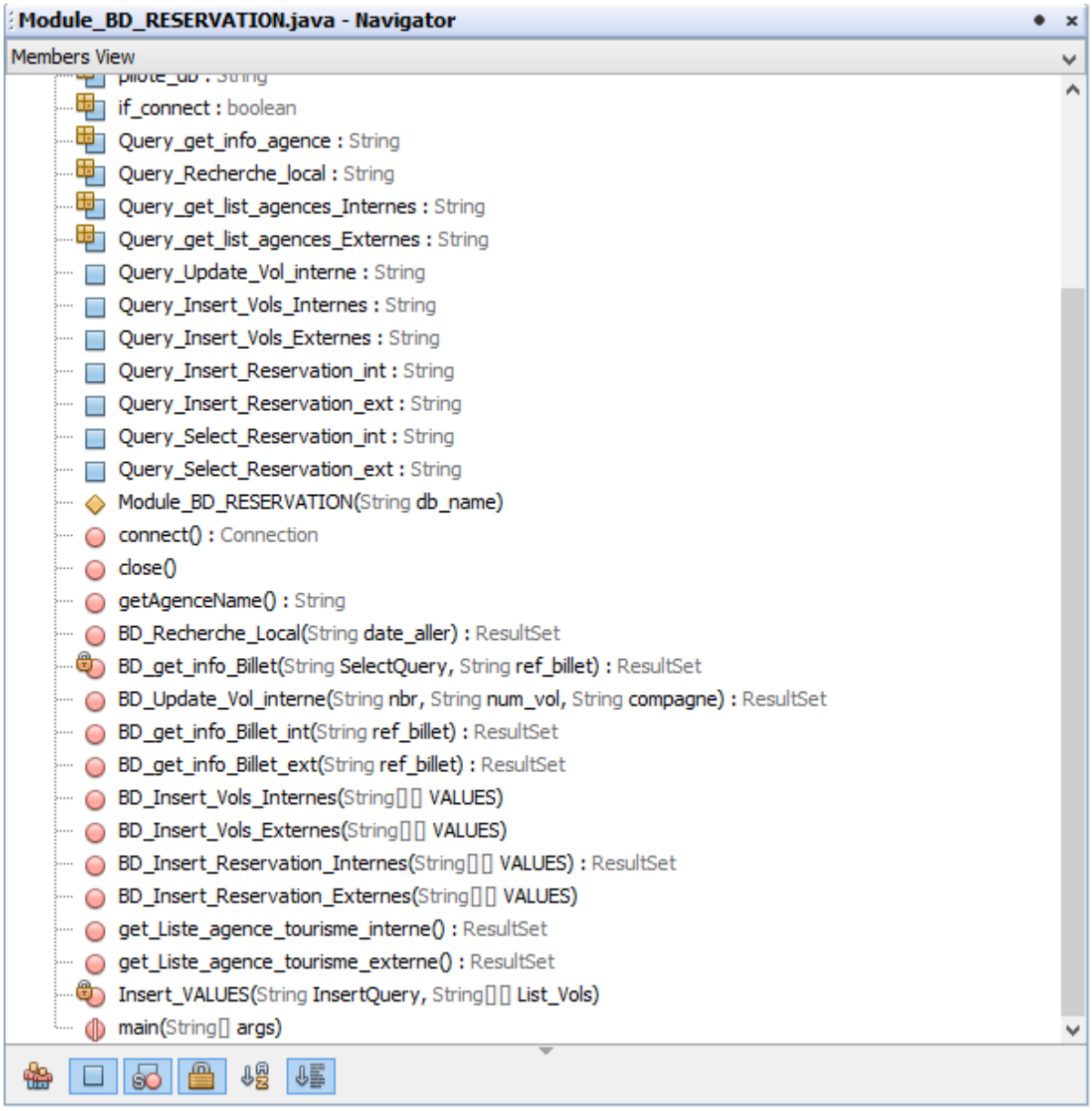


Figure IV.9: Pseudo code du Module BD Réservation pour un Agent Agence.

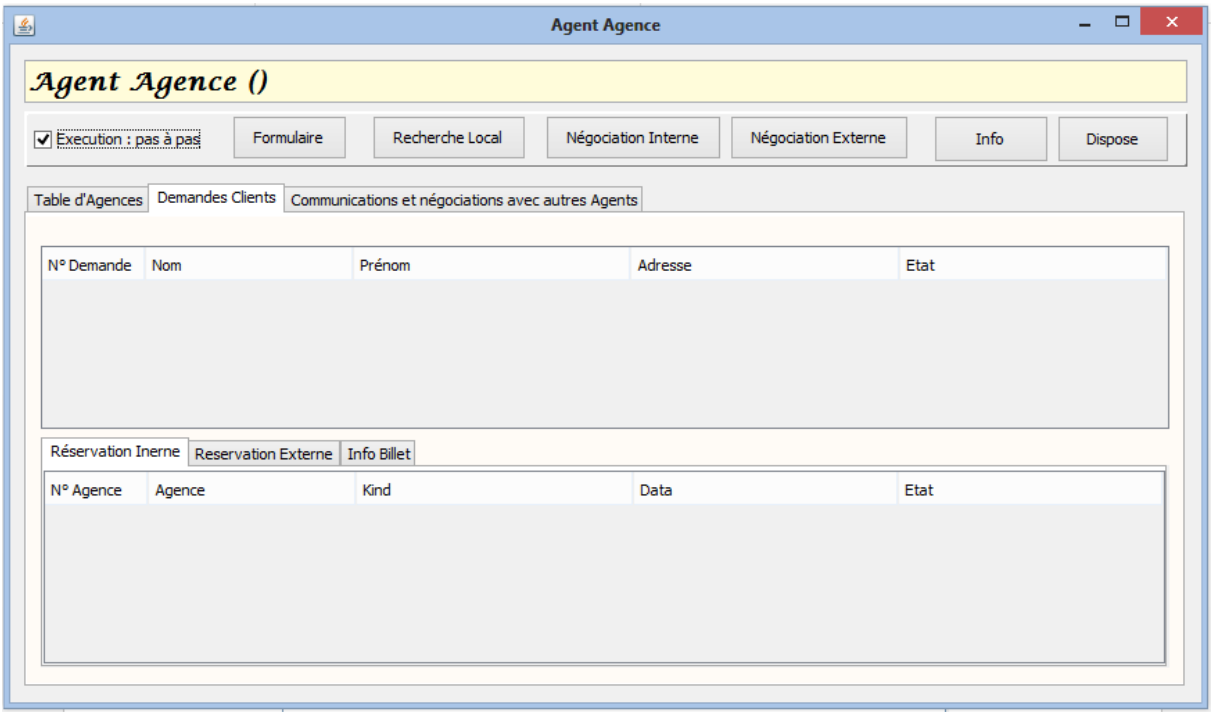


Figure IV.10: Panneau de contrôle de l'agent Agence.

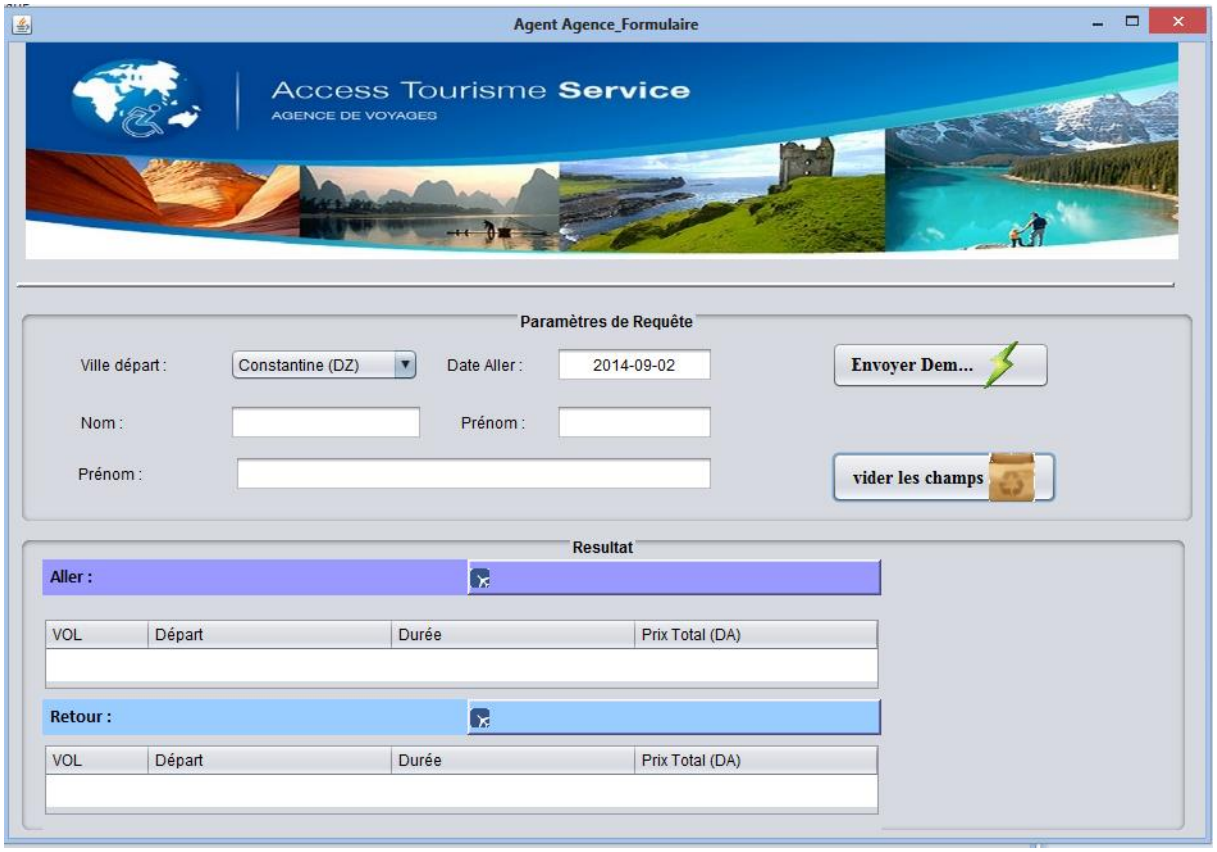


Figure IV.11: Formulaire de requête Voyageur fourni par l'agent Agence.

IV.4. 2 Agent Aéroport

Comme nous avons vu dans le chapitre de conception, l'Agent Aéroport ne communique qu'avec les Agents Agences de sa région et interagi avec sa Base de données « BD_VOL ».

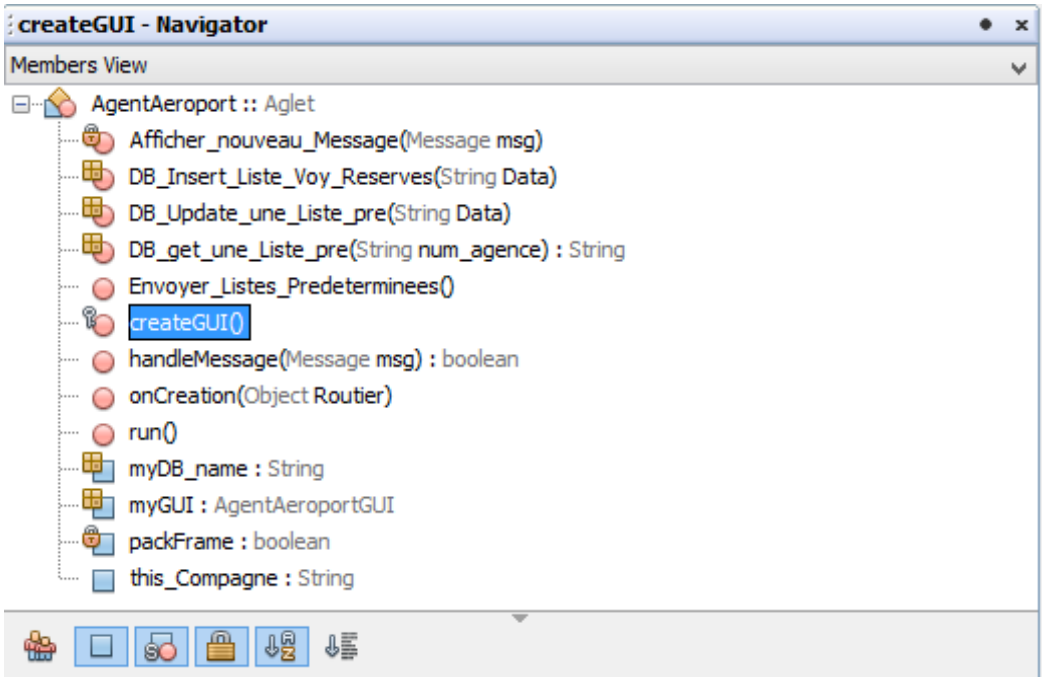


Figure IV.12: Pseudo code de l'Agent Aéroport.

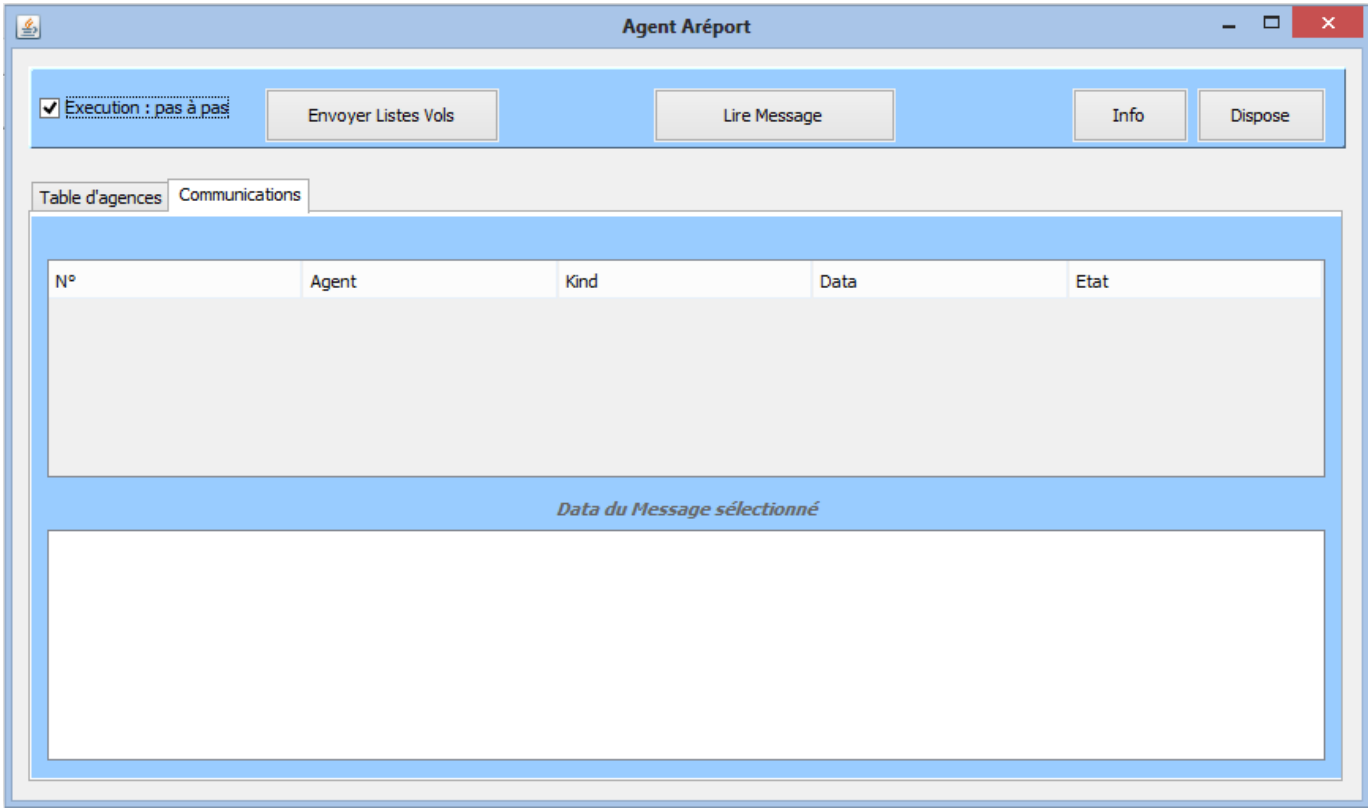


Figure IV.13: Panneau de contrôle de l'agent Aéroport.

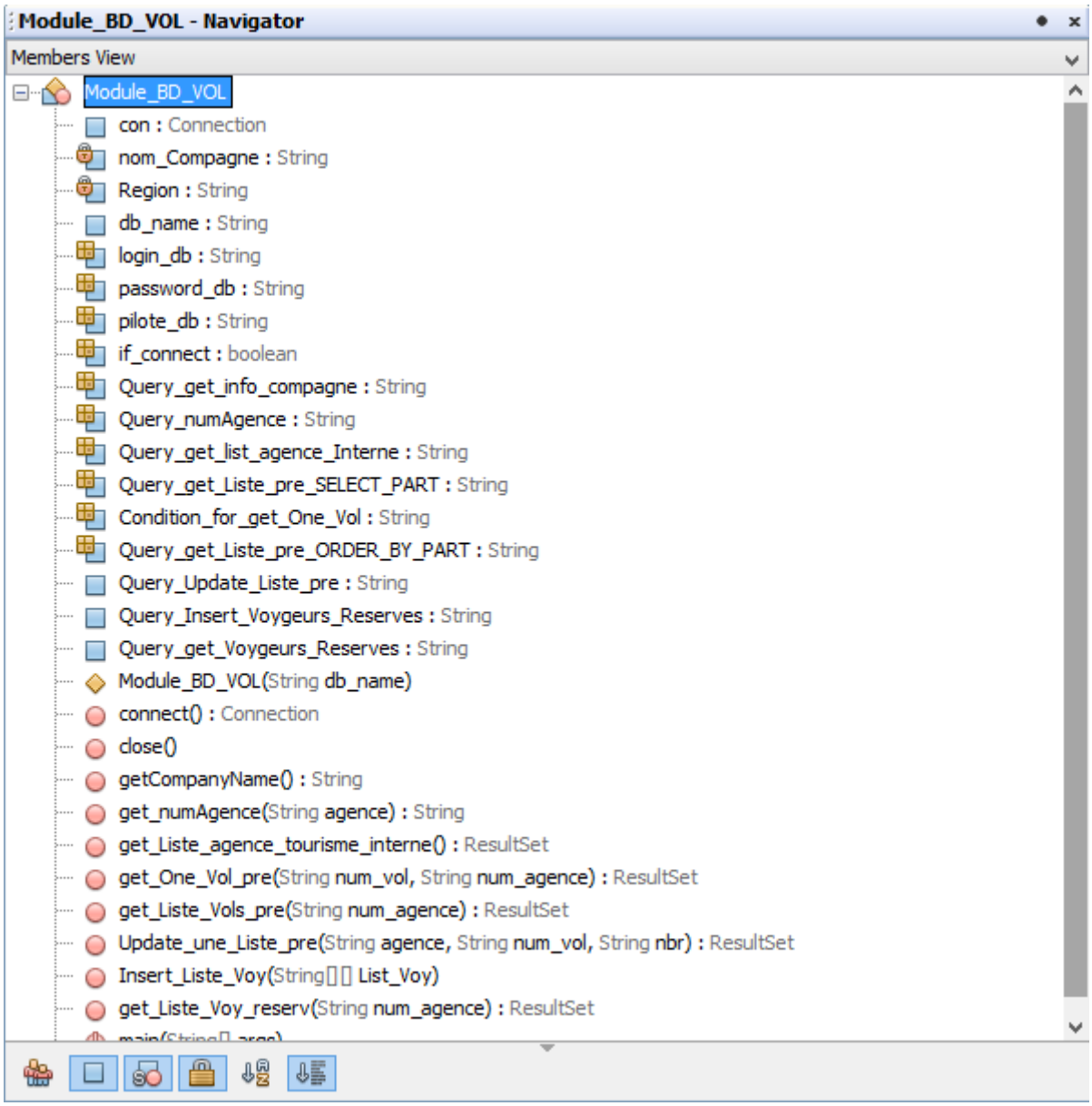


Figure IV.14: Pseudo code du Module_BD_VOL de l'Agent Aéroport.

IV.5. Résultats obtenus

Afin de montrer la validité, la fiabilité et l'extensibilité de notre architecture, on a intérêt à faire une étude de cas. D'où nous allons appliquer notre approche sur un exemple d'organisation des réservations de voyageurs dans la saison du Hadj en l'Algérie, cet exemple considéré comme un exemple réel pour simuler le travail pour une planification décentralisée.

On suppose le scénario suivant : Un voyageur (Hadj) habite à Biskra, il désir de réserver un billet de voyage dans une Date. Il prépare les arrangements de voyage. Il doit consulter une agence de tourisme (ex : ONAT, Biskra), il doit remplir un formulaire de demande de réservation, l'agent de l'agence prend le formulaire et lance le processus de la recherche d'un vol satisfaisant pour cette demande.

On a les Agents Utilisés dans le système :

- Agent Aéroport – Air Algérie, Constantine
- Agent Agence – ONAT, Biskra
- Agent Agence – ONAT, Constantine
- Agent Agence – ONAT, Annaba

} Région Est, Constantine

- Agent Aéroport – Saudi Arabian Airlines, Alger
- Agent Agence – ONAT, Alger
- Agent Agence – ONAT, Blida

} Région Nord, Alger

IV.5.1 Démarrage et Initialisation du système :

IV.5.1.1 Coté de la plateforme TAHITI AGLET

Pour exploiter le système, nous devons lancer la plateforme Tahiti Aglet puis les agents du système, on obtient :

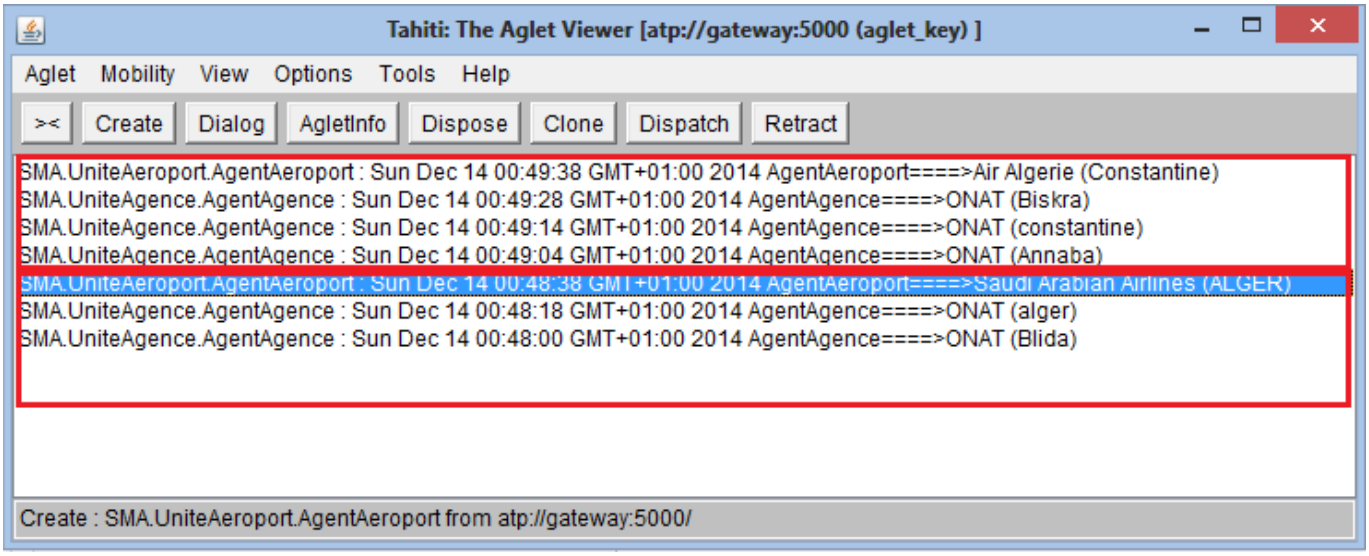


Figure IV.15 : Tahiti : (03) Agents Agence, (02) Agents Aéroport

IV.5.1.2 Coté d'Agent Aéroport Air Algérie, Constantine

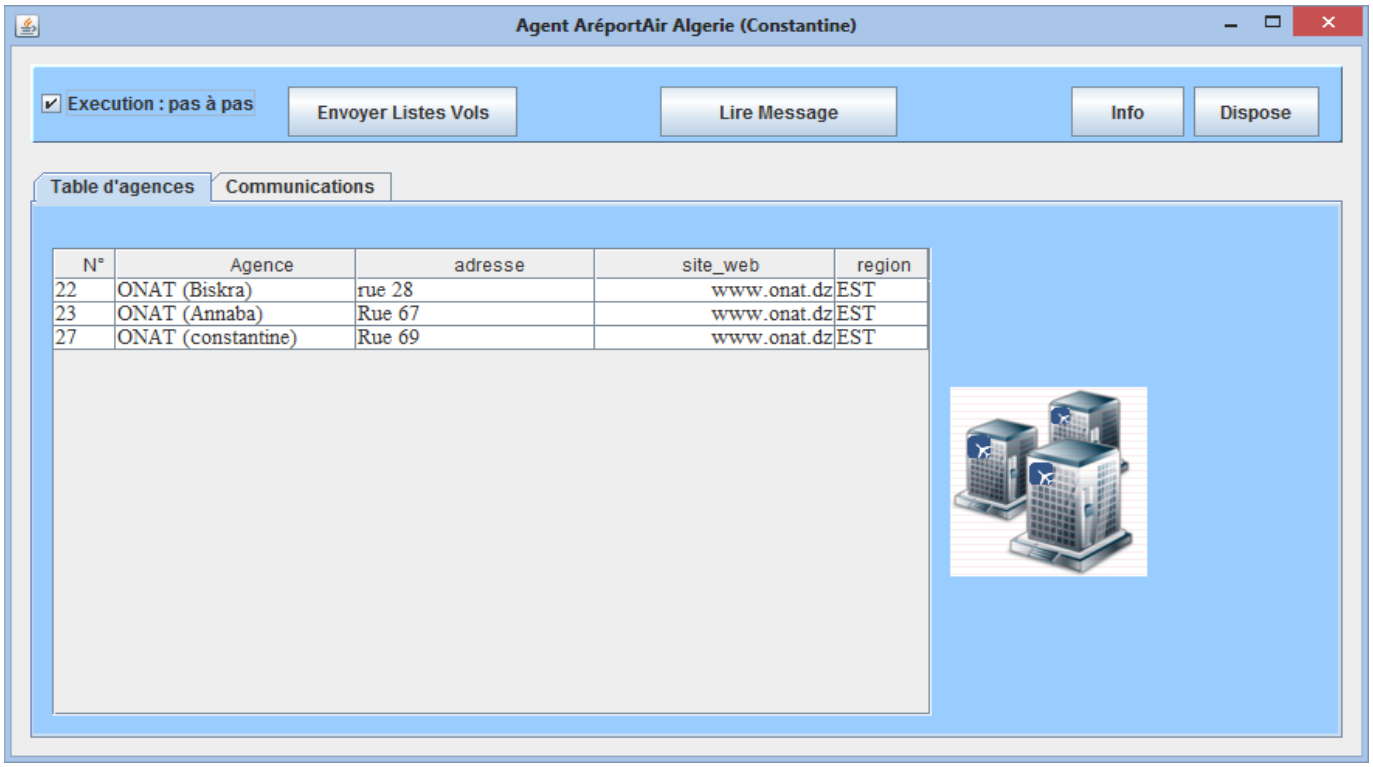


Figure IV.16 : Panneau de contrôle de l'Agent Aéroport – Air Algérie, Constantine

IV.5.1.3 Coté d'Agent Aéroport Saudi Arabian Airlines, Alger

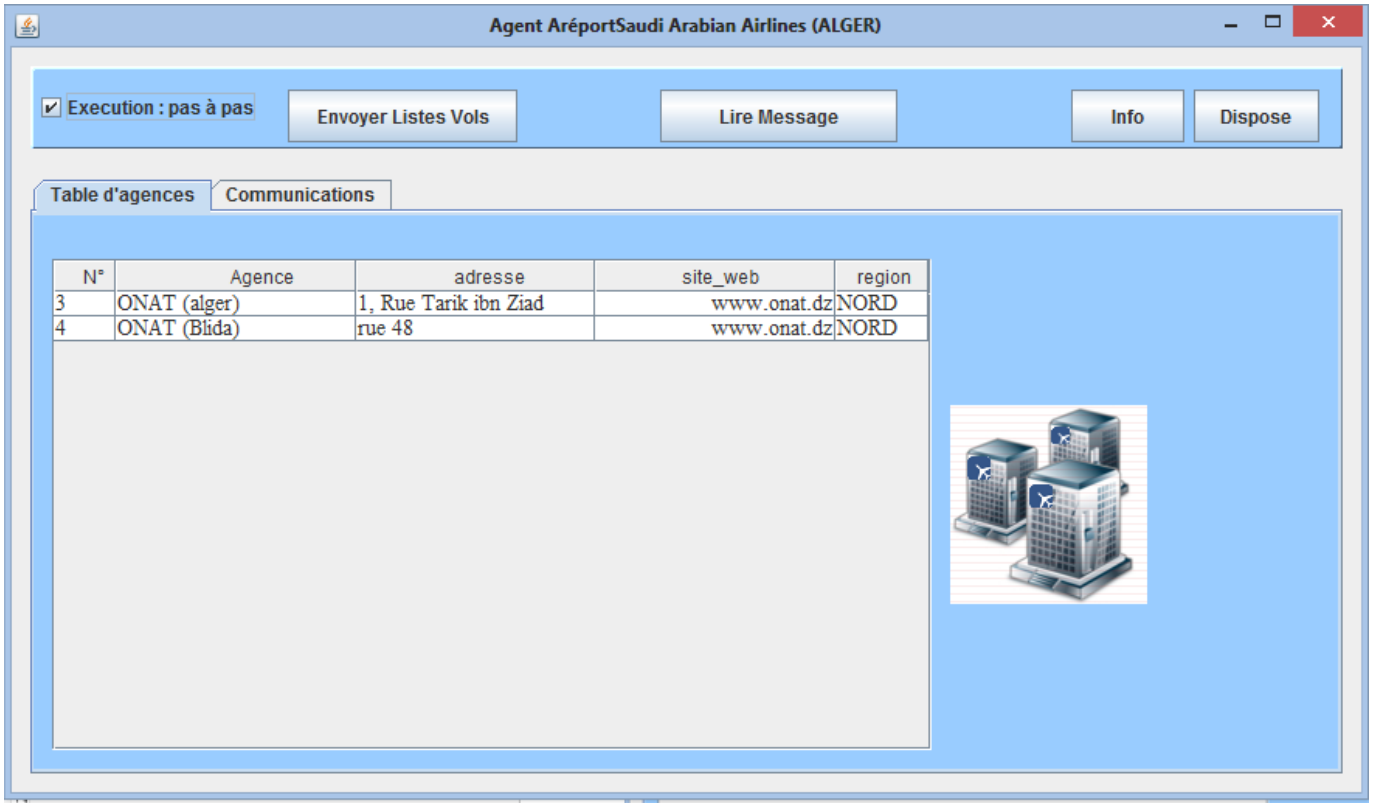


Figure IV.17 : Panneau de contrôle de l'Agent Aéroport – Saudi Arabian Airlines, Alger

IV.5.1.4 Formulaire De Voyageur de l'Agent Agence – ONAT, Biskra

Figure IV.18 : Formulaire De Voyageur pour Agent Agence – ONAT, Biskra

IV.5.1.5 Coté de Panneau de contrôle de l'Agent Agence – ONAT, Biskra

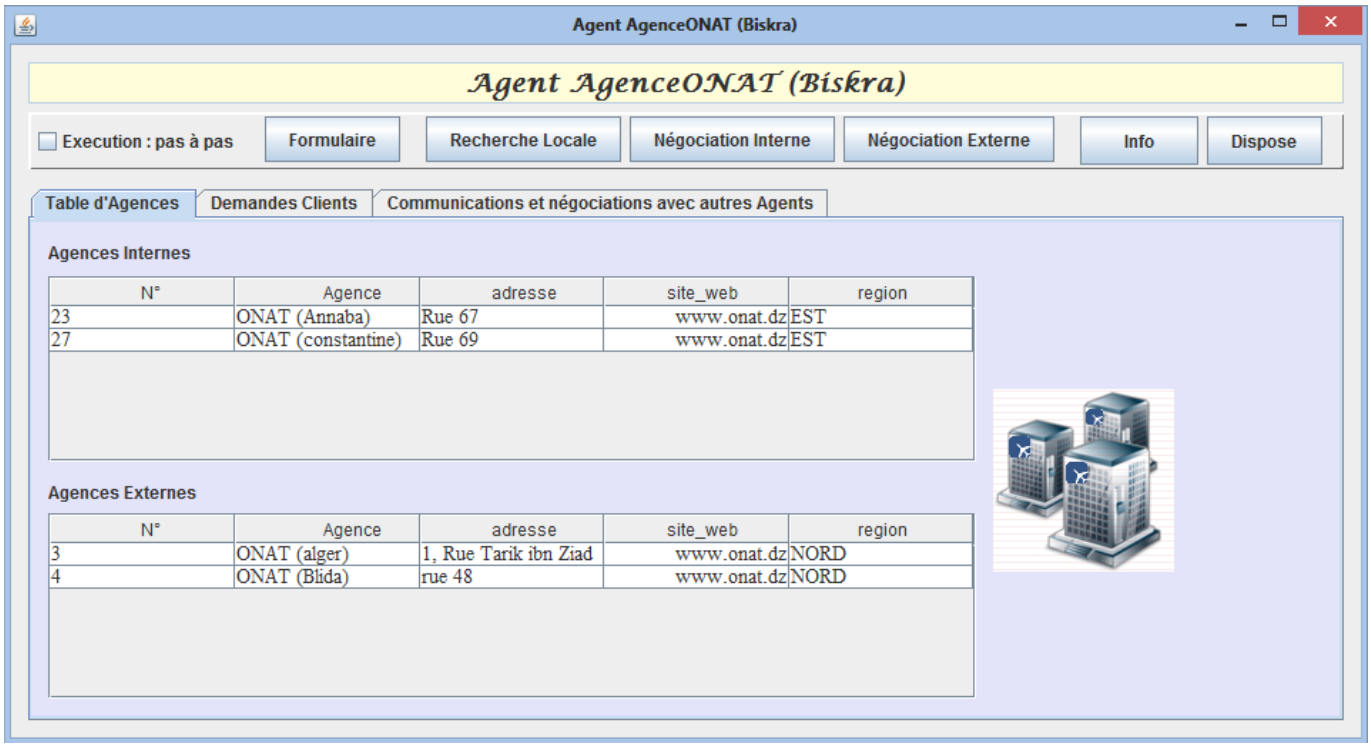


Figure IV.19 : Panneau de contrôle d'Agent Agence – ONAT, Biskra

IV.5.1.6 Coté de Panneau de contrôle de l'Agent Agence – ONAT, Constantine

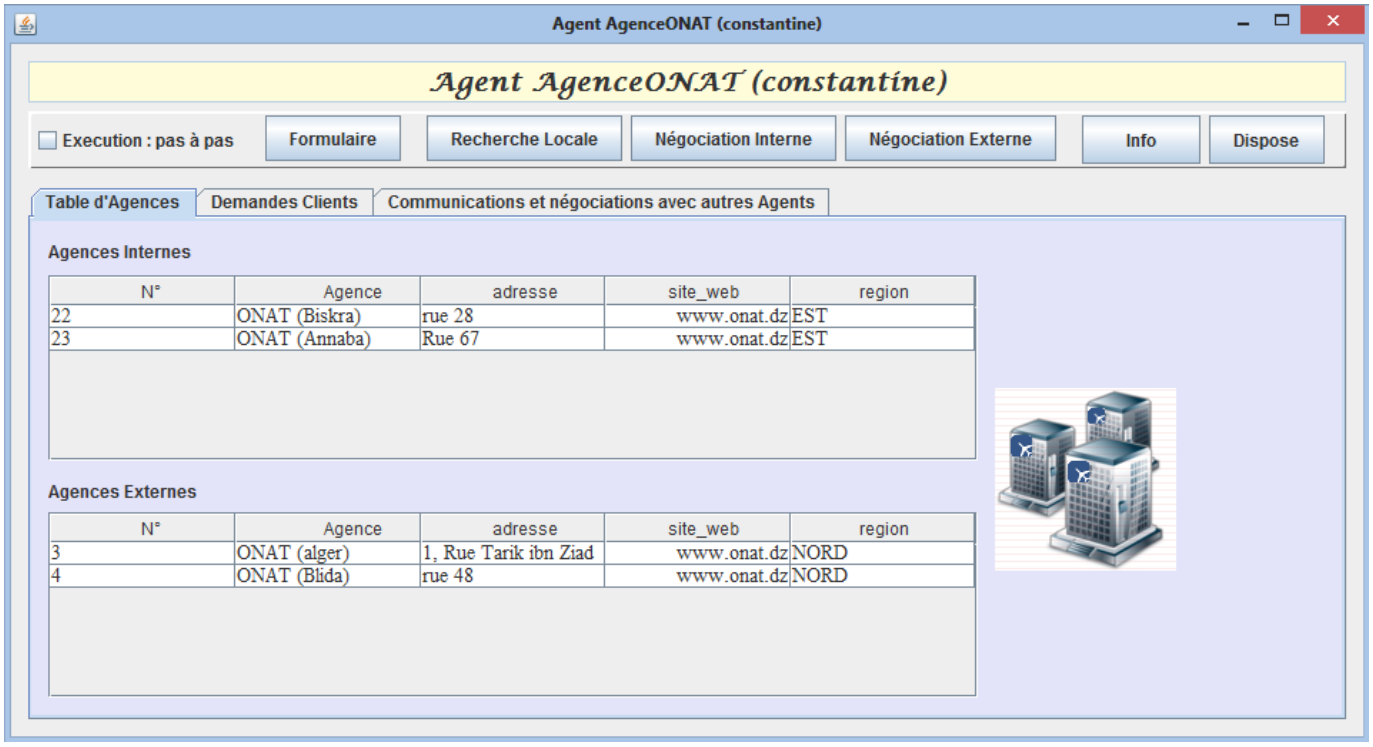


Figure IV.20 : Panneau de contrôle d'Agent Agence – ONAT, Constantine

IV.5.1.7 Coté de Panneau de contrôle de l'Agent Agence – ONAT, Annaba

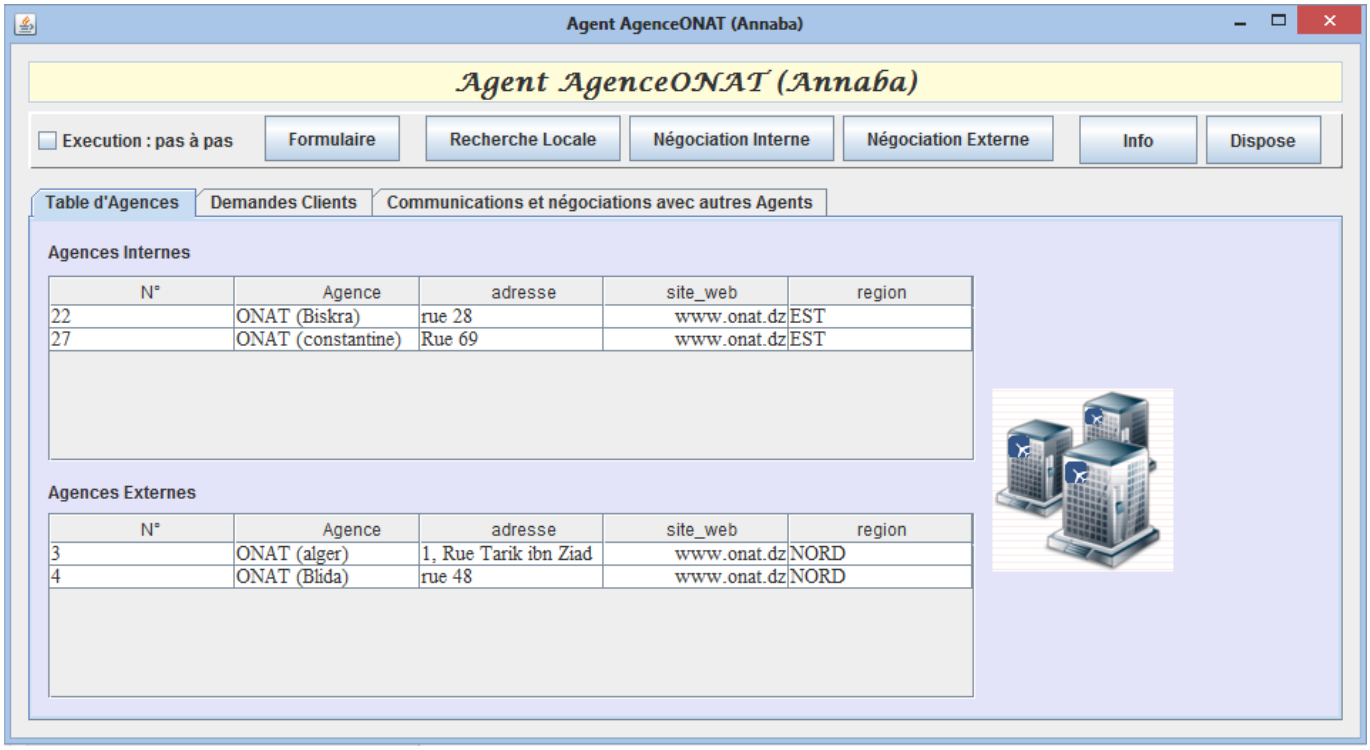


Figure IV.21 : Panneau de contrôle d'Agent Agence – ONAT, Annaba

IV.5.1.8 Coté de Panneau de contrôle de l'Agent Agence – ONAT, Alger

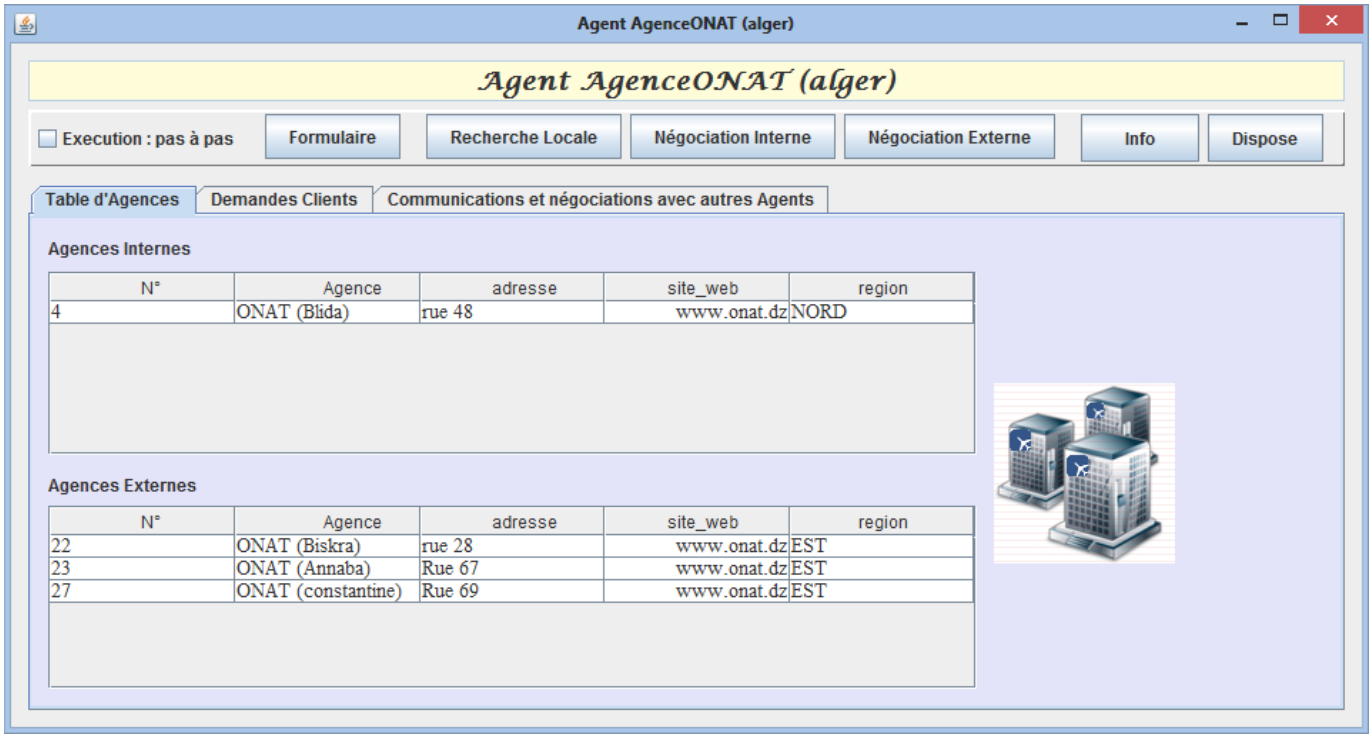


Figure IV.22 : Panneau de contrôle d'Agent Agence – ONAT, Alger

IV.5.1.9. Coté de Panneau de contrôle de l'Agent Agence – ONAT, Blida

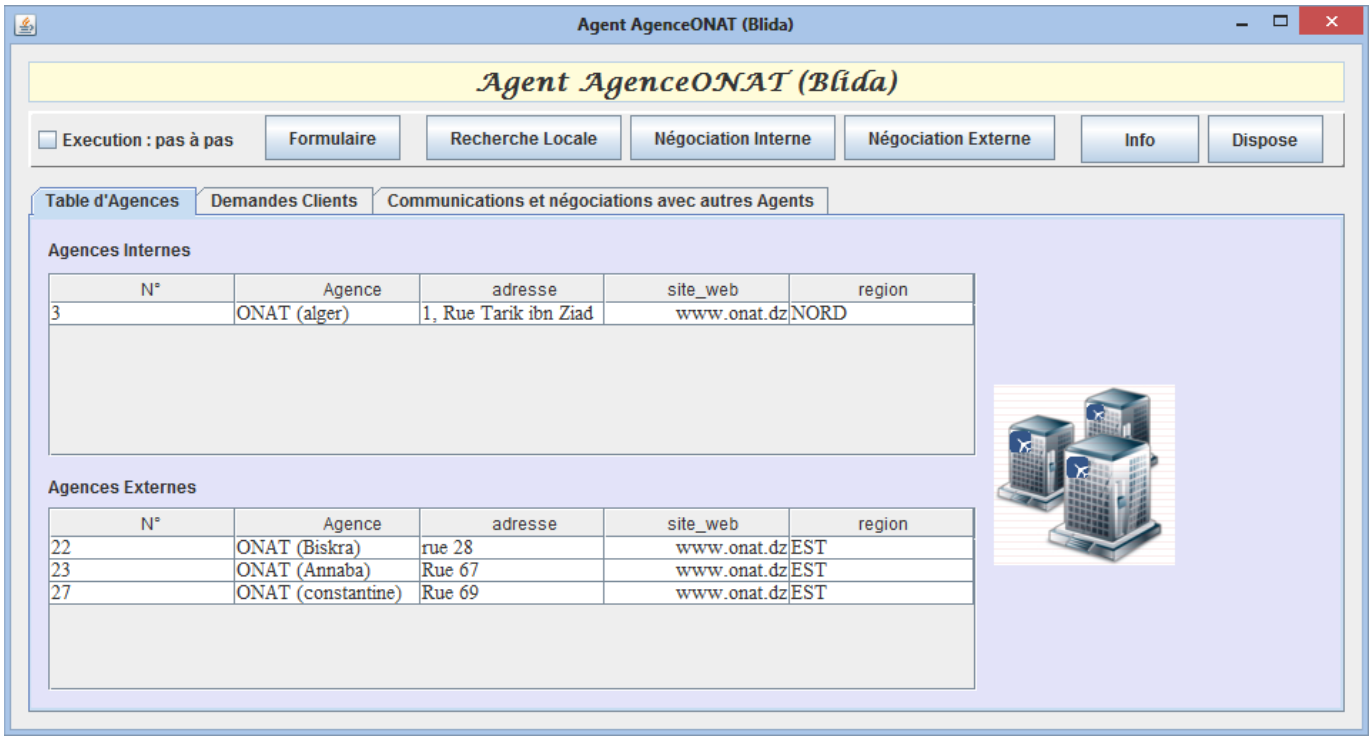


Figure IV.23 : Panneau de contrôle d'Agent Agence – ONAT, Blida

IV.5.2 Réservation Normale au niveau d'agence ONAT, Biskra (Cas 1)

IV.5.2.1 Coté d'Agent Aéroport Air Algérie, Constantine

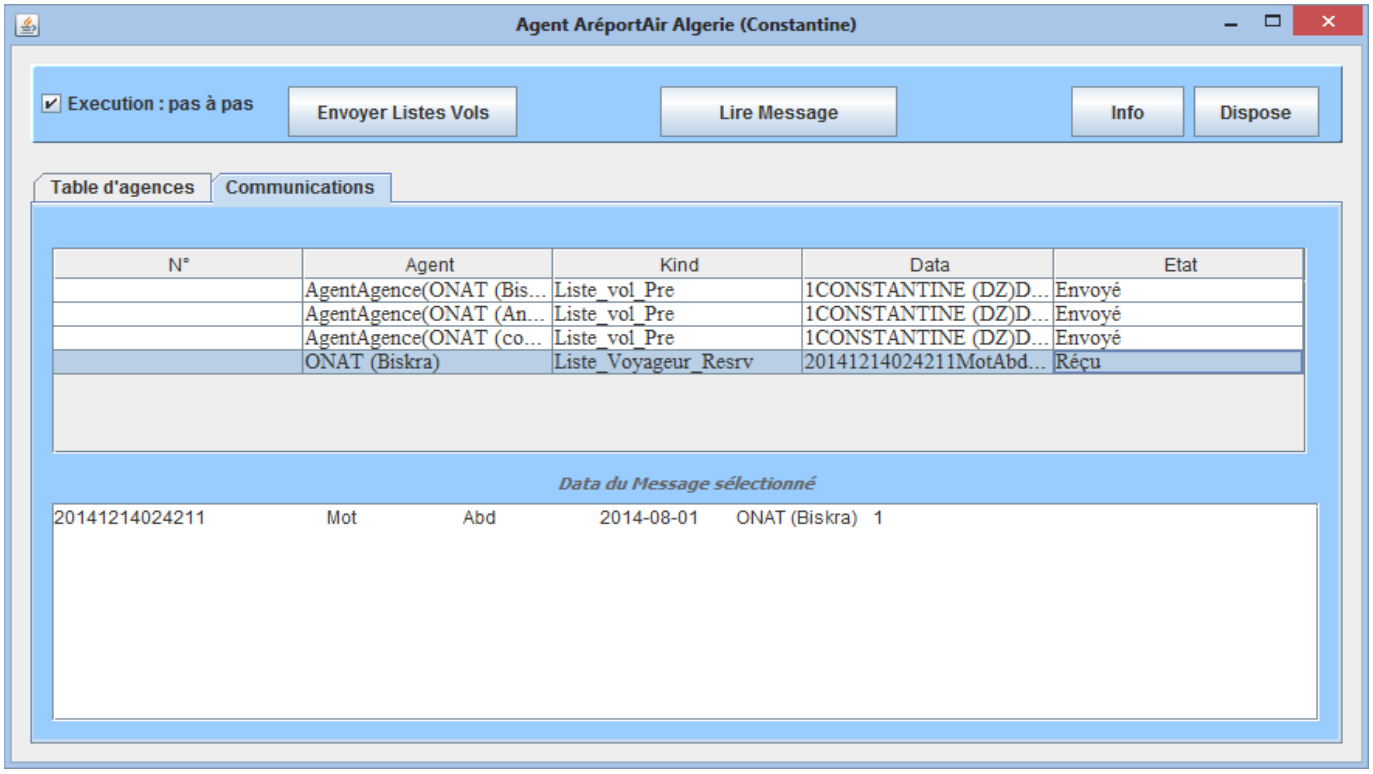


Figure IV.24 : Panneau de contrôle de l'Agent Agence – ONAT, Constantine (Cas 1)

IV.5.2.2 Coté d'Agent Agence– ONAT, Biskra

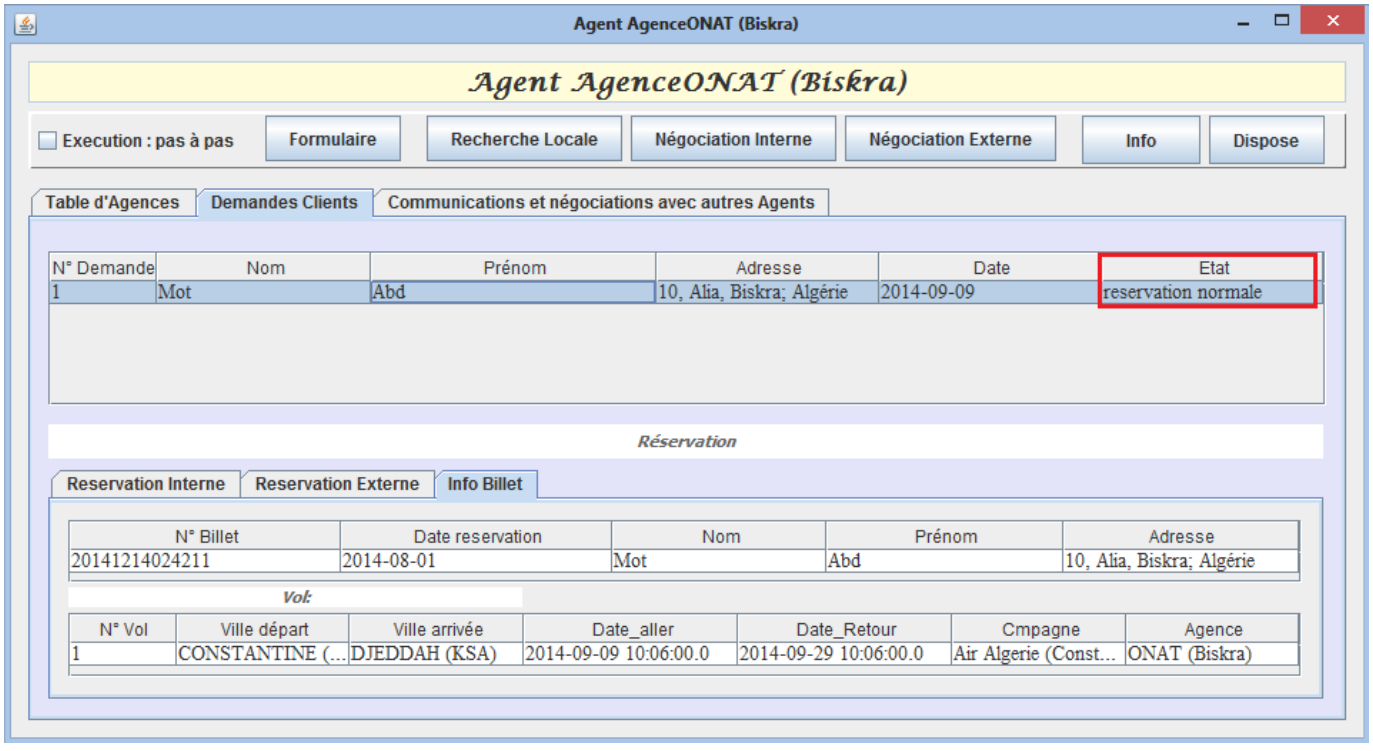


Figure IV.25 : Panneau de contrôle d'Agent Agence – ONAT, Biskra (Cas 1)

IV.5.2.3 Résultat Coté Interface Voyageur

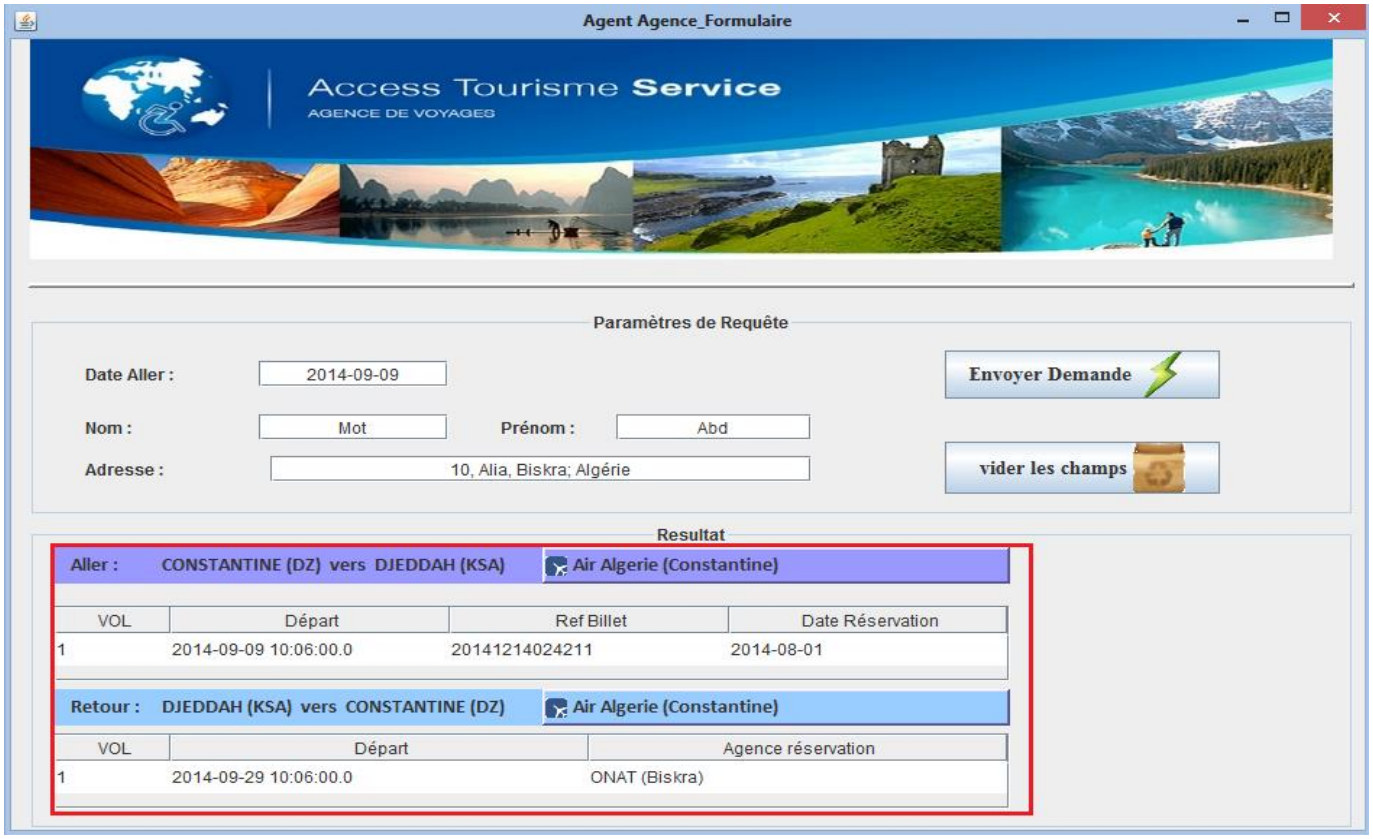


Figure IV.26: Résultat dans Interface Voyageur d'Agent Agence – ONAT, Biskra (Cas 1)

IV.5.3 Réservation Interne (Négociation avec ONAT Annaba/Constantine) (Cas 2)

IV.5.3.1 Coté d'Agent Agence – ONAT, Biskra

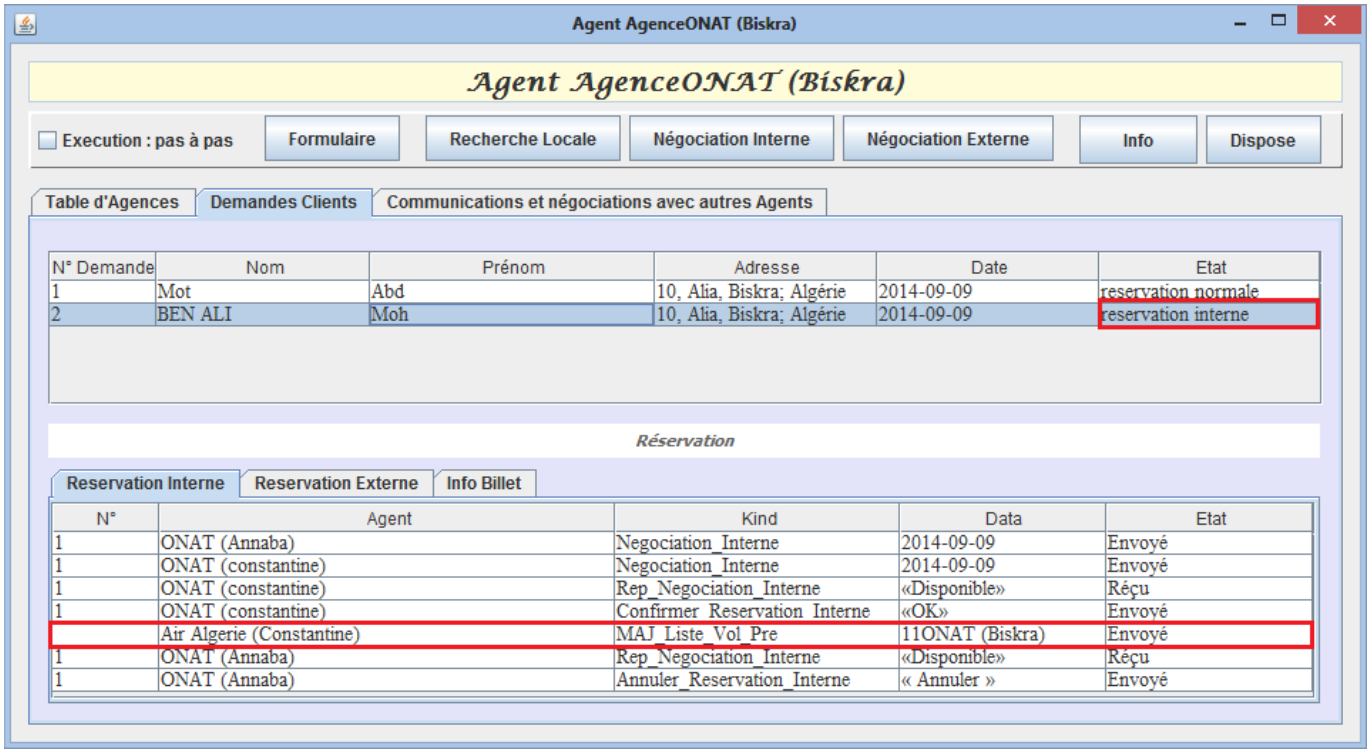


Figure IV.27 : Panneau de contrôle d'Agent Agence – ONAT, Biskra (Cas 2)

IV.5.3.2 Coté d'Agent Agence – ONAT, Constantine

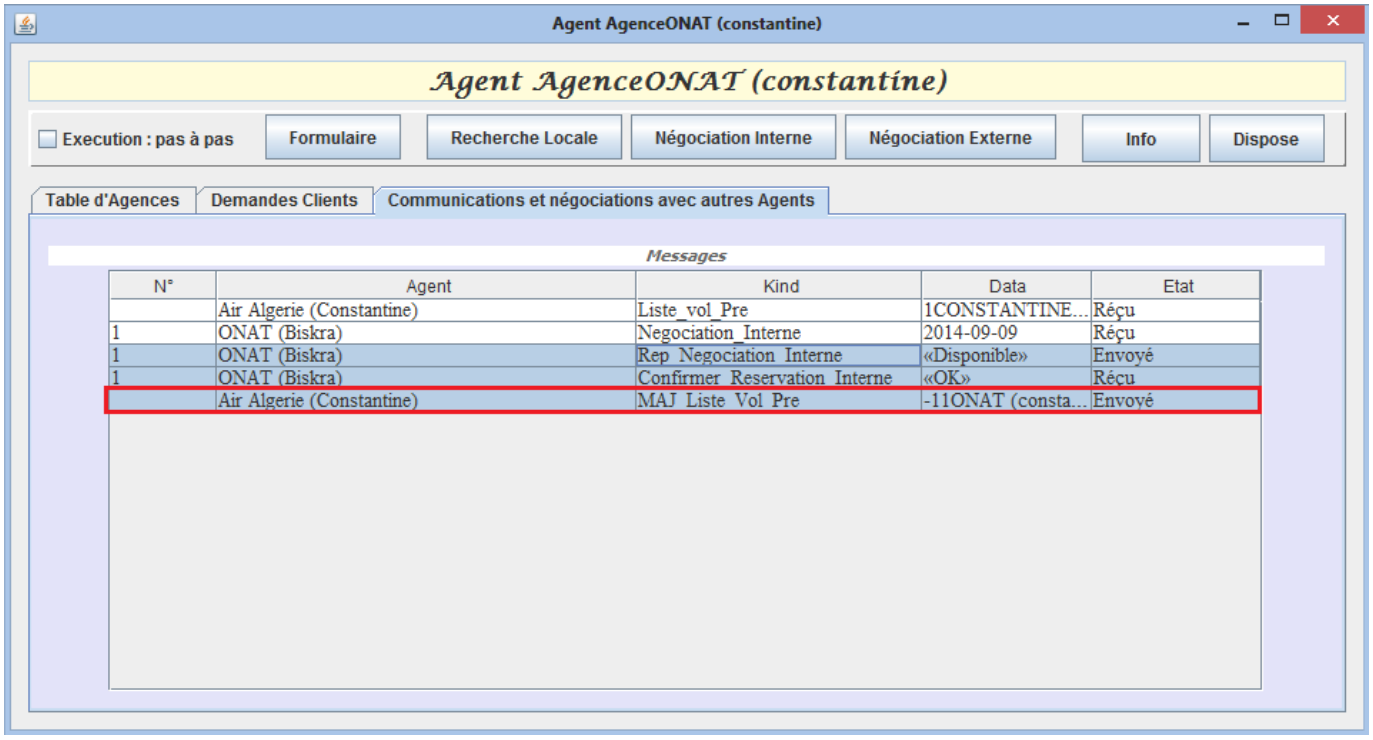


Figure IV.28 : Panneau de contrôle d'Agent Agence – ONAT, Constantine (Cas 2)

IV.5.3.3 Coté d'Agent Agence – ONAT, Annaba

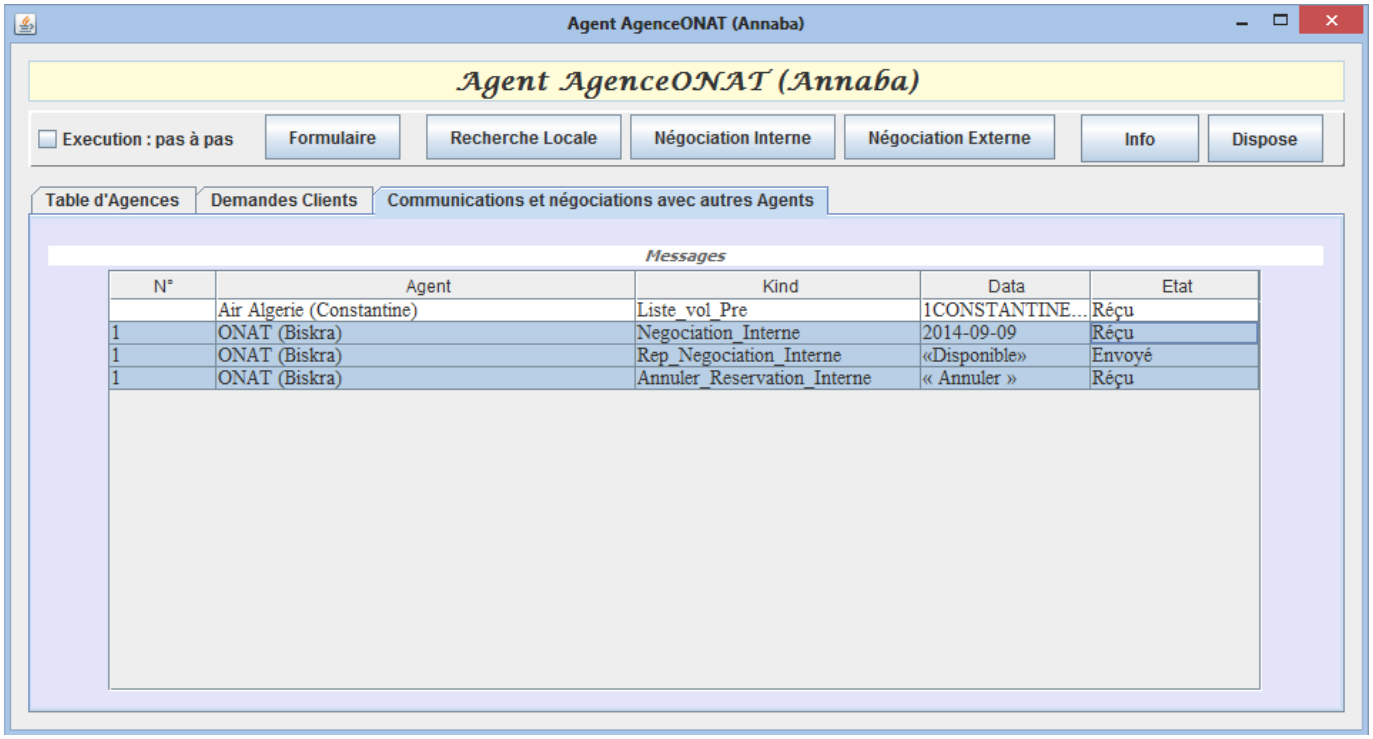


Figure IV.29 : Panneau de contrôle d'Agent Agence – ONAT, Annaba (Cas 2)

IV.5.3.4 Coté d'Agent Aéroport – Air Algérie, Constantine

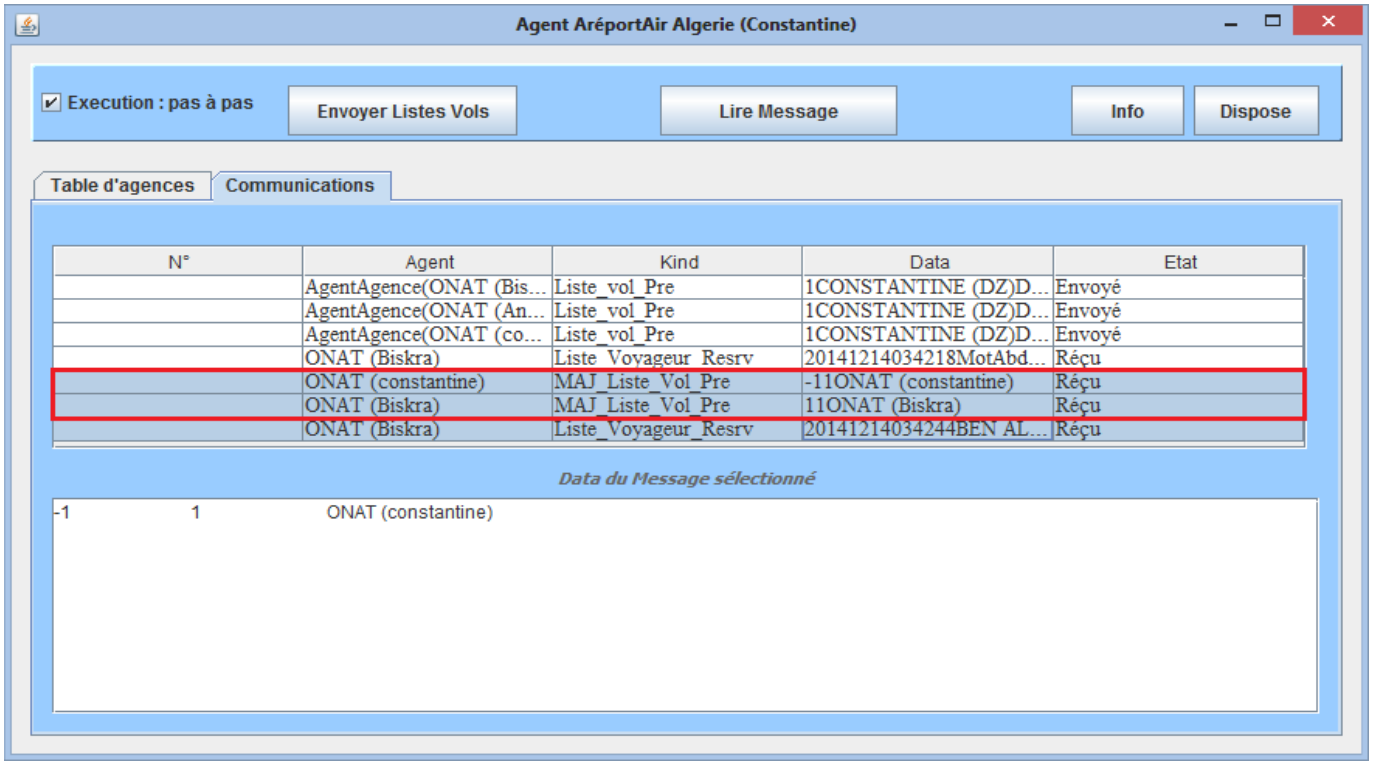


Figure IV.30 : Panneau de contrôle d'Agent Aéroport – Air Algérie, Constantine

IV.5.3.5 Résultat Coté Interface Voyageur

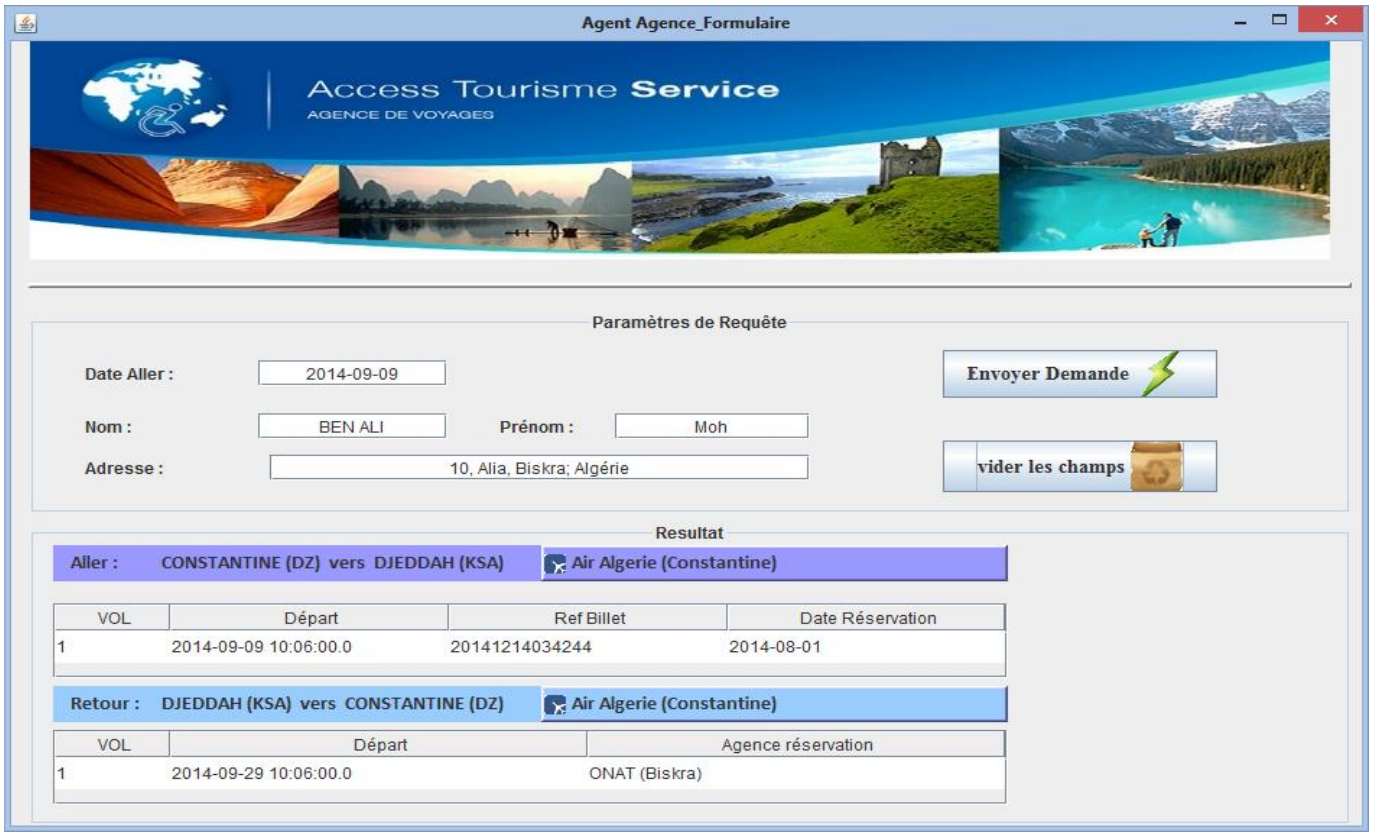


Figure IV.31: Résultat dans Interface Voyageur d'Agent Agence – ONAT, Biskra (Cas 2)

IV.5.4 Réservation Externe et (Négociation avec ONAT de Blida/Alger) (Cas 3)

IV.5.4.1 Coté d'Agent Agence – ONAT, Biskra

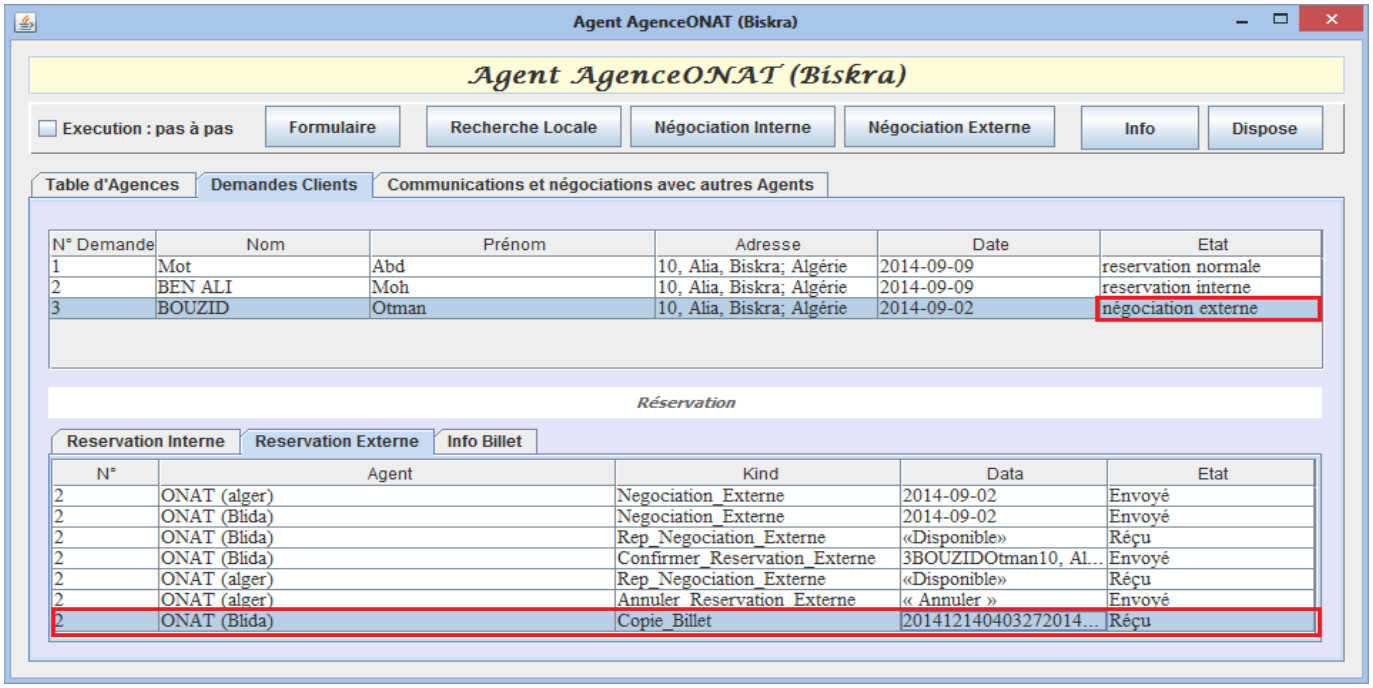


Figure IV.32 : Panneau de contrôle d'Agent Agence – ONAT, Biskra (Cas 3)

IV.5.4.2 Coté d'Agent Agence – ONAT, Blida

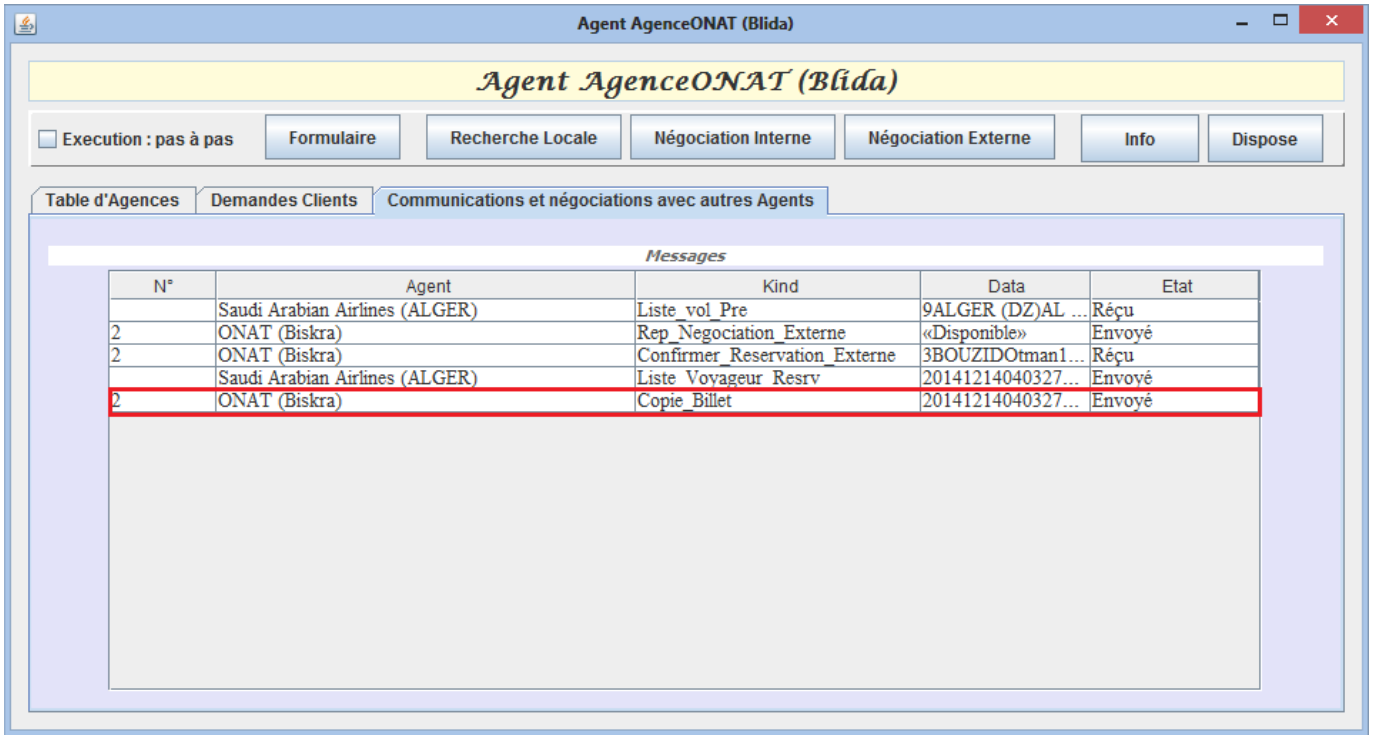


Figure IV.33 : Panneau de contrôle d'Agent Agence – ONAT, Blida (Cas 3)

IV.5.4.3 Coté d'Agent Agence – ONAT, Alger

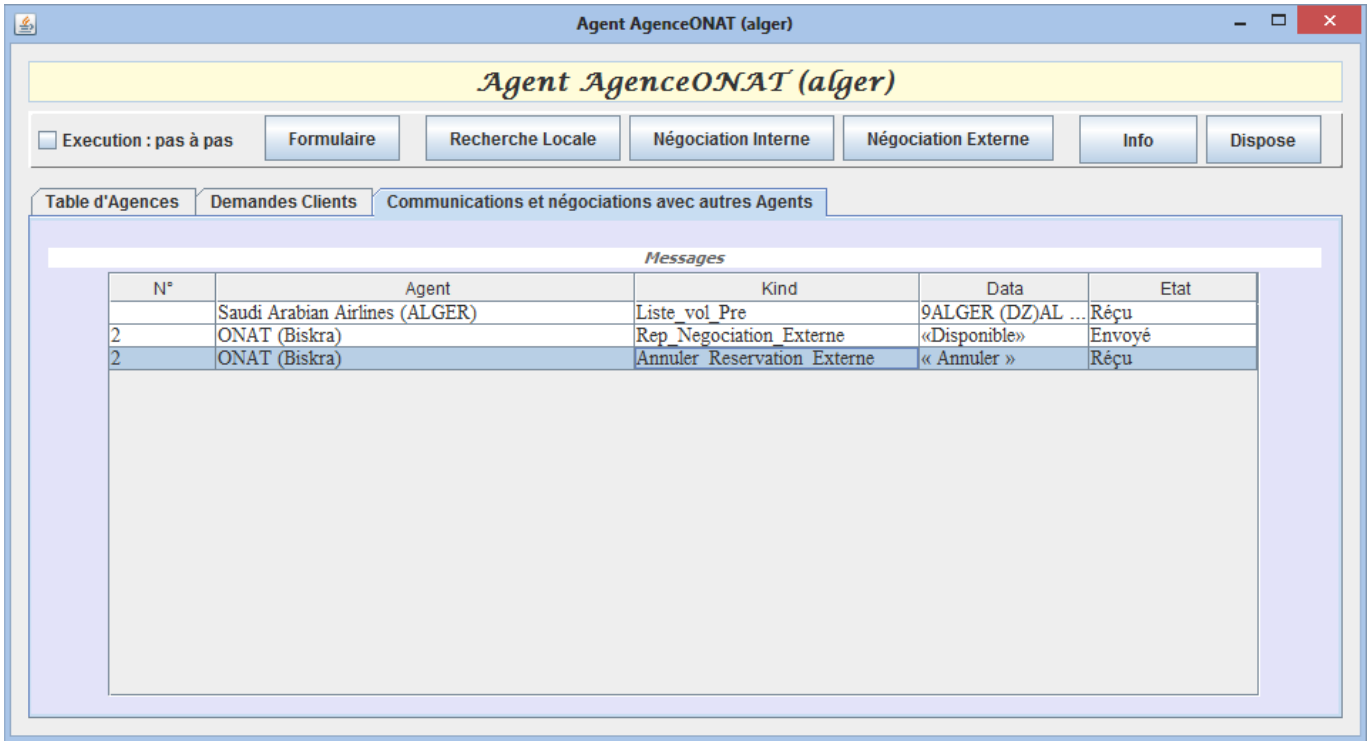


Figure IV.34 : Panneau de contrôle d'Agent Agence – ONAT, Alger (Cas 3)

IV.5.4.4 Coté d'Agent Aéroport – Saudi Arabian Airlines, Alger

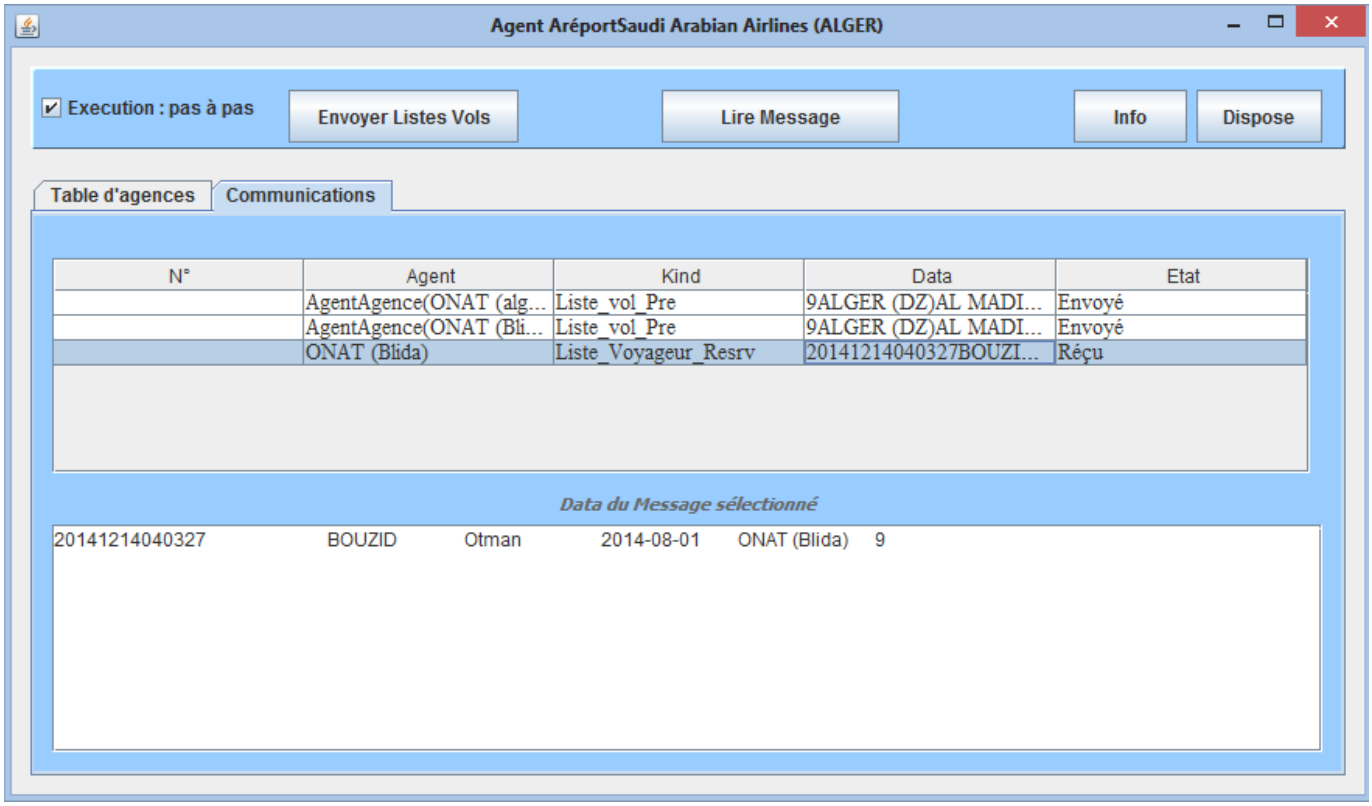


Figure IV.35 : Panneau de contrôle d'Agent Aéroport – Saudi Arabian Airlines, Alger (Cas 3)

IV.5.4.5 Résultat Coté Interface Voyageur

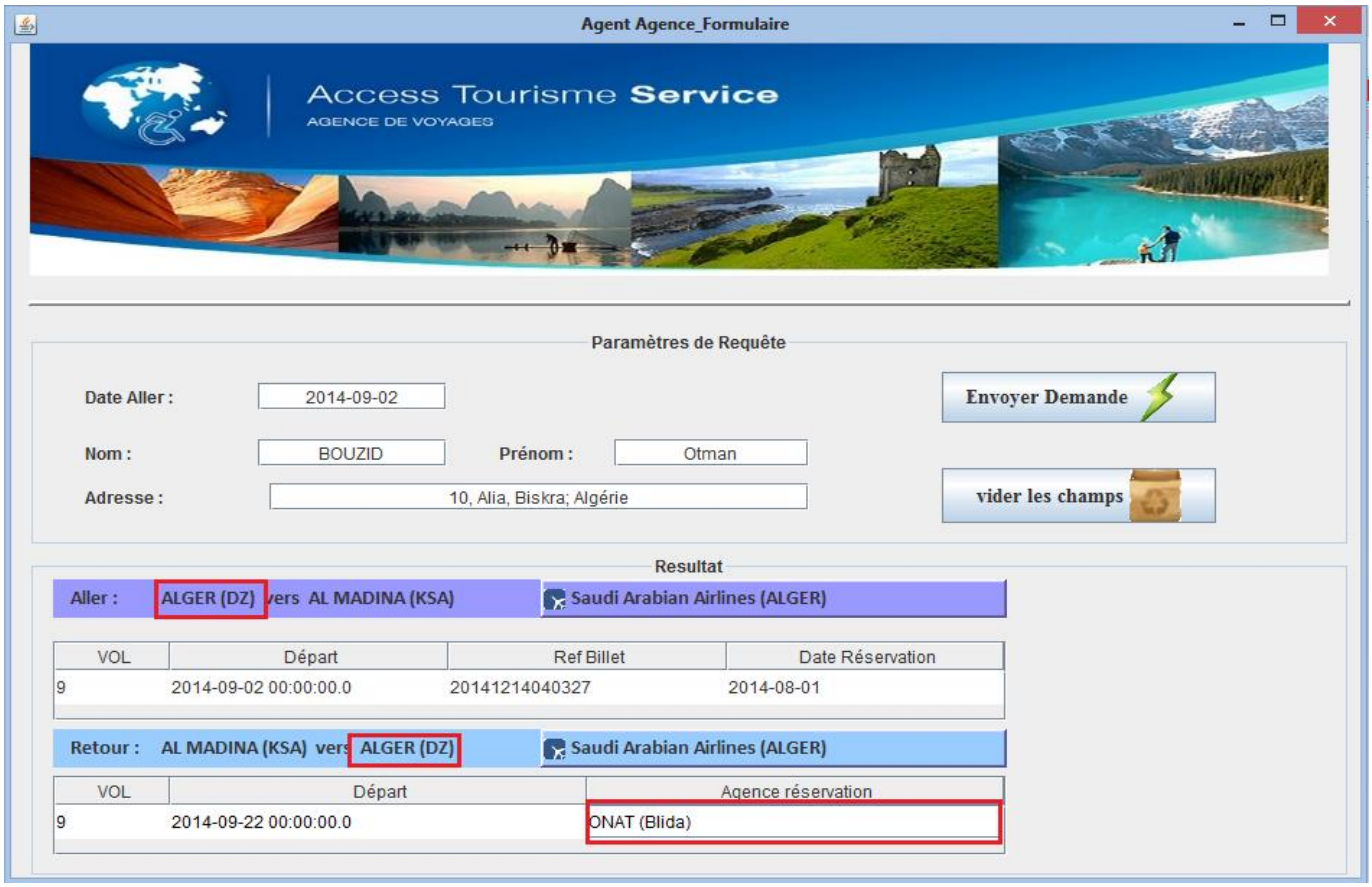


Figure IV.36 : Résultat dans Interface Voyageur d'Agent Agence – ONAT, Biskra (Cas 3)

IV.5.5. Plan de de Vols de Compagne

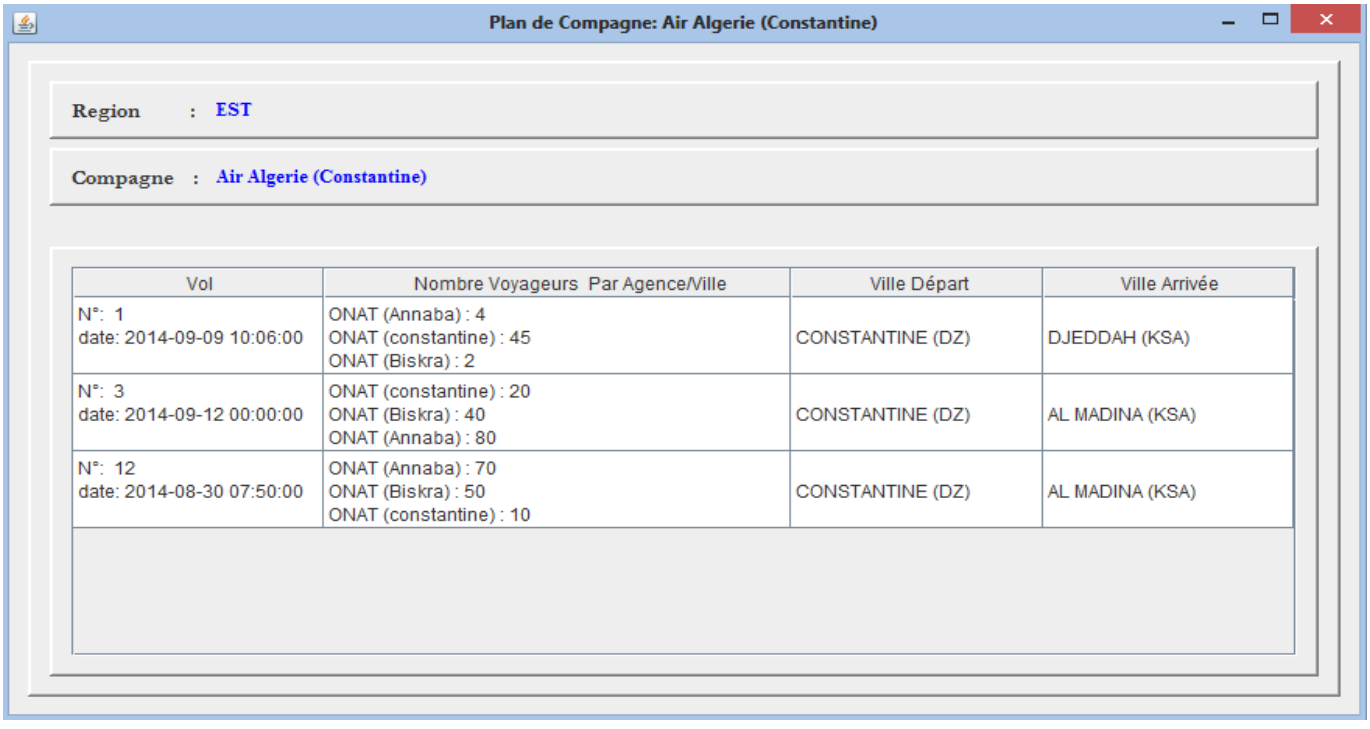


Figure IV.37 : Plan de Vols pour la compagne Air Algérie, Constantine, Région : Est

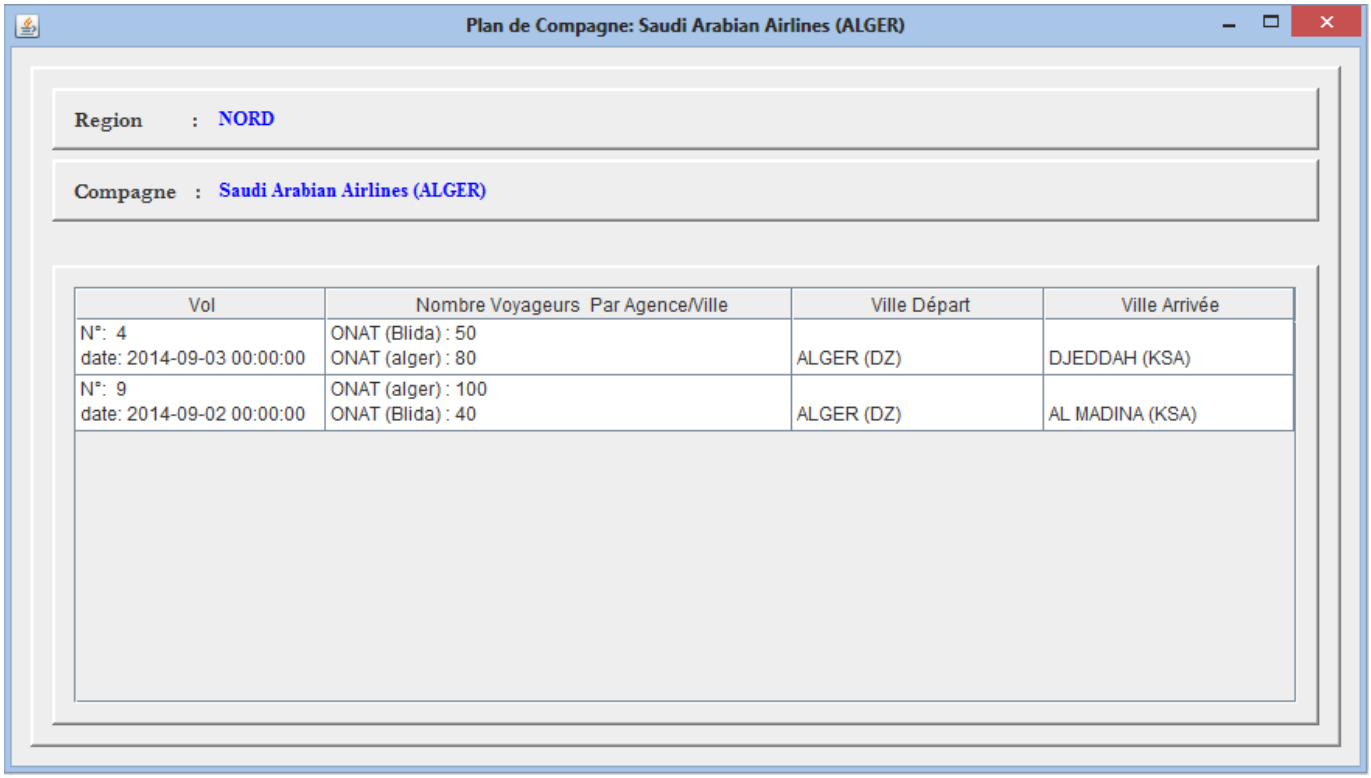


Figure IV.38 : Plan de Vols pour la compagne Saudi Arabian Airlines, Alger, Région : Nord

IV.6. Conclusion

Dans ce chapitre nous avons présenté l'implémentation de notre système, le langage JAVA s'avère pertinent pour l'implémentation de notre plateforme multi-agents. De même l'Aglet est l'outil pour le développement des agents. Ce qui nous a permis d'avoir les résultats attendus.

Conclusion générale

Tout au long de ce mémoire, nous avons présenté les différentes technologies nécessaires pour proposer une approche pour la planification décentralisée dans les systèmes multi-agents. Notre travail a mis l'emphase sur le problème de satisfaction de contraintes distribué (DCSP). Résoudre un tel problème revient à coordonner des agents de façon distribuée en se basant sur la communication et négociation. À cet effet, On a étudié le problème de « La gestion des réservations de voyageur au saison du Hadj dans l'Algérie » comme un exemple de DCSP, le problème de départ a été formalisé sous la forme d'un problème de satisfaction de contraintes distribué (DCSP pour Distributed Constraints Satisfaction Problem), enfin cette approche a été implémentée et testée dans un système multi-agent.

Afin de montrer le contexte de travail nous avons présenté au début une vision sur le paradigme d'agent et le système multi-agent en présentant leur concept et ses domaines d'application. Puis, nous avons fait des généralités sur les différents types de planification décentralisée comme, Planification décentralisée avec fusion de plans, Planification décentralisée par la satisfaction de contraintes distribués, Planification décentralisée par recherche dans l'espace d'états distribuée, …etc.

Ensuite, nous avons présenté en détail l'architecture du système, ses composantes et les idées adoptées. Nous avons montré les outils utilisés pour développer les composants du système.

Dans l'avenir nous espérons étendre ce travail à :

- Étendre l'architecture à l'adaptation avec d'autres problèmes similaires.
- Réimplémenter des solutions itératives (colonie de fourmis, Algorithme génétique, …) avec des SMA vue que la nature de ces heuristiques est de nature parallèle.
- Intégrer des protocoles et outils de sécurité (authentification, cryptage, …) pour améliore la performance du système.
- Intégrer technologie d'agents mobiles pour réduire le trafic sur le réseau et de diminuer la quantité d'informations échangées.

Bibliographie

[1]:Yves Demazeau & Antonio Rocha Costa, “Populations and Organizations in Open MAS”, 1st National Symposium on Parallel and Distributed AI, PDAI$96, Hyderabad, July 1996.

[2] : J. Ferber, “Les systèmes Multi-Agents Vers une intelligence collective”. Inter Edition, 1995, Paris

[3] : I. Jarras, B. Chaib-Draa, “Aperçu sur les systèmes Multi-Agents”. Série Scientifique du centre inter universitaire de recherche en analyse des organisations -CIRANO-2002

[4] : P. Blangi, “Etat de l'art sur les plates formes et les langages Multi- Agents Appliqués aux écosystèmes”. Master recherche:modélisation et simulation des systèmes complexes (2004/2005).

[5]: S.J. Russel and P. Norvig. Artificial Intelligence, A Modern Approach. Prentice Hall, 1995.

[6] : François Bourdon et Patrice Enjalbert, Abdel Illah Mouaddib, Serge Stinckwich, Thomas Lebarbe, Samir Saidani :«Systèmes Multi-Agents» université de C A E N 1999/ 2000.

[7] : Nicolas Dailly : « Systèmes multi-agents dans les EIAH » le 05/02/2007.

[8]: Weerdt, M., Clement, B.: Introduction to planning in multi-agent systems. Mult-iagent and

Grid Systems 5 (2009).

[9]: R. Durand, A. Cool and Z. Guessoum, Resource accumulation and subtainability of competitive advantage simulation and empirical analysis, Academy of management Conference, August 1999, Chicago.

[10]: http://set.utbm.fr/upload/gestionFichiers/planif_852.pdf

[11]: Zlatina L. M, “Planning in Mult-iagent Systems” MSc Thesis, Sofia University “St. Kliment Ohridski” Faculty of Mathematics and Informatics, 2002.

[12]: Shehory, O. and Kraus, S. (1998). Methods for task allocation via agent coalition formation. Artificial Intelligence.

[13]: Wellman, M. P., Walsh, W. E., Wurman, P. R., and MacKie-Mason, J. K. (2001). Auction protocols for decentralized scheduling. GEB: Games and Economic Behavior, 35(1-2), 271–303.

[14]: Shoham, Y., and Tennenholtz, M. (1995). On social laws for artificial agent societies: Off-line design. Artificial Intelligence.

[15]: Briggs, W. (1996). Modularity and Communication in Multi-Agent Planning. PhD thesis, University of Texas at Arlington.

[16]: Thangarajah, J., Padhgam, L., and Winikoff, M. (2003). Detecting and avoiding interference between goals in intelligent agents. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)

[17]:Edmund H. Durfee and Victor R. Lesser. Planning coordinated actions in dynamic domains. In Proceedings of the DARPA Knowledge-Based Planning Workshop, pages 18.1{18.10, December 1987

[18]: Keith S. Decker and Jinjiang Li. Coordinating mutually exclusive resources using GPGP. Autonomous Agents and Multi-Agent Systems, 3(2):113{157, 2000. URL http://www.cis.udel.edu/ decker/cv.html#PUBLICATIONS

[19]: Victor Lesser, Keith Decker, N. Carver, A. Garvey, D. Neimen, M.V. Prassad, and Thomas Wagner. Evolution of the GPGP domain independent coordination framework. Technical Report UMASS CS TR 1998-005, University of Massachusetts, 1998

[20]: Clement, B. J. and Barrett, A. C. (2003). Continual coordination through shared activities. In Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-03).

[21]: Michael P. George_. Communication and interaction in multi-agent planning. In A. Bond and L. Gasser, editors, Readings in Distributed Ar- ti_cial Intelligence, pages 200{204. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[22]: Michael P. George_. A theory of action for multi-agent planning. In Proceedings of the Fourth National Conference on Arti_cial Intelligence (AAAI-84), pages 121{125, Menlo Park, CA, August 1984. AAAI Press. Also published in [?], pages 205{209

[23]: Stuart, C. J. (1985). An implementation of a multi-agent plan synchronizer. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI- 85), pages 1031 1033, San

Mateo, CA. Morgan Kaufmann Publishers. Also published in Bond and Gasser (1988), pages 216–219.

[24]: Eithan Ephrati and Je_rey S. Rosenschein. Multi-agent planning as the process of merging distributed sub-plans. In Pro- ceedings of the Twelfth International Workshop on Distributed Ar- ti_cial Intelligence (DAI-93), pages 115{129, May 1993. URL http://www.cs.huji.ac.il/labs/dai/papers.html

[25]: Eithan Ephrati, Martha Pollack, and Je_rey S. Rosenschein. A tractable heuristic that maximizes global utility through local plan combination. In Victor Lesser, editor, Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pages 94{101, San Francisco, CA, 1995. AAAI Press, distributed by The MIT Press

[26]: I. Tsamardinos, M. E. Pollack, and J. F. Horty. Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In Proceedings of the Fifth International Con- ference on Arti_cial Intelligence Planning Systems (AIPS-00), pages 264{272, Menlo Park, CA, April 2000. AAAI Press

[27]: J. S. Cox and E. H. Durfee. Discovering and exploiting synergy between hierarchical planning agents. In Proceedings of the International Joint Conference on Autonomous Agents and Mult-iagent Systems, pages 281{ 288, 2003

[28] : Weerdt. M, Bos. A , Tonino. H, Witteveen. C, A resource logic for multi-agent plan merging, Delft University of Technology, Faculty of Information Technology and Systems, P.O. Box 356, 2600 AJ Delft, The Netherlands, 2002.

[29]: Jeffrey S. Cox, Edmund H. Durfee, Efficient Mechanisms for Mult-iagent Plan Merging, Department of Electrical Engineering and Computer Science University of Michigan Ann Arbor, MI 48109. 2004.

[30]: Gurnell.D, A Comparative Study of Approaches to Multi Agent Planning, A thesis submitted to the University of Birmingham for the degree of Doctor of Philosophy, 2005.

[31]: Kumar. V, Algorithms for Constraint- Satisfaction Problems: A Survey, AI Magazine Volume 13 Number 1 (1992).

[32]: Moyaux. T, Chaib-draa. B, D'Amours. S, Satisfaction distribuée de contraintes et son application à la génération d'un emploi du temps d'employés.

[33] : CLAIR. G, Gestion de production par système multi-agent auto-organisateur, Rapport de Master 2 Recherche Web Intelligence (WI), à l'Université Jean Monnet et l'École des Mines de Saint-Étienne 16 juin 2008.

[34] Monier. P, DBS multi-variables pour des problèmes de coordination multi-agents, Thèse de doctorat Pour obtenir le grade de Docteur de l'Université de VALENCIENNES ET DU HAINAUT-CAMBRESIS, 2012.

[35]: Shoham. Y, Leyton-Brown. K, MULT-IAGENT SYSTEMS Algorithmic, Game-Theoretic, and Logical Foundations, 2010.

[36]: Bertsekas, Distributed dynamic programming is discussed, 1982.

[37]: Guestrin , Distributed solutions to Markov Decision Problems, 2003.

[38]: http://fr.wikipedia.org/wiki/Th%C3%A9orie_des_jeux

[39]: http://www.oeconomia.net/private/cours/risqueincertitude/licencemass-tri3.pdf

[40]: http://theses.ulaval.ca/archimede/fichiers/25762/ch05.html

[41]: Mahmoud. B, Belahcene. B, Utilisation de la théorie des jeux dans les réseaux de radio cognitive pour l'accès dynamique au spectre, THESE Pour obtenir le diplôme de MASTER EN INFORMATIQUE Spécialité : : Réseaux et Systèmes Distribués (RSD,2013.

[42] : Khamoudj. CE, Classification non supervisée et théorie des jeux, LMCS, Ecole Supérieur d'Informatique -Alger- Oued Smar, 16309, El-Harrach Alger, Algérie.

[43] : http://alsoftware.pagesperso-orange.fr/french/formation/java/glossaire.html

[44] : Bettahar Aoued, « Les Aglets d'IBM », Université de Montréal, BETA08036508, Cours IFT6802 – H2003..

[45] : http://stromboli.it-sudparis.eu/~bernard/ipr/projets97-98/agents-rapport/aglet.htm

[46] : I. HOURI : Thèse << *Une approche agent mobile pour le e-commerce*>>, Master 2 académique en Informatique, juin 2011