République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 - Guelma

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Informatique Académique

Thème :

Un système intelligent à base d'ontologie pour la modélisation distribuée du dossier patient

Encadré Par :

Mr. BOURBIA Riad

Présenté Par :

BENRAZEK Alaeddine

Juin 2017

Remerciement

Je remercie Dieu « Allah »le tout puissant, qui « nous aime, nous protège, nous guide vers le bien, nous donne la chance et le bonheur » de m'a aidé à mener à bout ce travail, et à le fait concrétiser.

Je tiens à remercier Monsieur BOURBIA RIAD qui a bien voulu diriger ce mémoire ; de m'avoir accordé sa confiance et pour son aide et ses précieux conseils.

Sans oublier la personne qui donne sans compter les conseils et l'aides précieuses Pr. Dib lynda.

Mes remerciements et considérations aux membres du jury, qui ont bien voulu accepter d'examiner et juger mon travail.

Mes remerciements aussi à mes collègues dans la formation de Master, NOUAR NOUR EL HOUDA, KHOBIZI ABDENNOUR et tous les autres.

En fin, à tous ceux qui m'ont soutenue de prés ou de loin, notamment : à ma famille et à toutes les personnes qui ont contribués à l'élaboration de ce mémoire.

Dédicace

J'aimerais dédier ce modeste travail tout d'abord à mon âme sœur et ma vie « wafa », à ma grande sœur comme maman « Yasmina » et son mari « Lazher » et ses enfants « Iyade & AbdEl-Madjid », à mon grand frère comme papa « Yassine » et sa femme« Hannen » et ses enfants « Zahra, Omaima», et à mon Petit frère « Salah Eddine ».

Je le dédie également à mon grand-père, mes oncles et spécialement« Nassima », mes cousins« Dounia-Zed, Chahra-Zed et Hakim».

Je le dédie spécialement à ma chérie NOUR et aux membres de sa famille.

Je le dédie aussi à mes amis Abdennour, Semsem, Fouzi, Hayder, Bilel, Chamsou, Azou, Akrem, Khaled.

Et à toute âme généreuse, pleine de foi et d'espoir.

Que dieu les préserve tous.

BENRAZEK Alaeddine

Résumé

Résumé

Les réseaux de santé en général, sont supportés par des SI qui se structurent particulièrement autour du dossier médical. Les informations proviennent de nombreuses sources hétérogènes et distantes. De plus, l'aspect de coopération réunissant diverses compétences complémentaires n'est envisageable qu'en présence d'une communication fluide entre les intervenants

Les systèmes multi agents semble être une technologie prometteuse et s'adapte naturellement aux applications distribuées. Cependant, la mise en œuvre des applications du monde réel est généralement assez complexe. Une des principales propriétés des SMA, est la communication entre agents. Cette dernière permet aux agents d'interagir, d'échanger de l'information et de coordonner leurs activités. Seulement, il faut que les messages échangés soient compréhensibles par tous. Un moyen qui garantirait cette cohérence est l'utilisation d'une ontologie.

Ce mémoire présente une architecture basée SMA pour l'implémentation d'un système d'information médicale opérant dans un environnement dynamique distribué et intégrant une ontologie pour contrôler la communication inter-agent.

Mots-clés : Système d'information médical (SIM), Système multi-agent (SMA), Ontologie, JADE, PROTEGE.

Liste des figures

Liste des figures

Liste des Abréviations

Liste des abréviations

SIM	Systèmes d'Informations Médicaux
SMA	Systèmes Multi-Agents
SI	Systèmes d'Informations
DMP	Dossier Médical Patient
KQML	Knowledge Query and Manipulation Language
FIPA	Foundation for Intelligent Physical Agents
ACL	Agent Communication Language
AC	Actes Communicatifs
MaSE	Multi-Agent Software Engineering
MADKIT	Multi-Agents Developement Kit
JADE	Java Agent DEvelopment framework
CoMM	Corporate Memory Management
XML	Extended Markup Language
SGML	Standard Generalized Markup Language
RDF	Ressource Description Framework
OWL	Ontology Web Language
OIL	Ontology Inference Layer
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
SL	String-Encoded
LDB	Local Database
UML	Unified Modeling Language
MLD	Modèle Logique De Données
BDR	Base de Données Répartie
CSELT	Groupe de recherche de Gruppo Telecom, Italie.
DF	Directory Facilitator

Liste des abréviations

ACC	Agent Communication Channel
AMS	Agent Management System
EDI	Environnement de Développement Intégré
IBM	International Business Machines
SGBD	Système de Gestion de Base de Données
API	application program interface
OWLViz	OWL VisualiZation
JVM	Java virtual Machine
AID	Agent IDentifier
MTP	Message Transport Protocol

Liste des tableaux

Liste des tableaux

Sommaire

Sommaire

Chapitre I : Revue de la littérature

Chapitre II : Conception

Chapitre III : Implémentation

Introduction générale

Introduction Générale

Au fil des années, les environnements des Systèmes d'Information Médicaux (SIM) sont devenus plus complexes en raison de la diversité des besoins des acteurs de soins pour la prise en charge des patients. Les innovations grandissantes de l'industrie informatique et les nouvelles technologies de la communication ont permis l'augmentation de l'efficacité et de la réduction des coûts dans les SIM. L'enjeu essentiel du SIM est d'assurer l'échange de données entre les acteurs de soins. En effet, la coopération des SIM consiste à échanger des flux d'information et de rendre disponible les services propres à un système à ses différents partenaires (Zarour, 2012).

Le dossier médical est considérée comme une ressource importante à l'activité de soins, son informatisation permet de pallier à plusieurs problèmes et notamment le partage d'informations relatives aux patients entre les différents acteurs professionnels. Sachant que ces informations proviennent de diverses sources hétérogènes et distribuées, et elles sont produites et gérées de façon autonome par différentes institutions (Guerroui & al. 2016).

Le paradigme multi-agents propose des concepts particulièrement intéressants pour le développement des systèmes d'informations hospitaliers tels que l'autonomie de contrôle et la décentralisation. Seulement, le contexte médical impose que la communication entre agents doit être cohérente et que les acteurs aient la même sémantique des concepts échangés lors des discussions. Cette contrainte de compréhension, nous mène vers l'utilisation d'une ontologie dans les actes de communication inter-agent. Les ontologies permettent alors le partage de la compréhension et la communication dans des contextes particuliers et selon les besoins.

Dans le cadre de ce mémoire, nous proposons la conception et l'implémentation d'un système d'information hospitalier à base du paradigme multi-agents, opérant dans un environnement distribué et intégrant une ontologie qui valide l'aspect syntaxique et sémantique des messages échangés entre les différents agents. Cette contribution vise principalement à :

- Implémenter un outil SIM fiable.
- Garantir une prise en charge de qualité du patient lors de son parcours de soin.

- Informatiser le dossier patient et faciliter son partage.
- Rendre cohérent la communication inter-agent par la compréhension des messages échangés permet d'augmenter la qualité de l'information médicale manipulée.
- Avoir un outil d'aide à la décision.
- Avoir des données statistiques sur l'activité de l'institution médicale.

Organisation du mémoire

Ce mémoire et est organisé en trois chapitres présentant respectivement, l'état de l'art, notre proposition conceptuelle et enfin l'implémentation.

- Dans le chapitre I, en passe en revue la littérature liée aux notions dégagées de l'intitulé du projet, à savoir : l'informatique médicale, les systèmes multi-agents et l'ontologie.
- Le chapitre II présente notre contribution. Il s'agit d'une architecture basée agent et intègre une ontologie dans les actes de communication inter-agents pour la mise en place d'un Système d'Information Médicale (SIM) distribué dans une institution hospitalière.
- Finalement, le chapitre III, présente la partie réalisation projet. Une brève présentation des outils matériels et logiciels utilisés suivi d'un scénario d'exécution du parcours du patient et la description de l'ontologie utilisée dans les actes de communications entres agents.

Nous terminerons mon mémoire par une conclusion générale qui résume l'apport essentiel de notre travail et présente quelques perspectives de recherches.

Chapitre I :

Revue de la littérature

1. Introduction

Les systèmes multi-agents sont l'un des paradigmes technologiques les plus prometteurs dans le développement de systèmes logiciels distribués, ouverts et intelligents. Le paradigme multi-agents propose des concepts particulièrement intéressants pour le développement des systèmes d'informations hospitaliers tels que l'autonomie de contrôle et la décentralisation. Cependant les agents développés répondaient à des besoins bien spécifiques d'une application, entraînant une diversité et une hétérogénéité au niveau des agents développés. Il apparaît donc normal de faire communiquer ces agents hétérogènes à travers un langage dit «Langage de communication entre Agents ». Un tel langage commun doit avoir une **syntaxe** non ambiguë pour que les agents puissent « **parser**» les informations de la même manière. Il doit y avoir une **sémantique** bien définie pour que les informations puissent avoir le même sens pour tous les agents. Cette contrainte peut être levée en intégrant une ontologie dans les actes de communication entre agent. Dans ce chapitre, nous allons faire un tour d'horizon sur les concepts et définitions de l'état de l'art lié à notre thème, à savoir : Les Systèmes d'Informations Médicaux (SIM), les Systèmes Multi-Agents (SMA) et les Ontologies.

2. Les Systèmes d'Informations Médicaux (SIM)

Les Systèmes d'Informations Médicaux (SIM) sont avant tout des Systèmes d'Informations (SI). Nous commençons par définir ces derniers à travers un ensemble de définition.

2.1 Définitions

« Un système d'information (SI) est défini comme **un ensemble organisé de méthodes**, de moyens matériels (ordinateurs par exemple), **immatériels** (logiciels, brevets) et **humains** destinés à faciliter les fonctions de l'organisation et à lui fournir l'information utile pour atteindre ses objectifs » (Web 1).

En sciences de l'informatique, le SI est définit comme « l'ensemble des ressources matérielles, humaines et informatiques contribuant à la collecte, la mémorisation, la recherche, la communication et l'utilisation des données nécessaireset suffisantes pour un pilotage des opérations au sein du système de l'entreprise ». (Zarour, 2012).

Selon la littérature, Plusieurs définitions sont également attribuées au SIM :

Pour l'office d'évaluation technologique du congrès américain (DWC, 1977), un SIM est définit comme « un système assisté par ordinateur qui reçoit des données des patients normalement enregistrées, crées, et maintenues dans un dossier médical informatisé. Les données sont disponibles pour les utilisations suivantes : les soins des patients, la gestion administrative et commerciale, le suivi et l'évaluation des services de soins médicaux, la recherche épidémiologique et clinique, et la planification des ressources de soins médicaux ».

Dans (Xukai, 2007), « le SIM est une application informatique typiquement collaborative ou des gens tels que médecin, infirmier, chercheur, personnel d'assurance de santé, etc., partagent des informations du patient (incluant textes, images, données multimédia) et gérer collaborativement des tâches critiques via un réseau ».

2.2 Types d'information dans les SIM

Les SIM sont généralement caractérisés par la grande variété et le grand nombre de sources d'informations. Ces sources d'informations sont hétérogènes, autonomes et distribuées. Les flux d'information sont multiples. Ils concernent les fonctions soins, logistiques administratives et la fonction de gestion. Les informations échangées sont de natures différentes, elles dépendent de l'acteur et son domaine d'activité ainsi que l'action effectuée. Selon (Zarour, 2012), les informations dans les SIM sont classées en trois catégories :

- **Médicales** : concernent toute information médicale.
- **Organisationnelles** : les activités de l'ensemble des intervenants.
- **Communes** : rôles et fonctions hiérarchisés entre tous les acteurs.

2.3 Les acteurs impliqués

Les techniques modernes des soins desanté dépendent d'un nombre croissant de professionnels, sous forme d'équipe interdisciplinaire. La prise en charge d'un patient demande la collaboration de ces acteurs qui coopère selon un processus de soins prédéfini, la figure 1.1 nous donne un aperçu partiel sur ces acteurs (Zarour, 2012)

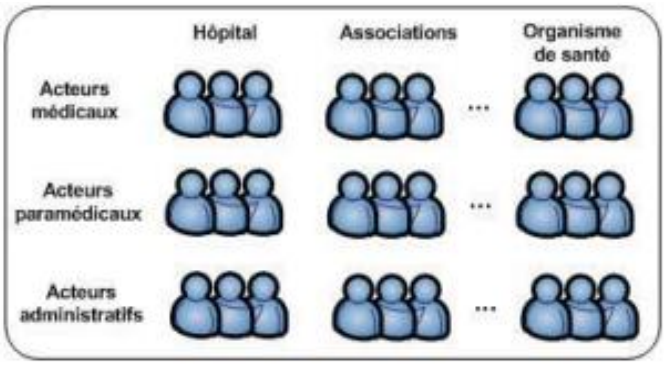


***Figure 1. 1** : Les acteurs d'un système d'information médical, Source (*Zarour, 2012*)*

2.4 Le Dossier Médical du Patient

Depuis plusieurs années, le dossier du médical fait l'objet d'une attention toute particulière. Il est considéré comme un critère majeur de la qualité des soins.

2.4.1 Définition

Selon F. Roger France (Venot, 2013), le Dossier Médical Patient (DMP) est « la mémoire écrite de toutes les informations concernant un malade, constamment mises à jour et dont l'utilisation est à la fois individuelle et collective ».

Dans (Zarour, 2012), on définit le Dossier Médical comme suit : « Le dossier du malade ne se résume pas à l'observation écrite du médecin (le dossier médical proprement dit) ou aux notes de l'infirmière (le dossier infirmier). Il englobe tout ce qui peut être mémorisé chez un malade, des données démographiques aux enregistrements électro-physiologiques ou aux images les plus sophistiquées. Compte tenu de ce rôle, le dossier du malade restera long temps l'outil principal de centralisation et de coordination de l'activité médicale ».

C'est un outil indispensable pour la pratique médicale. Il est intrinsèquement lié au processus de prise en charge du patient. Il contient des données d'origine et de nature différentes (administratives, médicales, paramédicales, sociales, etc.) qui sont générées, inférées, recueillies et notées par autant de type d'acteurs impliqués dans la prise en charge. Ces données sont autant de faits utiles correspondant aux diverses décisions et actions que nécessite l'état du patient.

2.4.2 Constitution du dossier médical et l'apport de son informatisation

Un dossier médical est constitué pour chaque patient hospitalisé dans un établissement de santé public ou privé. Ce dossier contient au moins les éléments suivants (Venot, 2013):

- La partie administrative du Dossier Médical Patient fournit les données d'identification du patient et les données sociodémographiques régulièrement

tenues à jour (suivi de l'identité de l'état civil, de la couverture sociale, du statut matrimonial, des employeurs, etc.).

- Le Dossier Médical Patient comporte les données de santé dont le volume et la complexité croit sous l'influence du développement des spécialités médicales et de leur technicité.

En effet, si le Dossier Médical Patient est destiné à la prise en charge globale d'un individu unique, son contenu doit répondre aux besoins spécifiques des différents professionnels de santé impliqués dans le processus de soin mais également dans des activités de recherche ou de santé publique. Au-delà d'une utilisation individuelle, le DMP est utilisé de façon collective pour caractériser une population à des fins de santé publique ou de recherche (épidémiologique, recherche clinique) ou dans le cadre de l'évaluation des pratiques professionnelles (Venot., 2013).Le Dossier Médical Patient est considéré de nos jours, comme un outil de collaboration et de coordination entre les différents types d'acteurs. Il permet une communication simple et une présentation unifiée de toutes les informations médicales. Les médecins utilisent le dossier médical plus comme un aide-mémoire de prise en charge que comme un outil de retranscription de l'ensemble des informations et activités réalisées (Zarour, 2012).

L'informatisation du dossier médicale du patient a pour objectif d'améliorer la qualité des soins. Elle est devenue aujourd'hui incontournable. La dématérialisation des données médicales permet de répondre aux besoins d'une médecine moderne, devenue de plus en plus technique et impliquant la plupart du temps une approche multidisciplinaire. L'informatisation du DMP contribue à remplir les objectifs suivants (Venot., 2013):

- Diminuer les temps d'accès et d'acheminement des informations médicales.
- Partager les données entre professionnels de santé.
- Répondre aux besoins sécurité et de traçabilité.
- Aider à la décision médicale et éviter les erreurs médicales.
- Réaliser des études ou produire des indicateurs de pratique professionnels et de santé publique.
- Permettre des activités d'enseignement.

3. Le concept d'agent et des Systèmes Multi-Agents (SMA)

3.1 Définition

Un Système multi-agents (SMA) est un système informatique constitué d'un ensemble d'entités appelées agents, qui sont en interactions (communication, coopération, coordination, négociation, ...) entre eux et avec leur environnement, via des modalités de perception et d'action, pour accomplir les tâches pour lesquelles elles ont été créés et qui dépassent leur capacité individuelle (Ferber, 1995).

3.2 Caractéristiques des agents

3.2.1 Notion d'agent

Plusieurs définitions de la notion d'agent existent dans la littérature. Nous avons opté ici pour la définition proposée par Ferber (1995), un agent peut être défini comme « *une entité autonome qui dispose de son propre environnement, qui est capable d'agir sur elle-même et sur cet environnement et qui peut communiquer avec d'autres agents dans un univers multi-agents. Un agent possède des compétences et offre des services, possède des ressources propres. Son comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents* ».

D'après la définition, on peut identifier pour un agent les caractéristiques suivantes :

- ***situé***– l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement;
- ***autonom****e* – l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;
- ***proactif*** – l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment;
- ***capable de répondre à temps***– l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis ;
- ***social*** – l'agent doit être capable d'interagir avec d'autres agents (logiciels ou humains) afin d'accomplir des tâches ou aider ces agents à accomplir les leurs.

3.2.2 Architecture interne d'un agent

L'architecture interne désigne l'ensemble des structures de données et des processus internes utilisés par un agent pour la prise d'une décision en vue de modifier l'environnement.

Selon Ferber, (1999), chaque agent peut être défini par un processus cyclique comportant trois phases successives (Figure 1.1):

1. la perception par l'agent de son environnement (entrées),
2. la délibération qui est l'ensemble des comportements d'un agent qui définissent la manière dont il réagit à son environnement,
3. les actions de l'agent sur son environnement (sorties).

La partie la plus essentielle de cette architecture interne, est la délibération qui consiste à la prise de décisions selon les caractéristiques comportementales de l'agent.

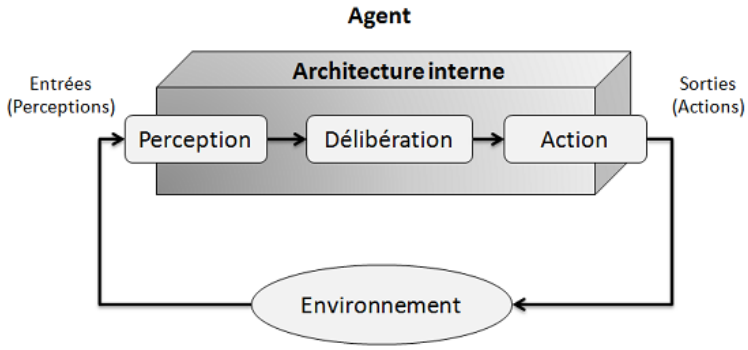


Figure 1.2 - *Interaction d'un agent avec son environnement, source (Belaqziz, 2014)*

3.2.3 Les classes d'agents

De manière générale, un agent peut être classé selon les points suivants (MaasKri et al, 2006):

A. Selon leur nature :

- ***Agents réactifs* :** Ce sont des agents qui n'ont pas une représentation explicite de leur environnement, ni de mémoire de leur passé, ni de but explicite et leur comportement est de type stimulus réponse.
- ***Agents cognitifs* :** Ce sont des agents qui possèdent une représentation explicite de leur environnement. Ils se basent sur la connaissance qu'ils ont de leur environnement et leur habileté à raisonner sur leurs connaissances.
- ***Agents hybrides* :** De façon habituelle, un agent n'est pas complètement de type réactif ni de type cognitif. C'est plutôt une combinaison de ces deux approches.

B. Selon l'utilisation :

- ***Agents collaboratifs*** : Ces agents ont des habiletés de coopération.
- ***Agents d'interface*** : Ces agents collaborent avec l'utilisateur pour effectuer certaines tâches.

- ***Agents pour la recherche d'informations*** : Ces agents effectuent, en premier lieu, une recherche d'informations parmi une collection de données et, en second lieu, procèdent à une analyse des informations utiles trouvées afin de découvrir de nouvelles connaissances.

C. Selon la technologie employée :

- ***Agents stationnaires***: Il s'agit du cas où l'agent s'exécute toujours sur la même machine.
- ***Agents mobiles*** : Ces agents s'exécutent sur différentes machines en se promenant d'un hôte à l'autre.

3.3 Caractéristiques des Systèmes Multi Agents

Les SMA sont caractérisés par :

3.3.1 L'environnement

Dans un système multi agents, on appelle environnement l'espace commun aux agents du système. Un environnement peut être (Benhawala, 2008) :

- **Accessible** si un agent peut, à l'aide des primitives de perception, déterminer l'état de l'environnement et ainsi procéder, par exemple, à une action. Si l'environnement est inaccessible alors il faut que l'agent soit doté de moyens de mémorisation afin d'enregistrer les modifications qui sont intervenues.
- **Déterministe**, ou non, selon que l'état futur de l'environnement ne soit, ou non, fixé que par son état courant et les actions de l'agent.
- **Episodique** si le prochain état de l'environnement ne dépend pas des actions réalisées par les agents.
- **Statique** si l'état de l'environnement est stable pendant que l'agent réfléchit. Dans le cas contraire, il sera qualifié de dynamique.
- **Discret** si le nombre des actions faisables et des états de l'environnement est fini.

Les caractéristiques de l'environnement influencent la façon dont on conçoit un agent car il faut tenir compte de l'évolution de l'environnement, de la capacité de l'agent de saisir cette évolution et de sa capacité à décider en conséquence.

3.3.2 L'interaction

En général, les interactions sont mises en œuvre par un transfert d'informations entre agents ou entre l'environnement et les agents, soit par perception, soit par communication. Par

la perception, les agents ont connaissance d'un changement de comportement d'un tiers au travers du milieu. Par la communication, un agent fait un acte délibéré de transfert d'informations vers un ou plusieurs autres agents (Tahri, 2009). Les conversations entre agents dans les SMA sont souvent structurées selon des schémas typiques appelés protocoles d'interaction. Il existe différents types de protocoles d'interaction, on cite :

- **La Coordination :** assure la cohérence et l'harmonie des interactions entre une communauté d'agents individuels. Elle permet donc de gérer les comportements hétérogènes et d'organiser les agents de la même manière, du fait qu'aucun agent ne possède une vue globale sur le système et parce qu'ils ont des capacités différentes, mais complémentaires.
- **La coopération :** prend place si un ensemble particulier d'agents converge vers un but global par l'achèvement de leurs propres buts locaux. Dans ce cas, la coopération est une attitude intentionnelle adoptée par les agents. La coopération permet à un agent d'accomplir des tâches qu'il ne peut pas réaliser tout seul et d'améliorer l'utilisation des ressources (Mazyad, 2013).
- **La négociation:** est un processus d'interaction entre un groupe d'agents, qui communiquent pour atteindre un accord mutuellement accepté dans un problème donné. Elle est considérée comme une méthode de coordination et de résolution de conflits.

3.3.3 L'organisation

L'organisation d'un système multi-agents est la manière dont le groupe d'agent est constitué, à un instant donné, pour pouvoir fonctionner. Cette organisation peut être statique ou dynamique. Son dynamisme est le fait que le groupe se réorganise à chaque fois en fonction de la tâche à accomplir.

3.3.4 La communication

La communication permet aux agents d'interagir, d'échanger de l'information et de coordonner leurs activités. Elle s'effectue soit par échange des messages directement, ou par l'accès à une base de données partagée contenant les informations nécessaires.

D'après la théorie des actes de langage (« ***Speech Act Theory***»)[1], chaque acte de communication multi-agents peut être décrit sous la forme d'un message, dont le type est

1Cette théorie est introduite en 1962, par le philosophe et le linguiste anglais J.-L.Austin dans son livre « *How to do thingswithwords*».

défini par le verbe performatif, tel que « ***Request***» ou « ***Inform*** ». Le contenu de ce message est décrit en utilisant les langages de communication (par exemple, *KQML*[2], *FIPA ACL*[3]).

- **FIPA-ACL :**

Ce langage est fondé sur la théorie des actes de langages. Sa spécification consiste en la définition d'un ensemble des types de messages (Tableau 1.1) et en description de leurs pragmatiques, c'est- à-dire, les effets sur les attitudes mentaux des agents émetteurs et des agents récepteurs.

Communicative Acts	Associated Meaning
accept-proposal	The action of accepting a previously submitted proposal to perform an action.
agree	The action of agreeing to perform some action, possibly in the future.
cancel	The action of cancelling some previously *request*ed action.
cfp	The action of calling for proposals to perform a given action.
confirm	The sender informs the receiver that a given proposition is true, where the receiver is known to be uncertain about the proposition.
disconfirm	The sender informs the receiver that a given proposition is false, where the receiver is known to believe, or believe it likely that, the proposition is true.
failure	The action of telling another agent that an action was attempted but the attempt failed.
inform	The sender informs the receiver that a given proposition is true.

Tableau 1.1 *: Exemple d'acte de communication dans FIPA-ACL*

La sémantique formelle de FIPA-ACL se compose de cinq niveaux (Maaskri, 2006) :

- ***Le protocole***, qui décrit les règles sociales des dialogues entre les agents ;
- ***Les actes communicatifs (AC)***, qui définissent le type de communication entre les agents (par exemple, la demande, la confirmation, etc.) ;
- ***La méta-information concernant le message*** (l'identification d'un agent émetteur et d'un agent récepteur, le contexte, etc.) ;
- ***Le langage du contenu***, qui décrit la grammaire et la sémantique associée, utilisées pour exprimer le contenu d'un message ;

[2]**KQML** (Knowledge Query and Manipulation Language)

[3]**FIPA ACL**, mis au point par la FIPA (Foundation for Intelligent Physical Agents).

- ***L'ontologie***, qui définit le vocabulaire et les significations des termes et des concepts, employées dans le contenu.

Le message FIPA-ACL est représenté sous la forme d'une liste, contenant le type de l'acte communicatif (par exemple, *INFORM, REQUEST*), le nom de l'agent émetteur et celui de l'agent récepteur, le contenu et le contexte du message, l'ontologie à utiliser pour interpréter ce contenu, et le protocole (FIPA, 1999) :

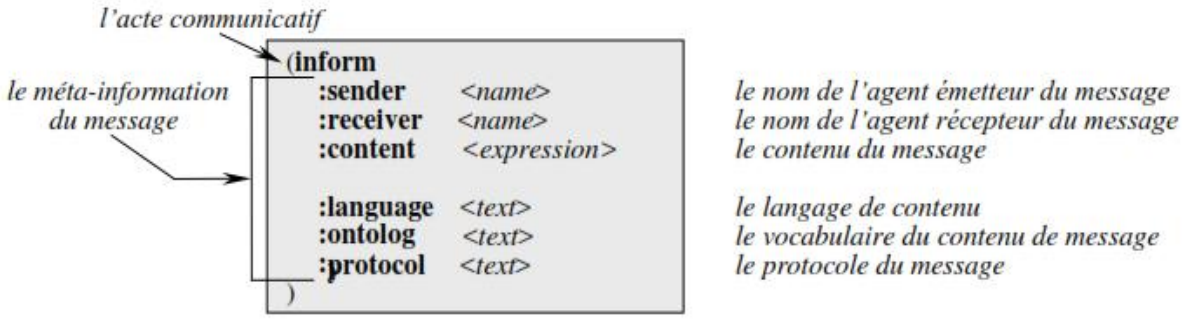


Figure 1.3 *: La structure d'un message FIPA-ACL*

3.4 Méthodologie de conception des Systèmes Multi Agents

La modélisation des SMA est actuellement un champ de recherche très ouvert. Quoique plusieurs méthodologies et approches aient été proposées, on n'a toujours pas fixé une méthode standard de développement malgré la multitude de travaux qui tentent de standardiser les méthodologies (Bernon, 2009).

Les méthodologies les plus connues sont :

- TROPOS
- Voyelles
- PASSI
- Prometheus
- Gaia
- ADELFE
- MESSAGE
- INGENIAS
- MaSE (Multi agent Software Engineering)

3.5 Plates-formes de développement des SMA (Azaiez, 2007)

Il existe une multitude de plates-formes multi-agents dédiées à différents modèles d'agent. Les plates-formes fournissent une couche d'abstraction permettant de facilement

implémenter les concepts des systèmes multi-agents. D'un autre côté, elle permet aussi le déploiement de ces systèmes. Ainsi, elles constituent un réceptacle au sein duquel les agents peuvent s'exécuter et évoluer. En effet, les plates-formes sont un environnement permettant de gérer le cycle de vie des agents et dans lequel les agents ont accès à certains services. Parmi les plateformes d'implémentation des systèmes multi agents les plus connues, on peut citer :

- MADKIT (Multi-Agents Developement kit)
- ZEUS
- JADE (Java Agent DEvelopment framework)
- Agent Builder.

Dans notre cas, nous avons utilisé la plateforme JADE, qui est reconnue comme étant l'une des meilleures plateformes open-source fournissant un environnement de développement des systèmes à base d'agents sous le langage Java, et utilisant les spécifications de la FIPA. Une description plus détaillée des fonctionnalités de cette plateforme est représentée dans le chapitre implémentation.

3.6 Domaine d'application des systèmes multi-agents

Les domaines d'application des SMA sont particulièrement riche : Support et aide à la décision, commerce électronique, systèmes manufacturiers, télécommunications, supervision de réseaux, robotique, simulation de systèmes sociaux et naturels, applications embarqués et sans fil, application internet, filtrage et recherche d'information, gestion des transports et du trafic, ingénierie médicale, jeux, e-learning (Mazyad, 2013).

4 Les ontologies

4.1 Notions et rôles d'une ontologie

La notion d'ontologie est utilisée dans des contextes différents allant de la philosophie, où le terme a été initialement introduit, à l'intelligence artificielle, et à l'ingénierie des connaissances. La définition la plus citée en littérature est celle de Thomas Grüber (Gruber, 1993) « ***une ontologie est une spécification explicite d'une conceptualisation*** ». Cette dernière a ensuite été complétée par Studer et ses collègues (Studer et al., 1998). Ils définissent une ontologie comme une « ***spécification formelle et explicite d'une conceptualisation partagée*** ».

Une autre définition plus au moins récente est introduite par (Roche, 2005) et qui résume les précédentes définitions : « ***définie pour un objectif donné et un domaine particulier, une***

***ontologie est pour l'ingénierie des connaissances une représentation d'une modélisation d'un domaine partagée par une communauté d'acteurs. Objet informatique défini à l'aide d'un formalisme de représentation, elle se compose principalement d'un ensemble de concepts définis en compréhension, de relations et de propriétés logique*s** ».

Nous pouvons donc d'abord conclure, qu'une ontologie est constituée d'un ensemble de concepts et de relations entre ces concepts et elle peut également comprendre des contraintes (Drame, 2014). Ensuite, pour la représentation Informatique, l'ontologie est vue comme un graphe orienté qui contient (Lakel, 2012) :

- Des **nœuds** représentant le vocabulaire d'un domaine particulier.
- Des **arcs** représentant les relations (ou rôles) nommées entre les concepts.

A partir de cette structure, la sémantique de chaque mot est déduite par les relations que ce mot possède dans l'ontologie, ce qui permet de restreindre les interprétations possibles. Les caractéristiques et les propriétés inhérentes des ontologies ont permis de diversifier leurs applications. Elles sont utilisées notamment, pour (Djillali, 2014):

- résoudre des problèmes de compréhension et faciliter le partage des connaissances entre personnes de spécialités différentes (Slodzian, 2000) ;
- assurer l'interopérabilité entre applications à base de connaissances (Gruber, 1991) ; elles jouent le rôle de format d'échange pour des systèmes informatiques devant coopérer ou communiquer (comme les services Web, les agents logiciels sur le Web, les systèmes multi-agents, et le « pervasive computing » (McGrath et al., 2003) ;
- accéder à des ressources hétérogènes, comme dans le système Picsel (Bidault et al., 2002) où l'ontologie unifie la représentation des données et permet, par le biais de médiateurs, d'interroger les différentes ressources de manière unifiée et transparente pour l'utilisateur ;
- permettre la réutilisation de modèles de connaissances (McGrath et al., 2003), puisque les ontologies d'un domaine donné peuvent être rendues accessibles et partagées sur le Web, et adaptées à une nouvelle application du même domaine en ajoutant de nouvelles instances ou des connaissances spécifiques; de plus des ontologies génériques sont disponibles et peuvent servir de point de départ pour des ontologies de domaine (Bergenti, 2002);
- faciliter la communication entre agents logiciels, comme dans le système ***CoMMA*** (Bergenti, 2002); l'ontologie offre alors une description commune des objets utilisés et

tient lieu de connaissance partagée; la standardisation des formats est alors une clef pour assurer leur interprétation par les agents ;

- annoter des ressources à l'aide de méta-données ; cette annotation est automatisée dans un système comme ***Magpie*** (Domingue, 2003) ; elle permet d'associer des objets formels et consensuels reflétant le sens de l'information qui se trouve dans la ressource annotée en vue d'offrir des services (navigation, recherche, etc.) sur ces ressources ; ainsi, on rend compte d'un point de vue sur les ressources ;
- améliorer les processus de recherche d'informations (Baziz, 2005) grâce à une indexation plus riche ou encore en introduisant une meilleure interprétation des requêtes utilisateur en langage naturel ; on peut ainsi dépasser les approches par mots-clefs en s'intéressant aux concepts manipulés par l'utilisateur ou en fournissant des réponses dégradées là où un système classique ne donnerait pas de résultat.

En fonction de l'usage qui en est fait, une ontologie peut être constituée de différents composants. Dans la section suivante, nous décrivons les principaux éléments constitutifs d'une ontologie.

4.2 Les composants d'une ontologie (Aouachria, 2012)

Une ontologie est constituée des éléments suivants (Gómez-Pérez, 1999) :

- **Les concepts (classes) :** sont utilisés dans un Large sens, ils peuvent être abstraits ou concrets, élémentaires ou composés, réels ou fictifs. Un concept peut aussi être la description d'une tâche, fonction, action, stratégie, processus, Raisonnement, etc. appartenant à un segment de la réalité, c'est-à-dire le domaine.
- **Les relations :** qui traduisent les associations existantes entre les concepts présents dans ce domaine.
- **Les fonctions :** soit les cas particuliers de relations dans lesquelles un élément de la relation, le (x-ième), est défini en fonction des (x-1) éléments précédents.
- **Les axiomes :** qui constituent des affirmations, acceptées comme des vérités, concernant des abstractions traduites de ce domaine par l'ontologie.
- **Les instances (modèles) :** qui désignent la description en extension de l'ontologie, véhiculant les connaissances d'un domaine.

4.3 Typologie des ontologies

Différentes typologies ont été proposées :

4.3.1 Typologie des ontologies selon l'objet de la conceptualisation (Guarino, 1998) :

- **Les ontologies de haut niveau** sont celles qui décrivent des concepts généraux (espace, temps, matière, objets, événements, actions, etc.) indépendants d'un problème ou d'un domaine d'application particulier.
- **Les ontologies de domaine** et **les ontologies de tâche,** expriment une conceptualisation qui les rend spécifiques à un domaine déterminé de la connaissance (comme la médecine ou les automobiles), ou une tâche, ou une activité générique (comme le diagnostic ou la vente), en spécialisant les concepts présentés dans les ontologies de haut niveau.
- **Les ontologies d'application** contiennent les définitions requises pour modéliser la connaissance dans une application, par exemple identifier les maladies du cœur, d'après une ontologie du domaine de la cardiologie. Ces concepts correspondent souvent aux rôles joués par des entités de domaine tout en exécutant une certaine activité, comme l'unité remplaçable ou le composant disponible.

4.3.2 Typologie des ontologies selon le formalisme utilisé pour les exprimer (Guarino, 1998) :

- **Ontologie Informelle :** l'ontologie est exprimée en langage naturel. Cela peut permettre de rendre plus compréhensible l'ontologie pour l'utilisateur, mais cela peut rendre plus difficile la vérification de l'absence de redondances ou de contradiction.
- **Ontologie Semi-informelle :** l'ontologie est exprimée dans une forme restreinte et structurée de la langue naturelle ; cela permet d'augmenter la clarté de l'ontologie tout en réduisant l'ambigüité.
- **Ontologie Semi-formelle :** l'ontologie est exprimée dans un langage artificiel définit formellement.
- **Ontologie Formelle :** l'ontologie est exprimée dans un langage artificiel disposant d'une sémantique formelle, permettant de prouver des propriétés de cette ontologie.

4.3.3 Typologie des ontologies selon leur granularité (Diallo, 2006) :

- **Granularité fine** : c'est lorsque l'ontologie est très détaillée. Elle permet de décrire précisément les connaissances. Souvent les ontologies de domaine, les ontologies de tâches et les ontologies d'application sont de ce type.
- **Granularité large** : dans le cas où l'ontologie est moins détaillée, on parle de granularité large (par exemple, les ontologies de haut niveau). Ce type d'ontologie

peut être partagée et raffinée dans d'autres types d'ontologies telles que les ontologies de domaine ou d'application.

4.4 La construction d'une ontologie

Nées des besoins de représentation des connaissances, les ontologies sont à l'heure actuelle au cœur des travaux menés en Ingénierie des Connaissances et visant à établir des représentations à travers lesquelles les machines puissent manipuler la sémantique des informations. Le processus de construction d'une ontologie doit respecter un certains nombres de critères (Gruber, 1993) (Aouachria, 2012):

- **La clarté :** La définition d'un concept doit faire passer le sens voulu du terme, de manière aussi objective que possible (indépendante du contexte).
- **La Complétude :** Une définition exprimée par des conditions nécessaires et suffisantes est préférée à une définition partielle (définie seulement par une condition soit nécessaire ou bien suffisante).
- **La cohérence :** Une ontologie cohérente doit permettre des inférences conformes à ces définitions.
- **L'extensibilité :** Il doit être possible d'ajouter de nouveaux concepts sans avoir à toucher aux fondations de l'ontologie.
- **Engagements ontologiques minimaux :** L'ontologie devrait spécifier le moins possible la signification de ses termes, donnant aux parties qui s'engagent dans cette ontologie la liberté de spécialiser et d'instancier l'ontologie comme elles le désirent.
- **Principe de distinction ontologique :** les classes dans une ontologie devraient être disjointes. Le critère utilisé pour isoler le noyau de propriétés considérées comme invariables pour une instance d'une classe est appelé le critère d'Identité.
- **Modularité :** Ce principe vise à minimiser les couplages entre les modules.
- **Réduire au minimum la distance sémantique** entre les concepts enfants de mêmes parents. Les concepts similaires sont groupés et représentés comme des sous-classes d'une classe, et devraient être définis en utilisant les mêmes primitives, considérant que les concepts qui sont moins similaires sont représentés plus loin dans la hiérarchie.
- **Normaliser :** il est préférable de normaliser les noms autant que possible.

Selon Fürst, le processus général de construction d'ontologies peut être découpé en 3 phases (Fürst, 2002) comme l'illustre la figure suivante :

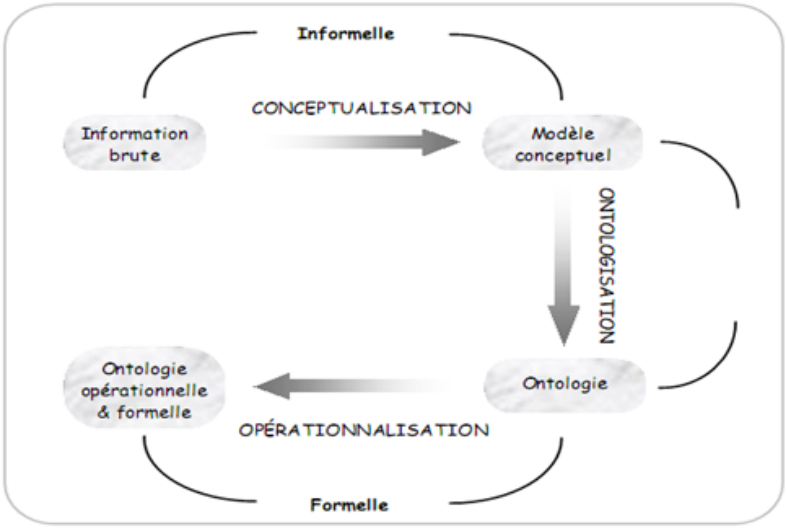


***Figure 1.4** : processus de construction d'ontologie. Source (Krama, 2014)*

- **La conceptualisation :** C'est l'identification des connaissances contenues dans un corpus représentatif du domaine considéré.
- **L'ontologisation :** C'est la formalisation, autant que possible, du modèle conceptuel obtenu à l'étape précédente. Une part des connaissances du domaine peut, à ce niveau, être abandonnée, du fait de l'impossibilité de lever certaines ambiguïtés, ou du fait des limitations de l'expressivité du langage de représentation d'ontologie utilisé.
- **L'opérationnalisation :** C'est la transcription de l'ontologie dans un langage formel et opérationnel de représentation de connaissances.

Une ontologie est toujours liée à une méthodologie de construction, à un outil de construction et avec un langage de représentation d'ontologie. Dans la section suivante, nous présentons un état de l'art des principales méthodologies, des langages de représentation et des outils de développement pour la construction des ontologies.

4.4.1 Méthodologie de développement d'une ontologie

Le développement d'une ontologie est une tâche très complexe. Une ontologie est toujours liée à une méthodologie de construction, à un outil deconstruction et avec un langage de représentation d'ontologie. Parmi celles existantes, nous citons :

- **La méthodologie « METHONTOLOGY »** (Fernández-López, *et al.*, 1997)
- **La méthodologie « On-To-Knowledge »** (Staab, *et al.*, 2001)
- **La méthodologie « OntologyDevelopment 101 »** (Noy, 2001)

4.4.2 Les langages de représentation et de manipulation d'ontologies

Plusieurs langages de représentation et de manipulation d'ontologies ont été développés. Dans cette section, nous faisons une rapide revue de ceux qui nous paraissent très représentatifs (Lakel, 2012) :

- **XML** (*eXtendedMarkupLanguage*) est un langage de description et d'échange de documents structurés, issu de SGML (*Standard GeneralizedMarkupLanguage*) et défini par le consortium Web. Il Fournit une surface syntaxique pour les documents structurés mais il ne pose aucune contrainte sémantique sur le sens de ces documents (Boutemedjet, 2004).
- **RDF (***Ressource description Framework*) est un modèle de données pour représenter les objets et les relations entre eux, fournissant une sémantique simple pour ce modèle qui peut être représenté dans une syntaxe XML.
- **RDF-Schéma** est un langage de définition de vocabulaire pour la description de propriétés et de classes représentées par des ressources RDF. RDF-S permet de définir des graphes de triplets RDF, avec une sémantique de généralisation/hiérarchisation de ces propriétés et de ces classes.
- **OWL** (*Ontology Web language*) ajoute du vocabulaire pour la description des propriétés et des classes, des relations entre classes (par exemple disjointness), des cardinalités et des caractéristiques de propriétés (par exemple symmetry). OWL est développé comme une extension du vocabulaire de RDF et il est dérivé du langage d'ontologies DAML + OIL.

4.4.3 Les outils de développement d'ontologies

Plusieurs outils existent sur le marché. On peut citer à titre d'exemple (krama, 2014) :

- **Protégé 2000**

C'est un éditeur d'ontologies distribué en open source par l'université en informatique médicale de Stanford. Le programme Protégé est basé sur la technologie Java ; il peut être étendu pour créer des applications à base de connaissances. Protégé possède une interface graphique qui offre un ensemble de fonctionnalités qui permet d'éditer les ontologies. Des extensions permettent de sauvegarder les ontologies crées sous le format OWL-DL, RDF et RDFS.

- **OntoEdit et sa suite OntoStudio :**

Il a été développé à l'Université de Karlsruhe en Allemagne. OntoStudio est un éditeur d'ontologies pour le Web sémantique, disponible dans des versions de freeware et professionnelle (payant). Il est un outil qui propose comme méthodologie On-To-Knowledge.

- **Ontolingua**

Il est créé à l'Université Stanford. Il consiste en un ensemble d'outils et de services qui supportent la construction en coopération d'ontologies, entre des groupes séparés géographiquement.

4.5 Apport des ontologies dans les SMA

Les ontologies ont été développées pour fournir des vocabulaires spécifiques dépendants du domaine d'application et employées pour contrôler la communication entre les agents. Une ontologie définit les concepts et les relations qui existent entre les mots d'un vocabulaire formel pour les agents qui l'utilisent. Les agents d'un système multi-agent partagent la même ontologie même s'ils ne possèdent pas la même base de connaissances.

4.5.1 Fonction de l'ontologie et du langage de contenu dans JADE

Dans Jade, quand un agent A communique avec un agent B, une quantité d'information est transférée à partir de A vers B au moyen d'un message ACL (*AclMessage*). A l'intérieur du message ACL, **I** est représenté comme une expression de contenu conforme à un langage de contenu approprié (par exemple FIPA-SL[4]) et codé dans un format approprié (Par exemple *String*).

Le support des langages de contenu et des ontologies fourni par Jade est conçu pour exécuter automatiquement toutes les conversions avant et après l'échange du contenu de la communication.

[4]**SL** : estun langage de contenu *String-encoded (l'expression* du contenu en **SL** est une chaîne de caractères) lisible pour l'homme et est probablement le langage de contenu le plus diffusé dans la communauté scientifique traitant les agents intelligents

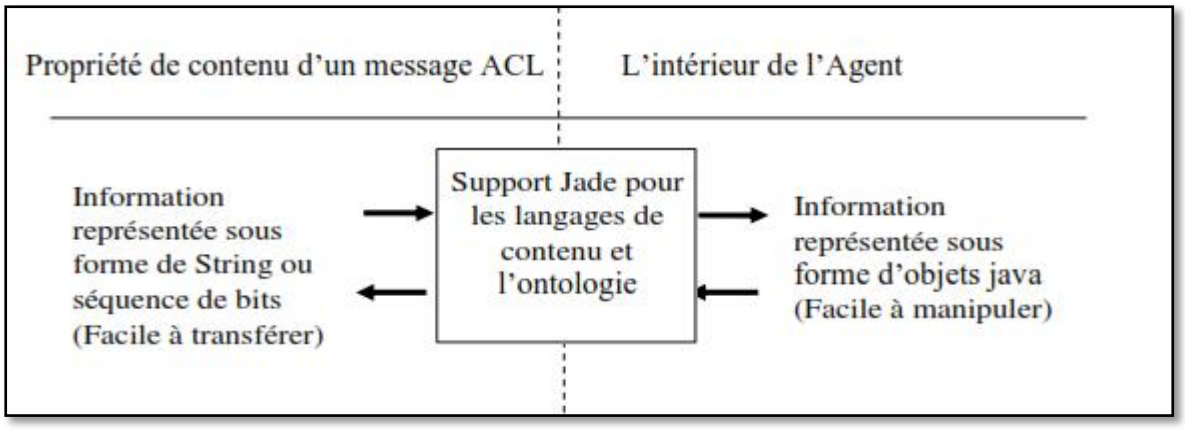


***Figure 1.5** : Le mécanisme de conversion de contenu de message suivant une ontologie*

4.5.2 Le modèle de contenu de référence

Pour que Jade exécute les contrôles sémantiques appropriés sur une expression de contenu donnée, il est nécessaire de classifier tous les éléments possibles dans le domaine de discours (les éléments qui peuvent apparaître dans une information valide employée par un agent comme contenu d'un message *ACL*) selon leurs caractéristiques sémantiques génériques. Cette classification est dérivée du langage ACL défini dans FIPA qui exige du contenu de chaque message (*ACLMessage*) d'avoir une sémantique appropriée selon la performative du message *ACL*. D'abord, nous devons faire la distinction entre Prédicat et terme :

- **Les prédicats** : sont des expressions qui indiquent quelque chose au sujet du statut du monde et peuvent être vraies ou fausses. Des prédicats peuvent être clairement employées par exemple comme contenu d'un message INFORM ou d'un message QUERY-IF, tandis qu'il ne semblerait pas raisonnable qu'il soit utilisés comme contenu d'un message de demande (REQUEST).
- **Les Termes** sont des expressions identifiant des entités (abstraites ou concrètes) qui existent dans le monde et qui font un sujet de raisonnement et de communication des agents. Ils sont classifiés en:
  - **Les expressions de concepts** sont des expressions qui indiquent des entités ayant une structure complexe qui peut être définie en termes de champs.
  - **Les Actions d'agents :** sont des concepts spéciaux qui indiquent les actions qui peuvent être effectuées par quelques agents. Il est utile de traiter des actions d'agents séparément puisque, à la différence des concepts normaux, ils sont les contenus significatifs de certains types de message *ACL* tels que REQUEST.

Une ontologie pour un domaine donné est un ensemble de schémas définissant la structure des prédicats, des actions d'agents et des concepts pertinents à ce domaine. Ainsi et pour récapituler nous pouvons dire que l'ontologie assure une sémantique commune à la terminologie utilisée dans les messages échangés entre les agents.

5. Conclusion

Dans ce chapitre, j'ai présenté une revue de littérature sur les notions théoriques en rapport direct avec le thème de notre mémoire. Cet état de l'art présente de manière plus ou moins détaillée des définitions et explications sur les SIM, le dossier médical, les concepts d'agents, de SMA et d'ontologie. Une attention particulière a été accordée à la communication dans les systèmes multi-agents et les langages qui sont employés (FIPA-ACL). Pour enrichir la sémantique des actes de langages, nous avons abordé les notions de base contribuant au développement de l'ontologie utilisée par les agents pour communiquer, ces notions sont propres à la plateforme JADE.

Chapitre II :

La Conception du Système

1. Introduction

Dans ce chapitre nous allons décrire notre contribution. Nous énumérons d'abord les objectifs à atteindre. Par la suite, nous présentons une description détaillée de notre approche de conception multi-agent, suivi par une modélisation des données collectées pour un système d'information médicale distribué. Et en fin de parcours, nous aborderons la construction de notre ontologie à intégrer dans les actes de communication inter-agents**.**

2. Objectifs

L'objectif global assigné est la mise en place d'un SIM distribué à base d'agents intégrant une ontologie comme base de dialogue dans la communication inter-agents. Cette dernière permet de coordonner et de contrôler les échanges entre plusieurs agents pour avoir un comportement collectif cohérent du système. Le système sera composé d'un ensemble de sous-système, où chacun est dédié à la réalisation d'une partie de la tâche globale de manière autonome afin de satisfaire ses propres objectifs locaux (le médecin au niveau du service assure la consultation, le traitement et le suivi, le laboratoire réalise les examens d'analyses,… etc.). La fonction clef de notre système est de faciliter le partage de l'information en permettant aux différents acteurs appartenant au SIM distribué l'accès aux données médicales.

Les sous objectifs à atteindre sont :

- Concevoir et implémenter un SIM distribué à base d'agent
- Assurer la communication et le transfert de données entres les sous-systèmes.
- Intégrer une ontologie qui coordonne et contrôle les échanges entre les agents.
- Accéderau dossier patient de manière instantanée.
- Disposer de données viables concernant l'activité des différents services hospitaliers.

3. Architecture Générale du système

La mise en œuvre de notre futur système suivra les étapes suivantes :

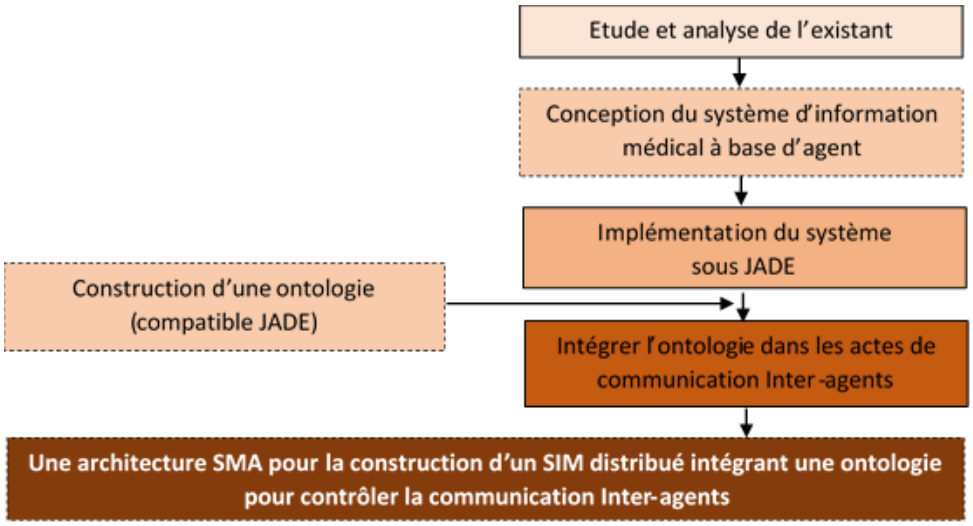


Figure 2.1 : *Etapes à suivre pour la réalisation de notre futur système*

3.1. Un Système d'Information Médicale à base d'agent

3.1.1. Description du système

Les SMA sont actuellement très largement utilisés, et particulièrement pour les applications distribués complexes nécessitant l'interaction entre plusieurs entités. L'identification des sous-systèmes opérationnels et par la suite les agents s'est effectuée en analysant le parcours du patient depuis son admission et jusqu'à sa sortie de l'établissement hospitalier. La figure suivante schématise un parcours typique d'un patient.

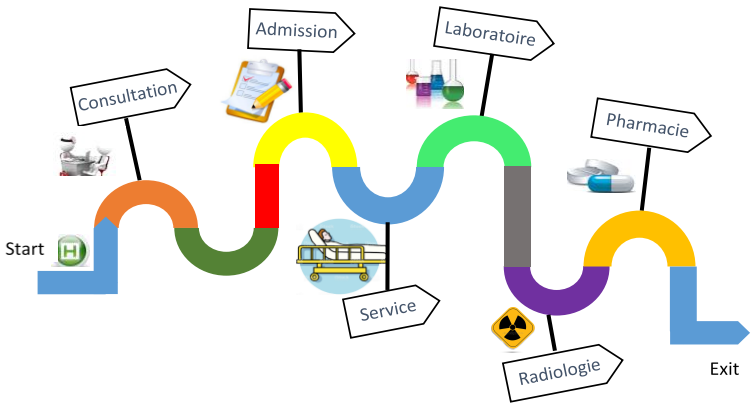


Figure 2.2 : *Parcours typique d'un patient*

A partir du parcours du patient, nous avons décelé les acteurs et les sous-systèmes suivant qui représentent des services de l'hôpital, à savoir :

- Service Urgence /Consultation Externe (Médecin Consultant)
- Le Bureau des Entrées (Agent Bureau Entrée)
- Un service Hospitalier (Médecin Chef Service, Médecin de Suivi)
- Un laboratoire d'analyse (Laborantin)

Pour une prise en charge efficiente du patient, ces services communiquent et s'échangent beaucoup d'informations traitées localement. La figure suivante dresse le flux des échanges des données entre les différents acteurs impliqués.

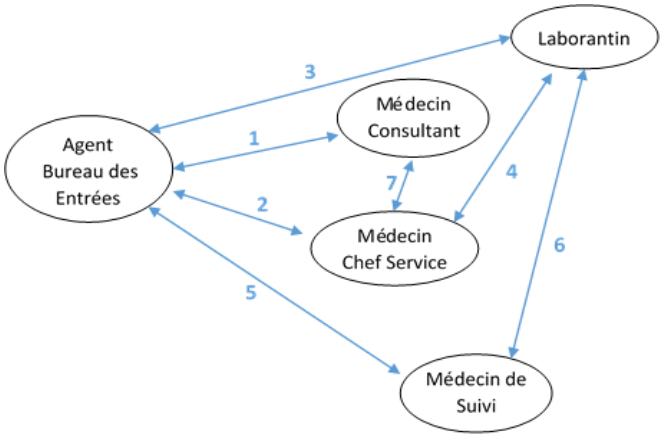


Figure 2.3 *: Flux d'échange de données entre les différents acteurs*

	Données échangées concernant
1	les consultations, les patients, les médecins consultants et les admissions.
2	les admissions, les patients, les médecins de suivis et le service.
3	les examens d'analyses et les patients.
4	les examens d'analyses et les laborantins.
5	les médicaments, les examens d'analyses prescrits, les admissions et les patients
6	les examens d'analyses, les laborantins et les examens d'analyses prescrits
7	les symptômes et les causes.

Tableau 2.1 *: Données échangées entre les différents acteurs*

3.1.2. Architecture à base d'agent

L'architecture proposée dans ce travail permet de relier les différents sous-systèmes (SI) distribués afin de satisfaire les buts organisationnels du SIM global. C'est une architecture extensible dans la mesure, où on peut rattacher d'autres structures (services hospitaliers).

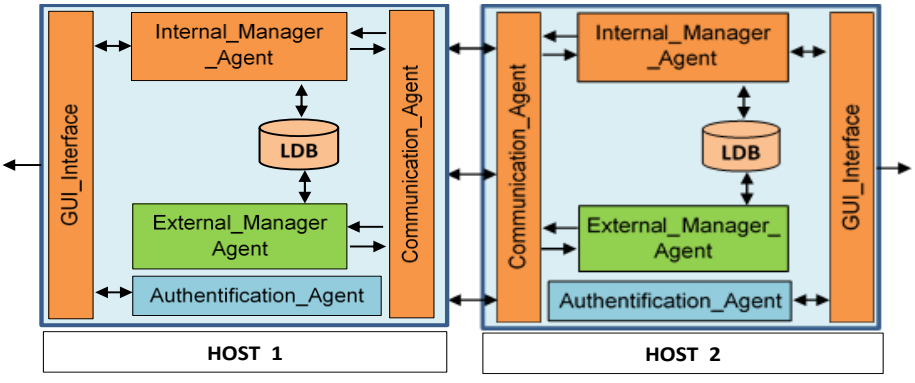


Figure 2.4 *: Structure interne à base d'agent d'une structure hospitalière*

Chaque hôte (exemple : service hospitalier) de notre système se compose de :

- Un certains nombres d'agents :
  - ***Agents d'authentification*** : responsable de l'authentification des acteurs humains
  - ***Agent de communication*** : permet d'établir la communication par envoi de message entre les différents sous-systèmes interconnectés.
  - ***Agent Gestion Interne*** : responsable de l'exécution des tâches internes associées à chaque sous-système.

    Exemple :

    IMA_Bureau des entrées : Gestion des Admissions, Facturation et Statistique.
  - ***Agent Gestion Externe*** : répond aux sollicitations extérieures
- **Interface de communication** : responsable de la liaison de l'utilisateur avec le système en présentant les fonctionnalités du système sous forme d'une interface graphique.
- La base de données locale contenant les données préparées et transformées.

3.1.3. Fonctionnement du système

Cette partie consiste à présenter notre application en se basant sur le langage UML (Unified Modeling Language) pour modéliser les interactions entre les acteurs. Parmi les objectifs d'UML : être indépendant des langages de programmation et être adapté à toutes les phases de développement. Parmi les diagrammes que propose UML, Nous utilisons : le diagramme des Cas d'Utilisation, de Séquences et de Classes.

3.1.3.1.Diagramme de cas d'utilisation

a. IMA_Urgence **b. IMA_Bureau des Entrées**

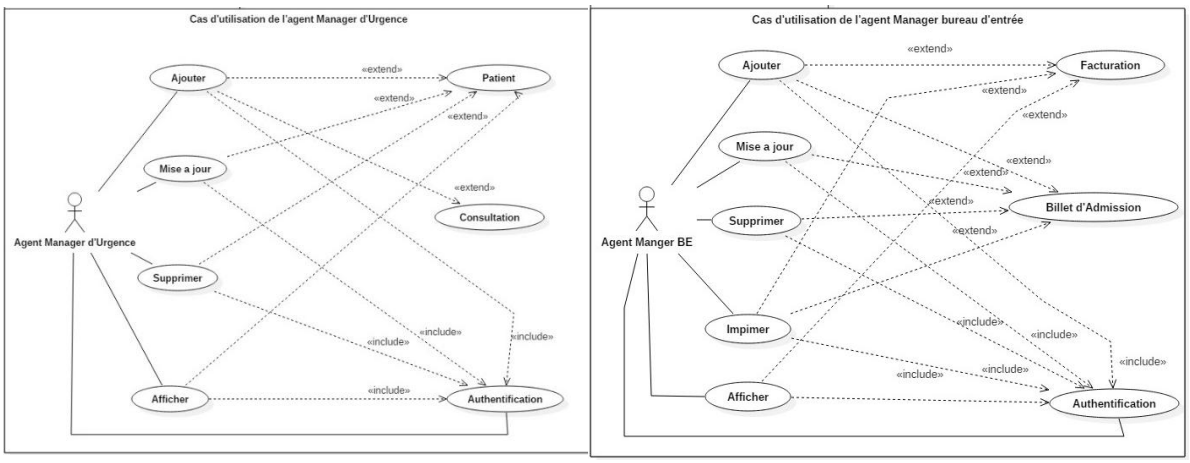


***Figure 2.5** : Diagramme cas d'utilisation d'IMA_Urgence (a) et d'IMA_Bureau des Entrées (b).*

c. IMA_ Médecin Chef Service **d. IMA_ Médecin de Suivi**

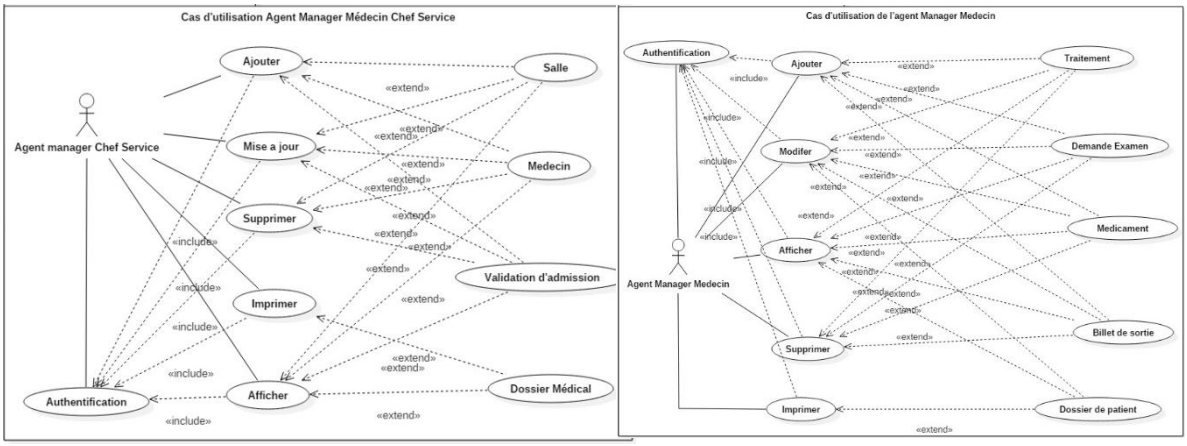


***Figure 2.6** : Diagramme cas d'utilisation d'IMA_Medecin chef service (c) et d'IMA_Medecin de suivi (d).*

e. IMA_ laboratoire

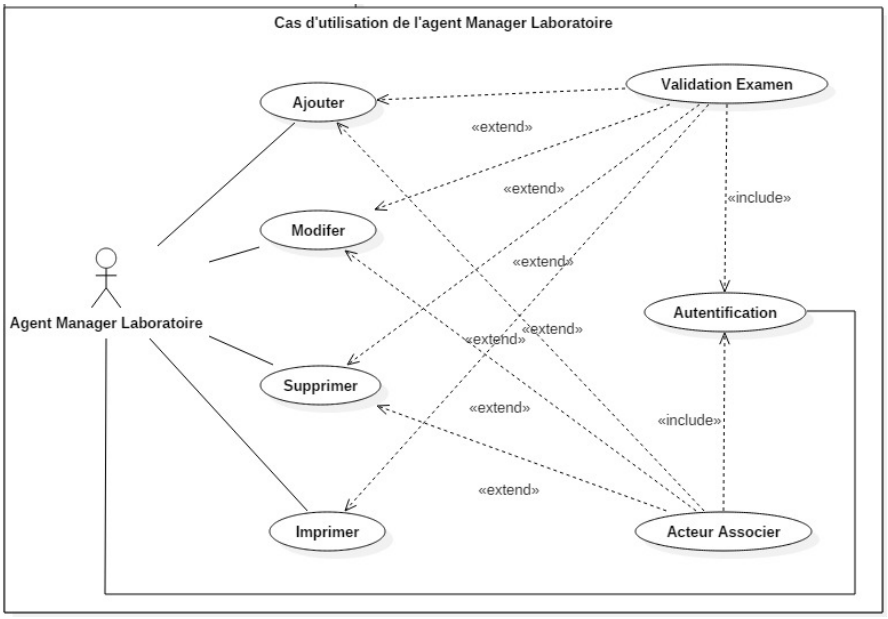


***Figure 2.7** : Diagramme cas d'utilisation d'IMA_Laboratoire.*

3.1.3.2.Diagramme de séquence

a. Diagramme de séquence : Création du dossier médical

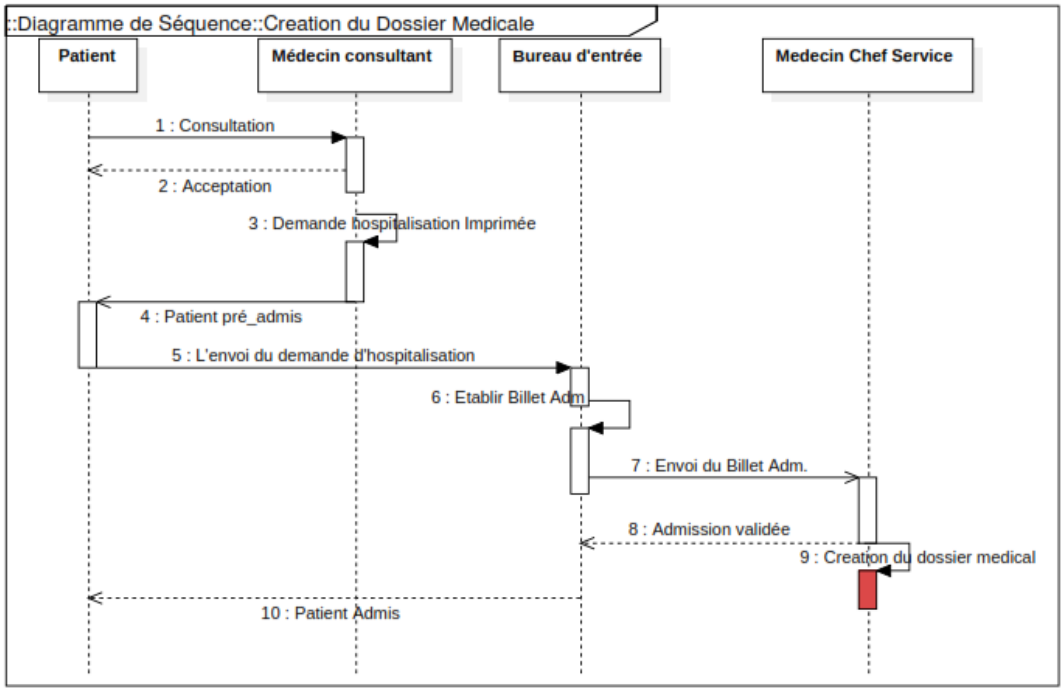


Figure 2.8 *: Diagramme de séquence de création du dossier médical.*

b. Diagramme de séquence : Suivi du patient

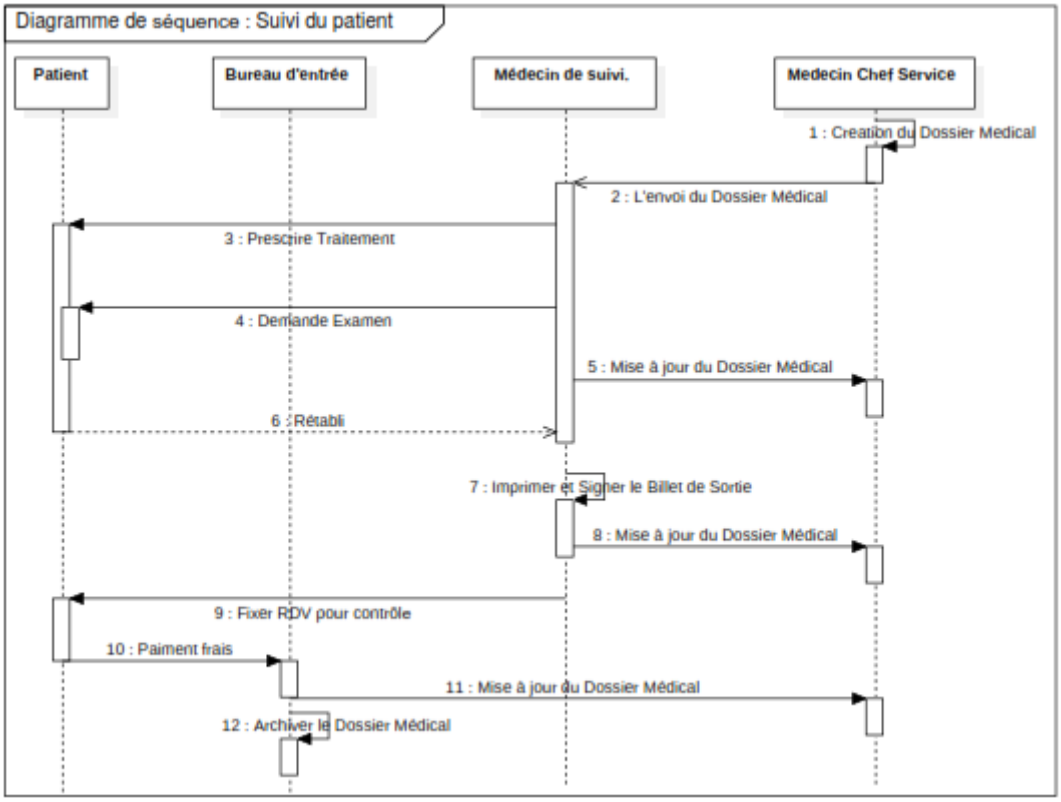


Figure 2.9 *: Diagramme de séquence de suivi du patient.*

c. Diagramme de séquence : Faire des examens d'analyses au Laboratoire

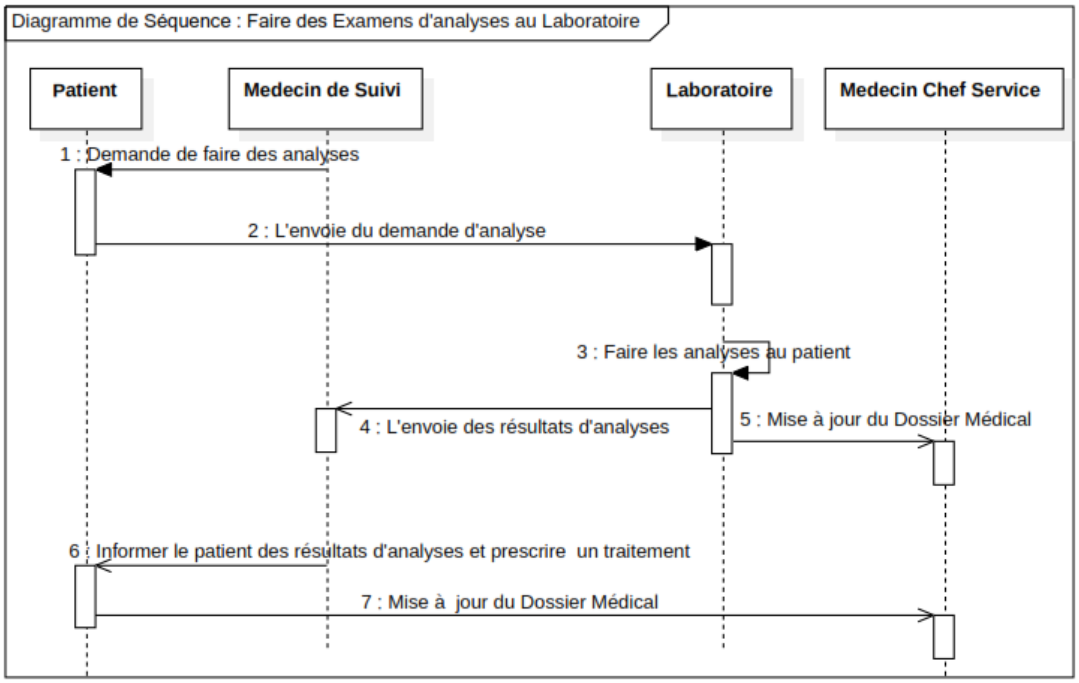


Figure 2.10 : *Diagramme de séquence des examens d'analyses au Laboratoire.*

3.1.3.3.Diagramme d'activité : Parcours du patient

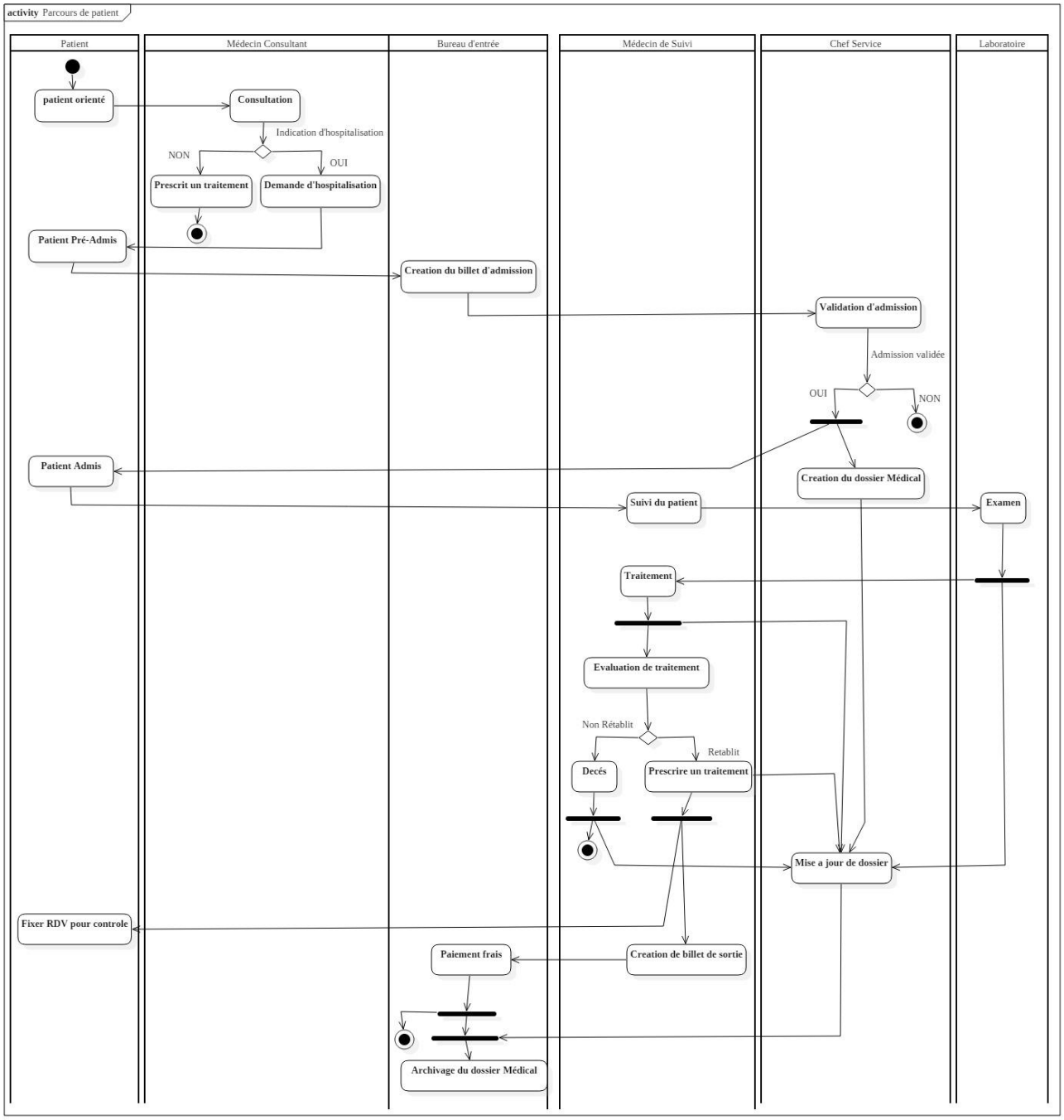


Figure 2.11 *: Diagramme d'activité du parcours du patient.*

3.1.4. Modélisation des données

Avant de concevoir un système informatique, il est essentiel de faire une analyse du domaine afin d'observer les différentes lacunes et de proposer une solution aux problèmes posés. L'analyse de l'existant constitue l'étape fondamentale de toute méthode de conception. Elle consiste à étudier toutes les procédures existantes au niveau des différents services de l'hôpital afin d'examiner la situation de gestion actuelle en vue de l'améliorer par des procédures et des méthodes bien adaptées.

Les données ont été collectées suite à un stage pratique effectué au niveau de l'hôpital « Ibn Zohr - Guelma » (Bourbia&al., 2016).Cet évènement riche en expérience avait pour objectif de comprendre le fonctionnement de l'établissement hospitalier et de réaliser l'étude :

- des différents postes de travail (bureau des entrées, chef service, laborantins, etc.…)
- du parcours des patients (de son admission à sa sortie de l'hôpital)
- des documents et fichiers existants (fiche navette, dossier médical, etc.…)
- des moyens de traitement et de circulation de l'information.

Cette phase a permis d'élaborer un dictionnaire de données contenant toutes les informations qui circulent au sein de l'établissement et qui s'est enrichi après l'ajout de nouvelles fonctionnalités au système.

3.1.4.1. Dictionnaire de données

Libellé	Mnémonique	Type de données	Taille
Identificateur Patient	Id _ Pat	N	6
Nom Patient	Nom_Pat	A	50
Prénom Patient	Prénom_Pat	A	50
Date Naissance Patient	DN_Pat	D	8
Sexe Patient	sexe _Pat	A	50
Téléphone Patient	tél_Pat	N	10
Mail Patient	mail_ Pat	AN	100
Photo Patient	Ph _ Pat	A	250
Groupe sanguin Patient	GS _ Pat	A	50
Etat matrimonial Patient	Etat Mat _ Pat	A	50
Numéro Admission	Num _Adm,	N	6
Identificateur médecin suivi	Id_med_suiv	N	6
Date Admission	Date _Adm	D	8
Heure admission	Heure _Adm	T	4
Diagnostic d'entrée	Diagnostic_entrée	A	100
Diagnostic de sortie	Diagnostic_sortie	A	100
Mode admission	Mode_Adm	A	30
Mode fin admission	ModeFin_Adm	A	100
Date fin admission	DateFin_Adm	D	8

Frais séjour	Frais_ Séjour	N	6
Identificateur Médecin	Id_ Med	N	6
Spécialité Médecin	Spécialité _ Med	A	50
Grade Médecin	grade _Med	A	50
Nom Médecin	Nom_ Med	A	50
Prénom Médecin	Prénom _Med	A	50
Date Naissance Médecin	DN _Med	D	8
Sexe Médecin	Sexe _Med	A	30
Téléphone Médecin	Tél_ Med	N	10
Mail Médecin	Mail_ Med	AN	100
Numéro Salle	Num _Salle	N	2
Nombre de lit de salle	Nombre Lit _ Salle	N	2
Nom de Salle	Nom _ Salle	A	50
Code de service	Code_ Service	N	6
Nom de service	Nom _ Service	A	50
Code Examen Prescrit	Code_ ExamPres	N	6
Date Examen Prescrit	Date_ ExamPres	D	8
Code Traitement	Code _ Trait	N	6
Nom Traitement	Nom _ Trait	A	100
Code Traitement Prescrit	Code _TraitPres	N	6
Date Traitement Prescrit	Date _TraitPres	D	8
Code Médicament	Code_ Médicament	AN	6
Libellé de Médicament	Libellé_ Médicament	A	100
Forme de Médicament	Forme_ Médicament	A	50
Frais de Médicament	Frais_ Médicament	N	4
Code Traitement Prescrit	Code _TraitPres	N	6
Quantité de Médicament	Quantité_ Médicament,	AN	50
Dosage de Médicament	Dosage_ Médicament	AN	50
Code Laboratoire	Code_ Lab	N	6
Nom Laboratoire	Nom _Lab	A	50
Description Laboratoire	Description	A	100
Identificateur ActeurAssocie	Id _Act	N	6

Spécialité Acteur Associe	Spécialité _Act	A	50
Nom Acteur Associe	Nom _Act	A	50
Prénom Acteur Associe	Prénom _Act	A	50
Date Naissance Acteur Associe	DN_ Act	D	8
Sexe Acteur Associe	Sexe _Act	A	50
Téléphone Acteur Associe	Tél _Act	N	10
Mail Acteur Associe	Mail_ Act	AN	100
Code Examen	Code_ Exam	N	6
Description Examen	Description _ Exam	A	60
Frais Examen	Frais_ Exam	N	4
Nom Examen	Nom_examen	A	50
Type Examen	Type_examen	A	60
Numéro consultation	Numéro_cons	N	6
Date consultation	Date_consultation	D	
Admis	admi	A	3
Code maladie	Code_maladie	N	6
Nom maladie	Nom_maladie	A	60
Description de maladie	Discription_maladie	A	100
Code symptômes	Code_Sym	N	6
Nom symptômes	Nom_Sym	A	60
Code causes	Code_Caus	N	6
Nom causes	Nom_Caus	A	60
Code symptômes relevé	Code_SymtoRele	N	6
Code causes relevé	Code_CauseRele	N	6

Tableau 2.2 :*Dictionnaire de données*

3.1.4.2. Diagramme de classe

Le diagramme de classes exprime la structure statique du système en termes d'objets et de relations entre ces objets. L'intérêt de ce diagramme est de modéliser les entités du système et de représenter de manière structurée l'ensemble des informations gérées par le domaine. Ces

informations sont regroupées dans des classes, elles sont décrites par des attributs et hiérarchisées grâce à des relations.

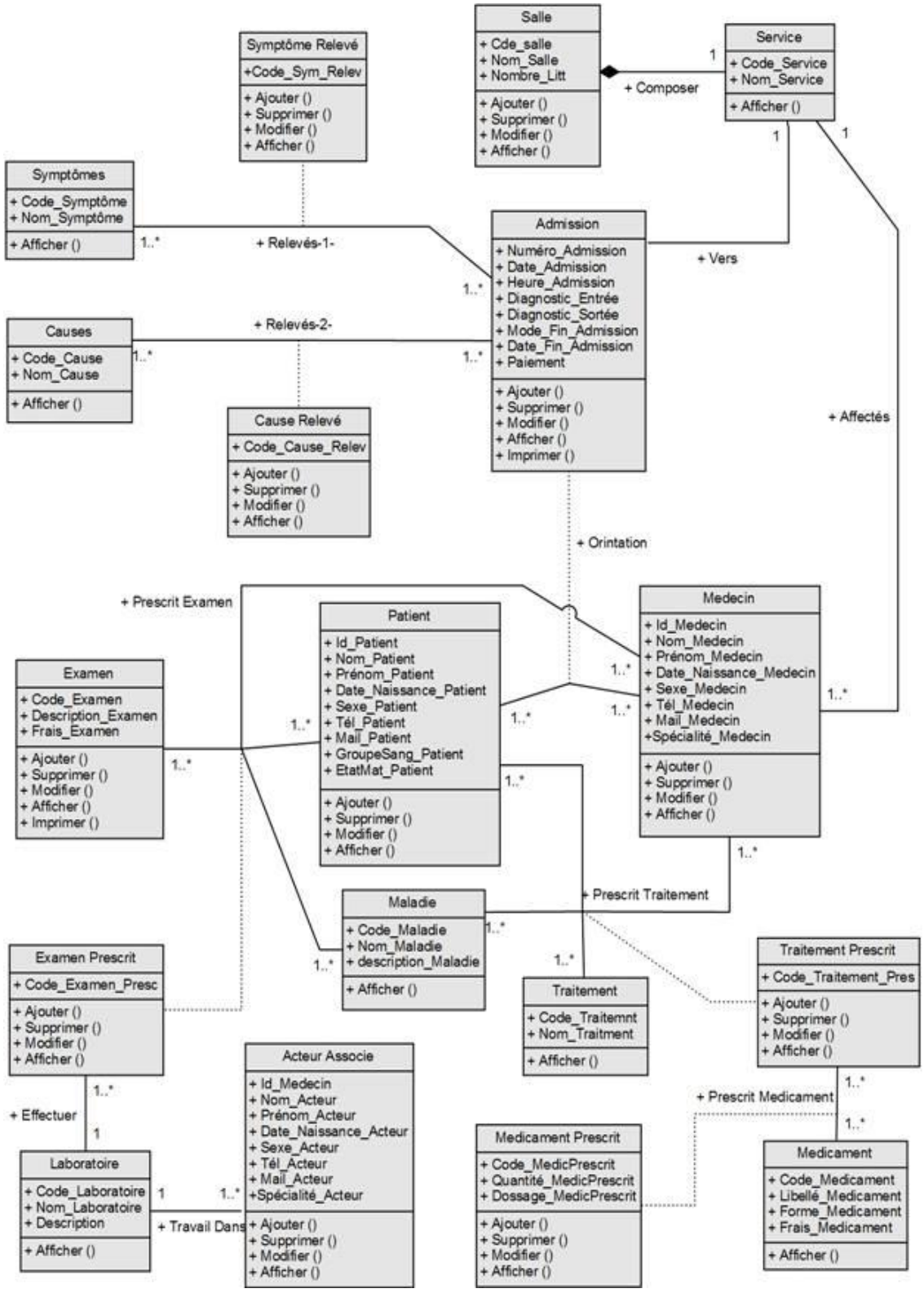


Figure2.12 :*Diagramme de classe.*

3.1.4.3. Passage de diagramme de classe vers le model relationnel (Schéma global)

a. Règle de passage

Dans cette section, Nous allons présenter les différentes règles de passages de diagramme de classe vers le model relationnelle (web 2)

- **Une classe se transforme en une relation** *:*

  Une table dans la base de données. Chaque attribut de la classe devient un attribut de cette relation, et dont une colonne de la table correspondante. L'identifiant de la classe devient la clé Primaire de la relation (elle est donc soulignée), et donc la Clé Primaire de la table correspondante.

- **Relation binaire aux cardinalités (X, 1) - (X, n), X=0 ou X=***1* :

  La Clé Primaire de la table à la cardinalité (X, n) devient une Clé Etrangère dans table à la cardinalité (X, 1).

- **Relation binaire aux cardinalités (X, n) - (X, n), X=0 ou X=1** :

  Il y a création d'une table supplémentaire ayant comme Clé Primaire une clé composée des identifiants des 2 entités. On dit que la Clé Primaire de la nouvelle table est la concaténation des Clés Primaires des deux autres tables. Si la relation est porteuse de donnée, celles-ci deviennent des attributs pour la nouvelle table.

b. Modèle logique de données

L'application des règles précédentes nous permis d'élaborer le Modèle Logique de Données (MLD).

Nom_table	Attributs
Patient	(Id_ Pat, GS _Pat, Etat Mat _Pat, Ph _ Pat, Nom_Pat,Prénom_Pat,DN_Pat, sexe _Pat, tél_Pat,mail_ Pat).
Admission	(Num _Adm, Id _Pat, Id _Med, Date _Adm, Heure _Adm, Mode_Adm, Diagnostic_ entrée,Diagnostic _sortie,ModeFin_Adm, DateFin_Adm, FraisSejour,*Code_Service**)
Médecin	(Id_Med , spécialité_Med , grade _ Med , Nom _ Med, Prénom _Med , DN _ Med , Sexe _ Med , Tél _Med , Mail_ Med , *Code_Service*)*
Salle	(Num_Salle, Nombre Lit _Salle, Nom _Salle, *Code_Service*).*
Service	(Code_ Service, Nom _Service).
Traitement	(Code _Trait, Nom _Diagn).
Médicament	(Code_ Médicament, Libellé_ Médicament, Forme _Médicament, Frais_ Médicament).
Laboratoire	(Code_ Lab,Nom_Lab, Description).

ActAss	(Id _Act, Spécialité _Act, Nom _Act, Prénom _Act, DN_ Act, Sexe _Act, Tél _Act, Mail_ Act, Code_ Lab*).
Examen	(Code_ Exam,nom_examen, type_examen,Description _Exam, Frais_ Examen).
Causes	(Code_Cause, Nom_Cause).
Sypmtômes	(Code_Sym, Nom_Sym).
Maladie	(Code_maladie, nom_maladie,_description_maladie)
Traitement_ Prescrit	(Code _TraitPres, Code_ Trait, Id_ Med, Id_Patient, Code_maladie,Date _TraitPres).
Médicament _Prescrit	(Code _TraitPres,Code_ Médicament, Code _TraitPres,Quantité _Médicament, Dosage_ Médicament).
Examen _Prescrit	(Code_ExamPres, Id_Med, Code_Exam, Date_ExamPres, Id_Patient,Code_maladie, Id_Act*,Code_Lab*).
Symptômes_releves	(Code_SymtoRele, Num _Adm, Code_Sym)
Causes_relevées	(Code_CauseRele, Num _Adm, Code_Cause)

Tableau 2.3 :*Model logique de données.*

3.1.4.4.Répartition de la base de données

- **Définition**

Une base de données répartie (BDR) est une base de données dont différentes parties sont stockées sur des sites, généralement géographiquement distants, reliés par un réseau. La réunion de ces parties forme la base de données répartie (Bouanani, 2013).

- **Méthodes de conception**

Deux approches fondamentales sont à l'origine de la conception des BDR :

a. Conception descendante (top down design)

On commence par définir un schéma conceptuel global de la base de données répartie, puis on le distribue sur les différents sites en des schémas conceptuels locaux. La répartition se fait donc en deux étapes, en première étape la fragmentation et en deuxième étape l'allocation de ces fragments aux sites. L'approche top down est intéressante quand on part du néant.

b. Ascendante (bottom up design)

L'approche se base sur le fait que la répartition est déjà faite, mais il faut réussir à intégrer les différentes BD existantes en une seule BD globale.

Concernant notre cas d'étude, nous avons choisi la méthode de conception descendante pour concevoir notre BDR.

- **La fragmentation**

La fragmentation est le processus de décomposition d'une base de données logique (telle qu'elle est vue par les utilisateurs) en un ensemble de "sous" bases de données. Cette décomposition doit être sans perte d'information (Mechid, 2011).

- **Les règles de fragmentation :** Les règles à appliquer sont :

- La complétude : Chaque élément du schéma global (R) doit se trouver dans un fragment.
- La reconstruction : On doit pouvoir recomposer (R) à partir de ses fragments.
- La disjonction : Chaque élément de (R) ne doit pas être dupliqué.

- **Types de fragmentation**

a. **La fragmentation horizontale :** la relation est divisée en plusieurs sous-relations contenant chacune un sous-ensemble des tuples (lignes) de la relation.
b. **La fragmentation verticale :** la relation est divisée en plusieurs sous-relations contenant chacune un sous-ensemble des attributs (colonnes) de la relation.
c. **La fragmentation mixte :** Elle résulte de l'application successive d'opérations de fragmentation horizontale et verticale sur une relation globale.

- **Allocation**

L'affectation des fragments sur les sites est décidée en fonction de l'origine prévue des requêtes qui ont servi à la fragmentation. Le but est de placer les fragments sur les sites où ils sont les plus utilisés, et ce pour minimiser les transferts de données entre les sites. L'allocation peut se faire avec réplication ou sans réplication. Sachant que la réplication favorise les performances des requêtes et la disponibilité des données, mais est coûteuse en considérant les mises à jour des fragments répliqués (Bouanani, 2013).

Suite à l'application des différents concepts vus dans les sections précédentes, notre BDR est présentée ainsi :

BD Local (1)		
Site	**Service « Urgence / Consultation Externe »**	
Modèle Logique Données	• **Patient_pour _consultation**(Id_ Pat, GS_Pat, EtatMat_Pat, Ph_ Pat, Nom_Pat, Prénom_Pat, DN_Pat, sexe_Pat, tél_Pat, mail_Pat). • **Médecin_Consultant** (Id_Med , spécialité_Med , grade _ Med , Nom _ Med,Prénom _Med , DN _ Med , Sexe _ Med , Tél _Med , Mail_ Med , Code_Service*).	**Opérations** - (Ajouter, Modifier, Supprimer, Afficher). - (Ajouter, Modifier, Supprimer, Afficher).

Tableau 2.4: *MLD associé au site « Service des Urgences / Consultation externe ».*

BD Local (2)		
Site	**Bureau des entrées**	
Modèle Logique Données	• **Admission**(Num _Adm, Id _Pat, Id _Med, Date _Adm, Heure _Adm, Mode_Adm, Diagnostic_ entrée,Diagnostic _sortie,ModeFin_Adm, DateFin_Adm, FraisSejour,Code _Service*). • **Patient_Admis** (Id_ Pat, GS_Pat, EtatMat_Pat, Ph_ Pat, Nom_Pat, Prénom_Pat, DN_Pat, sexe_Pat, tél_Pat, mail_Pat).	**Opérations** - (Ajouter, Modifier, Supprimer, Afficher, Imprimer). - (Ajouter, Modifier, Supprimer, Afficher, Imprimer).

Tableau 2.5: *MLD associé au site « Bureau des entrées ».*

BD Local (3)		
Site	**Service hospitalier**	
ModèleLogiqueDonnées	• **Service** (Code_ Service, Nom _Service). • **Salle** (Num_Salle, Nombre Lit _Salle, Nom _Salle, Code_Service). • **Traitement** (Code _Trait, Nom _Trait). • **Médicament** (Code_ Médicament, Libellé_ Médicament, Forme _Médicament, Frais_ Médicament). • **Médecin_service** (Id _Med , spécialité_Med , grade _Med , Nom_Med , Prénom_Med , DN_Med , Sexe_Med , Tél_Med , Mail_Med , Code_Service). • **Examen _Prescrit** (Code_ExamPres, Id_Med, Code_Exam, Date_ExamPres, Id_Pat, Id_Act, Code_Lab). • **Traitement_ Prescrit** (Code_TraitPres, Code_Trait, Id_Med, Date_TraitPres, Id_Pat). • **Médicament _Prescrit** (Code_TraitPres, Code_Médicament, Quantité_Médicament, Dosage_Médicament). • **Causes**(Code_Cause, Nom_Cause). • **Sypmtômes** (Code_Sym, Nom_Sym). • **Maladie** (Code_maladie,	**Opérations** - (Afficher) - (Créer, Modifier, Afficher) - (Afficher) - (Afficher) - (Ajouter, Modifier, Supprimer,Afficher) - (Ajouter, Modifier, Supprimer,Afficher) - (Ajouter, Modifier, Supprimer,Afficher) - (Ajouter, Modifier, Supprimer,Afficher) - (Afficher) - (Afficher) - (Afficher)

	nom_maladie,_description_maladie). • **Symptômes_releves**(Code_SymtoRele, Num _Adm, Code_Sym). • **Causes_relevées**(Code_CauseRele, Num _Adm, Code_Cause).	- (Ajouter, Modifier, Supprimer,Afficher) - (Ajouter, Modifier, Supprimer,Afficher)

Tableau2.6 : *MLD associé au site Service Hospitalier.*

BD Local (4)		
Site	**Laboratoire**	
ModèleLogique de Données	• **Laboratoire** (Code_Lab, Nom_Lab, Description_Lab). • **Act_Lab** (Id_Act, Spécialité_Act, Nom_Act, Prénom_Act, DN_Act, Sexe_Act, Tél_Act, Mail_Act, Code_Lab). • **Examen** (Code_ Exam, nom_examen, type_examen, Description_Examen, Frais_Examen).	**Opérations** - (Afficher) - (Ajouter, Modifier, Supprimer, Afficher) - (Créer, Modifier, Afficher, Imprimer)

Tableau2.7 :*MLD associé au site Laboratoire.*

3.2. Construction d'une ontologie pour contrôler la communication inter-agents

Dans un Système Multi-Agents, les agents interagissent et coopèrent pour, conjointement, réaliser une tâche ou atteindre un but commun. Une coordination s'avère donc indispensable afin d'améliorer le fonctionnement global du système. Par ce fait, pour que les agents se comprennent, la communication doit être conforme aux notions suivantes :

- La **syntaxe** : les agents doivent parler le même langage ;
- L'**ontologie** : les mêmes objets et les concepts doivent avoir la même signification pour tous les agents. les agents doivent partager une ontologie commune ;
- Le **langage déclaratif** : les agents doivent être capables de dialoguer entre eux : pouvoir s'informer, se poser des questions. Pour ce faire, les agents doivent spécifier l'état désiré dans un langage déclaratif.

FIPA acteur dans le domaine des Systèmes Multi-Agents a pour principale mission de mettre au point un standard pour la communication entre agents (FIPA 1999). Un de ses aboutissements est la norme FIPA-ACL (Chaib *et al*, 2002) (voir ch.1, section 3.3.4).

Un **Langage de Communication Agent** (ACL) doit être conçu pour l'échange entre agents d'informations, de connaissances ou de services. Le format utilisé pour l'échange des connaissances est fourni par un **langage de contenu**, indépendant du langage ACL (par exemple : KIF, FIPA-SL, Prolog, Clips). Le vocabulaire commun concerne les définitions précisées dans une **ontologie** (Figure 2.13).

Figure 2.13 : *Modèle des Langages de Communication entre Agents*

FIPA-ACL propose un système standard d'échange de messages entre agents. Nous exposons dans la figure suivante, la structure d'un message au format ACL :

([**performatif**, exemple REQUEST]
:**sender** [expéditeur, exemple agent-identifier :name Prince]
:**receiver** [destinataire, exemple agent-identifier :name Aviateur]
:**content** [contenu, exemple "((action(agent-identifier :name StExupery) (draw :object cercle)))"]
:**ontology** [ontologie du domaine, exemple PetitPrince]
:**language** [langage dans lequel est exprimé le contenu, exemple FIPA-SL]
:**reply-with** [code d'identification du message, exemple dessin])

Figure 2.14 : *La structure d'un message au format ACL (source : Ketata, 2006)*

Dans l'exemple ci-dessus, le Prince demande à l'Aviateur de dessiner un cercle. Ce message a été émis avec la performative ***Request,*** la structure standard du message apparaît clairement. Elle se compose des champs expéditeur (*:* ***sender***), destinataire (*:* ***receiver***), contexte (*:* ***ontology*****)**, message (*:* ***content*****)** et d'un certain nombre de champs métalinguistiques, en particulier le champ (*:* ***language***) qui spécifie le langage du contenu du message.

Le champ (*: ontologie*) sert quant à lui, à préciser l'ontologie décrivant la structure de la conversation entre agents.

Une ontologie pour un domaine donné est un ensemble de schémas définissant la structure des prédicats, des concepts et des actions d'agent qui sont pertinentes à ce domaine. L'exploitation du langage de contenu de JADE et de la prise en charge de l'ontologie pour permettre aux agents de communiquer et de raisonner sur les faits et les connaissances liés à un domaine donné est obtenue par les étapes suivantes (Bellifemine& al., 2007) :

1. Définir une ontologie comprenant les schémas pour les types ***de prédicat, d'action d'agent et de concept*** qui sont pertinents au domaine considéré.
2. Développer des classes Java pour tous les types de prédicat, d'action d'agent et de concept définis dans l'ontologie.
3. Choisir un langage de contenu approprié parmi ceux directement soutenus par JADE.

4. Enregistrer l'ontologie définie et le langage de contenu choisi pour l'agent.
5. Créer et manipuler l'expression de contenu comme des objets Java qui sont des classes développées dans l'étape 2 pour permettre à JADE de traduire ces objets Java en chaînes de caractères ou séquences de bits qui seront utilisées par le champ contenu d'un message ACL.

L'étude et l'analyse approfondie du flux d'échange de données (voir figure 2.3) entre services, acteurs et agents du système, nous a permis de mettre en évidence le contenu des discussions et la nature des données échangées. Cela, nous permettra d'établir un glossaire des termes utilisés pour la communication entre les agents du système. Le tableau suivant présente un extrait des informations transmises sur le réseau des sous-systèmes interconnectés.

		Message de communication inter-agent Type : Demande (Request) – Réponse (Inform)			
Agent Emetteur	**Agent Récepteur**	**Demande**	**Contenu msg. Demande**	**Réponse**	**Contenu msg. Reponse**
Ag. Com.[1] Bureau d'entrée	**Ag. Com.**Service médical	Liste des medecins	Object[] obj = { "Liste des medecins",};	Liste des medecins	Object[] obj = { "Liste des medecins", DbUtils.resultSetToTableModel(rs) };
Ag. Com. Bureau d'entrée	**Ag. Com.** Service médical	Nom Medecin	Object[] obj = { "Nom Medecin", id_med, };	Nom Medecin	Object[] obj = { "Nom Medecin", nom, prenom };
Ag. Com. Bureau d'entrée	**Ag. Com.** Service médical	FraisMedic.	Object[] obj = { "FraisMedic", id_pat}	FraisMedic.	Object[] obj = { "FraisMedic", sum };
Ag. Com. Bureau d'entrée	**Ag. Com.** Laboratoire	SommeEx	Object[] obj = { "SommeEx", ListCod, };	SommeEx	Object[] obj = { "SommeEx", sum };
Ag. Com. Laboratoire	**Ag. Com.** Service médical	Liste des demandes examens	Object[] obj = { "Liste des demandes examens"}	Liste des demandes examens	Object[] obj = { "Liste des demandes examens", DLM, DLM1, DLM2, DLM3, DLM4, DLM5,DLM6};
Ag. Com. Service médical	**Ag. Com.** Laboratoire	Liste NomEx	Object[] obj = { "ListNomEx GAD", ListCodeEX}	Liste NomEx	Object[] obj = { "ListNomEx GAD", DLM };
Ag. Com. Laboratoire	**Ag. Com.** Bureau d'entrée	Liste NomPat.	Object[] obj2 = { "ListNomPat", ListCodePat}	Liste NomPat.	Object[] objr = { "ListNomPat Medic", DLMnom, DLMprenom };

Tableau 2.8 *: Extrait des messages d'échanges entre agents*

[1] **Ag. Com. : Agent Communication**

A partir de ce tableau détaillé, l'identification des schémas de structure des prédicats, des concepts et des actions d'agent peut être entamée, sachant que par définition :

- **Prédicats** : expressions sur l'état du monde. Ils sont généralement utilisés dans les actes de communication de type « INFORM et QUERY-IF,.. »
- **Actions des agents** : expressions qui indiquent des actions que les agents peuvent. En règle générale, ils sont utilisés dans les messages de type « demande ».
- **Concepts** : expressions qui représentent des objets représentant une structure avec plusieurs attributs.

Nous utilisons le logiciel PROTÉGÉ pour illustrer les schémas de structure des prédicats, des concepts et des actions d'agent. La figure 2.15 montre alors la hiérarchie de l'ontologie construite manuellement.

Figure 2.15 : *Hiérarchie de l'Ontologie construite sous PROTÉGÉ.*

Nous donnons dans la figure 2.16, un aperçu de l'ontologie édité sous PROTÉGÉ et dans la figure 2.17 un extrait du code OWL généré par PROTÉGÉ.

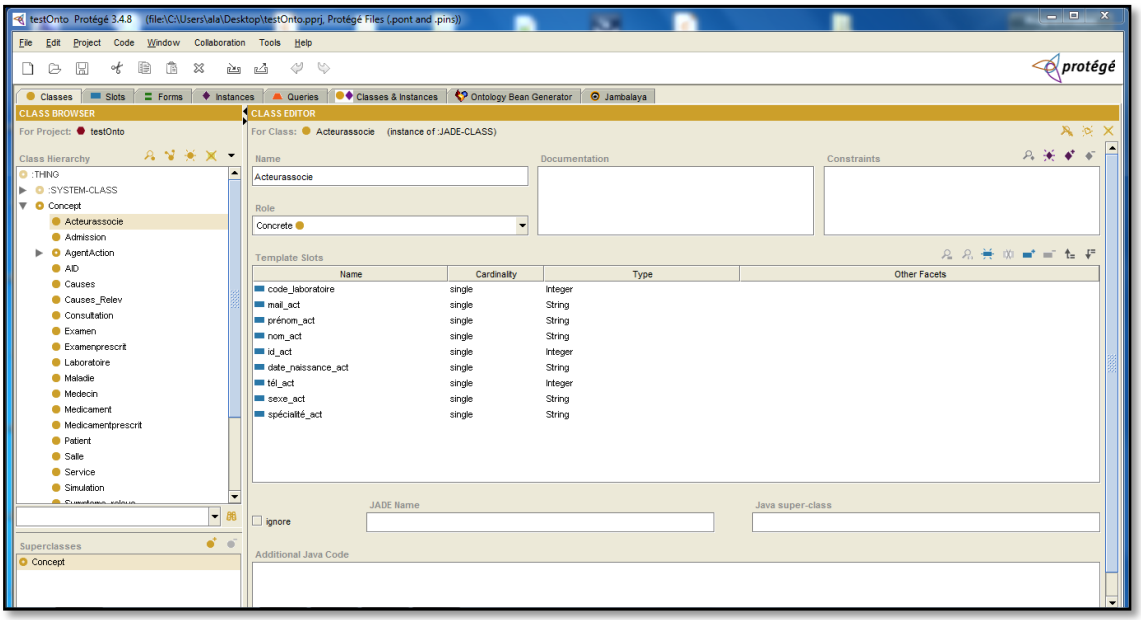


Figure 2.16 : *Ontologie édité sous l'outil PROTÉGÉ.*

```
import jade.content.onto.*;
import jade.content.schema.*;
public class OntoMedicaleOntology extends jade.content.onto.Ontology  {
  //NAME
  public static final String ONTOLOGY_NAME = "ontoMedicale";
  // The singleton instance of this ontology
  private static Ontology theInstance = new OntoMedicaleOntology();
  public static Ontology getInstance( ) {
     return theInstance;   }
   // VOCABULARY
    public static final String PATIENT_ID_PATIENT="patient";
    public static final String PATIENT_ID="Patient_ID";
    public static final String GET_PATIENT_PATIENT="patient";
    public static final String GET_PATIENT="Get_Patient";
    public static final String PATIENT_ID_PATIENT="id_patient";
    public static final String PATIENT_NOM_PATIENT="nom_patient";
    public static final String PATIENT="Patient";
  /*** Constructor  */
  private OntoMedicaleOntology(){
    super(ONTOLOGY_NAME, BasicOntology.getInstance());
    try {
    // adding Concept(s)
    ConceptSchema patientSchema = new ConceptSchema(PATIENT);
    add(patientSchema, onto-test.Patient.class);
    // adding AgentAction(s)
    AgentActionSchema get_PatientSchema = new AgentActionSchema(GET_PATIENT);
    add(get_PatientSchema, onto-test.Get_Patient.class);
    // adding Predicate(s)
    PredicateSchema patient_IDSchema = new PredicateSchema(PATIENT_ID);
    add(patient_IDSchema, onto-test.Patient_ID.class);
    // adding fields
patientSchema.add (PATIENT_NOM_PATIENT, (TermSchema)getSchema(BasicOntology.STRING),
ObjectSchema.OPTIONAL);
    patientSchema.add (PATIENT_ID_PATIENT, (TermSchema)getSchema(BasicOntology.STRING),
ObjectSchema.OPTIONAL);
    get_PatientSchema.add (GET_PATIENT_PATIENT, patientSchema, ObjectSchema.OPTIONAL);
    patient_IDSchema.add (PATIENT_ID_PATIENT, patientSchema, ObjectSchema.OPTIONAL);
```

Figure 2.17 : Fragment *du code OWL généré par PROTÉGÉ.*

4. Conclusion

Je présenté dans ce chapitre, notre conception détaillée d'une architecture d'un système d'information hospitalier opérant dans un environnement distribué et intégrant une ontologie pour contrôler la communication inter-agents. Le chapitre suivant, présente la phase d'implémentation de notre système.

Chapitre III :

Implémentation du Système

1. Introduction

Après avoir élaboré la conception de notre système, nous abordons dans ce chapitre le dernier volet de ce mémoire à savoir l'implémentation. Nous commençons par la présentation de l'environnement matériel et logiciel utilisé pour développer notre application, puis la présentation des fonctionnalités et quelque interface de notre système en suivant un scenario d'utilisation bien précis.

2. Environnement de réalisation

2.1. Environnement matériel

Le développement de l'application est réalisé via deux ordinateurs portables ayant les caractéristiques suivantes :

Caractéristique	Toshiba	Toshiba
Marque	Toshiba	Toshiba
Processeur	Intel(R) Core (TM) i3-3110M CPU @ 2.40GHz	Intel(R) Core (TM)i3 CPU M 380 @ 2.53GHz
RAM	4.00 Go	4.00 Go
Disque dur	102 Go	195 Go
Système d'exploitation	Windows 7, 64 bits	Windows 7, 64 bits

Tableau 3.1 : Environnement Matériel de développement.

2.2. Environnement Logiciel

2.2.1. Plateforme multi-Agent JADE

JADE est une plate-forme multi-agents développée en Java par **CSELT**[1] qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA.

JADE possède trois modules principaux, nécessaire aux normes FIPA (Miguel, 2013) :

- **DF** « *Directory Facilitator* » : fournit un service de « pages jaunes » afin de pouvoir mettre en relation un agent avec les compétences qui lui incombent. Un agent peut faire appel en tout temps au DF afin d'enregistrer ses comportements ou d'obtenir des informations sur d'autres comportements d'agents.

[1]**CSELT**: Groupe de recherche de Gruppo Telecom, Italie.

- **ACC** « *Agent Communication Channel* » : gère la communication entre les agents de la plate-forme, c'est la méthode de communication implicite qui offre un service fiable et précis pour le routage des messages.
- **AMS** « *Agent Management System* » : supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

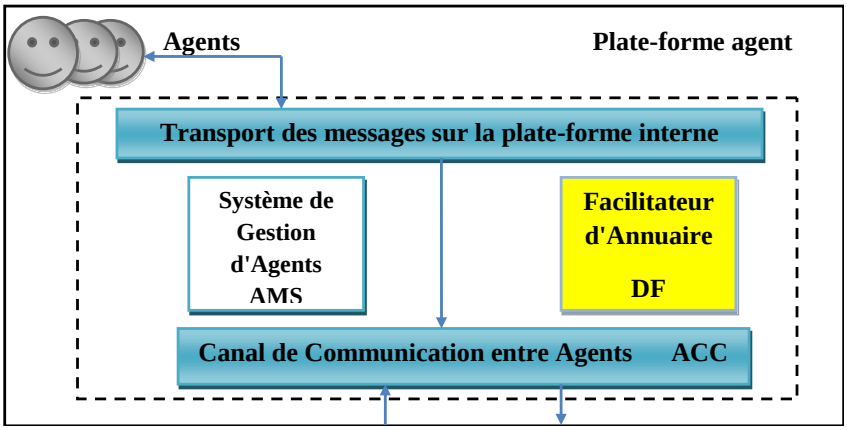


***Figure 3.1** : Les modules principaux pour JADE.*

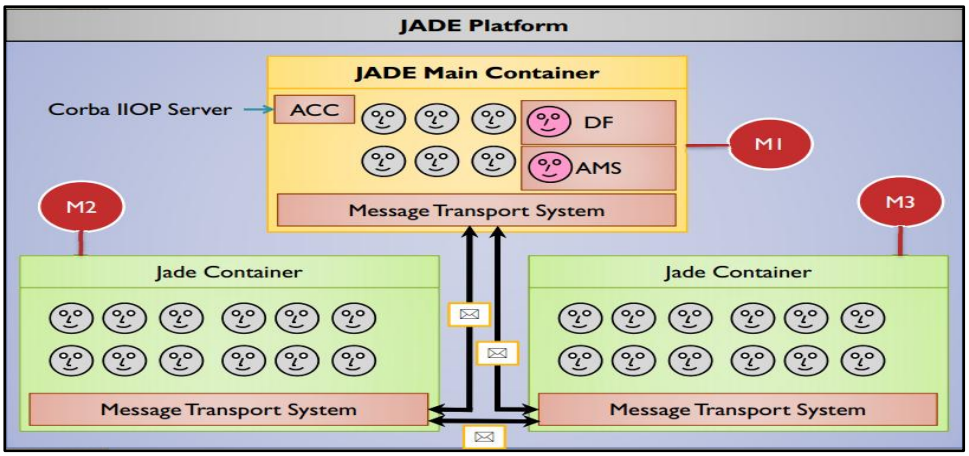


***Figure 3.2 :** Architecteur logicielle de JADE, Source (Web6)*

JADE est un middleware qui facilite le développement de systèmes multi-agents. Il contient :

- ***Un Runtime Environment*** est l'environnement d'exécution où les agents JADE peuvent "*vivre*" et qui doivent être actifs sur un hôte donné avant qu'un ou plusieurs agents puissent être exécutés sur cet hôte.
- ***Une bibliothèque de classes*** que les programmeurs doivent/peuvent utiliser (directement ou en les spécialisant) pour développer leurs agents.
- ***Un ensemble d'outils graphiques*** qui permettent d'administrer et de surveiller l'activité des agents dans la plateforme.

Chaque instance en cours d'exécution de l'environnement d'exécution JADE s'appelle ***Container***, puisqu'il peut contenir plusieurs agents. L'ensemble des conteneurs actifs s'appelle **une plate-forme**. Un seul conteneur principal spécial doit toujours être actif dans une plate-forme (***Main container)*** et tous les autres conteneurs s'inscrivent dès qu'ils commencent (Giovanni, 2009).

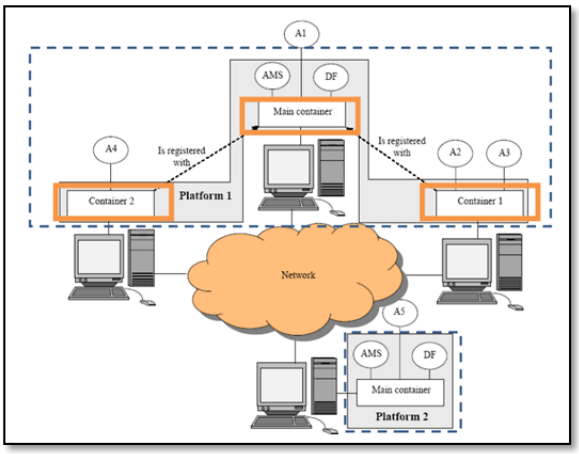


Figure 3.3 : *Containers et plates-formes, Source (Web6).*

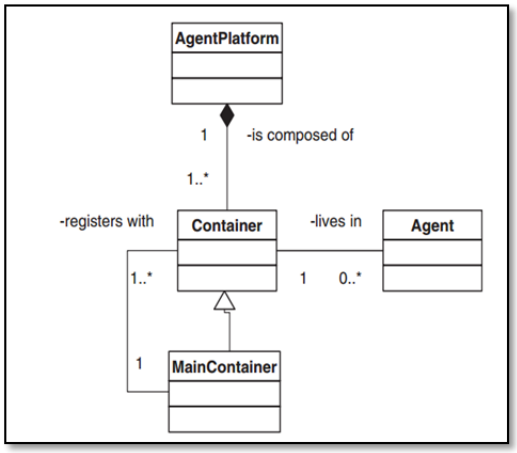


Figure 3.4 *: Relation entre les principaux éléments architecturaux.*

La plateforme multi-agent ***JADE***, offre les avantages suivants :

- Plate-forme assez facile à mettre en place.
- Disponibilité de packages sur lesquels nous nous sommes appuyés pour développer notre application.
- Documentation claire et complète.
- Licence gratuite.

Installation de JADE

Voici les étapes à suivre pour installer JADE : (Web 2)

1. Téléchargez le fichier JADE-all-3.6.zip de l'adresse suivante : http://jade.tilab.com/download.php.
2. décompressez le fichier.
3. Créer un projet qui s'appelle Agent Project

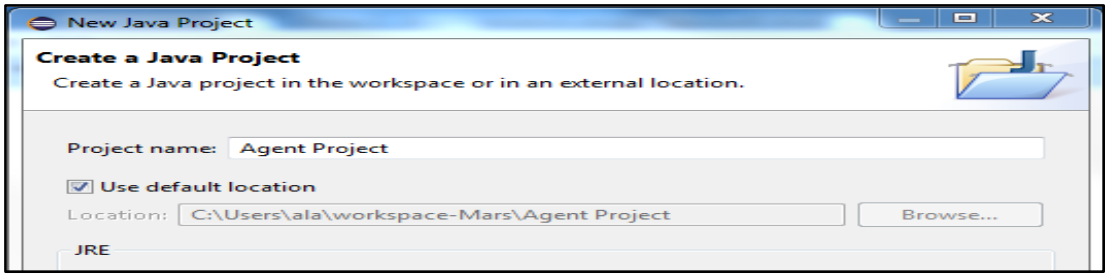


Figure 3.5 : *Création de projet.*

4. Après la création d'un projet, maintenant vous construisez un chemin de JADE. Dans l'Explorateur de packages, un Clic droit sur Projet → Cliquez sur BuildPath et après cliquez sur Libararies.
5. Dans Java BuildPath Cliquez sur AddExternal jar.
6. Choisissez le chemin ou vous avez enregistré JADE\bin\lib\jade.jar.

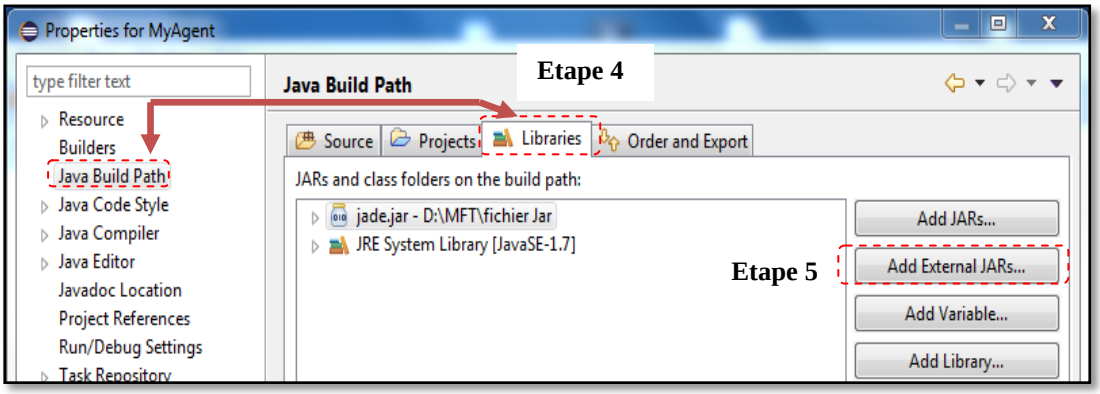


Figure 3.6 : *Importation de fichier jade.jar*

7. Après l'ajout du fichier jar, créer la classe *MyAgent*.
8. Après la création, importer *jade.core*.
9. On doit faire fonctionner Jade Environnement sous éclipse, un clic droit sur MyAgent.java, et cliquez sur Run → Configuration.
10. Dans Run Configuration, cliquez sur Java Application → New launch Configuration.
11. Cliquez sur le bouton « Search », et sélectionnez fichier *jade. Boot.*
12. Cliquez sur l'onglet Argument, et écrire ***–gui*** sur les arguments du programme, puis cliquez sur le bouton « A***pply*** », puis cliquez sur «***Run*** ».
13. Cliquez sur le bouton « ***Run*** » et après quelques secondes, vous verrez la fenêtre de la plate-forme jade, voir les figures suivant :

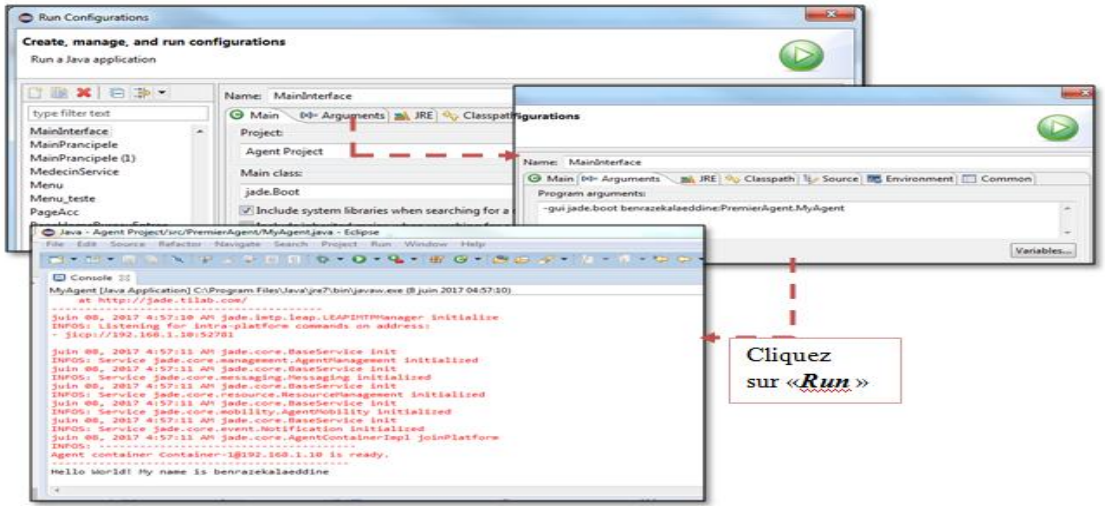


Figure 3.7 *: Configuration du projet avec JADE.*

2.2.2. Langage de programmation Java

Java C'est un langage très utilisé, notamment par un grand nombre de programmeurs professionnels, ce qui en fait un langage incontournable actuellement. Java est un langage de programmation moderne développé par Sun Microsystems (aujourd'hui racheté par Oracle) en 1995. Il ne faut surtout pas le confondre avec JavaScript (langage de scripts utilisé principalement sur les sites web). Le langage Java a l'avantage d'être ***modulaire***, ***rigoureux*** (la plupart des erreurs se produisent à la compilation et non à l'exécution) et ***portable*** (un même programme compilé peut s'exécuter sur différents environnements).

Eclipse est un EDI environnement de développement intégré, c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par **IBM**[2], est gratuit et disponible pour la plupart des systèmes d'exploitation. Au fur et à mesure que vous programmez, Eclipse compile automatiquement le code que vous écrivez, en soulignant en rouge ou jaune les problème qu'il décèle. Eclipse présente les points forts suivants (Web 4) :

- Une plate-forme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plug-ins.
- Plusieurs versions d'un même plug-in peuvent cohabiter sur une même plate-forme.
- Un support multi langage grâce à des plug-ins dédiés : Cobol, C, PHP, C#, ...
- Support de plusieurs plates-formes d'exécution : Windows, Linux, Mac OS X, ...
- Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT.
- La plate-forme est entièrement internationalisée dans une dizaine de langue sous la forme d'un plug-in téléchargeable séparément.
- Un historique local des dernières modifications.

2.2.3. Système de Gestion de Base de Données (SGBD)

Le **SGBD** est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la base de données. Parmi ceux qui existent, nous avons choisi **MySQL**. (Web 5)

MySQL est un système de gestion de bases de données relationnelles. Le SQL est

[2]**IBM** : International Business Machines

le langage standard pour les traitements de bases de données. MySQL est Open Source, Ce qui rend MySQL très intéressant et l'API (application program interface) dont il dispose. Vous pouvez en effet l'intégrer dans des applications écrites en : C, C++, Eiffel, Java, Perl, PHP, Python, Ruby et Tcl.

Parmi les points forts de ce logiciel :

- **Rapide** : Le serveur MySQL est très rapide. Des tests de performances sont disponibles sur le site de MySQL
- **Facile à utiliser** : MySQL est beaucoup plus simple à utiliser que la plupart des serveurs de bases de données commerciaux.
- **API diverses** : On peut effectuer diverses opérations sur une base MySQL en utilisant des interfaces écrits en C, Perl, C++, Java, Python, PHP.
- **Connexion et Sécurité** : MySQL dispose d'un système de sécurité permettant de gérer les personnes et les machines pouvant accéder aux différentes bases.
- **Portabilité** : MySQL tourne sur divers systèmes tels qu'Unix, Windows, Linux ou OS/2.
- **Distribution ouverte** : Les sources étant fournies, il est possible d'améliorer MySQL.

2.2.4. PROTÉGÉ 2000

PROTÉGE [3] a été développé par le *Stanford Medical Informatics* de l'Université de *Stanford*. Protégé est une plate-forme Open Source autonome, qui fournit un environnement graphique permettant l'édition, la visualisation et le contrôle (vérification des contraintes) d'ontologies. Le modèle de représentation de connaissances de PROTÉGÉ, est issu du modèle des frames. Ce dernier contient des classes (pour modéliser les concepts), des slots (pour modéliser les attributs des concepts) et des facettes (pour définir les valeurs des propriétés et des contraintes sur ces valeurs), ainsi que des instances des classes. L'interface est très complète ainsi que l'architecture logicielle extensible permettant l'insertion de plusieurs plug-ins offrant de nouvelles fonctionnalités, notamment des plug-ins pour gérer les représentations sous forme graphique, par exemple OWLViz[4] et la prise en charge de nouveaux langages.

Toutes ces caractéristiques ont participé à son succès et le rendent l'éditeur d'ontologie jouissant de la plus grande renommée à l'heure actuelle, servant de référence pour une importante communauté d'utilisateurs (Noy, *et al.*, 2000).

[3] **PROTÉGÉ** est disponible à l'adresse suivante : http://protege.stanford.edu/.
[4] **OWLViz** c'est un Plug-in Il permet de visualiser les hiérarchies de classe dans une ontologie OWL, ce qui permet de comparer la hiérarchie de classe affirmée et la hiérarchie de classes inférée.

2.2.4.1. Le Plugin OntologyBeanGenerator[5] :

Le plugin de générateur de beans d'ontologie est un widget PROTÉGÉ Tab qui génère des fichiers java représentant une ontologie qui peut être utilisée avec l'environnement JADE. Avec l'outil Beangenerator, vous pouvez générer des ontologies FIPA / JADE conformes aux projets RDF (S), XML et PROTÉGÉ.

- **Télécharger**

1) Télécharger la distribution binaire : **beangenerator_bin_Protege_3.2.1.zip**.
   Fonctionne avec PROTÉGÉ3.2.1 et versions ultérieures Compilé avec Jade 3.4.1.
2) Les ontologies abstraites JADE **abstractJadeOnt.zip** sont incluses dans la distribution binaire dans le sous-dossier des projets.
3) Les ontologies du générateur de beans sont incluses dans la distribution binaire dans le sous-dossier Exemples **beangenerator_examples.zip.**

- **Installation**

Décompressez le fichier téléchargé dans le répertoire des plugins PROTÉGÉ.

- **Configuration d'OntologyBeanGenerator**

1) Ouvrez un projet Protégé neuf ou existant.

2) Inclure / importer l'Ontologie abstraite de JADE

- Pour les ontologies de cadres, incluez le Simple JADEAbstractOntology.pprj dans votre projet.
- Pour les ontologies OWL, importez l'OWLSimpleJADEAbstractOntology.owl dans votre projet.
- Les ontologies JADE sont situées dans le répertoire de plugins (Protege_install_dir / plugins / nl.uva.psy.swi.beangenerator / projets))

Nous obtiendrons Maintenant, ce qu'on appelle un **modèle** que nous utiliserons pour créer notre Ontologie à intégrer dans les actes de communication inter-agent.

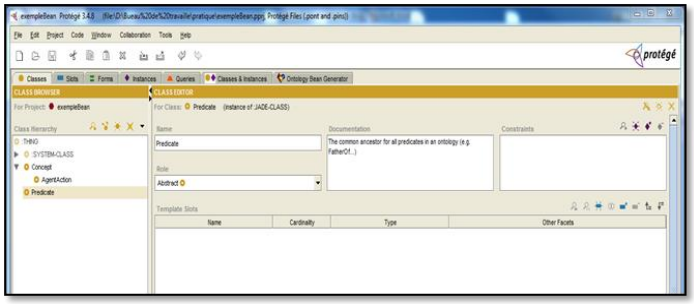


***Figure 3.8** : OWLSimpleJADEAbstractOntology.owl*

[5]**OntologyBeanGenerator** est disponible à l'adresse suivante :
https://protegewiki.stanford.edu/wiki/OntologyBeanGenerator_4.0

3. Présentation du système :

La figure 3.13 présente l'interface principale qui s'afficher au lancement du système. C'est un point de lancement des autres sous-systèmes, et elle contient aussi un panel de statistiques des agents actifs modélisés à travers différents diagrammes (Figures 3.14).

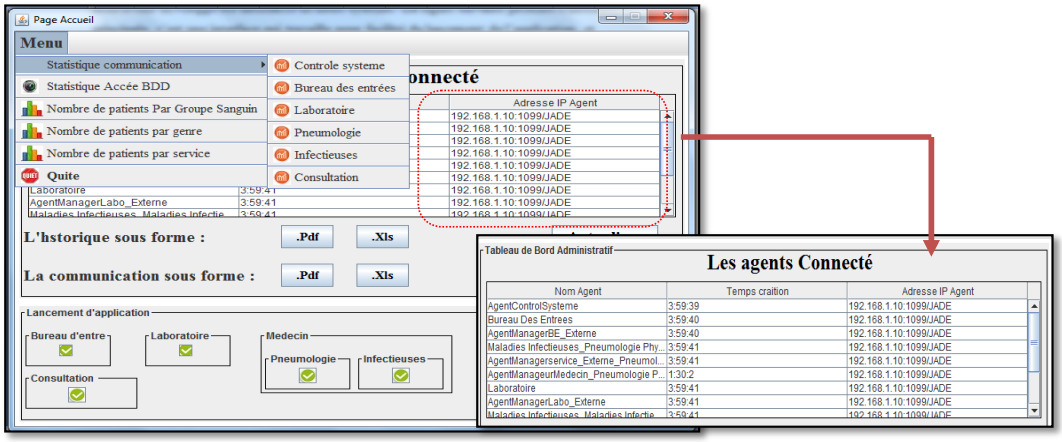


Figure 3.9 *: Main Interface*

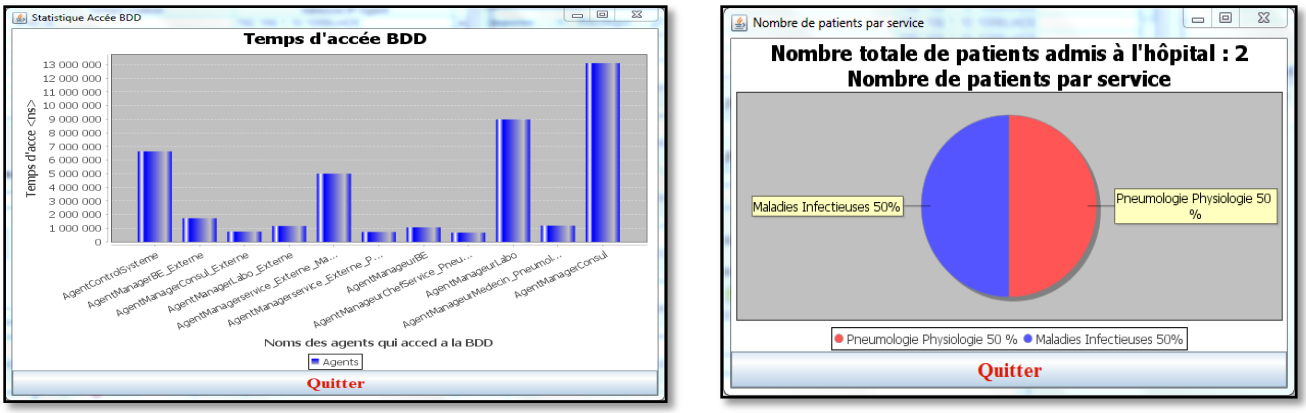


Figure 3.10 *: Interfaces statistiques*

L'accès aux services est sécurisé et il protégé par un *Code Système*. En plus, un *Code médecin* est nécessaire pour identifier le médecin qui à initier la session.

Figure 3.11 : *Page Connexion.*

L'idée est de présenter la suite des interfaces de notre application conformément au scénario présenté dans le diagramme d'activité (Ch. 2, section 3.1.3.3) qui décrit un parcours typique d'un patient. A travers ce scénario, nous allons exposer les fonctionnalités du système et les interactions inter-agents qui en découlent. Ces interactions seront illustrées en utilisons l'outil ***Sniffer_Agent*** qui permet de visualiser les échanges de messages entre agents.

La communication établit entre les agents des différents sous-systèmes suit un protocole bien défini qui permet d'assurer la cohérence des discussions. Dans notre cas, on a choisi le protocole " ***Requet Protocol*** " par ce que, c'est un protocole de *demande-réponse* simple, dans lequel un agent demande à un autre agent d'effectuer une action et les agents d'écoute doivent répondre avec une réponse appropriée (voir figure 3.17).

La réponse appropriée peut être, selon les cas, soit : ***not_understood*** « si l'agent de réception n'est pas en mesure de comprendre la demande », ***refuse*** avec une raison telle que incapable de satisfaire le désir, peut ***accepter*** d'exécuter la tâche. Si un agent accepte la demande, il doit répondre soit en envoyant le résultat direct, soit en envoyant la référence au résultat en utilisant des actes de communication tels que ***Inform*** et ***Inform-ref*** respectivement. Cependant, si l'agent ne parvient pas à exécuter la tâche, il utilise également l'acte de communication ***Failure*** et envoi la raison d'échec.

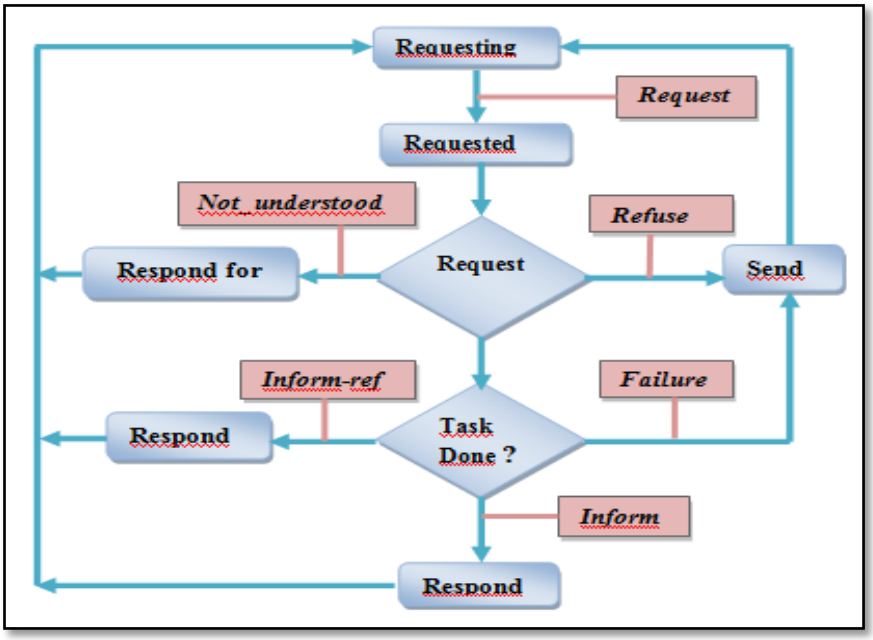
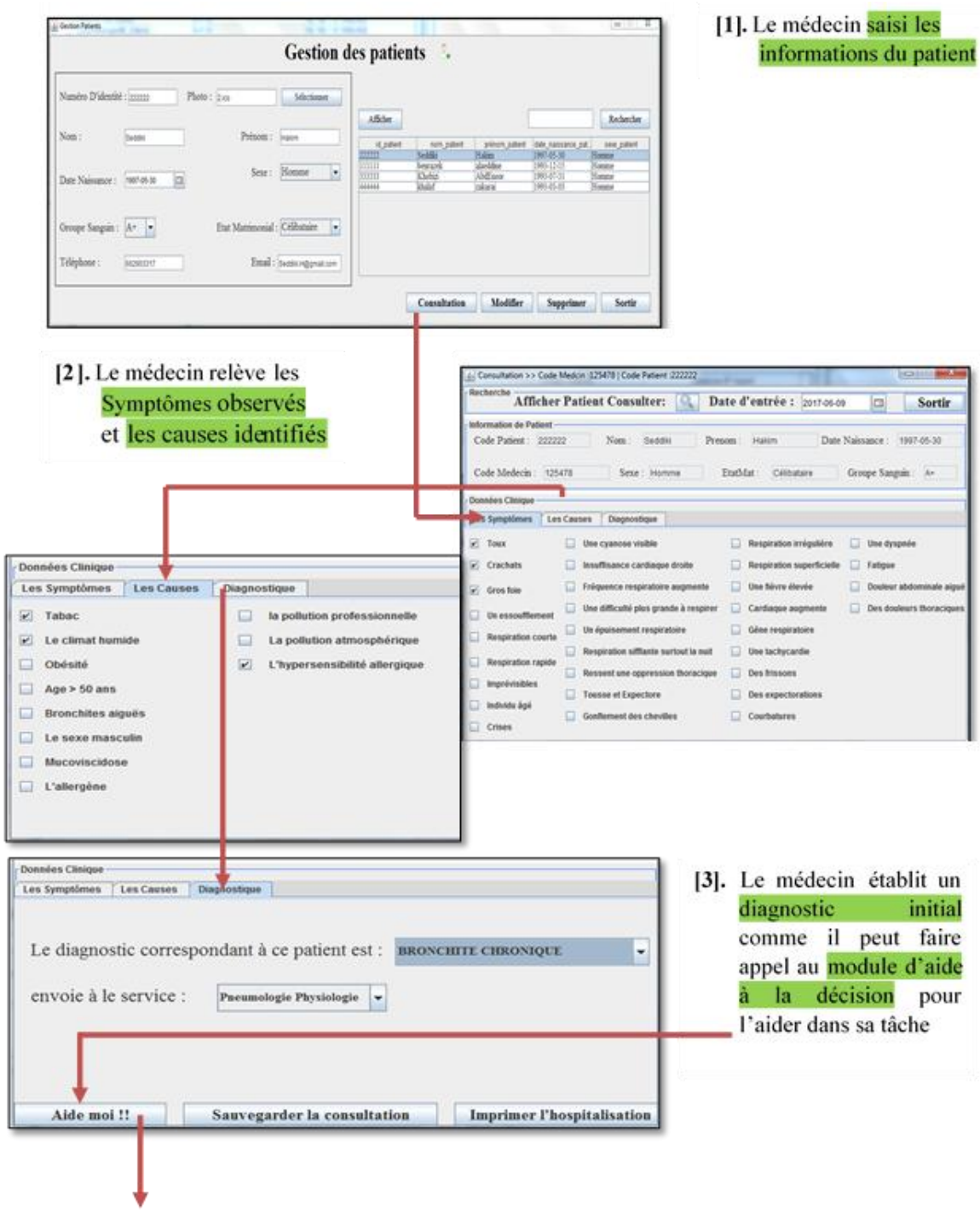


Figure 3.12 *: Protocole Request.*

Dans la section suivante, nous présentons le scénario « ***Parcours du patient*** » depuis son admission à l'hôpital et jusqu'à sa sortie.

Etape 1 : **Au niveau du service « Consultation externe / Urgence »**

Lorsque le patient arrive à l'hôpital, il passera au service « Consultation externe / Urgence » pour être consulter. Le médecin consultant, voit le patient, ajoute ses données dans la BDD temporaire, et fait un diagnostic initial selon les symptômes et les causes observés. Le médecin consultant dispose d'un module d'aide à la décision pour l'aider dans sa tâche de diagnostic. Le système opère sur les profiles similaires des patients ayant déjà étaient rétablit. Si l'état du patient nécessite d'être hospitaliser, le médecin consultant lui donne une lettre d'hospitalisation vers un des services de l'hôpital, sinon une simple ordonnance lui sera remis.



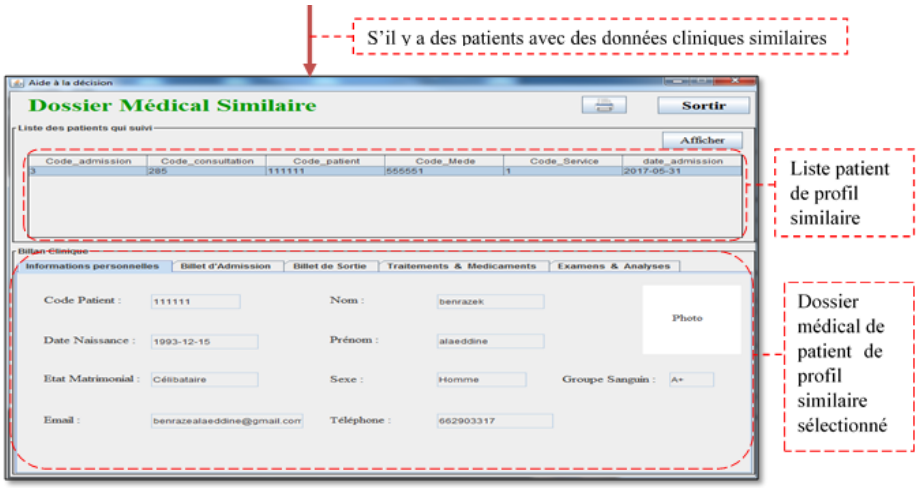


***Figure 3.13** : Scénario de consultation d'un patient.*

Etape 2 : **Au niveau Bureau des entrées**

Ensuite, le patient muni de sa lettre d'hospitalisations dirige vers le bureau des entrées pour lui remettre un billet d'admission.

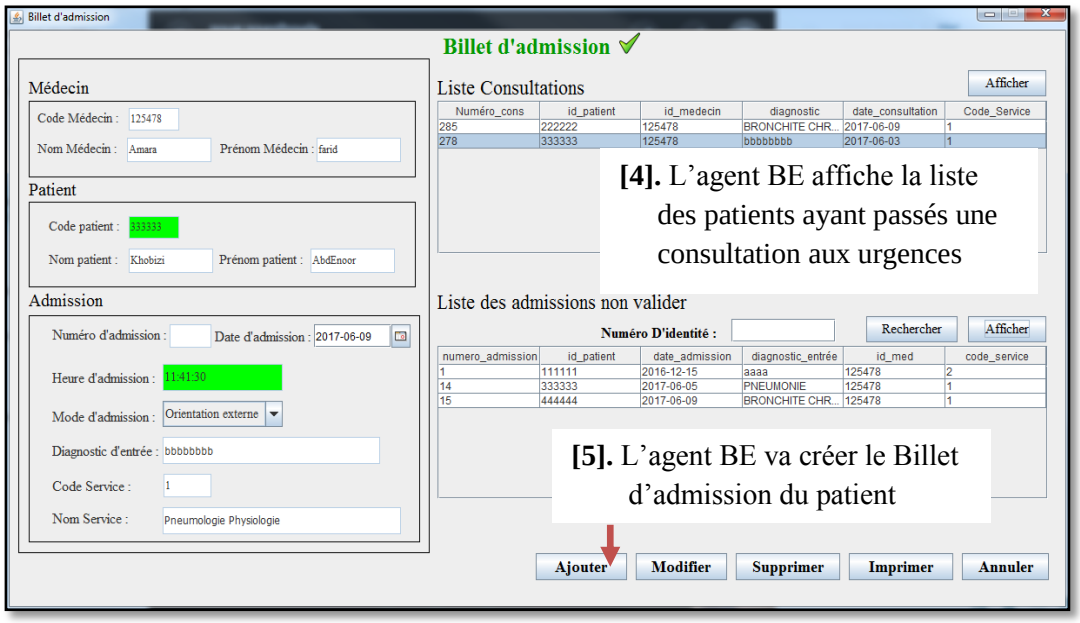
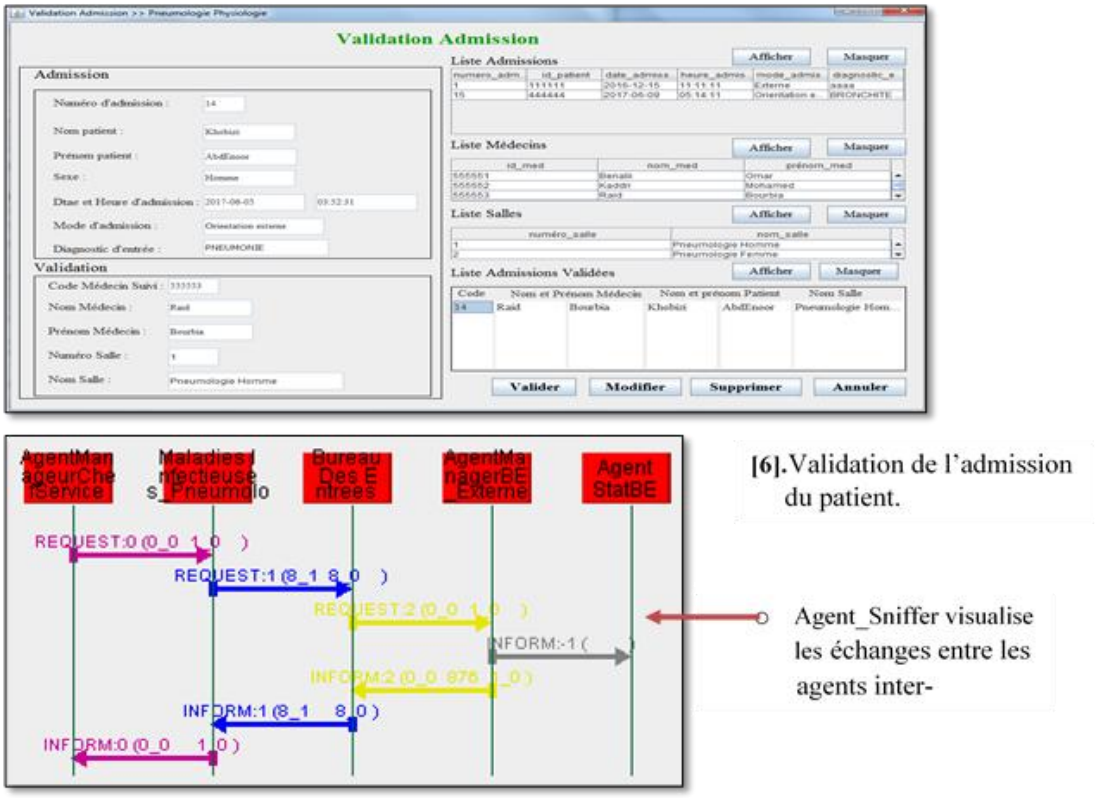


***Figure 3.14** : scénario de création d'admission*

Etape 3 : **Au niveau du Service Hospitalier**

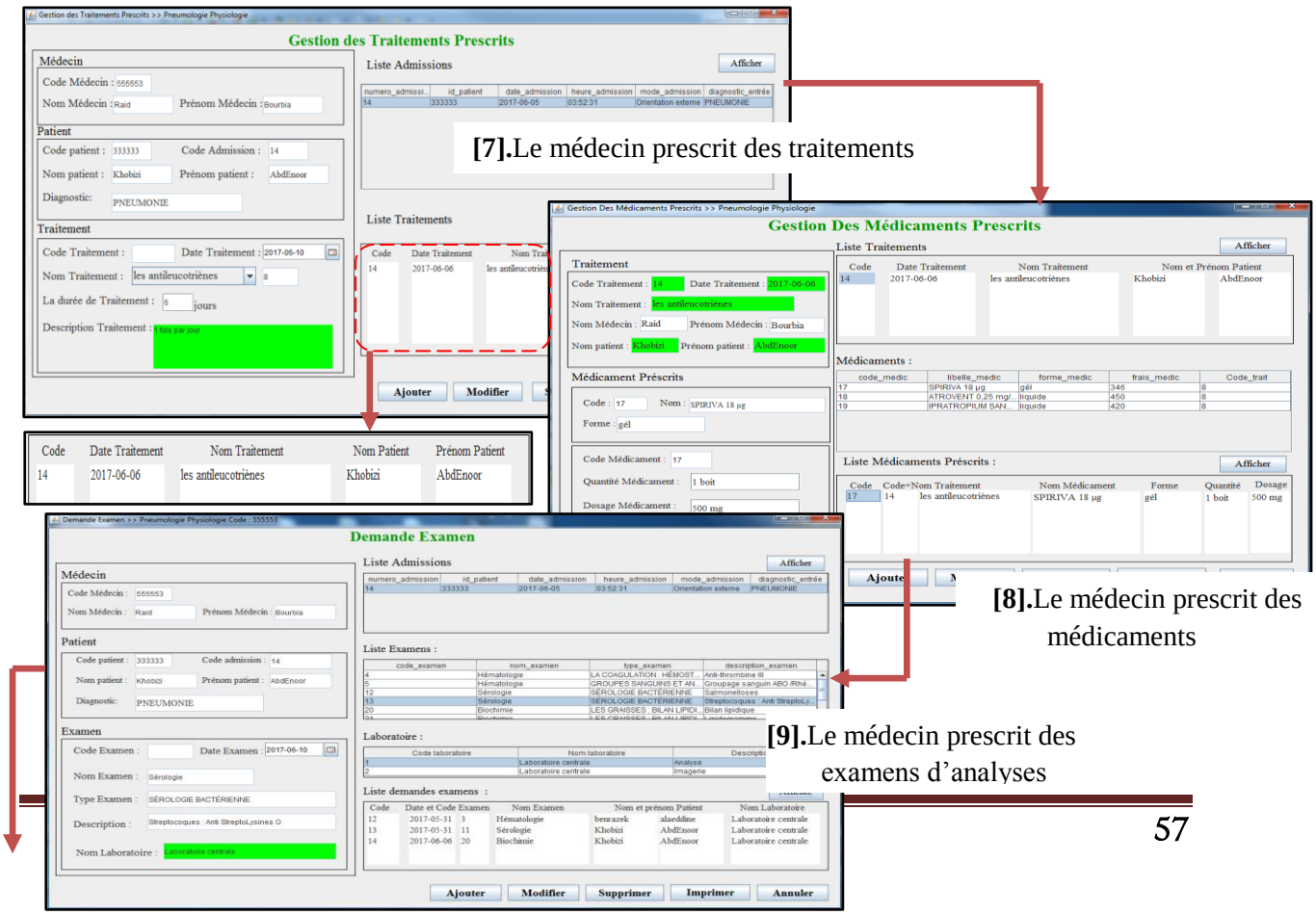
- **Acteur :** Médecin Chef service
- **Rôle :** Le Médecin Chef service valide l'admission du patient, en lui donnant un lit dans une salle du service hospitalier et en lui affectant un médecin spécialiste qui va le prendre en charge.



[6].Validation de l'admission du patient.

***Figure 3.15** : Validation de l'admission.*

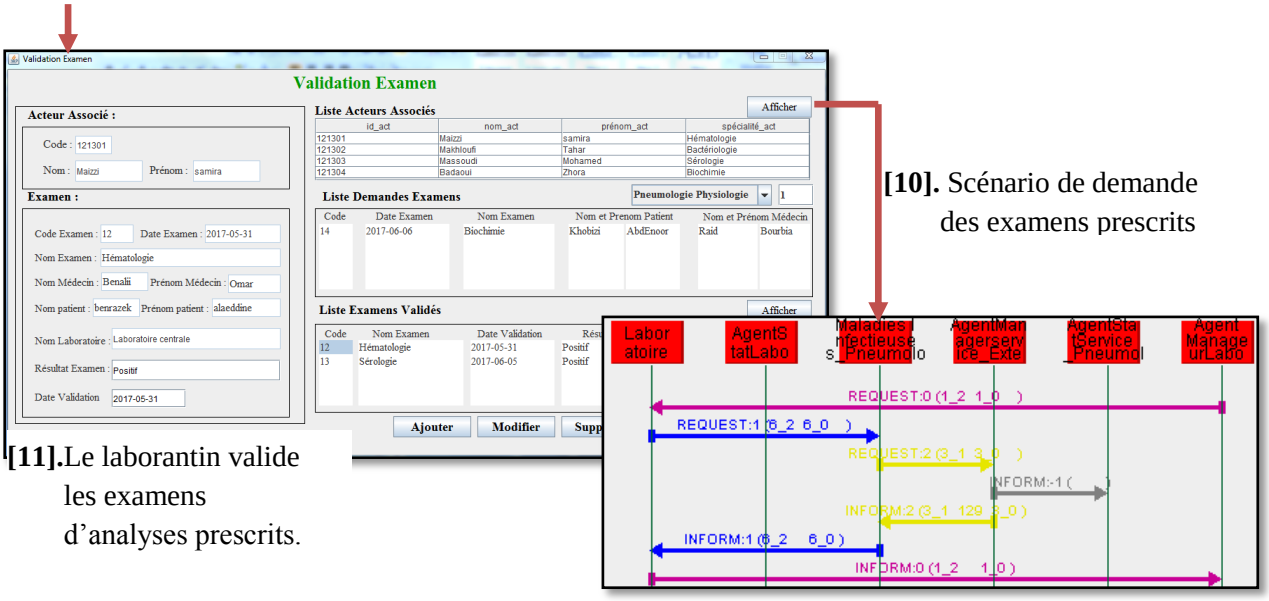
- **Acteur :** Médecin de suivi
- **Rôle :** Le Médecin de suivi prend en charge le patient en lui prescrivant des traitements, des médicaments et des examens d'analyses.



[7].Le médecin prescrit des traitements

[8].Le médecin prescrit des médicaments

[9].Le médecin prescrit des examens d'analyses



[10]. Scénario de demande des examens prescrits

[11]. Le laborantin valide les examens d'analyses prescrits.

Figure 3.16 *: Scénario de la prise en charge du patient*

Le médecin de suivi, peut à tout moment visualiser le dossier du patient. Ce dernier contient toutes les données médicales du patient ainsi que des données antérieures archivées.

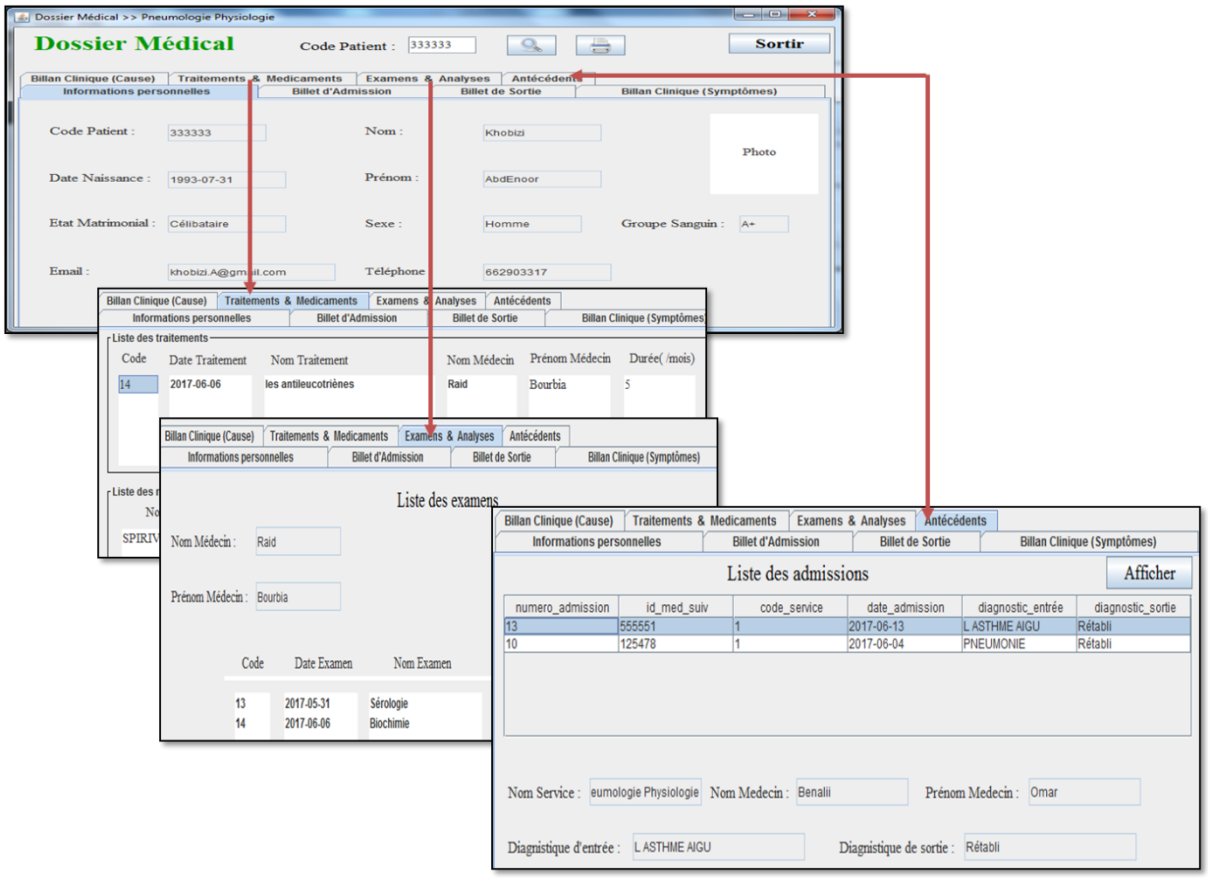
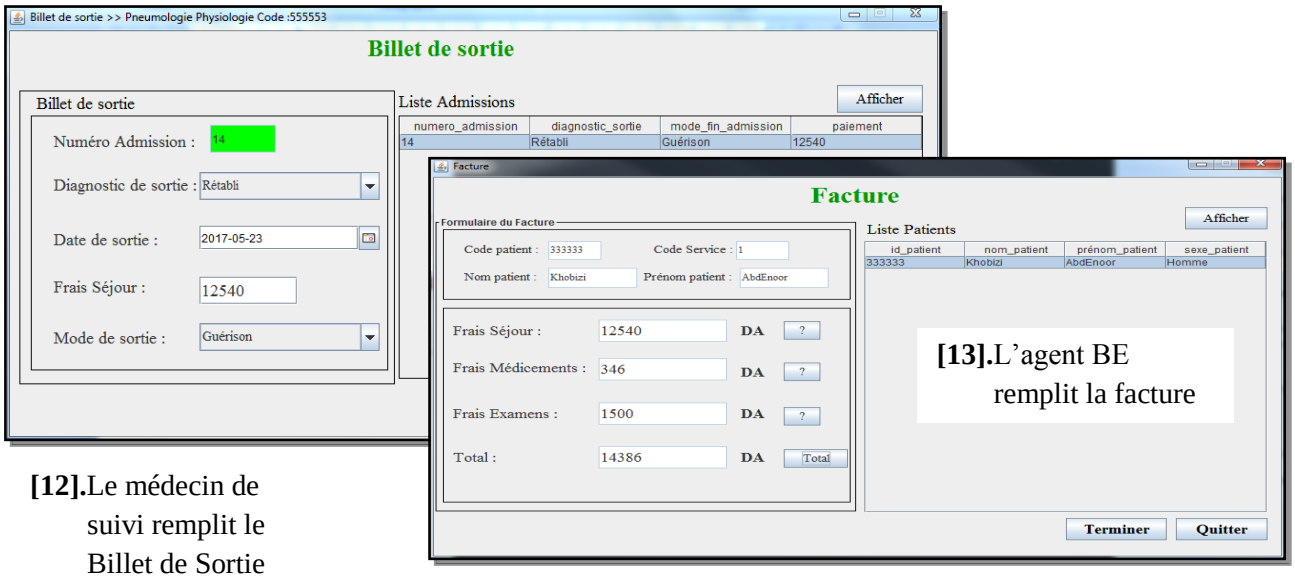


Figure 3.17 : Le dossier médicale

À la fin du traitement, le médecin remplirai le billet de sortie, avec le diagnostic de sortie du patient. Ce billet joue le rôle d'un permis de sortie de l'hôpital, et le patient retourne au bureau des entrées terminer les formalités et payer la facture.



[12]. Le médecin de suivi remplit le Billet de Sortie

***Figure 3.18** : Scénario de sortie du patient.*

4. Construction de l'ontologie

Une ontologie en JADE est une instance de la classe ***jade.content.onto.Ontology*** à laquelle les schémas définissant la structure des types de **prédicat**, **d'actions d'agent** et de **concept** concernant le domaine adressé. Ces schémas sont des instances des classes : ***PredicateSchema, AgentActionSchema*** **et** ***ConceptSchema*** incluses dans le package ***jade.content.schema.***

Comme une ontologie est fondamentalement une collection de schémas qui typiquement n'évoluent pas pendant la durée de vie d'un agent, il est plus pratique de déclarer l'ontologie comme un objet singleton et de définir une classe qui étend le package ***jade.content.onto.Ontology*** avec une méthode d'accès statique à cet objet singleton. Ceci permet de partager le même objet ontologie (et tous les schémas inclus) entre différents agents d'une même JVM.

- **Prédicats** : expressions qui disent quelque chose sur le statut du monde et peuvent être soit vraies soit fausses.
- **Actions des agents** : expressions qui indiquent des actions que les agents peuvent. En règle générale, ils sont utilisés dans les messages de type « demande ».
- **Concepts** : expressions qui représentent des objets avec plusieurs attributs.
- **Autres éléments** : primitifs (éléments atomiques sous forme de nombres ou des chaînes), agrégations (ensembles, listes d'autres termes), les expressions (identifier les entités pour lesquelles est remplie d'un prédicat) des variables.

Dans notre application, nous définissons :

- **Les concepts :**

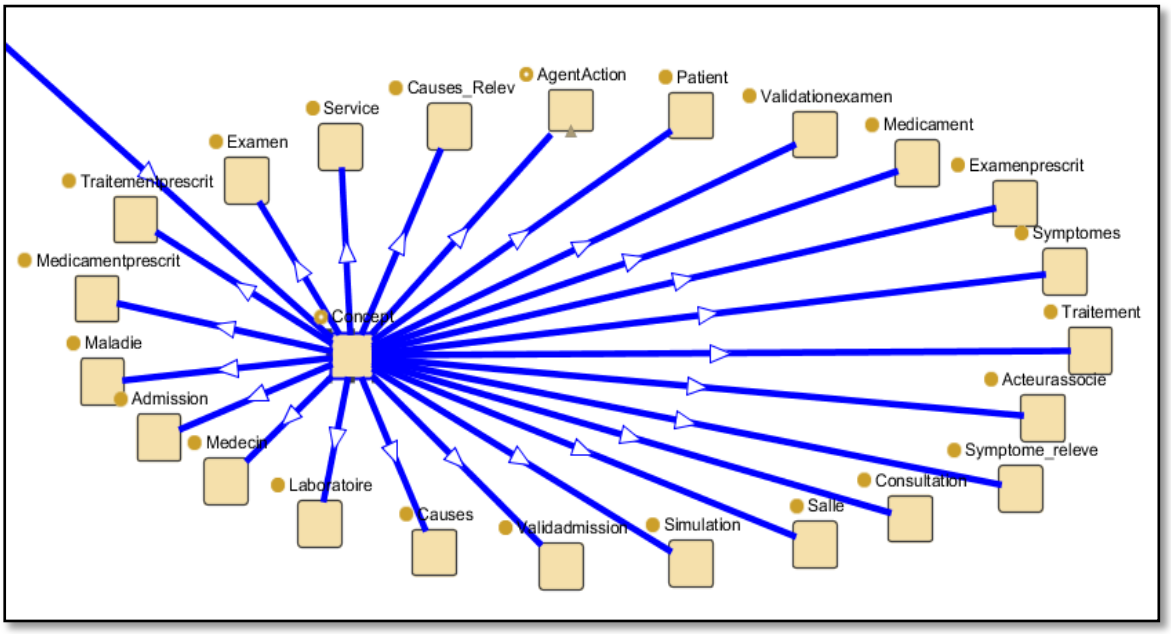


***Figure 3.19** : Les concepts visualisés par **Jambalaya**[6].*

- **Les prédicats :**

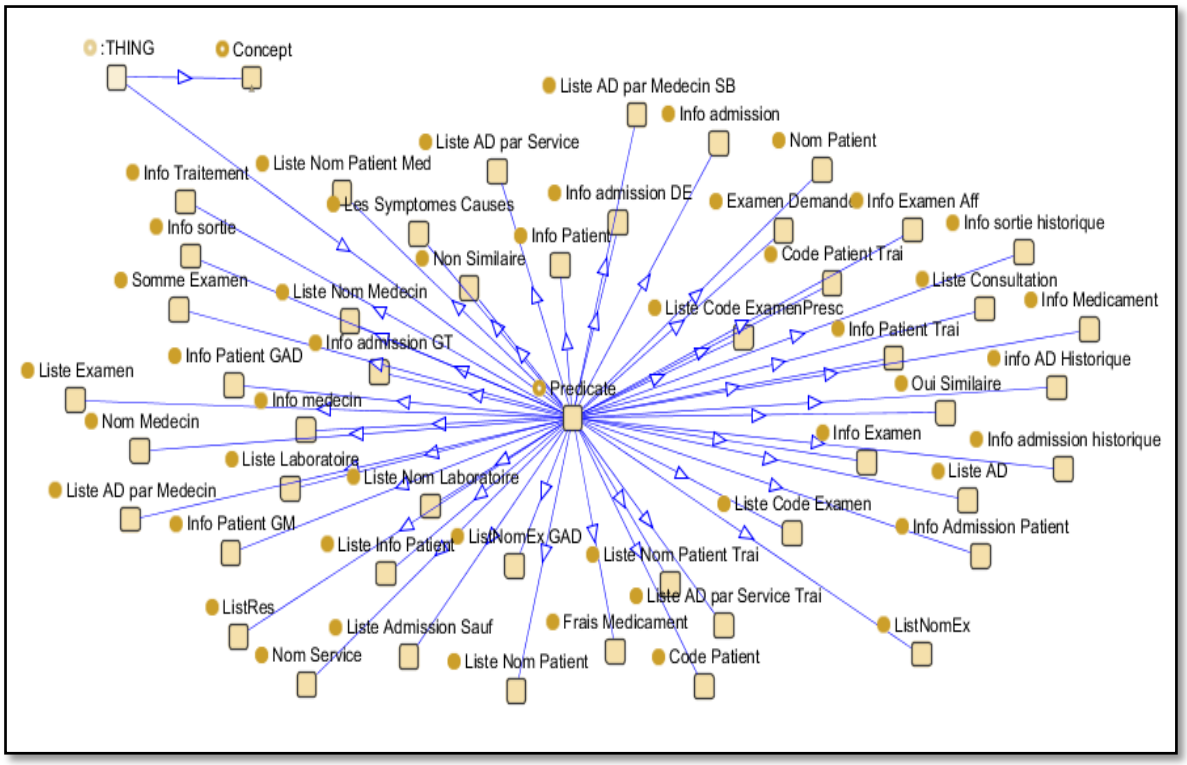


Figure3.20** : Les prédicats visualisés par **Jambalaya.

[6]**Jambalaya** : est un plug-in Protégé qui utilise la crevette pour visualiser les ontologies Protégé-Frames et Protégé-OWL, disponible sur le lien : https://protegewiki.stanford.edu/wiki/Jambalaya.

Les actions Agent :

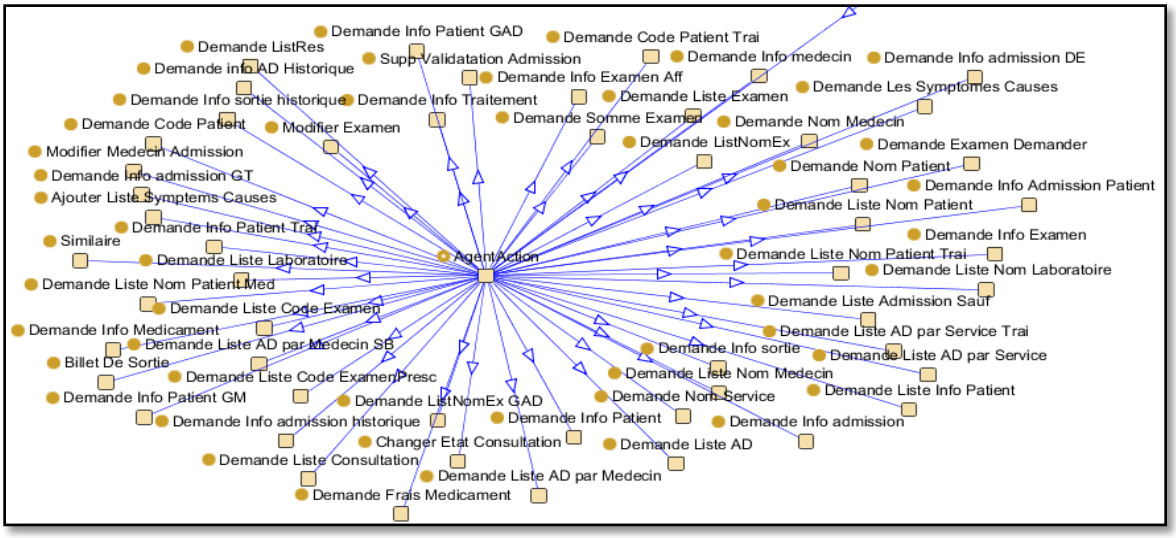


Figure 3.21** : Les actions Agent visualisés par **Jambalaya.

4.1. Développer des classes Java

Après avoir identifié la structure de l'ontologie, en va générer le code java (transformation de l'ontologie vers des objets) de cette ontologie Par le plugin 'OntologyBeanGenerator.

***Figure 3.22** : Génération de code java de l'ontologie par **BeanGenerator**.*

Voilà quelques classes (une classe pour chaque type) :

- **Classe Patient (*Concept*):**

```
Package Ontologie;
Import jade.content.Concept;

Public class Patient implements Concept {

      private String id_patient;
      public void setId_patient(String value) {
            this.id_patient = value; }
      public String getId_patient() { return this.id_patient;     }

      private String nom_patient;
      public void setNom_patient(String value) {this.nom_patient =
value; }
      public String getNom_patient() {    return this.nom_patient; }

      private String mail_patient;
      public void setMail_patient(String value) {
            this.mail_patient = value; }
      public String getMail_patient() {return this.mail_patient;}

      private String prénom_patient;
      public void setPrénom_patient(String value) {
            this.prénom_patient = value; }
      public String getPrénom_patient() {return this.prénom_patient; }
      …… …… ……
```

- **Classe Demande_Info_Patient (*Action Agent*)**

```
Package Ontologie;
Import jade.content.AgentAction;
Publicclass Demande_Info_Patient implements AgentAction {
      /*** Protegename: patient */
      private Patient patient;

      public void setPatient(Patient value) {
            this.patient = value;}
      public Patient getPatient() {return this.patient;}}
```

- **Classe Info_Patient(*Prédicat*)**

```
package onto-test.impl;
import jade.content.Predicate;

public class Info_Patient implements Predicate {
      /*** Protege name: patient */
      private Patient patient;
      public void setPatient(Patient value) {
            this.patient = value;
      }
      public Patient getPatient( ) {  return this.patient; }
```

Classe OntoMedicale (Ontologie Principale)

```
Package Ontologie;
Import jade.content.onto.*;
Import jade.content.schema.*;

Public class OntoMedicale extends Ontology {
	// NAME
	Public static final String ONTOLOGY_NAME = "ontoMedicale";
	// The singleton instance of this ontology
	Private static Ontology theInstance = new OntoMedicale();

	Public static Ontology getInstance() {
		Return theInstance;
	}
	// VOCABULARY
	Public static final String PATIENT_ID_PATIENT = "patient";
	Public static final String PATIENT_ID = "Patient_ID";
	Public static final String GET_PATIENT_PATIENT = "patient";
	Public static final String GET_PATIENT = "Get_Patient";
	Public static final String PATIENT_ID_PATIENT = "id_patient";
	Public static final String PATIENT_NOM_PATIENT = "nom_patient";
	Public static final String PATIENT = "Patient";
	/*** Constructor */
	Private OntoMedicaleOntology() {
		Super (ONTOLOGY_NAME, BasicOntology.getInstance());
		try {
			// adding Concept(s)
			ConceptSchemapatientSchema = new ConceptSchema(PATIENT);
			add(patientSchema, onto - test.Patient.class);
			// adding AgentAction(s)
			AgentActionSchemaget_PatientSchema = new
AgentActionSchema(GET_PATIENT);
			add(get_PatientSchema, onto - test.Get_Patient.class);
			// adding Predicate(s)
			PredicateSchemapatient_IDSchema = new
PredicateSchema(PATIENT_ID);
			add(patient_IDSchema, onto - test.Patient_ID.class);
			// adding fields
			patientSchema.add(PATIENT_NOM_PATIENT, (TermSchema)
getSchema(BasicOntology.STRING), ObjectSchema.OPTIONAL);
			patientSchema.add(PATIENT_ID_PATIENT, (TermSchema)
getSchema(BasicOntology.STRING), ObjectSchema.OPTIONAL);
			get_PatientSchema.add(GET_PATIENT_PATIENT, patientSchema,
ObjectSchema.OPTIONAL);
			patient_IDSchema.add(PATIENT_ID_PATIENT, patientSchema,
ObjectSchema.OPTIONAL);
		} catch (java.lang.Exceptione) {e.printStackTrace();}}}
……..
```

Toutes les classes de XXXSchema sont incluses dans le package ***jade.content.schema***. A partir du code ci-dessus nous pouvons voir que :

- Chaque schéma ajouté à l'ontologie est associé à une classe Java, par exemple le schéma pour le concept Patient est associé à la classe Patient.java. Tout en utilisant l'ontologie définie, les expressions indiquant des patients seront des instances de la classe Patient. Ces classes Java doivent avoir une structure appropriée.
- Chaque champ dans un schéma a un nom et un type, c.-à-d. les valeurs pour cette propriété doivent être conformes au schéma indiqué.
- Un schéma peut avoir un certain nombre de super-schémas. Ceci permet de définir des rapports de spécialisation/généralisation parmi des concepts.

4.2. Choisir le langage de contenu « FIPA-SL »

Le package ***jade.content*** inclut directement des **codecs** pour deux langages de contenu (le langage SL et le langage LEAP). Un codec pour un langage de contenu L est un objet Java capable de contrôler des expressions de contenu écrites en langage L. Dans la majorité des cas, un programmeur peut juste adopter un de ces deux langages de contenu et employer le codec relatif sans aucun effort additionnel. Le langage **SL** est un langage de contenu String-encoded (l'expression du contenu en SL est une chaîne de caractères) lisible pour l'homme et est probablement le langage de contenu le plus diffusé dans la communauté scientifique traitant les agents. FIPA-SL est une langue logique avec une syntaxe préfixée. Il consiste en un premier langage de calcul de prédicat d'ordre.

Un exemple d'information liée au patient codé avec la langue SL est comme suit :

(Patient: id 111111: nom "BENRAZEK": prénom "Alaeddine": date naissance "15-12-1993").

Cet ensemble de concepts et les symboles utilisés pour les exprimer sont connus sous le nom d'une ontologie. Contrairement aux langues de contenu généralement indépendantes du domaine, les ontologies sont généralement spécifiques à un domaine donné.

Toutes les actions d'agent dans le langage **SL** doivent être insérées dans la construction *d'ACTION* (inclus dans *BasicOntology* et implémenté par la classe *jade.content.onto.basic.Action*) qui associe l'action d'agent à *l'AID* de l'agent qui est prévu pour exécuter l'action. De ce fait l'expression :

(Demande_info_Patient (Patient: nom "BENRAZEK") (agent-identifier : AgentCommunicationBE))

Ne peut pas être employée directement comme contenu d'un message de (**demande/request**), même s'il correspond à une action d'agent dans le modèle de référence de contenu. En fait, la grammaire de SL ne le permet pas comme expression de premier niveau. L'expression suivante doit être employée à la place

(ACTION

(agent-identifier:Agent1)

(L'action

(Concept)

(agent-identifier :Agent2)))

((action

(agent-identifier

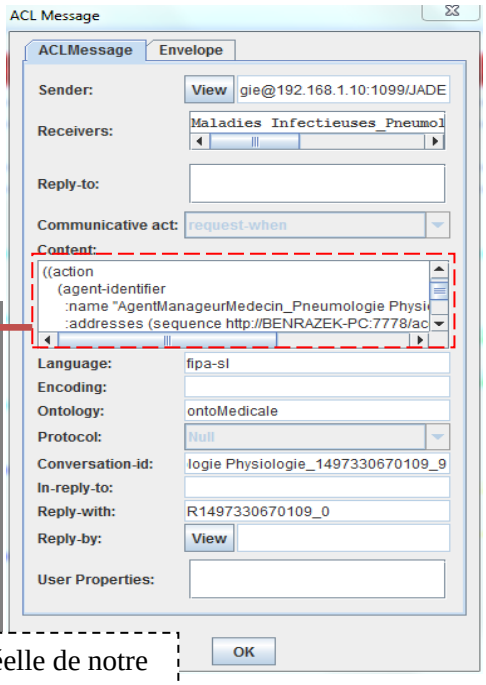
: name "AgentManageurMedecin_Pneumologie Physiologie@192.168.1.10:1099/JADE"

: addresses (sequence http://BENRAZEK-PC:7778/acc))

(Donner_Info_Patient_GM

: patient (Patient

: id_patient "333333"))))



Exemple réelle de notre système

4.3. Enregistrement des langages de contenu et des ontologies

Avant qu'un agent puisse réellement employer l'ontologie définie et le langage de contenu choisi, ils doivent être enregistrés par le gestionnaire de contenu de l'agent. Cette opération est effectuée pendant la création de l'agent (c.-à-d. dans la méthode ***setup ()*** de classe Agent).

Le code suivant montre cet enregistrement dans le cas de l'AgentCommunicationBE en supposant que le langage SL est choisi.

```
Public class AgentCommunicationBureauEntree extends Agent {
	Connection connection = null;
	private Codec codec = new SLCodec(); #
	private Ontology ontology = OntoMedicaleOntology.getInstance();#

	protectedvoid setup() {
		connection = SqlConnection1.dbConnector();
		getContentManager().registerLanguage(codec);#
		getContentManager().registerOntology(ontology);#
		addBehaviour(newCallForferservcie());
		addBehaviour(newCallForferservcieWhen());}
```

La figure suivante illustre le mécanisme de contrôle intégrer par l'ontologie dans les actes de communication. Si le message est conforme au modèle Fipa-Sl, alors on peut avoir comme réponse, soit :

- Un succès : réponse obtenu du coté destinataire et envoyé à l'émetteur.
- Un échec : le destinataire ne trouve pas les éléments de réponse.

Sinon, si le message n'est pas conforme, on aura une erreur généré du côté de l'ontologie.

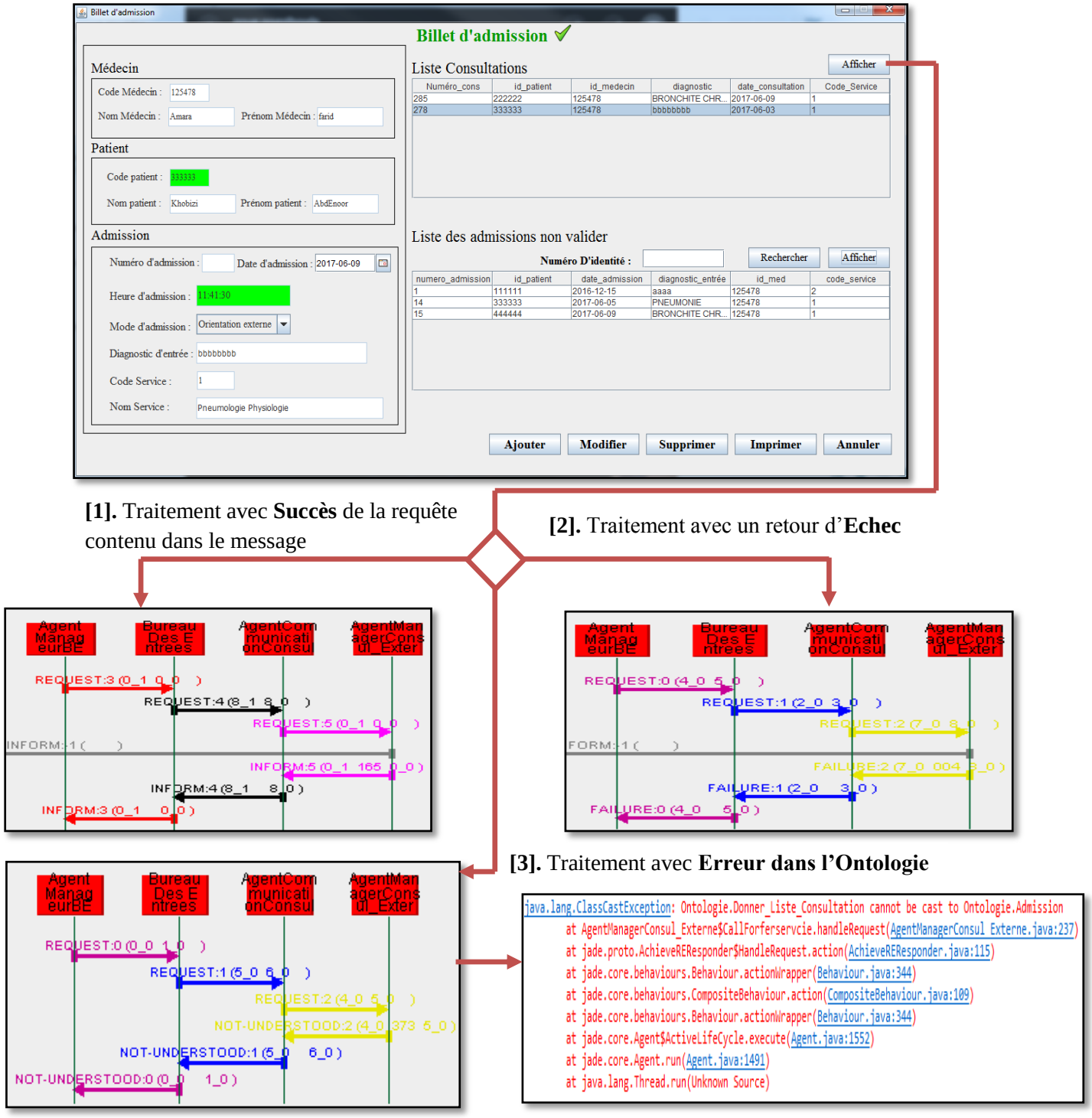


***Figure 3.23** : Validation des messages par l'ontologie*

5. Configuration du réseau

Dans cette partie, nous montrerons comment lancer plusieurs plates-formes JADE et faire communiquer les agents. Pour que cela fonctionne correctement, il est nécessaire que l'AID du destinataire d'un message inclue, à côté du nom du destinataire, au moins une adresse de transport de la plate-forme du récepteur. Notre système est distribué sur deux machines comme suit :

PC	**PC1**	**PC2**
Application	**Consultation** + **Bureau des entrées** + **Contrôle Système**	**Service Médical** (Médecin de suivi + Médecin Chef Service) + **Laboratoire**
Adresse IP	192.168.1.10	192.168.1.8
Base de données	bdd5.sql bdd1.sql, bdd.sql	bdd2.sql bdd4.sql, bdd3.sql

***Tableau 3.2** : Distribution des applications, des adresses et des BDD sur les différentes machines inter connectées.*

La communication inter-plateforme, c'est-à-dire la communication entre les agents résidant sur différentes plates-formes, repose sur des modules appelés ***MTP***[7]. Ce module est capable de transmettre les messages ACL selon les spécifications FIPA. De cette façon, les agents de JADE peuvent communiquer avec des agents vivant sur des plates-formes distantes, qu'il s'agisse d'autres plates-formes JADE ou de plates-formes différentes à condition qu'elles soient conformes à la norme FIPA.

FIPA spécifie comment transférer des messages ACL sur trois protocoles de transport bien connus : HTTP, IIOP (protocole de transport défini dans CORBA) et SMTP. JADE fournit des MTP adéquats pour HTTP et IIOP uniquement.

[7]**MTP** : Message Transport Protocol.

Les modules MTP peuvent être installés dans n'importe quel conteneur et un conteneur peut avoir zéro ou plus de modules MTP installés. Par défaut, JADE installe le MTP HTTP dans un conteneur principal et aucun MTP sur les conteneurs périphériques. MTP écoutant des messages provenant de plates-formes distantes sur le port 7778 a été activé comme indiqué ci-dessous.

```
mai 22, 2017 7:16:19 AM jade.core.Runtime beginContainer
INFOS: ----------------------------------
    This is JADE 4.3.3 - revision 6726 of 2014/12/09 09:33:02
    downloaded in Open Source, under LGPL restrictions,
    at http://jade.tilab.com/
----------------------------------------
mai 22, 2017 7:16:19 AM jade.imtp.leap.LEAPIMTPManager initialize
INFOS: Listening for intra-platform commands on address:
- jicp://192.168.1.10:1099

mai 22, 2017 7:16:20 AM jade.core.BaseService init
INFOS: Service jade.core.management.AgentManagement initialized
mai 22, 2017 7:16:20 AM jade.core.BaseService init
INFOS: Service jade.core.messaging.Messaging initialized
mai 22, 2017 7:16:20 AM jade.core.BaseService init
INFOS: Service jade.core.resource.ResourceManagement initialized
mai 22, 2017 7:16:20 AM jade.core.BaseService init
INFOS: Service jade.core.mobility.AgentMobility initialized
mai 22, 2017 7:16:20 AM jade.core.BaseService init
INFOS: Service jade.core.event.Notification initialized
mai 22, 2017 7:16:20 AM jade.mtp.http.HTTPServer <init>
INFOS: HTTP MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
mai 22, 2017 7:16:20 AM jade.core.messaging.MessagingService boot
INFOS: MTP addresses:
http://BENRAZEK-PC:7778/acc
mai 22, 2017 7:16:20 AM jade.core.AgentContainerImpl joinPlatform
INFOS: ----------------------------------------
Agent container Main-Container@192.168.1.10 is ready.
------------------------------------------------
```

Figure 3.24 *: Activation du protocole MTP sur le MainContainer.*

Les agents en cours d'exécution dans un conteneur sans MTP sont capables de communiquer avec des agents de la plate-forme à distance. De toute façon, JADE adresse les messages de manière directe à des agents étrangers qui hébergent un MTP approprié.

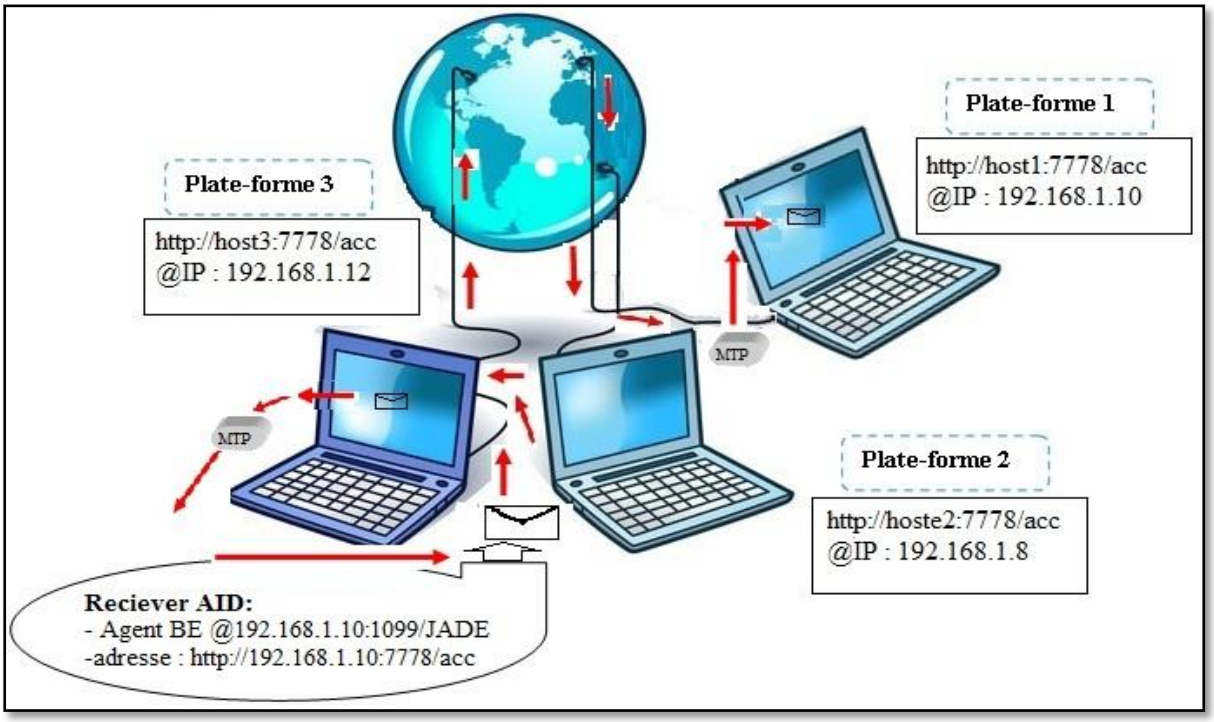


Figure 3.25 *: Message de routage vers/ à partir de modules* ***MTP****.*

Le scénario décrit à la figure 3.24 comprend trois plates-formes. Dans notre cas, on parle d'un scénario entre deux plates-formes qui sont : la plate-forme de bureau d'entrée et la plate-forme de l'agent Laboratoire et on va exprimer le déroulement du scénario entre ces 2 agents.

Lorsque l'agent A (Agent Communication laboratoire) envoie un message à l'agent B (Agent communication BE), il indique à la fois :

- Le nom du récepteur « Agent BE@192.168.1.10:1099/JADE ».
- l'adresse de transport pour livrer le message à la plate-forme B.

Lorsque JADE traite un tel message, il regarde d'abord, le nom du récepteur (Agent EB@192.168.1.10:1099/JADE) et détecte qu'il vit dans une plateforme distante. Ensuite, il obtient l'adresse (http://192.168.1.10:7778/acc). Nous voyons que MTP et http sont nécessaires dans la communication entre les agents de plates-formes distantes.

6. Conclusion

Dans ce chapitre j'ai essayé de mettre en œuvre l'ensemble des idées qui caractérise l'architecture proposée en se concentrant sur l'implémentation de la plate-forme distribué avec JADE et la création d'une ontologie qui j'ai intégrée dans les actes de communication Inter-agents pour que les agents puissent accéder à la sémantique des messages qu'ils s'échangent, j'ai présenté aussi les interfaces de notre système avec quelle simulation de communication entre les agents.

Conclusion générale

CONCLUSION GENERALE

Dans ce projet de fin d'étude, nous avons proposé une architecture basée sur les technologies agent dont le principe est de distribuer la complexité du système sur un ensemble d'entités communicantes, autonomes, réactives et dotées de compétences appelées agents. Elle intègre une ontologie dans les actes de communication inter-agents dont l'objectif est d'assurer la compréhension des messages échangés entre les acteurs intervenant dans la prise en charge du patient. Cette architecture permet de :

- Faciliter l'accessibilité aux informations médicales.
- Faciliter le partage du dossier médical qui constitue un outil de coordination.
- Faciliter la communication entre acteurs pour échanger et partager des informations.
- Obtenir une cohérence des conversations à travers les messages communiqués.

Nous avons utilisé une plate-forme de développement des SMA pour implémenter notre proposition conceptuelle de l'architecture d'un SIM répartie. JADE est la plate-forme choisie pour le déploiement de notre système. C'est une plate-forme qui prend en compte les spécifications FIPA et qui permet d'avoir un retour visuel des communications inter-agents. De plus, JADE permet d'intégrer des ontologies via son modèle de message FIPA-ACL afin de fournir un vocabulaire spécifique dépendant du domaine d'application et employée pour contrôler la communication entre les agents. L'outil PROTEGE est utilisé pour générer l'ontologie. En outre, nous avons utilisé MySQL comme SGBD pour la modélisation des données médicales partagées et distribuées.

Comme perspectives à notre travail, nous envisageons les points suivants :

- Intégrer des modules d'aide à la décision pour le diagnostic initial et au moment de la prescription des traitements et médicaments.
- Rendre le système interopérable avec des SI existants hétérogènes.
- Le maintien de la confidentialité et de la disponibilité des données médicales ;
- l'accès sécurisé aux données par les professionnels de santé.

Bibliographie & Webographie

Bibliographie

1. Aouachria M., 2012. ‘Une approche basée ontologie pour la réutilisation des connaissances dans le processus d’affaires (Business Process)’. Mémoire de Magistère. Université Hadj Lakhdar Batna
2. Azaiez, S. (2007). *Approche dirigée par les modèles pour le développement de systèmes multi-agents.* Thèse de Doctorat, Université de Savoie. Retrieved from : https://tel.archives-ouvertes.fr/tel-00519195.
3. Baziz M., 2005. ‘Indexation conceptuelle guidée par ontologie pour la recherche d’information’. Thèse de doctorat, Université Paul Sabatier, 2005.
4. Belaqziz, S., 2014, ‘‘Une approche d’aide à la décision pour la gestion d’un système d’irrigation gravitaire modélisation multi-agents, télédétection et optimisation par algorithme évolutionnaire’’, Thèse de doctorat. Faculté des Sciences et Techniques de Marrakech, Maroc, 2014.
5. Bellifemine F., Caire G., & Greenwood Dominic (2007). Developing multi-agent systems with JADE.
6. Benhawala, F. (2008). *L'adoption d'une approche organisationnelle pour la conception et la réalisation d'un système multi agents d'acquisition coopérative d'information.* Mémoire de Maitrise Université de la Manouba.
7. Bergenti F., Poggi A, Rimassa G. Et Turci P., 2002. ‘Comma: a multi-agent system for corporate memory management’. AAMAS’02: Proceedings of the first international joint conference on Autonomous agents and multi-agent systems, pages 1039–1040.
8. Bernon C., G. M. P., Picard G. (2009). Méthodes orientées agent et multi-agent. In H.-. Lavoisier (Ed.), *Technologies des systèmes multi-agents et applications industrielles* (Vol. Chapitre 2).
9. Bidault A., Froidevaux C., Gagliardi H., Goasdoué F., Reynaud C, Rousset M-C., Et Safar B., 2002. ‘ Construction de médiateurs pour intégrer des sources d’information multiples et hétérogènes’, le projet picsel. Revue I3.

10. Bouanani, H. (2013). *Suivi et gestion répartie des clients d'une banque Cas de la Banque BADR.* Mémoire de Master, Université Abou BakrBelkaid– Tlemcen.
11. Bourbia R., Dib L. &Benrazek A., 2016." An agent-based architecture for a distributed MIS », 13th ACS/IEEE International Conference on computer Systems andApplications (AICCSA 2016) Agadir, Morocco.
12. Boutemedjet S., 2004. "Web Sémantique et e-Learning ", Cours IFT6251, Université de Montréal,
13. Chaib-Draa B. and F.Dignum, 2002. « Trends in Agent Communication Language », Journal: Computational Intelligence,
14. Diallo, G., 2006. Une architecture à base d'ontologies pour la gestion unifiée des données structurées et non structurées. Thèse : Université Joseph Fourier – Grenoble I.
15. Djilali, C., 2014. 'Une approche basée ontologie pour l'apprentissage sémantique d'un agent'. Mémoire de Magistère présenté l'université Kasdi Merbah Ouargla.
16. Domingue J., Dzbor M. Et Motta E., 2003. 'Handbook on Ontologies', Chapter 27. Semantic Layering with Magpie. Springer, 2003.
17. Drame, k., 2014. 'Contribution à la construction d'ontologies et à la recherche d'information : application au domaine médical', thèse de doctorat présentée l'université de bordeaux,
18. DWC, (1977). Office of Technology Assessment, *Policy Implications of Medical Information Systems*. OTA,. Congress of the United States, Washington, D.C., November 1977. http://www.fas.org/ota/reports/7708.pdf.
19. Ferber J. (1995). *Les systèmes multi-agents : vers une intelligen cecollective*, InterEditions, ISBN : 2-72-96-0572-X.
20. Ferber, J., 1999. Multi-agent Systems: An Introduction to Distributed Artificial Intelligence. Addison Wesley. ISBN: 0-201-36048-9.
21. Fernández-López, M., Gómez-Pérez, A., & Juristo, N. 1997. ''METHONTOLOGY: From Ontological Art Towards Ontological Engineering''. *Spring Symposium on Ontological Engineering of AAAI* , 33–40
22. FIPA, 1999. Specifications part 2: Agent Communication Language, rev. 0.1. URL: http://www.fipa.org/spec/FIPA99.html.

23. Fürst F., 2002.'L'ingénierie ontologique' .Rapport de recherches No 02-07 Octobre 2002, Institut de Recherche en Informatique de Nantes – France.
24. Guarino, N., 1998. 'Formal Ontology and Information Systems'. In Proc. of Formal Ontology and Information Systems, Trento, Italy. IOS Press, 1998. http://www.loa-cnr.it/Papers/FOIS98.pdf
25. Guerroui F.Z., Berkani H.et Bourbia R., 2016. " Un suivi médical intelligent des patients ''. Mémoire de master, département d'informatique. Université Guelma.
26. Gómez-Pérez, A., 1999. 'Ontological Engineering: A state of the Art'. Expert Update. British Computer Society. Autumn, 1999.
27. Gruber,T.R., 1991. 'The role of common ontology in achieving sharable, reusable knowledge bases'. KR'91 : Principles of Knowledge Representation and Reasoning,
28. Gruber, T.R., 1993. 'Toward Principles for the Design of Ontologies Used for Knowledge Sharing', in: International Journal of Human-Computer Studies - Special issue: the role of formal ontology in the information technology, Volume 43 Issue 5-6, PP. 907 - 928.
29. Ketata W. &Lejouad-Chaari W., 2006. »Une Ontologie pour la Réutilisation des Interactions dans un Système Multi-Agents. https://www.researchgate.net/profile/Wided_Lejouad_Chaari/publication/228462410_Une_Ontologie_pour_la_Reutilisation_des_Interactions_dans_un_Systeme_Multi-Agents/links/0046353839466c47c3000000/Une-Ontologie-pour-la-Reutilisation-des-Interactions-dans-un-Systeme-Multi-Agents.pdf
30. Krama I., 2014. ''Conception Et Réalisation D'un Système Intelligent De Recherche D'information Réglementaire'. Mémoire de Master en Informatique, UniversiteKasdiMerbah Ouargla.
31. Lakel, K., 2012. 'Système multi-agents pour la construction d'ontologies', Mémoire de Magistère présentée à l' Universite Des Sciences Et De La Technologie d'oran Mohamed Boudiaf.
32. Maaskri M. & Gouasmi M., (2006). ''Intégration d'ontologie dans les actes de communication inter-agents'', Université Ibn Khaldoun Tiaret. Diplôme d'ingénieur d'état en Informatique.

33. Mazyad, H. (2013). *Une approche Multi-agents à Architecture P2P pour l'apprentissage collaboratif.* Thèse de Doctorat, Université du Littoral Côte d'Opale. Retrieved from https://tel.archives-ouvertes.fr/tel-00845225
34. McGrath R. E, Ranganathan A., Campbell R. H. Et Mickunas D., 2003 ' Use of ontologies in pervasive computing environments'. Rapport technique UIUCDCS-R-2003-2332 UILU-ENG-2003-1719, University of Illinois, Department of Computer Science, Avril 2003.
35. Meshid, S. (2012). *Support de cours Base de Données Réparties (BDR*), Master 2 : Système Informatique Distribué, Département de Physique. Infotronique, Université M'hamedBougara-Boumerdes
36. Noy N. etMcGuinness D. L., 2001, "Ontology Development 101: A Guide to Creating Your First Ontology". Technical Report KSL-01-05 Stanford: Knowledge Systems Laboratory.
37. Roche, C., 2005. 'Terminologie et ontologie'. Langages 157, 48–62.DOI **:** 10.3917/lang.157.0048, Éd**:** Armand Colin. http://www.cairn.info/revue-langages-2005-1-page-48.htm
38. Staab, S., Schnurr, H. P., Studer, R., & Sure, Y. (2001). Knowledge Processes and Ontologies. *IEEE Intelligent Systems, 16* (1), 26–34.
39. Slodzian M., 2000, ' L'émergence d'une terminologie textuelle et le retour du sens'.
40. Studer, R., Benjamins, V.R., Fensel, D., 1998.' Knowledge Engineering: Principles and Methods'. Data & Knowledge Engineering 25 (1998) 161-197.
41. Tahri, S. (2009). *Analyse hydrodynamique d'un piston par simulation de SMA.* Mémoire de Magistère, Université de Hassiba Benbouali – Chlef.
42. Vandecasteele, A. 2012, 'Modélisation ontologique des connaissances expertes pour l'analyse de comportements à risque - Application à la surveillance maritime - ', Thèse de doctorat, Institut des MINES ParisTech.
43. Venot A., B. A., Quantin C. (2013). *Informatique médicale, e-Santé : Fondements et applications*: © Springer-Verlag France.

44. Xukai, Z., Yuan-Shun, D., Bradley, D., Mingrui Q. (2007). *Dependability and Security in Medical Information System*. J. Jacko (Ed.): Human-Computer Interaction, Part IV, HCII 2007, LNCS 4553, pp. 549–558, 2007.

45. Zarour, K. (2012). *L'interopérabilité des systèmes d'information médicaux : une approche basée agent.* Thèse de Doctorat, Université Mentouri de Constantine.

46. Giovanni C, 2009. '*Jade programming for beginners*',(TILAB, formerly CSELT).

47. Gauthier, P, Laurent, V, 2013. '*Initiation à la programmation orientée-objet avec le langage Java*',158 coursFauriel42023 Saint-Étienne Cedex 02

48. Noy, N. F., Fergerson, R. W., &Musen, M. A. 2000. The knowledge model of Protege-2000: Combining interoperability and flexibility. In R. Dieng, & O. Corby (Ed.), *12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00)* (pp. 17–32). (Lecture Notes in Artificial Intelligence LNAI 1937) SpringerVerlag.

49. Dimple,J. Ankit,J. Aarti S ,2015 '*A Review of FIPA Standardized Agent Communication Language and Interaction Protocols'*Journal of Network Communications and Emerging Technologies (JNCET) Volume 5, Special Issue 2, December (2015).

Webographie

Web1 :http://www.maxicours.com/soutien-scolaire/information-et-gestion/1restg/184096.html, date de consultation : 30 Mai 2017.

Web 2:http://docplayer.fr/215300-Regles-de-transformation-du-mcd-au-mld-mrd.html, date consultation : 30Mai 2017.

Web 3: http://djug.developpez.com/java/jade/creation-agent/#LV, date de consultation: 01 Mai 2017.

Web 4 :http://www.enseignement.polytechnique.fr/informatique/profs/ Julien.Cervelle/eclipse/date de consultation 01 Mai 2017.

Web 5 :https://www.jmdoudoux.fr/java/dejae/chap001.htm, date de consultation : 01 Mai 2017.

Web 6 : http://sql.sh/sgbd, date consultation : 01 Mai 2017.

Web7: https://fr.slideshare.net/mohamedyoussfi9/systmes-multi-agents-concepts-et-mise-en-oeuvre-avec-le-middleware-jade**,** date de consultation 03 Mai 2017