République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : SI.

Sujet :

Vers une ontologie à base de contenu et usage pour la recherche d'information dans des sources distribuées

Présenté par : Assia ABDERRAHIM.
Ilhem NOUAS.

Promotrice : M^me^ MELLAH.
Encadreur : M^me^ MELLAH.

Organisme d'accueil : Centre de Recherche sur l'information Scientifique et Technique (CERIST).

Soutenue le: date soutenance, devant le jury composé de :

Président

Examinateur

Examinateur

- 2006/2007 -

Remerciements

Tout d`abord nous remercions Dieu pour nous avoir guidé vers le bon chemin de la lumière et du savoir, pour nous avoir donné du courage et de la volonté afin de pouvoir réaliser ce mémoire.

Nous tenons à remercier nos parents pour leur soutient.

Nous tenons à remercier les membres du jury pour avoir eu l'obligeance d'accepter et d`apprécier notre travail.

Nous tenons à remercier notre promotrice M^{me} H.MELLAH pour son aide, sa compréhensibilité, sa disponibilité.

Notre plus profonde reconnaissance s'adresse à :

Merci Amine, Lilou, Lynda, Mejdi,Nadji, Tarek et enfin Japoni et Boualem

A tous ceux qui ont contribués de prés ou de loin dans la réalisation de ce projet.

Dédicace

A mes parents
A mes sœurs et frère
A mon amie Ilhem et sa famille
A tous mes amis(es)

Assia ABDERRAHIM

Dédicace

Je tiens à dédier ce mémoire à la mémoire de mon père رحمه الله

Ainsi qu'a ma grande mère رحمها الله

Qui nous ont quitté récemment qu'ils reposent en paix inchallah.

« Bien que les fleurs se fanent, meurent et disparaissent, leurs précieux parfums demeurent toujours. Tout comme ces fleurs éclatantes, ceux que nous aimons ne meurent jamais; ils demeurent avec nous à jamais »

A mon père qui m'a toujours appris les vraies valeurs de la vie et qui m'a soutenu et encouragé tout au long de sa vie.

A ma très chère et précieuse mère pour son soutient et sa patience tout au long de mes études et pour sa persévérance de m'avoir encouragé pour finir ce travail malgré les événements.

A mon très cher frère, pour son courage, je te souhaite une grande réussite et comme avait dit papa allah yarhmou « Le jour viendra inchallah », j'espère qu'on volera un jour avec toi.

A tous les membres de ma famille proches et éloignés
A mon binôme et meilleure amie Assia ainsi que toute sa famille

A mes très chers amis

A mon adorable promotion

A toute personne qui me connait.

Ilhem NOUAS

RESUME

Aussi longtemps qu'Internet poursuivra son évolution, nous continuerons à être submergés par des données, sans que celles-ci soient toutefois structurées. La recherche d'information dans ce cadre devient une tâche difficile et les méthodes traditionnelles de recherche sur Internet ou sur des bases de données s'avèrent de plus en plus limitées.

Notre projet s'inscrit dans le cadre général de cette problématique, et consiste à développer une ontologie à base de contenu et d'usage pour la recherche d'information dans des systèmes distribués à base d'une architecture multi-agents.

Les systèmes d'informations coopératifs basés sur les agents logiciels apportent de solutions prometteuses à cet épineux problème, quand aux ontologies, elles apportent une qualité sémantique à la recherche.

ABSTRACT

As long as Internet usage increases, we will continue to be submerged by unstructured data. As a result, search of information on the net or databases on the net, using traditional approaches, will soon prove impractical.

The work presented here is done in this framework and consist of developing an ontology, based on content and usage, for information search in a multiple agents distributed system architecture.

Cooperative information systems based on software agents seek to answer difficult questions in this area whereas ontologies bring the semantic quality required by the search.

MOTS-CLES : Système Multi-Agent, Recherche d'Information, Ontologie, Internet

Table des matières

Chapitre III: Les Systèmes Multi-Agents

Chapitre V : Implémentation & Test

LISTE DES FIGURES :

Chapitre V : Implémentation et Test

LISTE DES TABLEAUX :

GLOSSAIRE

A

- **ACC :** «Agent Communication Channel » gère la communication entre les agents.
- **ACL :** « Agent Communication language », le langage de Communication de la plate-forme JADE.
- **AMS** : « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.
- **API** : «Application Programming Interface» C'est donc une interface de code source fournie par un système informatique ou une bibliothèque logicielle.

B

- **Bibliothéconomie** : est l'ensemble des techniques de gestion et d'organisation des bibliothèques.

C

- **CERN** : « Conseil européen pour la recherche nucléaire » est le plus grand centre de physique des particules du monde.
- **Consortium :** est une collaboration temporaire entre plusieurs acteurs à un projet ou programme dans le but d'obtenir un résultat.

D

- **DAML+OIL** langages prédécesseurs d'OWL (DARPA Agent Markup Language -oil)
- **DF :** « Director Facilitor » fournit un service de « pages jaunes» à la plate-forme.

F

- **FIPA :** Fondation pour les agents physiques intelligents, FIPA est une association internationale d'entreprises et d'organisations destinées à la création de normes dans la conception des agents.
- **Framework:** est un espace de travail modulaire. C'est un ensemble de bibliothèques, d'outils et de conventions permettant le développement

rapide d'applications

H

- **Homonymies :** Qui s'écrit ou se prononce de façon identique, mais diffère par le sens. Dans le premier cas, on parle d'homographe, dans le second, d'homophone.
- **HTML :** « HyperText Mark-up Language ».Langage de programmation des pages Web.
- **HTTP:** (HyperText Transfer Protocol), Méthode utilisée pour transporter des pages HTML de la toile mondiale.
- **hyperlien** ou **lien hypertexte** ou simplement **lien**, est une référence dans un système hypertexte permettant de passer automatiquement d'un document consulté à un document lié.

I

- **IAD** : Intelligence artificielle distribuée.
- **IA :** Intelligence artificielle.
- **IBM: «** International Business Machines Corporation », est une société multinationale américaine présente dans les domaines du matériel informatique, du logiciel et des services informatiques. née le 15juin 1911
- **IDE**: « Integrated Development Environment », environnement de développement logiciel, en informatique.
- **IIS**: Internet Information Services

J

- **JADE : «** Java Agent DEvelopement framework » , plate-forme multi-agent.
- **JAT :** « Java agent template », plates-formes de développement.
- **JDT: «** Java Development Tools ».
- **Jena :** est un framework Java pour le développement des applications Web Semantique .
- **JESS :** « Java Expert System Shell» est un programme permettant la manipulation de systèmes experts d'ordre 1.
- **JVM :** «Java Virtual Machine»
- **JSP :** La technologie JSP (JavaServer Pages) est une extension de la notion de servlet permettant de simplifier la génération de pages web dynamiques.

K

- **KIF:** « Knowledge Interchange Format » est un langage de contenu utilisé dans le cadre des systèmes multi-agents.
- **KQML :** « Knowledge Query and Manipulation Language », Langages de communication inter-agents.

L

- **LISP:** « LIst Processing Language. »
- **Logique applicative :** La logique applicative d'un logiciel correspond au noyau fonctionnel de l'application c'est à dire l'ensemble des traitements réalisés par l'application : procédures de calcul et d'accès aux données.

M

- **Métadonnées :** Une **métadonnée** (du grec *meta* "après" et du latin *data* "informations") est une donnée servant à définir ou décrire une autre donnée quel que soit son support (papier ou électronique).

O

- **OCML :** Operational Conceptual Modelling Language
- **OKBC :** Open Knowledge Base Connectivity
- **OSGi:** « Open Services Gateway initiative »,est un organisme de normalisation fondée en mars 1999.
- **OWL :** « Web Ontology Language» Le langage d'ontologie Web OWL est conçu pour décrire des classes et leurs relations, lesquelles sont inhérentes aux documents et applications Web.

P

- **Page Rank :** Le ***PageRank*** ou *PR* est le système de classement des pages Web utilisé par le moteur de recherche Google pour attribuer l'ordre des liens dans les résultats de recherche. Le mot *PageRank* fait aussi référence à Larry Page cofondateur de Google et inventeur de ce principe. Ce mot est une marque déposée.
- **PDE**: Plug-in Development Environment

R

- **RDF :** « Resource Description Framework », un format informatique permettant de stocker des graphes.
- **RI :** Recherche d'information.

S

- **SBCs :** Systèmes à base de connaissances
- **SMA :** Système multi agents.

- **Serveur Web :** Processus logiciel chargé d'interpréter des requêtes de demande de pages et de renvoyer les pages demandées.
- **Servlet :** Application java exécutée sur un serveur Web. permet de générer dynamiquement une page web.
- **SWT:** Standard Widget Toolkit (est une bibliothèque graphique libre pour Java).

V

- **vie artificielle** : Discipline cousine de l'intelligence artificielle, cherchant à créer des êtres vivants artificiels, en particulier des organismes capables d'évoluer selon des règles qui n'ont pas été rédigées par le concepteur initial. Elle est controversée, surtout parce qu'elle travaille énormément sur les virus, qui sont les formes de vie les plus primitives aussi bien dans la nature que dans les ordinateurs, et qui posent pas mal de problèmes. Abrégé en VA, a-life en anglais.

W

- **W3C :** « World Wide Web Consortium », est un consortium fondé en octobre 1994 pour promouvoir la compatibilité des technologies du World Wide Web telles que HTML, XHTML, XML

X

- **XOL** : XML-Based Ontology Exchange Language
- **XML :** (eXtensible Markup Language) Meta-langage extensible permettant de structurer des données.

INTRODUCTION GENERALE

Préambule

Aujourd'hui, le Web est un outil essentiel pour quiconque tente d'obtenir rapidement de l'information. Le Web représente une source immense de données pour se renseigner, s'informer, rechercher, apprendre et découvrir de nouvelles connaissances. On pense même que c'est le Web qui fut l'élément décisif qui nous fit passer d'une société industrielle à une société d'information. Une société d'information est une société où la création, la distribution et la manipulation d'information est devenue une activité économique et culturelle significative. Le Web permet donc de mieux répondre à notre soif toujours plus grandissante d'information et de connaissances. Mais en apportant une solution, il apporte par le fait même un nouveau problème. Comment pouvons-nous trouver cette information parmi des milliards de documents ?

Problématique

Souvent, les compagnies et les organisations possèdent des centaines ou même des milliers de documents dans leur réseau d'entreprise et ont de la difficulté à trouver facilement l'information dont elles ont besoin.
Pour faciliter la recherche d'information, ces organisations font souvent appel à des techniques déjà disponibles avec les moteurs de recherche conventionnels tels que Google, Altavista, Lycos, et autres.

Les moteurs de recherche (search engine en anglais) font partie des outils d'aide à la recherche d'informations sur le Web au même titre que les annuaires de recherche et les métamoteurs. Il en existe plusieurs dizaines à ce jour: tous aident l'internaute à s'y retrouver dans les milliards de pages qui composent le World Wide Web, sites professionnels et autres pages personnelles qui naissent et disparaissent quotidiennement du réseau mondial.

Mais ces solutions basées uniquement sur la recherche par mots-clés ne permettent de résoudre qu'une partie de l'énorme complexité du problème de la recherche d'information.
Nous remarquons l'inexistence de la notion sémantique des résultats présentés par les moteurs de recherche, ceci peut poser de nombreux problèmes car il est très fréquent qu'une recherche ne nous retourne pas les documents dont nous avions besoin (manque de pertinence).

Dans l'optique de la résolution des problèmes d'absence de sémantique et de pertinence des résultats, nous inclurons à une ontologie de contenu qui traite un contenu informationnel, la notion d'USAGE qui proposera à l'utilisateur les différents usages possible de son information ,de sa recherche , ainsi nous parviendrons à nous rapprocher plus de son but. Et c'est en combinant cette notion d'usage à l'ontologie de contenu médical, que nous raffinerons les résultats de sa recherche.

Pour cela nous allons développer un système de recherche d'information basé sur une ontologie d'USAGE MEDICAL qui servira de « base sémantique » à un système multi agents pour la recherche d'information.

Il sera question, dans un premier temps de développer une ontologie de contenu médicale qui traitera des spécialités médicales auxquelles nous inclurons la notion d'USAGE MEDICAL qui contiendra les usages les plus fréquents lors d'une recherche concernant ces spécialités. Nous avons traité deux spécialités médicales auxquelles nous attribuons les usages possibles ex : traitement, symptôme, etc.

Il sera question, dans un second temps, de penser à une architecture multi agents pour la recherche d'information dans les sites web et l'exploration de l'ontologie ainsi développée afin de traiter sémantiquement les requêtes des utilisateurs.

Approche méthodologique de travail :

Afin d'apporter une contribution, aussi bien dans le cadre de la compréhension de la problématique de recherche d'information, que dans la possibilité de traiter cette question par l'utilisation d'un système multi agents et d'une ontologie, nous avons présenté un état de l'art qui étudie les différentes technologies utilisées à savoir les « ontologies » et les « systèmes multi agents », pour cela :

On s'est penché avant tout sur la recherche d'information et des outils de recherche actuels, leurs avantages et inconvénients.

On a présenté par ailleurs une étude approfondie des ontologies en offrant différentes définitions, en présentant leur importance et leur domaine d'applications .Aussi nous avons recensé les méthodologies de construction d'ontologies les plus représentatives et nous avons choisi la méthodologie la plus appropriée. Cette dernière fera l'objet d'une conception détaillée de notre ontologie intitulé « OntoMed », que nous implémenterons par la suite avec un outil de construction d'ontologie.

Nous étudierons également les systèmes multi agents, en étalant les concepts et techniques de base des systèmes multi agents, les domaines d'applications, les mo-

dèles des systèmes multi agent et autres notions, nous aborderons aussi l'apport des systèmes multi agents dans l'ingénierie ontologique.

Nous concevrons un système de recherche basé sur une ontologie et une architecture multi agents que nous définirons, puis nous arriverons au final à l'implémentation de ce système avec le langage de programmation JAVA.

Présentation du mémoire :

Après avoir présenté une introduction générale exposant la problématique, nos objectifs, notre mémoire est structuré comme suit :

- **Etat de l'art :** traite les chapitres suivants :

Chapitre I : ce chapitre est consacré à la recherche d'information partant de sa définition à ses outils de recherche.

Chapitre II : ce chapitre contient une petite introduction au web sémantique, les technologies mises en jeu, ainsi que l'étude des notions de base des ontologies, les définitions, typologies et autres concepts de bases y sont détaillés.

Chapitre III : celui-ci décrit les notions de bases des systèmes multi agents, les domaines d'application, notion de coopération, plate forme multi agents en passant par la définition des agents, les types d'agents ...etc.

- **Conception, Implémentation et test**

Chapitre IV : Conception

Se divise en 2 parties :

- Partie1 : cette étape consiste en la conception de notre ontologie médicale intitulée « OntoMed » suivant la méthodologie « METHONTOLOGY ».

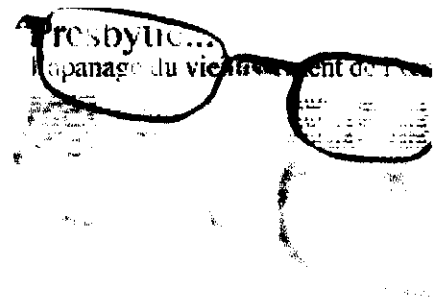
- Partie 2 : décrit la conception de notre système global intitulé M.S.E (Medical Search Engine) incluant l'architecture multi agents que nous avons proposé.

Chapitre V : Implémentation et test

Cette partie présente la réalisation effective de notre système de recherche basé sur une architecture multi agent, et une ontologie médicale que nous développerons selon des outils bien spécifique que nous utiliserons pour l'opérationnalisation du système.

CHAPITRE I

LA RECHERCHE D'INFORMATION



LA RECHERCHE D'INFORMATION

I. INTRODUCTION:

Abrégée en **RI** ou **IR** (*Information Retrieval* en anglais), la recherche d'information est un domaine historiquement lié aux Sciences de l'information et à la bibliothéconomie qui ont toujours eu le souci d'établir des représentations des documents dans le but d'en récupérer des informations, à travers la construction d'index. L'informatique a permis le développement d'outils pour traiter l'information et établir la représentation des documents au moment de leur indexation, ainsi que pour rechercher l'information. On peut aujourd'hui dire que la **recherche d'information** est un champ transdisciplinaire, qui peut être étudié par plusieurs disciplines, approche qui devrait permettre de trouver des solutions pour améliorer son efficacité.

I.1. Définition :

- Action, méthodes et procédures ayant pour objet d'extraire d'un ensemble de documents les informations voulues (d'après l'AFNOR, 1979). Dans un sens plus large, toute opération (ou ensemble d'opérations) ayant pour objet la recherche, la collecte et l'exploitation d'informations en réponse à une question sur un sujet précis.
- « La recherche d'information consiste à mettre en correspondance le besoin d'information d'un utilisateur avec des documents (appartenant à une collection) jugés pertinents » (Van Rijsbergen 86).
- La recherche d'information consiste à rechercher l'information dans des documents qui peuvent avoir différentes formes :
  - les métadonnées qui décrivent les documents.
  - Les bases de données qu'elles soient relationnelles ou mises en réseau par des liens hypertexte, l'internet, et les intranets, pour le texte, le son, les images, les données **[Web10].**

II. TYPOLOGIE DE LA RECHERCHE D'INFORMATION [Web05]:

Il existe plusieurs manières de rechercher l'information :

II.1. La recherche par navigation :

La recherche par navigation arborescente:

La recherche d'information correspond dans ce cas à une démarche qui part du

général pour aboutir à l'information la plus spécifique, et qui se présente comme un chemin à travers les divisions ou rubriques, puis les subdivisions ou sous-rubriques, par une succession de menus ou de dossiers.
C'est le cas d'une recherche dans un plan de classement ou dans les annuaires thématiques sur Internet.

La recherche par navigation hypertextuelle

Ce type de recherche correspond à une navigation dans un réseau de nœuds et de liens créés par des associations entre des mots et entre des documents : on accède à l'information recherchée en suivant les liens existants.
C'est le cas des sites web, mais aussi des hypertextes sur CD-ROM.

II.2. La recherche par interrogation :

C'est une recherche par mots clés à partir d'une requête formulée en langage naturel, ou une recherche guidée à l'aide d'une interface. Elle porte sur des **métadonnées** concernant un document (titre, auteur…), ou directement sur le **texte intégral** qui le constitue.

La recherche par interrogation utilise des operateurs, qui sont divisés en catégories : troncature, operateur booléens, Opérateurs numériques, Opérateurs de proximité (*cf. ANNEXE C*).

III. LES OUTILS DE LA RECHERCHE D'INFORMATION SUR LE WEB :

Dès son apparition, le web a vu se développer de nombreux outils de recherche destinés à faciliter le repérage et le signalement des documents publiés. Mais, à partir du milieu des années 90, l'ouverture d'internet à toute la société a conduit à un développement prodigieux de ce service. Une masse considérable d'information hétérogène est depuis disponible. Parallèlement sont apparus des outils de recherche nombreux et variés prétendant tous être le meilleur et fournir le plus rapidement possible l'information la plus pertinente.

Ils se divisent en deux types, de par leur fonctionnement et leur utilisation (Lardy01) :

III.1. Les outils humains : ces outils se limitent principalement aux annuaires (ou répertoires thématiques).

Ils reposent sur une procédure humaine d'identification, de validation et d'intégration des ressources internet : Des personnes, membres de l'annuaire (comme pour Yahoo, http://fr.yahoo.com) ou internautes participant à un annuaire (comme pour l'Open Directory, http://dmoz.org), spécialisées par domaine, sont chargées de ces

différentes tâches (même si une assistance par des outils automatisés est fréquente).

III.2. Les outils automatisés : ce sont les outils qui reposent sur une procédure automatisée pour la sélection, le traitement et la récupération des ressources internet: ils utilisent des logiciels ou "robots" pour effectuer ces tâches. C'est le cas des **moteurs de recherche** et des **méta-moteurs**.

III.2.1. Les annuaires de recherche

Autre appellations : répertoires / directories, annuaires, catalogues thématiques, listes thématiques, subject trees, portails...

Un annuaire se présente comme un répertoire séparé en catégories et voir en sous catégories et contient un certains nombres de liens (des adresses web en l'occurrence). Un **annuaire** est en général gérer par un humain.**[Web01]**

La première page de ces annuaires regroupe les principales catégories. La navigation s'opère hiérarchiquement du général au particulier. L'annuaire vous propose une sélection de sites dans la catégorie correspondant à ce que vous recherchez, exemple d'annuaire Lycos France, pour plus d'exemple d'annuaire (*cf. ANNEXE C)* .

Sachant que les répertoires sont souvent mis à jour suite à des soumissions, les moteurs de recherche sont amenés à visiter régulièrement les différentes pages que composent les annuaires. Le moteurs dans ce cas, mettent en mémoire les différentes pages visitées et visitent également les différents liens trouvés sur cette même page. Les sites présents dans le répertoire sont donc visités à leurs tours et indexés plus rapidement par les moteurs de recherche.

III.2.1.1. Avantage et inconvénient des annuaires

Avantage [Web09]

- La sélection des sites étant faite par des spécialistes, cela valide certaines ressources et permet d'éviter le bruit (l'information non pertinente)
- Les sites étant classés en catégories hiérarchisées, cela facilite la navigation.
- La plupart de ces annuaires permettent aussi une interrogation par mot-clé

Inconvénient [Web02]

- Mise à jour moins rapide que les robots : délai fréquent de 2 mois, avant l'inscription d'un nouveau site dans la base
- Couverture géographique et linguistique des ressources plus aléatoire (intervention humaine)
- Sélection de sites et non de pages web
- Accès direct par équation booléenne plus limité que pour les robots (intégralité des documents pas indexée, requêtes souvent limitées, pas d'équations de recherche complexes)
- Subjectivité dans les choix et les classements des ressources

Ainsi, un annuaire rend possible l'obtention rapide d'une sélection de sites par rapport à un thème précis, avec un court résumé indicatif de chaque URL proposée.

III.2.2. Les moteurs de recherche

Les moteurs de recherche sont des bases de données constituées automatiquement grâce aux logiciels robots, qui scrutent à intervalles réguliers les serveurs déclarés sur Internet. Ils indexent mot à mot les documents localisés, permettant ainsi des interrogations par mots-clés **[Web01]**

Parmi les moteurs de recherche les plus connus : Google, Altavista. (*cf. ANNEXE C*)

III.2.2.1. Avantage et inconvenant des moteurs de recherche

Avantage [Web03]

- **Beaucoup d'information**
  Les moteurs de recherches donnent accès à une masse considérable d'informations.
- **Informations précises**
  Ils permettent d'accéder rapidement à des informations précises.
- **Pertinence des résultats**
  Ils classent, en général, par degré de pertinence les réponses obtenues, ce qui, malgré des résultats souvent étonnants, permet de ne consulter que les premières pages des résultats de recherche.
- **Recherches complexes possibles**
  Ils permettent, pour la plupart, de faire des recherches assez complexes en utilisant la logique booléenne.

Inconvénient [Web03] [Web04]

- **Contrôle des informations plus ou moins adéquat**
  Les bases sont souvent si importantes que le contrôle des informations est plus ou moins adéquat. C'est pourquoi l'on retrouve beaucoup d'adresses URL périmées dans les résultats de la recherche.

- **Interrogation complexe**
  Les interfaces et les techniques de recherche varient d'un moteur à un autre: ce qui peut sembler rendre l'interrogation complexe à l'internaute débutant. Cependant malgré ces différences, la logique de recherche demeure la même.
- **Résultats parfois décevants**
  Les résultats de la recherche peuvent sembler décevants ou surprenants puisque l'indexation est automatique et que le repérage se base, de façon générale, sur les titres des pages, leur contenu et les métadonnées qui y sont associées.

- **Couverture du web**
  Aucun moteur de recherche ne couvre la totalité du web (plus gros indexes : environ 3 milliards de documents).
- **Un robot ne possède pas l'intelligence d'un humain en indexant des mots clés sans en comprendre le sens.**
  On tombe parfois sur des pages qui contiennent bien le mot-clé tapé mais sans rapport avec l'objet de notre recherche
- **Incapacité de distinguer les homonymies (la plupart des moteurs)**
  Par exemple pour le mot « orange » ils nous donnent : l'orange (le fruit), orange (la couleur) et Orange (l'opérateur de Téléphonie).

- **Ils indexent non pas des mots mais des "chaines de caractères"** ce qui fait qu'ils indexent **aussi des mots avec des fautes d'orthographes**
- **La pertinence des résultats dépend de la manière dont le robot est conçu et de la forme de la question**

III.2.3. Les méta-moteurs

Les méta-moteurs permettent d'interroger en parallèle plusieurs outils de recherche. Les plus récents suppriment les doublons et reclassent les résultats selon leur propre méthode. Utiles pour obtenir rapidement un panorama (vision) général des documents disponibles correspondant à un mot-clé, ils ne permettent pas d'effectuer des recherches complexes. Ils sont soit consultables directement en ligne, soit disponibles sous forme de logiciel à télécharger, avec des fonctionnalités plus performantes **[Web01].**

Parmi les méta-moteurs nous citons : Kartoo, Copernic (*cf. ANNEXE C*)

III.2.3.1.Avantage et inconvenant des méta-moteurs [Web03]

Avantages

- **Recherche rapide**
  La recherche est plus rapide que si on faisait la même recherche dans plusieurs moteurs différents séparément.
- **Recherche exhaustive**
  Les méta-moteurs peuvent être utiles dans le cas de recherches simples lorsqu'on veut s'assurer de faire la recherche la plus exhaustive possible. Aucun moteur de recherche ne couvrant plus de 60% du Web.

Inconvénients

- **Temps de réponse plus long**
  Le temps de réponse est plus long puisque la requête est transmise à plusieurs moteurs et que les résultats sont ensuite traités.
- **Moins de possibilités**
  Les subtilités propres à chacun des moteurs utilisés séparément ne peuvent être exploitées à leur maximum (par exemple la recherche avec des opérateurs de proximité).
- **Redondance**
  Si les doublets (homonyme) ne sont pas éliminés, il y a beaucoup de redondance

- **Difficulté d'interpréter les requêtes des utilisateurs**
  Les syntaxes d'interrogation diffèrent d'un moteur à l'autre (pour les requêtes complexes, il est conseillé d'utiliser directement un moteur de recherche)

IV. LES METHODES DE CLASSEMENT DES RESULTATS DE RECHERCHE

Les moteurs de recherche ont développé des méthodes de tri automatique des résultats.

Dans la pratique, aucune méthode de tri n'est parfaite mais cette variété offre à l'utilisateur la possibilité de traquer l'information de différentes manières ; elle augmente donc ses chances d'améliorer ses recherches.

On peut considérer trois grandes méthodes de tri :

IV.1. Tri par pertinence [Web01]

Les résultats d'une requête sont affichés selon un ordre déterminé par le calcul d'un score pour chaque réponse. La pertinence est basée sur les 5 facteurs suivants, appliqués aux termes de la question :

- Le **poids d'un mot dans un document** est déterminé par sa place dans le document : il est maximum pour le titre et le début du texte ; à l'intérieur du texte, il est plus important si le mot est en majuscule.
- La **densité** est basée sur la fréquence d'occurrence du terme dans un document, par rapport à la taille du document. Si deux documents contiennent le même nombre d'occurrences, le document le plus petit sera donc favorisé.
- Le **poids d'un mot dans la base** est basé sur la fréquence d'occurrence du terme dans toute la base de données. Les mots peu fréquents dans le corpus sont favorisés. Les mots vides sont soit éliminés, soit sous-évalués.
- La **correspondance d'expression** est basée sur la similarité entre l'expression de la question et l'expression correspondante dans un document. Un document contenant une expression identique à celle de la question reçoit le poids le plus élevé.
- La **relation de proximité** est basée sur la proximité des termes de la question entre eux dans le document. Les termes proches sont favorisés.

Cette technique est utilisé par Altavista, Voila, AlltheWeb.

IV.2. Tri par popularité : [Web07]

Les limites du tri par pertinence ont conduit à rechercher, à partir de principes tout à fait différents, d'autres méthodes indépendantes du contenu des documents. Connues sous le nom de tri par popularité, ces méthodes différentes sont au nombre de deux :

IV.2.1. La méthode basée sur la Co-citation : [Web07]

Lancé en 1998 par deux étudiants de l'Université de Stanford, Google classe les pages grâce à la combinaison de plusieurs facteurs dont le principal porte le nom de **Page Rank**.

Page Rank utilise le nombre de liens pointant sur les pages. Plusieurs moteurs de recherche ont intégré depuis cette fonctionnalité, ainsi le tri est indépendant du contenu.

Il faut cependant noter que la technique de Google a aussi des aspects négatifs : ainsi les pages récentes et donc inconnues sont défavorisées tout comme celles au contenu très pointu qui n'intéresse qu'un public restreint.

IV.2.2. La méthode basée sur la mesure d'audience [Web07]:

La société DirectHit a été fondée en avril 1998 et propose de trier les pages en fonction du nombre de visites qu'elles reçoivent.

Le fonctionnement, en règle générale, est en tâche de fond sur un moteur existant. À chaque consultation d'un internaute, il va noter sur quel lien celui-ci a cliqué et quel était le rang de ce lien. Il calcule ensuite combien de temps l'utilisateur met avant de revenir sur la page de résultats. S'il ne revient pas, il en « déduit » que le site proposé était pertinent. Son adresse sera alors mieux classée dans les résultats suivants, lors d'une interrogation sur le même mot clé. Les interrogations, la façon d'interroger et de naviguer des internautes, vont alors enrichir la base de données de DirectHit. Cette méthode, comme la précédente, pénalise les pages récentes.

IV.3. Tri par calcul dynamique de catégories [Web01]

Cette méthode effectue un classement des documents trouvés dans des dossiers (clustering) constitués automatiquement en fonction des réponses. Un dossier peut lui-même être constitué de sous-dossiers.
Vivisimo Clustered search results http://www.vivisimo.com en est un exemple.

En conclusion nous dirons qu'aucune de ces méthodes n'est idéale, le contenu très hétérogène des pages ne facilite pas les choses.

La tendance actuelle est de mixer différentes approches pour ne pas être trop dépendant d'une seule méthode.

V. EVALUATION DE LA RECHERCHE :

Pour caractériser la qualité d'un moteur de recherche, on utilise souvent quatre critères : la précision (opposée au bruit) et le rappel (opposé au silence).

V.1. Le bruit (noise) [Web06] :

En documentation, c'est **l'ensemble des documents non-pertinents affichés suite à une recherche documentaire**.

On a coutume de dire qu'en documentation, le bruit vaut mieux que le silence, puisque l'utilisateur peut ensuite trier parmi les réponses obtenues celles qui

l'intéressent.
Cependant, le bruit peut être un facteur de découragement pour les utilisateurs occasionnels, et de perte de temps pour tous.

Ce phénomène est dû par exemple aux ambigüités du langage, un même mot pouvant avoir des sens différents.

V.2. Le silence [Web06] :

Documentairement parlant, c'est **l'ensemble des documents pertinents non affichés lors d'une recherche documentaire**.
C'est le plus grave des problèmes car il est indétectable pour celui qui interroge.

Il peut être **causé par une indexation automatique, une indexation trop pointue, des fautes de frappe ou l'emploi de synonymes** (*par exemple, programme, logiciel, application informatique employés alternativement dans différentes références*) **ou de plusieurs mots de la même famille**.

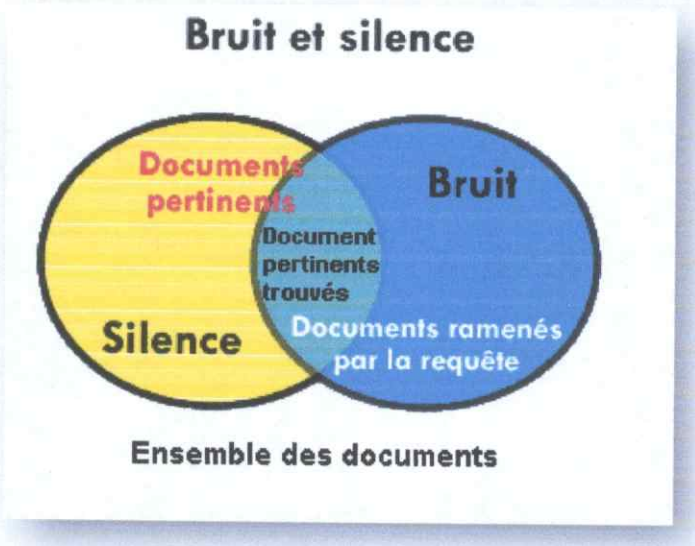


Figure I.1 : une représentation graphique du bruit et du silence :

L'évaluation de la qualité de la recherche documentaire s'évalue par ces quatre critères :

$$precision = \frac{le\ nombre\ de\ documents\ pertinents\ trouvés}{\text{Le nombre total de documents trouvés}}$$

SILENCE = 1-RAPPEL

BRUIT = 1-PRECISION

Pour un moteur d'indexation et de recherche idéal, RAPPEL=PRECISION= 1.

VI. LA RECHERCHE D'INFORMATION BASEE SUR LES ONTOLOGIES

La Recherche d'Information (RI) est une activité par laquelle un utilisateur accède à un granule d'information (un ensemble de documents, un document, une partie de document, un composant XML, une donnée) à partir d'un besoin qu'il spécifie.

Les systèmes de RI (SRI) développés depuis le début des années 50 reposent essentiellement sur des approches statistiques et des approches linguistiques de bas niveau. Ces approches prennent uniquement en compte le niveau lexical, parfois le niveau syntaxique, du contenu textuel des granules afin d'identifier les mots permettant de retrouver les granules répondant aux besoins de l'utilisateur. Un enjeu actuel de la RI, comme du Web avec le Web Sémantique, est de s'appuyer sur des connaissances pour enrichir les systèmes en apportant une couche sémantique **[20]**.

Cette dernière peut être mise en œuvre via l'utilisation des ontologies dont la structure permet de stocker à la fois des éléments (des termes, des entités, ou des concepts) et les relations entre ces éléments. Les ontologies sont des outils extrêmement puissants en sémantique, car les informations qu'elles contiennent facilitent toute une série de tâches réputées difficiles en sémantique appliquée.

L'utilisation des ontologies peut surpasser les limitations des modèles de recherche classique en donnant de meilleur résultats du fait qu'il recherche des concepts qui sont représentatif du document , contrairement aux autres méthodes d'indexation et de recherche , qui se contentent de la présence ou pas d'un mot dans le document et de la fréquence d'apparition de ce mot dans le document.
Ainsi, le résultat d'une recherche donnée par les moteurs de recherche basé sur une ontologie est plus adapté aux besoins de l'utilisateur, du fait qu'il retrouve le concept recherché et non le terme recherché **[20]**.

VII. RECHERCHERCHE D'INFORMATION BASEE SUR LES AGENT INTELLIGENTS :

Faisant suite aux méta-moteurs, une nouvelle génération d'outils est apparue. Ce sont des logiciels ou des programmes qui vous permettent de rechercher toute sorte de documents (fichiers, information...) sur différents supports (votre disque dur, réseau, internet...) **[Web05]**.

Ces outils se veulent intelligents, c'est à dire qu'ils s'efforcent d'offrir des fonctions évoluées (lancement automatique et périodique de la recherche, récupération de données) par rapport aux moteurs de recherche, notamment dans le classement des réponses **[Web05].**

Ces outils sont nommés Agent intelligents, ces derniers sont flexible et modulaires, et sont actuellement la seule solution aux problèmes liés à l'architecture centralisée des systèmes classique de la recherche d'information.

L'architecture multi agent des systèmes de recherche d'information est élaborée sur trois types d'agents :

- Agents fournisseurs qui offrent des services aux utilisateurs et à d'autres agents.
- Agents demandeurs qui demandent un service précis aux agents fournisseur, utilisent les informations fournies et interagissent avec les agents intermédiaires.
- Agents intermédiaires qui lient les agents demandeurs et les agents fournisseurs.

D'autres agents peuvent participer, c'est selon le fonctionnement de l'application.

Conclusion :

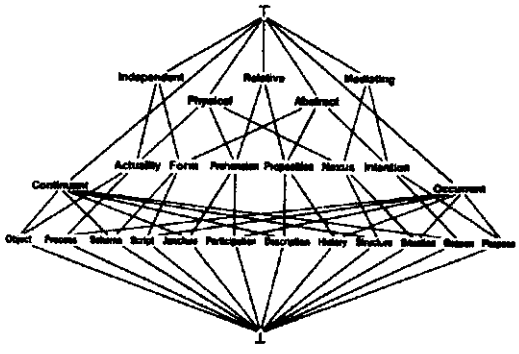
Nous avons présenté dans ce chapitre les principaux notions et concepts de la recherche d'information(RI). Nous citons entre autre l'apport des ontologies et les systèmes multi agents dans celle ci.

Les systèmes de recherche d'information (SRI) sont conçus à l'origine pour retrouver les documents traitant d'un sujet donné. Or, la majorité des systèmes actuels, se contentent de chercher les documents qui contiennent les mêmes mots que ceux de la requête. Ceci est évidemment insuffisant dans la mesure où l'utilisateur du système ne connaît pas à priori le vocabulaire que l'auteur a utilisé dans le document.

Des approches basées sur les ontologies, sont présentées par des chercheurs comme solution au problème de la représentation de l'information en RI. Dans le chapitre suivant, nous dressons un état de l'art sur la notion d'ontologies pour la recherche d'information en effleurant le web sémantique.

CHAPITRE II

LES ONTOLOGIES POUR LA RI



LES ONTOLOGIES POUR LA RI

1ER PARTIE : WEB SEMANTIQUE

I. Introduction au Web Sémantique

Au mois d'août 1991, un employé du CERN (Conseil européen pour la recherche nucléaire), Tim Berners-Lee, publie un message sur le forum de discussion « comp.sys.next.announce » annonçant la création d'une application appelée le WorldWideWeb. Cet outil allait littéralement initier une révolution dans le domaine de l'accès à l'information.

Aujourd'hui, le Web est basé principalement sur des documents écrits en langage HTML. Le HTML est très utile quand vient le temps de mettre en page un bloc de texte avec du contenu multimédia comme des images par exemple. Cependant, ce langage est centré sur la présentation du texte et non pas sur son contenu.

En 2002, Berners-Lee, soumet un nouveau concept à la communauté scientifique : « Le Web sémantique ». Le Web sémantique vise à pallier le manque de sémantique des documents du Web traditionnel. Selon (Berners-Lee et al., 2001) : « Le Web sémantique est une extension du Web tel qu'on le connaît dans lequel on donne à l'information un sens bien défini, permettant ainsi aux ordinateurs et aux gens de mieux travailler en coopération ». **[1]**

L'expression Web sémantique, fait d'abord référence à la vision du Web de demain comme un vaste espace d'échange de ressources entre êtres humains et machines permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de services variés.

Espace virtuel, il devrait voir, à la différence de celui que nous connaissons aujourd'hui, les utilisateurs déchargés d'une bonne partie de leurs tâches de recherche, de construction et de combinaison des résultats, grâce aux capacités accrues des machines à accéder aux *contenus* des ressources et à effectuer des *raisonnements* sur ceux-ci. **[2]**

II. En quoi consiste le Web sémantique ?

Le Web sémantique reste entièrement fondé sur le Web "classique" et ne le remet pas en cause, cela reste un moyen de publier et consulter des documents, mais les documents traités par le Web sémantique contiennent non pas des textes en langage naturel (français, espagnol, chinois, etc.) mais des informations formalisées pouvant être traitées automatiquement par des agents logiciels. Ceci est réalisé par l'annotation du contenu du Web, par la description de propriétés et de relations. Cette annotation est réalisée avec un langage raisonnablement expressif, possédant une sémantique bien définie et permettant le partage ainsi que la réutilisation des

données au travers de différentes applications. Le consortium W3C a définit un cadre général pour ce type de descriptions basé sur RDF (Ressource Description Framework) pour la représentation de connaissance et OWL (Ontology Web Language) pour la modélisation sémantique de connaissances. **[9]**

Le Web sémantique est composé principalement des couches suivantes : XML+ des schémas XML (la couche syntaxe), de RDF+ des schémas RDF (la couche donnée) du langage OWL (couche ontologie), La couche logique, La couche de preuve.
Les recherches, pour le moment, se concentrent principalement sur les trois premières couches du Web sémantique.

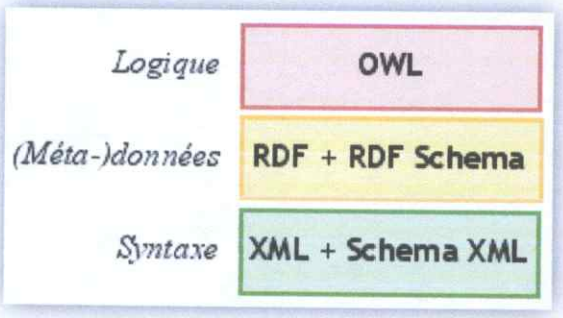


Figure II.1 : Couche du web sémantique

III. Ontologies pour le Web sémantique [10]

Le Web actuel est essentiellement syntaxique, la structure des ressources étant bien définie, mais leur contenu restant inaccessible aux traitements machines, seuls les humains étant capables de l'interpréter.

Il serait par conséquent possible avec le Web sémantique d'instaurer des services automatiques supportant l'utilisation du sens de l'information contenue dans les pages Web. Cette vision peut donc être perçue comme étant une autre solution au problème de l'immense quantité d'information disponible dans le Web. Pour que cette idée puisse se concrétiser, les différents outils Web auront besoin de schémas décrivant divers domaines de la connaissance pour ainsi être capable par exemple de lier deux concepts dans deux pages Web différentes. Une **ontologie** vise à répondre à ce problème.
De ce fait en associant aux ressources du Web des entités ontologiques comme références sémantiques, permettra aux différents agents logiciels d'accéder et d'exploiter directement le contenu des ressources et de raisonner dessus (Berners-Lee et *al.* 2001).

2EME PARTIE: DE LA NOTION DE THESAURUS A CELLE D'ONTOLOGIE

I. THESAURUS [3]

Un thésaurus est une sorte de dictionnaire hiérarchisé. Les termes sont reliés par des relations sémantiques. Il est organisé alphabétiquement, et propose généralement les définitions pour les termes.
Les termes d'un thésaurus servent à représenter ou à annoter des documents.
En plus des relations de spécialisation (relation verticale) présentes dans une taxonomie, un thésaurus élargit le contexte d'un terme en ajoutant d'autres relations : terme interdit (synonyme), terme préfère (relations horizontales).

II. TAXINOMIE [3]

Le mot taxinomie vient du grec taxis, rangement, et de nomos, loi. Le Petit Robert définit la taxinomie comme étant :

(1) l'étude théorique des bases, lois, règles, principes d'une classification

(2) une classification d'éléments. Le terme taxonomie fut inventé, sous cette orthographe, par Augustin Pyrame de Candolle pour définir la théorie des classifications. L'orthographe fut corrigée en taxinomie par Émile Littré mais l'autre forme reste pourtant très répandue.
Dans notre cas une taxonomie désigne une liste de termes définis généralement par une communauté, Il s'agit d'une hiérarchie de termes, organisée généralement avec la relation de spécialisation / généralisation. D'autres relations sont utilisées comme la composition mais dans une taxonomie, un seul type de relation est représenté.

Exemples de taxonomies
Relation de division

- Monde
  - Afrique
  - Europe
    - France
    - Italie

Cet exemple montre une décomposition hiérarchique ou division du monde en continents et chaque continent en pays.

III. ONTOLOGIE [3]

III.1. Définitions :

Le terme « ontologie », construit à partir des racines grecques ontos (ce qui existe, l'existant) et logos (le discours, l'étude), est un mot que l'informatique a emprunté à la philosophie au début des années 90. En d'autres termes, l'Ontologie serait la *Science ou théorie de l'être*. Bien que ce soient les Grecs qui aient inventé cette

science, ils ne l'avaient pas appelé Ontologie, le terme étant beaucoup plus récent (XVIIe siècle) que la discipline qu'il désigne (EncyclopædiaUniversalis00). La discipline elle-même a évolué en se rapprochant des sciences cognitives et de l'IA, il y a seulement une vingtaine d'années.

Dans les écrits scientifiques contemporains, le terme ontologie recouvre deux usages dont le premier appartient à la philosophie classique et le second plus récent, aux autres sciences cognitives. De ce fait, la convention veut que la notation *Ontologie* (avec un *O majuscule*) soit attribuée au domaine issu de la philosophie et *ontologie* aux autres.

L'ontologie permet de fixer une sémantique aux objets primitifs de la représentation d'un domaine.

« *An ontology is an explicit specification of a conceptualization.* » *Gruber (1993)*

Reprenant les travaux de T.R. Gruber (1993) et ceux de M. Uschold et M. Grüninger (1996), J. Charlet propose une définition rigoureuse et affinée de ce qu'est une ontologie (Charlet, 2002).

« Une ontologie implique ou comprend une certaine vue du monde par rapport à un domaine donné. Cette vue est souvent conçue comme un ensemble de concepts – e.g. entités, attributs, processus –, leurs définitions et leurs interrelations. On appelle cela une conceptualisation. »
[. . .]

«**Conceptualisation** » réfère à une abstraction d'un phénomène du monde obtenue en identifiant les concepts appropriés à ce phénomène.
L'ontologie possède les caractéristiques suivantes :
1) partagée, 2) explicite, et 3) formelle

« **Partagé** » signifie que l'ontologie capture la connaissance consensuelle.
« **Formel** » signifie que l'ontologie est interprétable par une machine (machine-readable)
« **Spécification explicite** » signifie que les concepts de l'ontologie et les contraintes liées à leur usage sont définis de façon déclarative.

III.2. À quoi sert une ontologie ? [Web12]

- ***Communication entre humains*** : Un besoin existe de partager la signification de termes dans un domaine donné

- Toute activité humaine spécialisée développe son propre jargon (langue de spécialité) sous la forme d'une terminologie et d'une conceptualisation associée spécifiques.
- L'existence de tels jargons entraîne des problèmes de compréhension et des difficultés à partager des connaissances entre les acteurs de l'entreprise, les services d'une entreprise, les entreprises d'une industrie, qui font des métiers différents.
- Fondamentalement, le rôle des ontologies est d'améliorer la communication entre humains, mais aussi entre humains et ordinateurs et finalement entre ordinateurs.

- ***Communication entre humains et organisations*** : Vers un vocabulaire standardisé

  - L'existence de vocabulaires différents au sein d'une entreprise (ex : bureau d'études, bureau des méthodes) ou d'une industrie (ex : constructeur automobile, équipementier) constitue un frein à la collaboration et aux partenariats. Les enjeux touchent donc directement la compétitivité de l'entreprise.

  - Pour l'entreprise, l'ontologie sert à :
    - améliorer la *compréhension* entre les employés,
    - favoriser la *diffusion* des informations et leur *exploitation*,
    - promouvoir une *nouvelle approche de conception* des systèmes d'information (réutilisation de codes, interopérabilité des logiciels).

- ***Conception et utilisation des systèmes d'information*** : Des apports pour l'ingénierie des systèmes d'information (Guarino, 98)

  - *Spécification* ; *Acquisition des connaissances* : une ontologie peut aider à l'analyse des besoins et à définir les spécifications d'un SI.
  - *Réutilisation* ; *Partage* : une ontologie peut être, ou peut devenir suite à une traduction, un composant réutilisable et/ou partagé par plusieurs logiciels.
  - *Fiabilité* ; *Maintenance* : une ontologie peut servir à améliorer la documentation d'un logiciel et/ou à automatiser des vérifications de cohérence (SBCs), réduisant les coûts de maintenance.
  - *Interopérabilité* : en jouant le rôle d'un format d'échange, l'ontologie permet à des systèmes d'information, basés sur des paradigmes de modélisation et des langages d'implantation différents, de coopérer.

- ***Vers une meilleure exploitation de sources d'information :***

  - *Recherche* : une ontologie peut jouer le rôle de méta-data servant d'index dans un répertoire d'information.

- *Intégration* : dans une application "entrepôt de données", une ontologie peut jouer le rôle d'un schéma conceptuel commun reliant entre elles plusieurs sources d'information hétérogènes.
- *Interface Homme-Machine* : la visualisation de l'ontologie permet à l'utilisateur de comprendre le vocabulaire utilisé par le SI et de mieux formuler ses requêtes.
- *Requêtes* : une ontologie linguistique peut permettre de comprendre les requêtes (représentation du contenu) de l'utilisateur formulées en langue naturelle, l'amélioration de la pertinence des résultats par la Considération de la sémantique des termes.

III.3. Domaine d'application des ontologies [7]

Dans plusieurs domaines, les ontologies sont considérées comme des solutions aux problèmes sémantiques et de gestion de l'information, parmi les domaines qui ont bénéficiés de l'apport des ontologies pour l'amélioration des solutions logiciels en termes d'efficacité et de performance nous citons :

- Intégration intelligente d'information
- Recherche d'information
- Portails du Web sémantique et communauté du web
- L'éducation
- Knowledge management
- E-commerce
- Traitement du Langage Naturel (TLN)

III.4. Constituants d'une ontologie

Comme nous l'avons dit précédemment, les ontologies rassemblent les connaissances propres à un domaine donné. En représentation des connaissances, ces ontologies existent sous la forme de concepts et de relations, et permettent d'en fixer la sémantique selon un degré de formalisme variable.
Les concepts et les relations de l'ontologie sont organisés sous une forme hiérarchique qui admet une relation de subsomption. Nous détaillons ci-dessous les différents composants de l'ontologie considérée en tant qu'objet informatique.

III.4.1 Concepts :

Les connaissances portent sur des objets auxquels on fait référence à travers des concepts (classes). Un concept peut représenter un objet matériel (par exemple, un comprimé de médicament), une notion (par exemple, la quantité) ou bien une idée (Uschold & King, 1995). **[3]**

L'ensemble des propriétés d'un concept s'appelle sa compréhension ou son *intension*, et l'ensemble des objets ou êtres qu'il englobe, son *extension*. Prenons un concept volontairement anonyme noté *C*, nous pouvons lui associer : **[Web11]**

- ***une intension*** : c'est un ensemble de propriétés qualitatives ou fonctionnelles communes aux individus auxquels le concept s'applique, et permettant de définir le concept, par exemple : « *C* est une sous-catégorie de véhicules de transports automobiles, conçus et aménagés pour le transport d'un petit nombre de personnes (7 ou moins) ainsi que d'objets de faible encombrement, et dotés d'au minimum trois roues » ; **[Web11]**
- ***une extension*** : un ensemble d'entités qui entrent dans cette catégorie, par exemple : {la Twingo de Rose, la Kangoo d'Olivier, la 306 d'Isabelle, la Clio d'Alain…}.

Pour exprimer, communiquer un concept, nous choisissons une représentation symbolique, souvent linguistique et verbale, parfois iconique. (Peirce distingue par exemple trois types de signes : l'indice, l'icone, le symbole.) **[Web11]**

Figure II.2 : types de signes

III.4.2. Relations [3]

Les relations représentent un type d'interaction entre les concepts d'un domaine. Elles lient les concepts primitifs (ou simples) entre eux pour construire des représentations conceptuelles complexes que nous appelons concepts définis. Elles sont caractérisées par un terme, voire plusieurs, et une signature qui précise le nombre d'instances de concepts que la relation lie, leurs types et l'ordre de ces concepts, c'est-à-dire la façon dont la relation doit être lue.

Par exemple, la relation « diagnostique » lie une instance du concept « personnel Médical » et une instance du concept « pathologie », dans cet ordre.

Des exemples de relations binaires sont :

« observe-par », « associe-a », « qualifie-de », ou encore « connecte-a ». G. Kassel (2002) et N. Guarino et al. (1995) formalisent les principales relations jugées utiles à la modélisation d'une ontologie : « instance-de », « sorte-de », « appartenance-a », « dépendance » . . .

La relation de subsomption « est-un » (is-a) a un statut particulier car elle structure la hiérarchie ontologique. Un concept C1 (concept père) subsume un concept C2 (con-

cept fils) si toute propriété sémantique de C1 est également une propriété sémantique de C2 et si C2 est plus spécifique que C1.

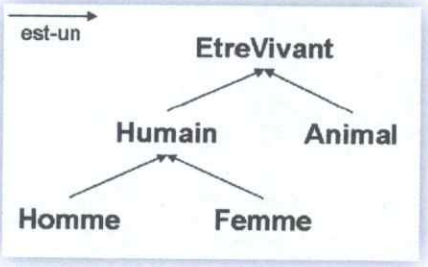


Figure II.3: représentation de la relation « est-un »

Ainsi, l'extension d'un concept est forcément plus réduite que celle de son concept père. Son intension est par contre plus riche.

Exemple :

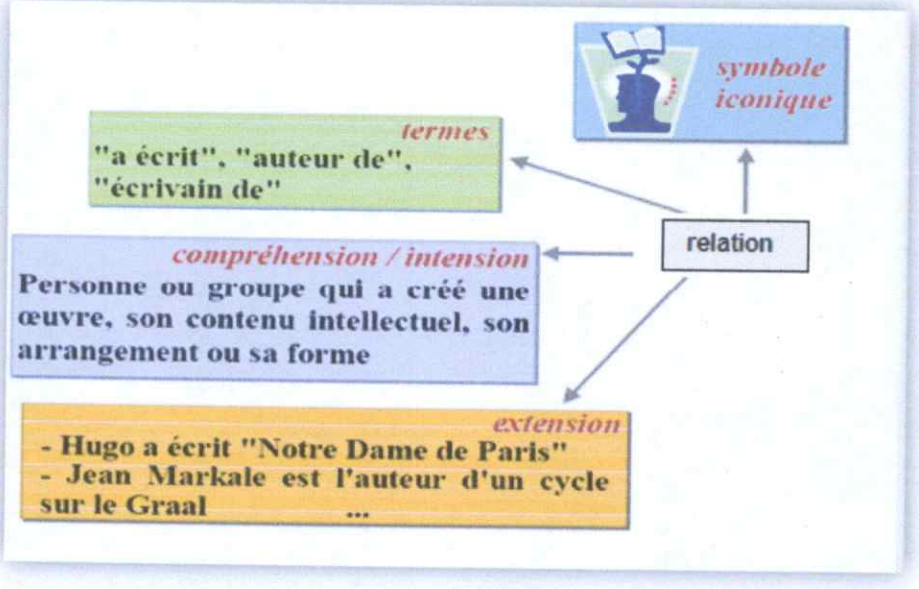


Figure II.4 : exemple représentant une relation

III.4.3. Les instances

Sont utilisées pour représenter des éléments spécifiques, ce sont des définitions extensionnelles du concept. Chambre d'hôtel, chambre d'écho, chambre des députés, chambre d'enregistrement, chambre noire et chambre funéraire sont des instances du concept "chambre".

III.4.4. Les axiomes

Constituent des assertions (affirmations), acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie. Ils peuvent intervenir dans la définition des concepts ou des relations ou alors sous forme de règles.

III.5. Typologie d'ontologie et leurs schématisations

III.5.1. Typologies d'ontologies [5]

Les ontologies peuvent être subdivisées en plusieurs dimensions, nous en retenons quatre :

- Typologie selon l'objet de conceptualisation : (Gomez-Pérez, 1999), (Guarino, 1997), (Mizoguchi *et al.*, 1992, 1995, 1998) ;
- Typologie selon le niveau du formalisme : (Uschold et Gruninger, 1996) ;
- Typologie selon le niveau de détail : (Guarino, 1998).

Nous présentons dans cette section ces différentes typologies.

III.5.1.1. Typologie selon l'objet de conceptualisation

Selon (Van Heijst *et al.*, 1997), on peut distinguer les types d'ontologies suivants :

- ***Ontologies du domaine :*** (MizoguchiAl00).

Cette ontologie régit un ensemble de vocabulaires et de concepts qui décrivent un domaine d'application du monde cible. Elle permet de créer des modèles d'objets du monde cible.

La plupart des ontologies existantes sont des ontologies du domaine. Selon Mizoguchi, l'ontologie du domaine caractérise la connaissance du domaine où la tâche est réalisée. Dans le contexte de la formation à distance, un domaine serait par exemple : le téléapprentissage.

- ***Ontologies applicatives :***

Ces ontologies contiennent des connaissances du domaine nécessaires à une application donnée, elles sont spécifiques et non réutilisables. Par exemple, dans le contexte du e-Learning, une application peut être : la formation de Statistiques et Probabilités.

- ***Ontologies génériques ou ontologies de haut niveau (top-ontologies) :***

Ces ontologies expriment des conceptualisations valables dans différents domaines. Ce type d'ontologies est fondé sur la théorie de l'identité, la méréologie (*theory of whole and parts role*) et la théorie de la dépendance. Son sujet est l'étude des catégories des choses qui existent dans le monde. Comme les concepts de haute abstraction tels que les entités, les évènements, les états, les processus, les actions, le temps, l'espace, les relations, les propriétés, etc.

- ***Ontologies générique*** (GómezPérez99), (GómezPérez99a), (VanHeigjstAl97).

Cette ontologie aussi appelée, méta-ontologies ou *core ontologies*, véhicule des connaissances génériques moins abstraites que celles véhiculées par l'ontologie de haut niveau, mais assez générales néanmoins pour être réutilisées à travers différents domaines.

Deux exemples de ce type d'ontologies sont :

- l'ontologie méréologique (Borst97) contenant des relations, *Partie-de*
- l'ontologie topologique contenant des relations, *Associé-à*.

- ***Ontologies de tâches*** (MizoguchiAl00).

Ce type d'ontologies est utilisé pour conceptualiser des tâches spécifiques dans les systèmes, telles que les tâches de diagnostic, de planification, de conception, de configuration, de tutorat (assistance), soit tout ce qui concerne la résolution de problèmes. Elle régit un ensemble de vocabulaires et de concepts qui décrit une structure de résolution des problèmes inhérente aux tâches et indépendante du domaine.

Deux exemples d'utilisation de l'ontologie de tâche dans le domaine de l'éducation sont les suivants :

- l'ontologie de formation par ordinateur - *Computer Based Training Ontology* (JinAl99) - qui régit un ensemble de concepts spécifiques à un système d'apprentissage inhérent à des ontologies de tâche ;
- l'ontologie des objectifs d'apprentissage - *Learning Goal Ontology* (InabaAl00) - qui décrit les rôles des apprenants et des agents dans le cadre d'un apprentissage collaboratif.

III.5.1.2. Typologie selon le niveau de détail

On peut distinguer les ontologies selon le niveau de description utilisé (Psyché *et al.* 2003) :

Granularité fine : ce niveau correspond à des ontologies très détaillées, elles possèdent ainsi un vocabulaire plus riche capable d'assurer une description détaillée des concepts pertinents d'un domaine ou d'une tâche. Ce niveau de granularité peut s'avérer utile lorsqu' il s'agit d'établir un consensus entre les agents qui l'utiliseront ;

Granularité large : ce niveau correspond à des vocabulaires moins détaillés. Par exemple, les scénarios d'utilisation spécifiques où les utilisateurs sont déjà préalablement d' accord à propos d'une conceptualisation sous-jacente **[8],**. Les ontologies de haut niveau possèdent une granularité large, compte tenu que les concepts qu'elles traduisent sont normalement raffinés subséquemment dans d'autres ontologies de domaine ou d'application **[8].**

III.5.1.3. Typologie selon le formalisme utilisé

On peut distinguer les ontologies selon le formalisme utilisé pour les exprimer. Selon Uschold et Gruninger (1996), on peut distinguer quatre sortes d'ontologies :

Informelle **:** l'ontologie est exprimée en langue naturelle. 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 contradictions.

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'ambiguïté.

Semi-formelle : l'ontologie est exprimée dans un langage artificiel défini formellement.

Formelle **:** l'ontologie est exprimée dans un langage artificiel disposant d'une sémantique formelle, permettant de prouver des propriétés de cette ontologie. L'intérêt d'une ontologie formelle est la possibilité d'effectuer des vérifications sur l'ontologie : complétude, non-redondance, consistance, cohérence, etc.

Uschold et Gruninger (1996) expliquent que "*le degré de formalisation exigé du langage pour l'ontologie dépend étroitement du degré d'automatisation dans les diverses tâches impliquant l'ontologie. Si une ontologie est une aide à la communication entre personnes, alors la représentation de l'ontologie peut être informelle du moment qu'elle est précise et qu'elle capture les intuitions de chacun. Cependant, si l'ontologie doit être employée par des outils logiciels ou des agents intelligents, alors la sémantique de l'ontologie doit être rendue beaucoup plus précise*".

Selon Studer (1998), "*il y a différents types d'ontologie et chaque type remplit un rôle différent dans le processus de construction du modèle du domaine*".

III.5.2. Schematisation des typpologie d'ontologie vue par certain auteurs :[6]

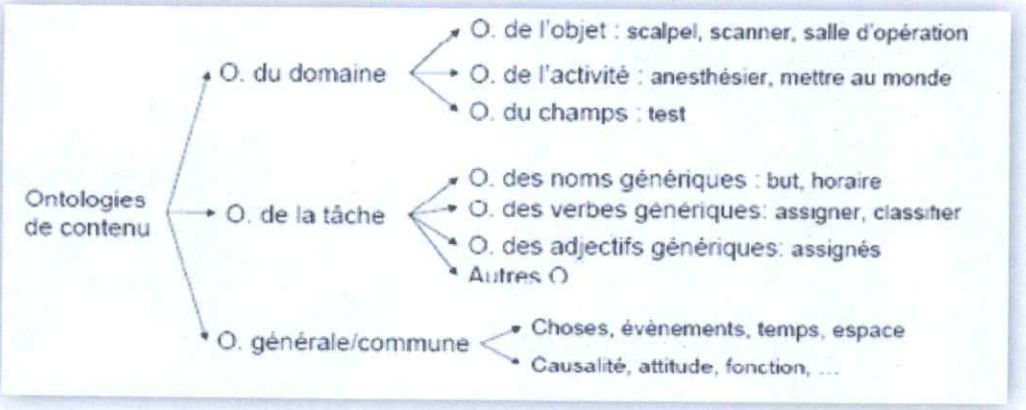


Figure II.5: type d'ontologie introduite par Mizoguchi et al (1992,1995)

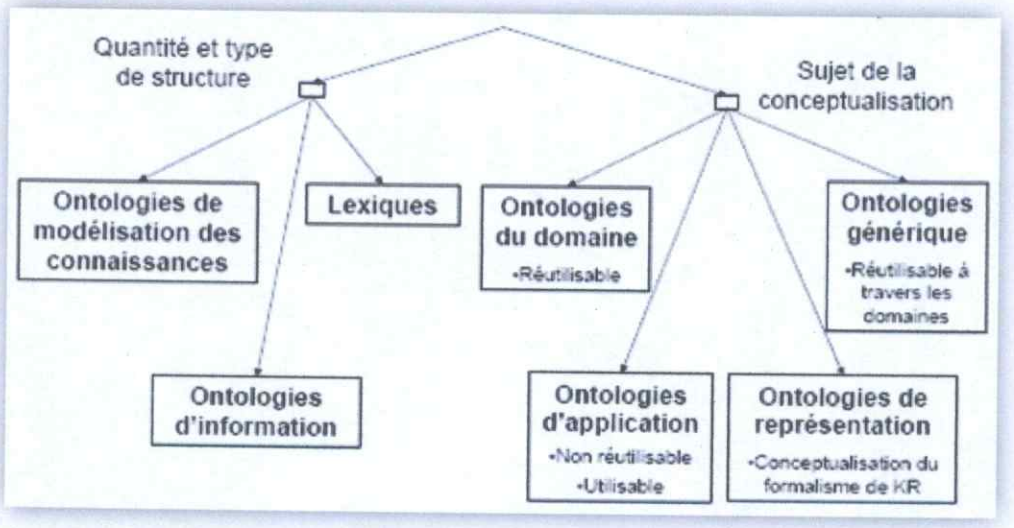


Figure II.6 : type d'ontologie introduite par van Heijst et al. (1997)

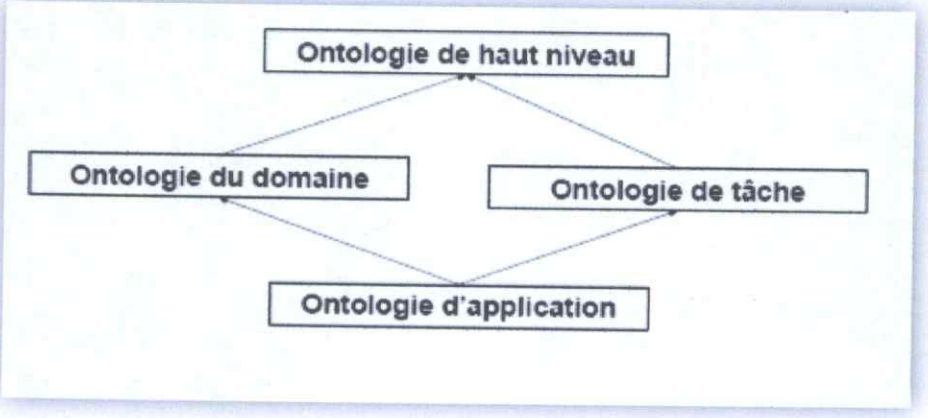


Figure II.7 : type d'ontologie introduite par Guarino N (1998)

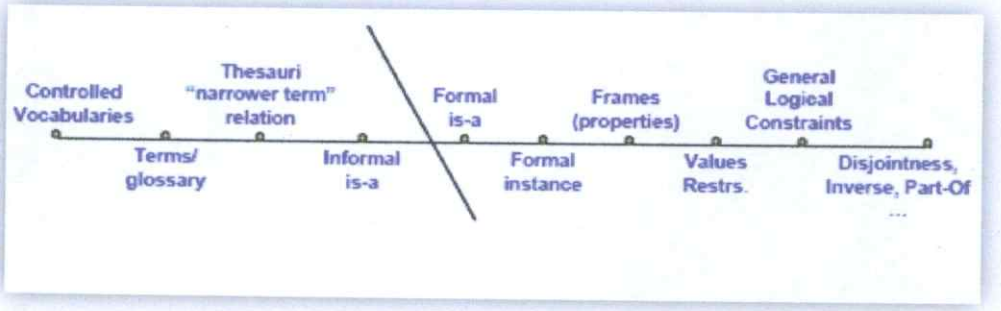


Figure II.8: type d'ontologie introduite par Lassila et McGuiness (2001)

III.6. Méthodes et méthodologies de l'ingénierie ontologique [7]

III.6.1. Les catégories des méthodes et méthodologies

Le recensement des méthodologies de l'ingénierie ontologique a permis de dénombrer au total 33 méthodologies existantes à l'heure actuelle.
Elles sont réparties sur 6 catégories :

- La construction de nouvelles ontologies à partir de zéro.
- La réingénierie d'ontologies existantes.
- La construction de nouvelles ontologies par la fusion ou l'intégration d'ontologies existantes.
- La construction collaborative d'ontologie.
- L'évolution d'une ontologie.
- L'évaluation d'une ontologie.

III.6.1.1 Méthodes de construction de nouvelles ontologies à partir de zéro :

On présente dans cette partie les méthodes les plus connues pour la construction d'ontologies à partir de zéro.

- **Méthode de Cyc** : 3 phases ont été adoptées pour la construction de la base de connaissance Cyc
  1. La première phase consiste à coder manuellement la connaissance explicite et implicite, sans utiliser le langage naturel ni l'étude des systèmes.
  2. La deuxième phase propose la codification des connaissances avec des outils utilisant la connaissance déjà stocké dans le Cyc.
  3. La troisième phase délègue aux outils la majorité du travail, les développeurs recommanderons seulement aux outils d'extraire les connaissances à partir de leurs différentes sources et ils expliqueront les parties les plus difficiles du texte.

- **Méthode d'Uschold et King** : conçue pour la modélisation des processus d'entreprise, d'Uschold et King proposent les directives suivantes :
  1. Identification des objectifs et du contexte (le but, l'utilisation et les utilisateurs potentiel de l'ontologie)
  2. Construction de l'ontologie : 3 étapes :

     2.1) capture de l'ontologie

     2.2) codage de l'ontologie

     2.3) intégration d'ontologies existantes

  3. Evaluation

  4. Documentation

- **Méthodologie de Grüninger et Fox** : Grüninger et Fox décrivent dans (Grüninger Fox95) la méthode adoptée dans leur expérience du développement des ontologies « TOVE » au sein du laboratoire « The Entreprise Integration Laboratory » à l'université de Toronto. Les étapes proposées pour la construction d'une ontologie sont :
  1. Scénario de motivation
  2. Question informelles de compétence
  3. Spécification de la terminologie de l'ontologie dans un langage formel
  4. Spécification des axiomes de l'ontologie dans un langage formel
  5. Théorèmes de complétude

- **Méthode KACTUS** : examiner la faisabilité de la réutilisation de connaissances dans des systèmes techniques complexes et le rôle des ontologies à la soutenir.
  Les trois étapes à suivre sont :
  1. La spécification de l'application.
  2. La conception préliminaire basée sur des catégories appropriées au plus haut niveau ontologiques.
  3. Raffinage de l'ontologie et structuration.

- **METHONTOLOGIE :**
  METHONTOLOGIE a été développée dans le laboratoire d'intelligence artificielle à l'UPM (*Université Polytechnique de Madrid*) et est utilisée pour la construction d'ontologie à partir de zéro aussi bien que pour la réutilisation ou la ré-ingénierie d'ontologies existantes [Gomez-Perez et al 02].
  Ses étapes sont les suivantes :
  1. La spécification
  2. La conceptualisation
  3. La formalisation

4. L'implémentation
5. La maintenance

- **Méthode SENSUS :**
  C'est une ontologie conçue pour être utilisée dans le traitement de langage naturel, développée à l'ISI (*Information Sciences Institute)* pour fournir une structure conceptuelle universelle qui rentrera dans le développement des systèmes de traduction automatique (Knight et al 94).
  Son contenu actuel a été obtenu en extrayant et fusionnant l'information des diverses sources électroniques de connaissances.
  SENSUS possède plus de 70 000 concepts organisés dans une hiérarchie selon leur niveau d'abstraction.
  Les différentes étapes de la méthode sont :
  1. Identifier, avec l'aide d'un expert dans le domaine, un ensemble de termes appelés graines (*seeds*).
  2. Lier manuellement les graines à l'ontologie SENSUS.
  3. Inclure tout les concepts sur le chemin des graines vers la racine de SENSUS dans l'ontologie.
  4. Ajouter les termes qui pourraient être appropriés dans le domaine et qui n'apparaissent pas encore.
  5. Pour chaque nœud qui figure dans plusieurs chemins ente les graines et la racine de l'ontologie SENSUS, le sous- arbre entier de ce nœud est parfois ajouté, basé sur l'idée que si beaucoup des nœuds dans un sous-arbre ont été capturés pour être intégré, alors les autres nœuds de ce sous arbre vont probablement être intégré aussi. Cette étape est manuelle, puisqu'elle exige une compréhension du domaine pour le choix des graines.

- **Méthodologie On-To-Knowledge :** cette méthodologie a été développée pour introduire et maintenir des applications de gestion de connaissance à base d'ontologie dans les entreprises (Staab et al 01).
  Ses étapes sont :
  1. Etude de faisabilité
  2. Démarrage
  3. Raffinage
  4. Evaluation
  5. Maintenance et évolution

III.6.1.2 Méthode de réingénierie d'ontologies :

Méthode de réingénierie d'ontologie développée par THE ONTOLOGIE GROUP à l'UPM (université polytechnique de Madrid), cette méthode se compose en 3 étapes :

1. Reverse engineering
2. La restructuration
3. Forward engineering

III.6.1.3. Méthodes de construction collaborative d'ontologies :

La plupart des méthodologies ne s'intéressent qu'au contenu des ontologies, en se sens qu'il n'est pas pris, sérieusement, en considération la notion de collaboration et de travail collectif et coopératif.

- **CO4 :**

CO4 est une infrastructure (fondation) permettant la construction, en collaboration, d'une base de connaissance à travers le web. La base consensuelle de connaissance est censée représenter le consensus dans une communauté du domaine à modéliser .Elle est accessible par les clients HTTP et peut être consultée ou éditée par les personnes autorisées.

Les coopérateurs ne modifient pas directement la base de connaissance, mais leur propre zone de travail. Dans CO4, le système considère chaque personne comme une base de connaissance.

Pour construire une base de connaissance consensuelle, les bases de connaissance individuelles doivent être liées. Toutes les bases sont organisées dans un arbre dont les feuilles sont des bases de connaissances des utilisateurs et dont les nœuds intermédiaires sont appelés des bases de connaissances de groupe. Chaque base de connaissance de groupe représente la connaissance consensuelle parmi ses fils (appelé des bases de connaissance d'abonné).

Quand les abonnés sont suffisamment confiants de quelques connaissances, ils peuvent les soumettre à leur base de connaissance de groupe. Cette proposition est alors soumise aux autres abonnés comme un appel de commentaires. Les utilisateurs doivent répondre par une des trois réponses possibles : « accepter » quand ils considèrent que la connaissance doit être intégrée dans la base de connaissance consensuelle, « rejeter » dans le cas contraire et « défier » quand ils proposent un autre changement.

- **$(KA)^2$ method :**

L'initiative $(KA)^2$ *(Knowledge Annotation of the Knowledge Acquisition community)* a pour objectif la construction d'ontologies développées conjointement par des groupes de personnes se trouvant dans différents emplacements et utilisant les mêmes langages et modèles. L'ontologie $(KA)^2$ va former une base pour annoter

les documents de la communauté d'acquisition de connaissances sur le Web pour permettre un accès intelligent à ces documents (Studer99).

Le modèle conceptuel actuel de l'ontologie $(KA)^2$ consiste en sept ontologies liées : une ontologie d'organisations, une ontologie de projets, une ontologie de personnes, une ontologie de publications, une ontologie d'événements, une ontologie de sujets de recherche et une ontologie de produits de recherche.

On peut distinguer deux sortes de développeurs dans le processus collaboratif et distribué utilisé pour construire ces ontologies. Ils ont été appelés : *Ontopic agents* et *Ontology coordinating agents*.

La majorité des ontologies $(KA)^2$ (organisations, projets, personnes, publications, événements et produits de recherche) ont été développées par les agents de coordination de l'ontologie.

Quant au développement de l'ontologie de sujets de recherche, les agents de la coordination de l'ontologie et les agents *Ontopic* y participent conjointement.

III.6.I.4. Méthode de fusion et d'intégration d'ontologies :

- **La méthode FCA-Merge pour la fusion d'ontologies** : ses étapes :
  1. Analyse linguistique et génération de concepts
  2. Génération de treillis de concept
  3. Dérivation de l'ontologie à partir du modèle de concepts

- **ONIONS :**

ONIONS *(ONtological Intégration Of Naïve Sources)* est une méthodologie pour l'analyse conceptuelle et l'intégration ou la fusion ontologique de terminologies.

ONIONS aspire à fournir une axiomatisation vaste, une sémantique claire et une profondeur ontologique aux terminologies de domaine qui vont être intégrées ou fusionnées.

L'analyse ontologique et la fusion de terminologies avec ONIONS sont effectuées dans deux phases : la réingénierie d'ontologies et la fusion d'ontologies.

La phase de la réingénierie consiste en l'extraction, le formatage, l'analyse et la formalisation des données conceptuellement appropriées des sources.

Si les ontologies du domaine obtenues dans le processus de la réingénierie sont hétérogènes (le cas habituel), ils peuvent être logiquement intégrées ou fusionnées en créant l'union de leurs vocabulaires et en comparant l'axiomatisation de chacun aux autres. Cependant, des problèmes peuvent surgir durant le processus d'intégration ou de fusion d'ontologies comme la redondance (con-

cepts avec des noms distincts, équivalence...), l'incomplétude de l'ontologie résultante, etc.

La méthodologie ONIONS fournit trois techniques pour la résolution de ces problèmes : l'extraction de la synonymie, l'enrichissement de la taxinomie et la gestion de la polysémie (plusieurs sens pour un mot).

- **PROMPT :**

PROMPT est un algorithme de fusion et d'alignement d'ontologies qui est pensé pour être semi-automatisé. Il est implémenté dans un *plug-in* pour Protégé-2000, et exécute quelques tâches automatiquement et dirige l'utilisateur dans l'exécution d'autres tâches pour lesquelles son intervention est exigée. Le *plug-in* détermine des inconsistances possibles, et suggère des méthodes pour corriger ces inconsistances. PROMPT est basé sur un modèle de connaissance extrêmement général et peut donc être appliqué à travers des plates-formes diverses.

PROMPT tient compte de particularités différentes dans les ontologies sources pour faire des suggestions et chercher des conflits. Ces particularités incluent : les noms de classes et des *slots,* la hiérarchie de classe, l'attachement de *slots* aux classes, les facettes et les valeurs de facettes.

- **The MOMIS methodology :**

MOMIS (Mediator environment for Multiple Information Source) suit une approche sémantique pour l'intégration de l'information, basée sur un schéma conceptuel, des sources de l'information.

Le processus d'intégration consiste en la création d'un schéma virtuel et global de toutes les sources de données en adoptant les étapes suivantes :

1. Construction d'un thesaurus commun
2. Regroupement des classes
3. Classe globale et tables de configuration

III.6.1.5. Méthode d'évaluation des ontologies :

L'évaluation des ontologies est une activité clé dans les processus d'ingénierie et de réingénierie. En effet, il est nécessaire de posséder des outils et/ou méthodes permettant l'évaluation du travail accompli, surtout dans le cas des ontologies, car si l'erreur commise pendant les phases du processus de création ou de modification n'est pas détectée, les conséquences peuvent être lourdes et difficilement réversibles : invalidité de l'ontologie, propagation de l'erreur à travers les liens de

l'ontologie et détection de l'erreur quasi impossible dans le cas des ontologies de grande taille.

- **L'approche de Gomez-Perez pour l'évaluation des taxonomies :**

Gomez-Perez a développé une méthode de « prévention » pour minimiser les risques de créer des erreurs pendant la phase de construction des taxonomies. Cette approche, basée sur une expérience d'évaluation des ontologies d'ONTO-LINGUA Server, repose sur les primitives de l'ontologie *(Frame Ontology)* et l'ensemble des erreurs qui pourraient être commises selon une étude faite par Gomez-Perez. (Gomez-Perez02)

Erreurs lors de la conception de taxonomies :

Gomez-Perez classifie ces erreurs en : erreurs d'incomplétude, erreurs d'inconsistance et erreurs de redondance.

- **Erreurs d'incomplétude :**

  Classification de concept incomplète : ce type d'erreurs apparaît quand le concept est classifié sans être expliqué et explicité.

  Ex : le concept instruments de musique est classifié en ne prenant compte que les instruments à cordes et à air, il y a dans ce cas omission des instruments à percussion par exemple.

  Erreurs de partition : l'ontologiste peut définir un ensemble de sous classes d'une classe donnée en oubliant de réunir en partition les sous classes non disjointes. Ce genre d'erreurs peut amener des incohérences lors de la création d'instances ou dans la définition des autres sous classes.

- **Erreurs d'inconsistance :**

  Erreurs de circularité : ce genre d'erreur se matérialise quand une classe se trouve définit comme la généralisation ou la spécification d'elle-même. Selon le nombre de relations en cause de cette circularité ces erreurs peuvent être classées en : erreurs de distance zéro (une classe avec elle-même) erreurs de distance 1 ou erreurs de distance N.

  Erreurs de partition : ces erreurs apparaissent lors de mauvaises définitions de partitions ou de d'assignations incorrectes d'instances à des classes formant des partirions. Les liens erronés devront être éliminés et les instances mal placées doivent être replacés en prenant en considération toute la partition.

- **Erreurs de redondance :**

La redondance est la redéfinition d'expressions déjà explicitement définies, ou la définition d'expressions qui peuvent être inférées en utilisant d'autres expressions.

- **OntoClean :**

Cette méthode d'évaluation d'ontologies a été développée par *thé ontology group* au CNR de Padova (Italie). Le but de Ontoclean est de « nettoyer » l'ontologie de toutes les relations *Subclass-of* erronées ou fausses dans la taxonomie, cette évaluation se fait suivant des notions philosophiques tels que : la rigidité, l'unité, l'identité. Ces notions sont normalement appliquées aux propriétés, mais ont été étendues aux concepts.

III.7. Cycle de vie d'une ontologie [8]

Les ontologies étant destinées à être utilisées comme des composants logiciels dans des systèmes informatiques, leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel.

Les ontologies doivent être considérées comme des objets techniques évolutifs et possédants un cycle de vie. Ce dernier comprend :

- Une étape initiale d'évaluation des besoins, où l'on doit préciser toutes les questions ou besoins auxquelles l'ontologie doit répondre.
- Une étape de construction : une version initiale de l'ontologie sera construite en suivant une méthodologie la plus adapté à notre cas.
- Une étape de diffusion où l'ontologie est mise à la disposition des utilisateurs (humains ou machines) auxquelles elle est destinée.
- Une étape d'utilisation où l'ontologie est pleinement exploitée par ses utilisateurs et après chaque utilisation significative, l'ontologie et les besoins sont réévalués, éventuellement en sollicitant l'aide d'un expert du domaine, et l'ontologie peut être étendue et, si nécessaire, en partie reconstruite

Nous pouvons dire que le cycle de vie d'une ontologie est donc un processus itératif.

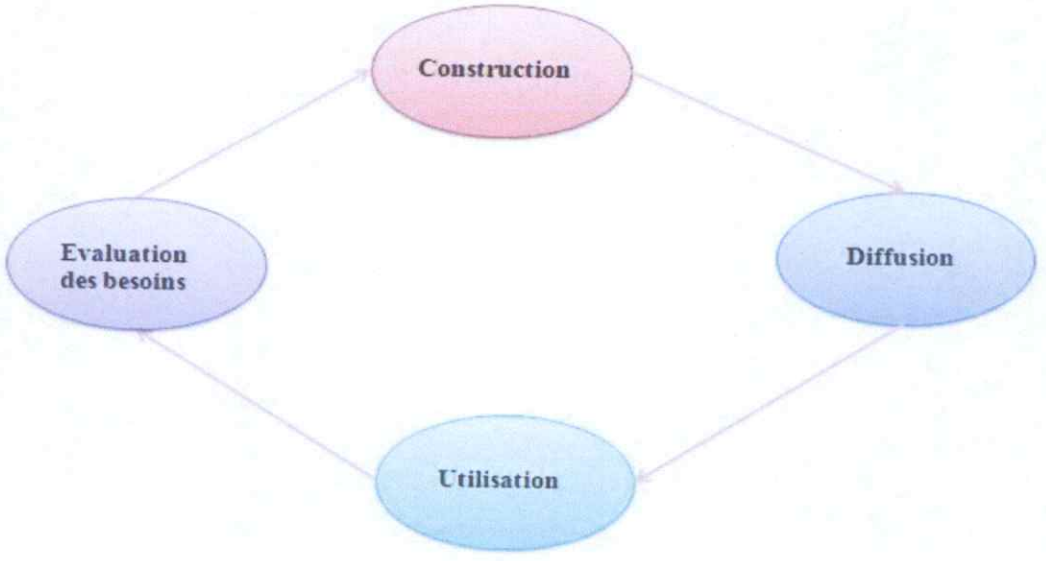


Figure II.9: Cycle de vie d'une ontologie

III.8. Outils pour les ontologies

III.8.1. Langages de construction d'ontologies [6]

III.8.1.1. Les langages traditionnels :

Dans les années 1990, les ontologies étaient construites en utilisant des techniques de modélisation d'IA fondées sur :

1) La *logique de 1er ordre* : LISP, KIF (Genesereth M.& Fikes R. E., 1992)

2) Les « *Frames* » combinés avec de la logique de 1er ordre : Cycl (Lenat&Guha, 1990), OCML (Motta E., 1999) et Flogic (Kifer M., 1985)

3) Les *logiques de description* : Loom (MacGregor R., 1991)

4) OKBC au aussi été créé comme un protocole d'accès aux ontologies implémentée dans un langage de *Frames*.

III.8.1.2. Les langages du web :

Le boom de l'Internet a mené à la création de langage d'implémentation de l'ontologie exploitant les caractéristiques du Web.
Ces langages sont des langages de balisage (ou *markup languages*)
Quelques-uns de ces langages tels que SHOE, XOL, OIL et OIL+DAML sont fondés sur la syntaxe XML que nous spécifierons ultérieurement.

Le W3C travaille aussi à créer des standards et des recommandations de langages Web à utiliser pour implémenter des ontologies.
Le W3C a d'abord proposé RDF, un langage de type réseau sémantique pour décrire les ressources Web, et RDF *Schema* , une extension de RDF. La combinaison de RDF et RDF *Schema* est connu sous le nom de RDF(S) puis le consortium a développé OWL qui est le successeur de DAML+OIL et s'inspire des logiques de description.

III.8.1.3. eXtended Markup Language et XML Schema [9]

L'eXtended Markup Language (XML) est un langage de description et d'échange de documents structurés, issu de SGML (Standard Generalized Markup Language) et défini par le consortium Web.
XML permet de décrire la structure arborescente de documents à l'aide d'un système de balises permettant de marquer les éléments qui composent la structure et les relations entre ces éléments. XML ne pose aucune contrainte sémantique sur la description des informations, il ne constitue donc pas un langage de modélisation d'ontologies à lui seul. Avec XML nous pouvons décrire une Personne Toto âgée de 37 ans qui habite 12 rue des pins.

La syntaxe en XML :
<Personne id='Toto'>
<Adresse>12 rue des pins</Adresse>
<Age>37</Age>
</Personne>

XML Schema (XML-S) est un outil de définition de grammaires caractérisant des arborescences de documents (notion de validité syntaxique). Avec les schémas XML, il est possible de contraindre la structure arborescente d'un document mais pas la sémantique des informations contenues dans ce document.

III.8.1.4. Resource Description Framework et RDF Schema

Le Resource Description Framework (RDF) est un modèle pour la représentation de métadonnées à propos de ressources. Cette représentation est faite sous la forme d'un triplet :

- Sujet : la ressource que l'on définit
- Prédicat : la propriété de la ressource, qui est une liaison étiquetée et orientée du sujet vers l'objet.
- Objet : la valeur de la propriété pouvant être une autre ressource ou bien un littéral.

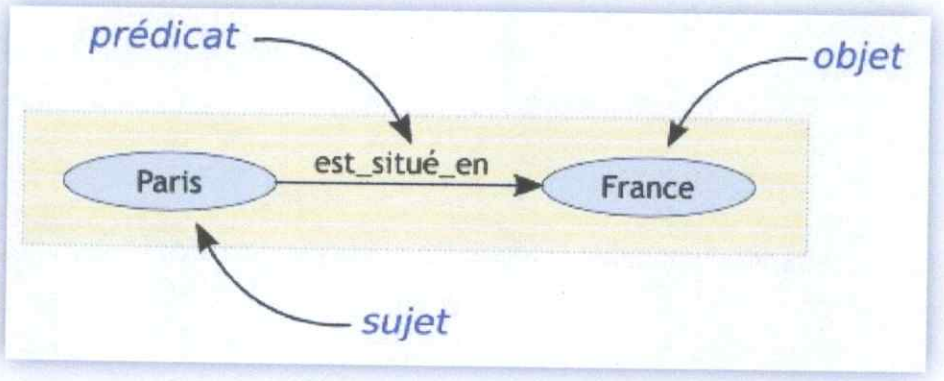


Figure II.10 : Exemple de triplet RDF

RDF est à la base exprimé sous la forme d'un graphe orienté mais on peut le décrire par la syntaxe XML. On parle alors de RDF/XML qui par abus de langage peut être appelé RDF. En prenant l'exemple défini pour XML, nous pouvons définir avec RDF (codeII.1) et un graphe (Figure II.11) : Toto qui possède les propriétés adresse et âge.

Si RDF fournit une capacité d'échange de connaissances, il ne permet pas à l'utilisateur de définir le vocabulaire des termes à utiliser, ni d'établir la sémantique des objets utilisés.

```
<rdf :RDF>
<rdf :Description about='Toto'>
<rdf :Property about='adresse'>
12 rue des pins
</rdf :Property>
<rdf :Property about='age'>
37
</rdf :Property>
</rdf :Description>
</rdf :RDF>
```

Code II.1: Représentation RDF/XML de Toto 37 ans qui habite au 12 rue des pins

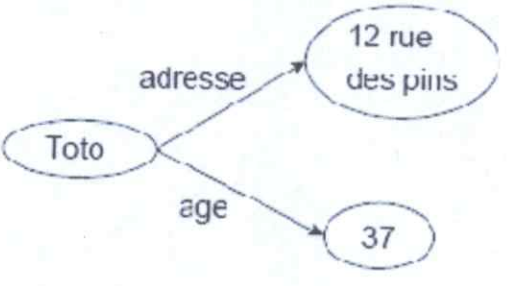


Figure II.11 : Représentation graphique de Toto 37 ans qui habite au 12 rue des pins

RDF schéma ou RDFS est un langage permettant de définir des propriétés sémantiques pour les ressources par un schéma. Dans un schéma on peut définir de nouvelles ressources comme des spécialisations d'autres ressources. Les schémas con-

traignent aussi le contexte d'utilisation des ressources. Avec RDFS de nouvelles notions sémantiques apparaissent.

Sur l'exemple de Toto, avec le RDF schéma nous définissons le concept de Personne (code II.2) avec les types des objets de chaque propriété, une taxinomie de concepts ou hiérarchie de subsomption (figure II.12) ainsi que l'instance Toto (code II. 3). Nous avons vu que RDF et RDFS permettent de définir sous la forme d'un graphe de triplets des données ou des méta-données. Cependant, il est impossible de raisonner sur ces représentations car la sémantique (hors subsomption) reste très limitée. C'est ce manque de sémantique que OWL comble par l'apport d'un vocabulaire plus riche.

```
<rdf :RDF>
<rdfs :Class rdf :about='Animal'>
<rdfs :subClassOf rdf :resource='Thing'/>
</rdfs :Class>
<rdfs :Class rdf :about='Personne'>
<rdfs :subClassOf rdf :resource='Animal'/>
</rdfs :Class>
<rdf :Property about='age'>
<rdfs :domain rdf :resource='Animal'/>
<rdfs :range rdf :resource='xsd :integer'/>
</rdf :Property>

<rdf :Property about='adresse'>
<rdfs :domain rdf :resource='Personne'/>


<rdfs :range rdf :resource='xsd :string'/>
</rdf :Property>
</rdf >
```

Code II.2: Représentation RDFS du concept Personne

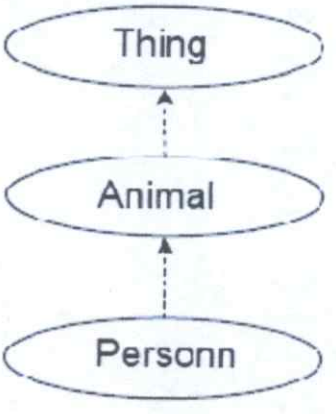


Figure II.12 Taxinomie des concepts

```
<Personne rdf :ID='Toto'>
<age rdf :resource='37' />
<adresse rdf :resource='12 rue des pins' />
</Personne>
```

Code II.3 : Représentation RDFS de Toto

III.8.1.5. Ontology Web Language

OWL (Ontology Web Language) est un langage fondé sur la syntaxe RDF/XML et héritier des travaux de DAML+OIL.
OWL n'est pas simplement une extension de RDF, il introduit l'aspect sémantique lui manquant, comme les outils de comparaison de propriétés et de classes (identité, équivalence, contraire, cardinalité, symétrie, etc.). Ainsi par ses primitives plus riche, OWL offre une capacité d'interprétation plus grande à la machine que RDF et RDFS.
OWL se décompose en trois sous-langages OWL Lite, OWL DL et OWL Full, offrant des capacités d'expression croissantes et donc destinés à des différentes utilisations.

OWL Lite : est le sous langage de OWL le plus simple car son utilisation est contrainte.
Voici la liste des constructeurs proposés par OWL Lite :

Tableau II.1 : liste des constructeurs proposés par OWL

Catégories	**Constructeurs**
RDF Schema	Class, rdfs :subClassOf, rdf :Property, rdfs :subPropertyOf, rdfs :domain, rdfs :range, Individual
(In)Equalité	equivalentClass, equivalentProperty, sameas, differentFrom, AllDifferent, distinctMembers
Restrictions	onProperty, allValuesFrom, someValuesFrom
Cardinalités (0 ou 1)	minCardinality ,maxCardinality, cardinality
Intersection	intersectionOf
Propriétés	SymmetricProperty, FunctionalProperty, Objectproperty, DatatypeProperty, inverseof, TransitiveProperty, InverseFonctionalProperty

Nous illustrons ces constructeurs par un exemple d'ontologie décrite en OWL Lite

Tableau II.2 : illustration des constructeurs OWL Lite

Catégories	Exemple
RDF Schema	Personne :subClassOf (Thing) Femme : subClassOf (Personne) Enfant : subClassOf (Personne)
(In)Equalité	Fille : equivalentClass (intersectionOf (Femme, Enfant))
Intersection	Parent : intersectionOf (Personne,
Restrictions	Restriction (minCardinality (1), onproperty (aEnfant)))
Propriétés	aParent : ObjectProperty (Enfant, Personne) aEnfant : inverseOf (aParent)

OWL DL : est plus complexe que OWL Lite, il est fondé sur la logique de description (d'où son nom OWL Description Logics). Malgré sa complexité OWL DL garantit la complétude des raisonnements (calculabilité des inférences) et leur décidabilité (leur calcul se fait en une durée finie).Voici la liste des constructeurs ajoutés par OWL DL par rapport OWL Lite :

Tableau II.3 : liste des constructeurs ajoutés par OWL DL

Catégories	Constructeurs
Axiome de Classe	oneOf (enumération), dataRange, disjointWith
Expressions booléennes	unionOf, complementOf
Cardinalités (0, n)	minCardinality,maxCardinality, cardinality
Individu cible d'une propriété	hasValue

Nous illustrons ces constructeurs par un exemple d'ontologie décrite en OWL DL

Tableau II.4 : illustration des constructeurs OWL DL

Catégories	Exemple
Axiome de Classe	Gender : oneOf (Male,Female)
Expressions booléennes	Tante : intersectionOf (Femme , unionOf

Individu cible d'une propriété	(aneuve, aNiece)) Homme: intersectionOf (Personne, has-Value (sexe,Male))

OWL Full : est la version la plus complexe d'OWL, mais également celle qui permet le plus haut niveau d'expressivité. Son utilisation n'est contrainte que par le langage RDF, mais elle ne garantit pas la complétude et la décidabilité des calculs liés à l'ontologie.
La syntaxe ne subit pas de modification par rapport à OWL DL mais OWL Full permet une utilisation sans contrainte sur l'utilisation des constructeurs.

Exemple : utiliser un concept à la place d'un individu
Adulte : IntersectionOf(Personne, hasValue(age,Majorite)), Majorite étant un concept remplaçant l'individu de type Age et de valeur 18
Les 3 niveaux d'OWL présentent une hiérarchie sur la validité des ontologies :
– une ontologie OWL Lite valide est également une ontologie OWL DL valide
– une ontologie OWL DL valide est également une ontologie OWL Full valide
OWL est un langage basé sur du XML, nous permettant de décrire de manière riche des informations.

III.8.2. Editeur d'ontologie [11]

Il existe actuellement de nombreux outils d'édition d'ontologies parmi lesquels nous citons :

III.8.2.1. ONTOEDIT (ONTOLOGY EdiTOR)

Permet l'édition des hiérarchies de concepts et de relations et l'expression d'axiomes algébriques portant sur les relations, et de propriétés telles que la généricité d'un concept. Des outils graphiques dédiés à la visualisation d'ontologies sont inclus dans l'environnement. ONTOEDIT intègre un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs.

III.8.2.2. ONTOLINGUA

Une ontologie est directement exprimée dans un formalisme également nommé ONTOLINGUA, qui constitue en fait une extension du langage KIF (Knowledge Interchange Format).

Le Serveur ONTOLINGUA est un ensemble d'outils et services qui supportent la construction d'ontologies partagées entre groupes distribués. L'architecture du serveur de l'ontologie permet l'accès à une bibliothèque d'ontologies, à des traducteurs vers des

langages (tel que Prolog) et à un éditeur permettant de créer et de parcourir des ontologies. Les éditeurs distants peuvent parcourir et modifier les ontologies, tandis que les applications distantes ou locales peuvent accéder à n'importe quelle ontologie de la bibliothèque en utilisant le protocole OKBC (Open Knowledge Based Connectivity).

III.8.2.3. PROTEGE

Protégé-2000 est un éditeur qui permet de construire une ontologie pour un domaine donné, de définir des formulaires d'entrée de données, et d'acquérir des données à l'aide de ces formulaires sous forme d'instances de cette ontologie. Protégé est également une librairie Java qui peut être étendue pour créer de véritables applications à bases de connaissances en utilisant un moteur d'inférence pour raisonner et déduire de nouveaux faits par application de règles d'inférence aux instances de l'ontologie et à l'ontologie elle même (méta-raisonnement).

Dans le contexte du web sémantique des « plugin » pour les langages RDF, DAML+OIL et OWL ont été développés pour Protégé. Ces « plugin » permettent d'utiliser Protégé comme éditeur d'ontologies pour ces différents langages, de créer des instances et les sauver dans les formats respectifs.

Il est également possible de raisonner sur les ontologies en utilisant un moteur d'inférence général tel que JESS, ou des outils d'inférence spécifiques au web sémantique basés sur des logiques de description [DL] tels que RACER .Ces deux outils peuvent être facilement intégrés à Protégé.

Conclusion :

Nous avons donc vu au cours de ce chapitre les notions de bases des ontologies, les différentes méthodologies de constructions d'ontologies, leurs domaines d'application qui est dans notre cas la recherche d'information ainsi que les principaux outils d'édition d'ontologies.

L'ontologie est une clef de voûte des systèmes multi-agents utilisant une communication de haut niveau, nous allons entamer dans le chapitre suivant la notion de système multi agents et dans notre cas, on aura un agent qui exploitera l'ontologie pour la recherche d'information.

CHAPITRE III

LES SYSTEMES MULTI -AGENTS

LES SYSTEMES MULTI-AGENTS

L'ORIGINE DES AGENTS ET DES SMA [Web13]

Elle provient d'une part, de l'intelligence artificielle distribuée (IAD), et d'autre part de la vie artificielle. Les premières applications de la vie artificielle sont apparues quasiment en même temps que l'informatique, avec Von Neumann et ses automates cellulaires, ou encore Mc Culloch et ses neurones formels.

L'IAD est apparue vers la fin des années 1970. Hewitt confronté à un problème de résolution de théorèmes proposa une solution en 1977 avec des entités actives appelées acteurs et considéra la résolution comme une confrontation de points de vue.

D'un autre côté, Erman et al (1980) élabora l'idée du tableau noir (" blackboard ") avec le projet HERSAY II. Le tableau noir très utilisé est un emplacement réservé pour la transition des informations entre les différents agents. Chaque agent peut venir consulter le tableau noir. Ce sont donc à partir de ces premières réalisations que nous avons vu apparaître l'idée de système multi-agents.

Nous vous présentons ce concept de système multi-agents, mais dans un premier temps, nous vous présentons l'agent individuellement.

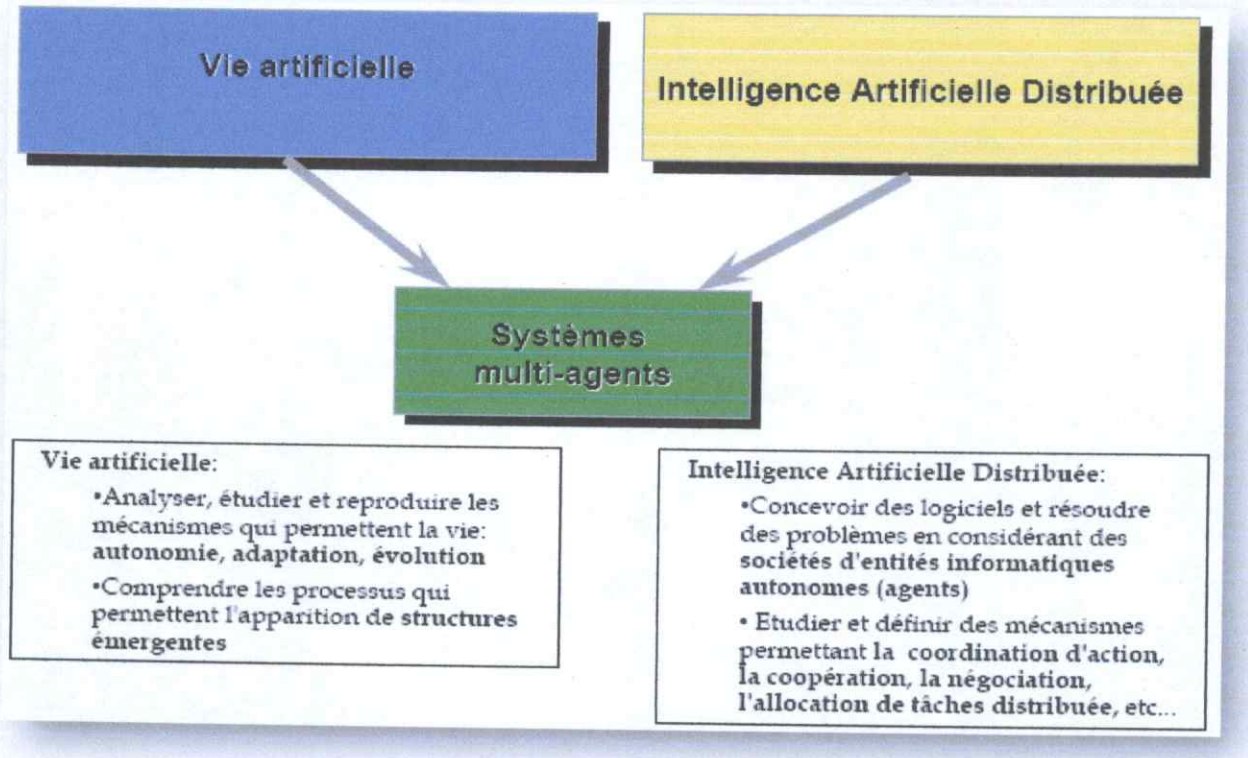


Figure III.1 : où se situe les systèmes multi-agents

I. L'AGENT

I.1. Définition : Les agents sont des composants logiciels et / ou matériels capables à des degrés différents :

- D'être autonome : Fonctionnement automatique.
- De communiquer : Echanger des informations avec d'autres programmes ou des hommes.
- D'apprendre : Capables de réagir avec un environnement, de s'adapter aux circonstances, de prendre une décision ou d'enrichir eux-mêmes leur propre comportement, sur la base d'observations qu'ils effectuent.

En informatique, un **agent** est l'équivalent d'un robot logiciel. C'est un programme qui accomplit des tâches à la manière d'un automate et en fonction de ce que lui a demandé son auteur **[Web14].**

Dans le contexte d'Internet, les agents intelligents sont liés au Web_sémantique, dans lequel ils sont utilisés pour faire à la place des humains les recherches et les corrélations entre les résultats de ces recherches. Ceci se fait en fonction de règles prédéfinies. Ils sont capables d'une certaine autonomie, en particulier de dialoguer entre eux.

Ferber a donné une définition minimale d'agent **[Web13]** :

On appelle agent une entité physique ou virtuelle :

1. qui est capable d'agir dans un environnement
2. qui peut communiquer directement avec d'autres agents.
3. qui est mue par un ensemble de tendance (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
4. qui possède des ressources propres
5. qui est capable de percevoir (mais de manière limitée) son environnement,
6. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
7. qui possède des compétences et offres des services,
8. qui peut éventuellement se reproduire,
9. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

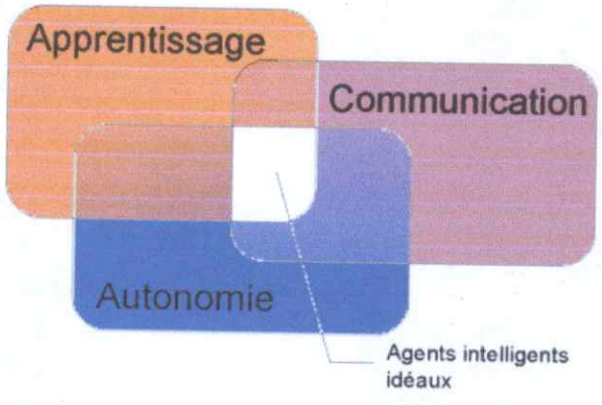


Figure III.2 : caractéristiques d'un agent idéal

I.2. Typologie des Agents [Web13]:

Dans le domaine de l'intelligence artificielle distribuée, il existe deux grandes catégories d'agents.

I.2.1. Les agents réactifs :

Les agents réactifs n'ont pas de représentation de leur environnement, de leur monde si ce n'est que très partiellement. Les actions consistent à réagir à des informations leur parvenant (comportement behavioriste). L'interaction de plusieurs agents permet l'émergence d'une intelligence. Les sociétés d'insectes comme les fourmis, les termites sont les exemples les plus anciens d'agents réactifs (Bonabeau 1994).

Un agent réactif peut être modélisé par un simple objet doté d'un comportement et d'un moyen de communication avec les autres agents. Il n'a pas de connaissance sur les autres, de capacité de raisonner sur les messages qu'il reçoit, ou de développer des stratégies de contrôle (CAR 84) (DEM & MUL 91).

Un agent réactif est plutôt centré comportement que connaissance.

I.2.2. Les agents cognitifs :

Ils ont en général une représentation précise de leur environnement, et chaque agent possède des connaissances comprenant des informations, un savoir-faire. De plus ces agents sont dits intentionnels car ils ont des buts (Ferber 1995). Ils doivent être intelligents. L'intelligence est définie le plus souvent en intelligence artificielle par la capacité d'apprentissage, et la capacité à s'adapter. Et l'agent intelligent doit faire preuve d'autonomie dans son intelligence. Il doit donc avoir des désirs et des intentions. L'agent cognitif est donc intéressant individuellement et collectivement, tandis que l'intérêt d'un agent réactif est uniquement l'interaction avec les autres agents. L'agent cognitif reprend en fait les différentes méthodes de l'intelligence artificielle auxquelles vient s'ajouter un comportement social.

I.2.3. Etude comparative [12] :

Le tableau ci-dessus résume les différences entre les agents cognitifs et réactifs (LAB & LEJ 93). C'est l'école cognitive qui, jusqu'à maintenant, a donné lieu aux applications les plus avancées.

Tableau III.1 : comparaison entre agent cognitif et agent réactif

AGENTS COGNITIFS	AGENTS REACTIFS
Représentation explicite de l'environnement	Pas représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son historique
Agents complexes	Fonctionnement stimulus/réponse
Petit nombre d'agents de forte granularité	Grand nombre d'agents de faible granularité

I.3. Propriétés d'un agent [Web13]:

> Tout d'abord un agent doit faire preuve d'autonomie, c'est à dire qu'il doit pouvoir agir seul, prendre une décision seul, en fonction d'informations lui provenant d'autres agents, de l'environnement. Il gère lui-même son état interne en fonction des informations qui lui parviennent. Nous pouvons détailler plus précisément cette propriété avec les propriétés pro-activité, et réactivité.

> En plus de l'autonomie, l'agent évolue de façon autonome mais aussi de façon **continue**. Cette propriété découle directement de la notion d'autonomie. S'il doit y avoir un démarrage, et un arrêt régulièrement, il n'y a plus d'autonomie

> **La réactivité** est aussi une propriété primordiale pour un agent. C'est à dire que l'agent reçoit des informations de son environnement, et il doit être capable de réagir en conséquence.

> Un agent ne peut être autonome que s'il est capable de **pro-activité**. C'est à dire qu'il peut avoir un comportement piloté par des buts en prenant l'initiative.

> Une autre caractéristique très importante est l'**efficacité**. C'est-à-dire la capacité à résoudre le problème, à atteindre ses buts. Des estimations sont faites pour pouvoir jauger de l'efficacité d'un agent.

> Ensuite, un agent doit faire preuve d'**adaptation**. En fait, il sait résoudre un certain nombre de problèmes, il est intéressant parfois qu'il soit capable de résoudre un nouveau problème à partir de son expérience. Pour ce faire, l'agent doit être capable d'apprentissage.

> **La robustesse** est la propriété directement complémentaire à l'adaptation. C'est la capacité justement que l'agent n'évolue pas trop vite, sinon il va être influencé par de faux signaux. L'agent doit donc avoir un juste équilibre entre adaptation et robustesse.

> La communication est indispensable à l'agent avec d'autres agents (comportement social) ce qui amène des difficultés majeures en terme de normalisation, de **capacité à communiquer**, de langages, de données, de priorités.

> Les agents ne possèdent pas forcément toutes les propriétés. De plus, il existe de nombreuses autres propriétés, le but ici est juste de vous donner une meilleure idée, une meilleure vision d'un agent.

I.4. Etats internes d'un agent [Web13]:

Il existe de nombreux types d'états internes (Müller [97] et Ferber [97]), nous décrirons les plus connus :

- croyance : les croyances décrivent l'état du monde du point de vue d'un agent. Elles peuvent donc être erronées.
- But : Les buts sont les résultats essentiellement de pulsions ou de demandes. Une possibilité de la réalisation d'un but n'est pas prise en compte. Les buts peuvent s'exprimer soit en termes d'attirance, soit en termes de répulsion.
- Pulsion : Les pulsions sont les sources internes de construction de buts. Elles sont l'expression des demandes du système conservatif (manger, se reproduire).
- Demande : Ce sont les requêtes provenant d'autres individus.
- Intention : Cela décrit ce qui permet à un agent de passer à l'acte.

Tout comme les propriétés, un agent n'a pas obligatoirement tous les états internes, ceci est particulièrement vrai pour les agents réactifs...

II. LE SYSTEME MUTI-AGENTS :

II.1. Définition :

Les systèmes multi-agents sont l'une des branches les plus prometteuses de l'I.A. Selon FERBER & GHALLAB (1988) : Un système multi-agents est une communauté d'agents autonomes travaillant en commun, selon des modes parfois complexes de coopération, conflit, concurrence, pour aboutir à un objectif global : la résolution d'un problème, l'établissement d'un diagnostic.

Ferber [1997] nous donne une excellente définition d'un système multi-agents. Il est composé des éléments suivants : **[Web13]**

1. Un environnement E, dans notre cas, c'est l'espace où peuvent se déplacer les agents.
2. Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que pour tout objet, il est possible, à un moment donné, d'associer une position dans E.
3. Un ensemble A d'agents qui sont des objets particuliers ,lesquels représentent les entités actives du système.
4. Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.
5. Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer, et manipuler des objets de O. Cela correspond à la capacité des agents de percevoir leur environnement, de manger, etc.
6. Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers

II.1.2 Domaines d'applications [12] :

Les systèmes multi-agents connaissant une grande expansion. Ils ont touché une grande variété de domaines d'applications tels que :

- les systèmes multi-experts d'aide à la décision ;
- l'enseignement intelligent assisté par ordinateur ;
- la gestion des activités de production ;
- l'imagerie médicale ;
- les travaux urbains ;
- le transport routier, aérien et ferroviaire ;
- le diagnostic de pannes sur les réseaux électriques ;
- la télémédecine ;
- la simulation du comportement des fourmis et l'étude de l'émergence des structures
- l'interprétation des images satellitaires, la traduction automatique ;
- la robotique ;
- etc.

II.2. Modèles des systèmes multi-agents: [12]

Nous distinguons trois classes de systèmes multi-agents (FER 89) (HAT 91) (KHO 94) Systèmes à tableau noir, systèmes d'acteurs et les systèmes physiquement distribués.

II.2.1 Les systèmes à tableaux noirs :

Le modèle de tableau noir :

Dans un système à tableau noir, appelés aussi **sources de connaissances,** sont des modules interdépendants renfermant chacune une partie de la connaissance du domaine d'application traité. Un agent, dans de tels systèmes, n'a connaissance ni du problème complet, ni de l'état global de la solution. Il réagit uniquement auxquelles il a accès et ne peut pas raisonner sur sa coopération avec les autres. La communication entre ces différents agents se fait à travers une zone de données commune, appelée **tableau noir.**

Ainsi le système de contrôle est centralisé. Le modèle du tableau noir résout les problèmes d'une façon incrémentale et opportuniste.

Tout système basé sur le modèle de tableau noir est composé de trois éléments (HAY 85) (ENG 88)(FER 89)(HAT 91): les sources de connaissances (agents), le tableau noir et le mécanisme de contrôle.

- Les sources de connaissances (SC) : les connaissances du domaine nécessaire à la résolution d'un problème sont réparties en modules indépendants appelés "sources de connaissances (knowledge sources)".Chaque source de connaissance fournit des éléments d'informations contribuant ainsi à la résolution du problème. Les sources de connaissances(SC) peuvent être des procédures, des règles.
- Le tableau noir : contient les données, les solutions partielles et la solution globale du problème à résoudre. Ces éléments de la solution sont constamment mémorisés dans cette zone de données commune accessible aux différents agents.
- Le mécanisme de contrôle : ce mécanisme supervise les changements qui surviennent dans le tableau noir et décide des actions à effectuer à la prochaine étape. Il gère les conflits d'accès entre les agents

Avantages : les architectures à base de tableau noir présentent plusieurs avantages :

- La séparation entre l'espace de solution (tableau noir), les connaissances qui opèrent sur cet espace (CS ou agents) et les mécanismes qui gèrent ces connaissances (le contrôle) confèrent au système une très grande modularité et donc une connaissance aisée.
- Le mécanisme de contrôle tente, à chaque étape du processus de résolution, de choisir la meilleure source de connaissance à appliquer et/ou de se focaliser sur une région du tableau noir. Ceci procure un gain en efficacité.

Inconvénients : Les architectures à base de tableau présentent l'inconvénient du contrôle qui est une lourdeur dans la procédure de traitement (manque de souplesse et d'efficacité).

II.2.2 Les systèmes d'acteurs :

A l'inverse des systèmes à tableau noir, les systèmes d'acteurs relèvent d'une distribution totale des connaissances et du contrôle (FER 89). Ces systèmes se caractérisent par :

- Un traitement local : un agent ne peut manipuler que sa base de connaissance locale, envoyer des messages aux autres agents qu'il connaît (accointances), et créer de nouveaux agents. De ce fait, les connaissances et le comportement sont distribués parmi les agents, qui effectuent des taches en parallèle et de manière indépendante.
- La seule structure de contrôle est l'envoi de message.

Le modèle acteur :

Le modèle acteur est une extension du modèle objet par ajout de la notion d'activité : chaque objet, appelé **acteur,** est un agent actif, autonome, communiquant librement avec ses semblables. Les acteurs évoluent dans un univers dynamique, où des activités se créent, se déroulent et s'achèvent. Ils participent à ces activités, et communiquent entre eux pour se confier ou se déléguer des taches et se transmettre des résultats (GER 90).

II.2.3 Systèmes physiquement distribués :

Les systèmes physiquement distribués sont composés d'agents plus complexes que précédemment, dotés de capacités de raisonnement élaborés (GER 90).

Généralement, ces agents sont localisés sur des sites physiques différents. Cela implique la mise en œuvre de stratégies de coordination très élaborées entre les agents. Ces derniers doivent en effet être capables de planifier intelligemment leurs activités locales en fonction de celles des autres.

Diverses approches ont été envisagées pour implanter de tels mécanismes.

II.2.4 Tableau comparatif :

Les principaux critères de comparaison (Tableau III.2) dans les systèmes multi-agents se résument dans les deux points suivants :

- taille, complexité, autonomie, degré de contribution de chaque agent, connaissances du problème à traiter et des autres agents.
- nombre d'agents en interaction, types de communication, mécanismes de contrôle et de coordination.

Tableau III.2 : Tableau comparatif des principaux critères de comparaison des SMA

Agents Critères	Source de connaissances	Acteur, Objets	Agents physiquement distribués
Granularité	Moyenne	Moyenne	Grande
Nombre	Moyenne	Grand	Petit
Communication	Partage d'information	Envoi de messages	Envoi d'information
Contrôle	Centralisé	Distribué	Distribué
Degré de contribution à la résolution	Moyen	Moyen	Elevée
Connaissance du problème global et d'autrui	Absente	Moyenne	Elevée

III. LA COMMUNICATION ET LA COOPERATION DANS UN SMA [12]

III.3.1. La communication :

III.3.1.1. Définition :

La communication est le moyen que possèdent les agents pour se mettre en relation. Elle est la base de la résolution coopérative de problèmes.

La communication est l'ensemble des processus physiques et psychologiques par lesquels s'effectue l'opération de mise en relation d'un ou plusieurs agents (émetteur) avec un ou plusieurs agents (récepteur), en vue d'atteindre certains objectifs (BOU 93)(BAU 92).

III.3.1.2. Modèles de communication :

Les modes et protocoles de communication établissent les moyens et les styles de communication entre les agents. Il existe principalement deux modèles de communication :

- **communication par envoi de messages.**
- **communication par partage d'information.**

Ces deux modèles ont été combinés pour donner naissance à un hybride (LAB & LEJ 93)(KHO 94)(BAU 92)

III.3.1.2.1. Communication par partage d'informations :

Ce type de communication est utilisé quand il y'a recouvrement des domaines d'expertises de chaque agent. Il suppose également que les agents ne possèdent qu'une connaissance limitée sur les domaines d'activité des autres agents. Ce mode de communication est adopté dans les systèmes à tableau noir. Dans ce mode de communication, les agents ne sont pas en liaison directe mais communique via le tableau noir (figure III.3)

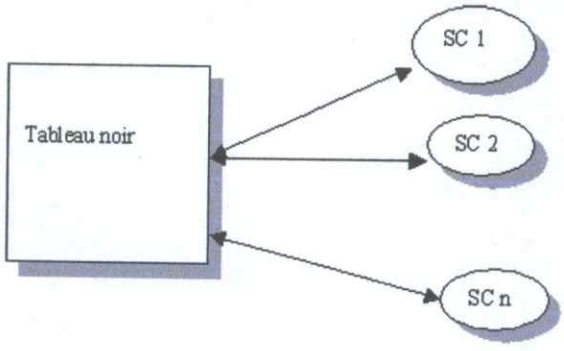


Figure III.3: Communication par partage d'informations via le tableau noir

III.3.1.2.2. Communications par envoi de messages :

Ce mode de communication est utilisé dans les systèmes d'acteurs. Dans ce cas, les agents sont en relation directe (figure III.4).Chaque agent, appelé aussi acteur, échange des messages directement et explicitement avec les autres agents qu'il connaît (ses accointances).

Ceci se fait d'une manière asynchrone. Ce message peut être une requête, une information, une suggestion, etc.

Le message est de la forme: (***Expéditeur, destinataire, type de message, message)***

Les propriétés qui caractérisent un message sont les suivantes:

Numéro: numéro du message.

Expéditeur: l'agent qui envoi le message.

Destinataire: l'agent à qui adressé le message.

Renvoyer: l'agent à qui le message doit être retourné.

Type: l'importance du message (urgent, normal).

Nature: une requête, une réponse, une information.

Si-échec: l'action à effectuer en cas d'échec.

Condition: la condition de traitement du message.

Etat: traité ou non traité.

Diffusion: point à point, diffusé.

Contenu: le corps du message.

Résultat attendu: résultat renvoyé.

Deux modes de transmission sont possibles:

- transmission point à point : l'agent émetteur du message connaît et précise l'adresse de ou des agent(s) destinataire(s).
- transmission par diffusion: le message est envoyé à tous les agents du système.

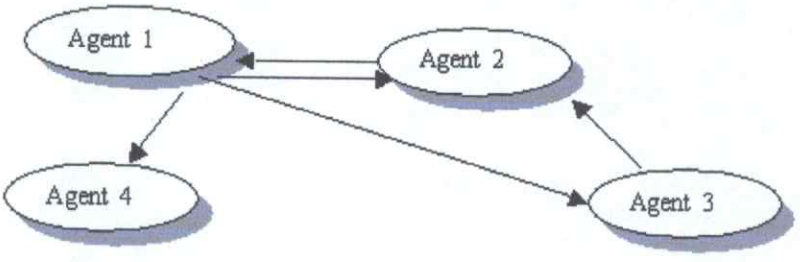


Figure III.4 : Communications par envoi de messages

III.3.2. La coopération :

III.3.2.1. Définition :

La coopération est une caractéristique très importante dans les systèmes multi-agents. En effet, une résolution distribuée d'un problème est le résultat de l'interaction coopérative entre un ensemble d'agents. Plusieurs études en I.A.D se sont rattachées à définir ce concept de coopération.

La coopération consiste à faire en sorte que des agents travaillent à la satisfaction d'un but commun, ou tout du moins à la satisfaction de tous les buts individuels (FER 95).

III.3.2.2. Modèles de coopération :

La coopération entre agents peut se faire de deux manières (KHO 94).La première suppose qu'il existe une hiérarchie entre les agents. La seconde au contraire, suppose qu'il n'y a pas de hiérarchie entre les agents.

➢ cas où il existe une hiérarchie entre les agents (figure III.5):

Dans ce cas, il existe trois modes de coopération:

1- Le mode " COMMANDE "

Un agent superviseur décompose un problème en sous problèmes qu'il répartit entre les agents. Ceux-ci les résolvent et renvoient les solutions à l'agent superviseur.

2- Le mode "APPEL D'OFFRES"

L'agent superviseur décompose un sous problèmes, dont il diffuse la liste aux agents. Chaque agent qui le souhaite envoi une offre. L'agent superviseur choisit parmi celles-ci et distribue les sous-problèmes. Le système fonctionne ensuite en mode commande.

3- Le mode "COMPETITION"

L'agent superviseur décompose un problème en sous-problèmes, dont il diffuse la liste des sous-problèmes aux agents, comme dans le mode appel d'offre. Chaque agent résout un ou plusieurs sous-problèmes et envoie les résultats correspondants à l'agent superviseur qui à son tour fait le tri.

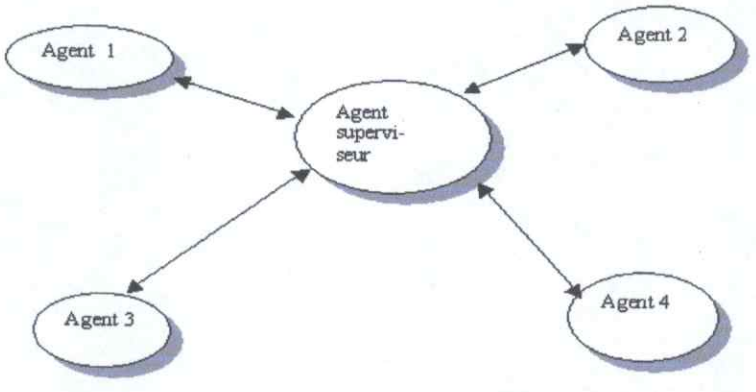


Figure III.5 : mode coopération (hiérarchie entre les agents)

➢ Cas où Il n'existe pas de hiérarchie entre les agents (figure III.6):

1- Coopération par partage de taches:

Le problème est distribué entre les différents agents. Les agents travaillent indépendamment les uns des autres. Chaque agent dispose de ressources et de compétences nécessaires pour accomplir la tache qui a été assignée. Le contrôle est dirigé par les buts, et les agents sont représentés par les taches qu'ils se sont engagés à exécuter.

2- Coopération par partage de résultats:

Les agents ne peuvent accomplir leurs taches de manière indépendante. Ils sont appelés à se transmettre des résultats partiels. Le contrôle est dirigé par les données, les agents sont représentés par des sources de connaissances; la problématique réside dans la communication des résultats.

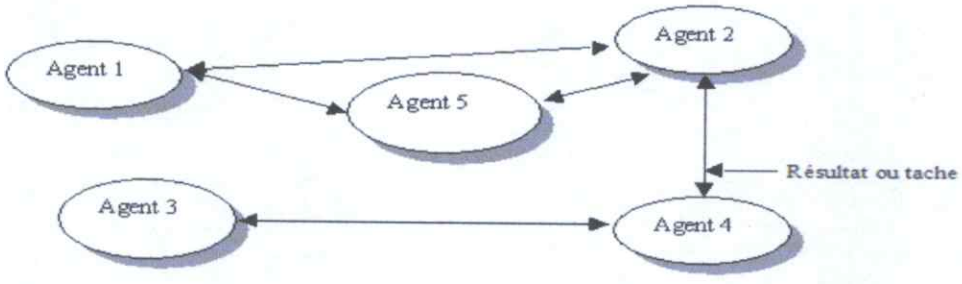


Figure III.6 : coopération par partage (pas de hiérarchie entre les agents)

III.3.3. Les approches de coopérations :

La coopération entre agents peut se faire suivant plusieurs approches: la négociation, la planification, et la coopération fonctionnellement exacte.

III.3.3.1. La planification: nous distinguons deux approches de planification: la planification centralisée et la planification distribuée (CAM & ART & STE 93)(KHO 94)(BAU 92).

III.3.3.1.1. La planification centralisée : Elle suppose une vue globale du problème. Un agent central gère les conflits entre les agents et établit un plan pour l'ensemble des agents. Le plan spécifie les actions que doivent effectuer les différents agents. La planification centralisée utilise trois types de plans pour coordonner la coopération entre agents:

- plan local: c'est le plan généré et suivi localement par un agent individuel.
- Plan partiel global: il représente la stratégie adoptée par le groupe pour atteindre un but global. Il contient des informations sur le but global, les étapes principales vers ce but, ainsi que les moyens utilisés pour intégrer les solutions partielles générées par les différents agents.
- Plan synthétique: c'est une représentation synthétique du plan local, transmise au sein du groupe d'agents à des fins de coordination.

III.3.4. La négociation :

Généralement, les agents coopèrent pour arriver à résoudre un problème donné. Dans certaines situations, cette coopération n'est plus possible. En effet, les agents

peuvent avoir des buts communs ou poursuivre des buts individuels et sont amenés à se partager des données communes et des ressources en nombre limité. Ceci peut entrainer des conflits de ressources, la génération de solutions concurrentes pour un même problème, etc.

Ils peuvent prendre également des décisions de contrôle conflictuelles (décision sur tâche à exécuter). Les agents sont amenés donc à détecter ces situations de conflits, à les éviter et les résoudre. Le processus qui permet aux agents de résoudre leurs conflits s'appelle **négociation.**

Il existe plusieurs méthodes de négociation (CAR 84) (CHE 92)(KHO 94): négociation centralisée, négociation distribué et la négociation pour l'allocation de tâches, appelé aussi réseau contractuel

III.3.4.1. Négociation centralisée : le processus de négociation centralisée suppose que l'agent superviseur dispose d'une vue globale du problème. Il détecte les conflits entre les agents grâce à sa vue globale du problème, et il prend en charge la résolution de ces conflits entre les agents.

La négociation suppose l'existence d'un agent arbitre qui reçoit les buts en conflits des différent agents.

III.3.4.2. Négociation distribuée: dans la négociation distribuée, le processus de détection et de résolution de conflits est totalement distribué entre les agents. Chaque agent peut détecter les situations de conflits à partir des informations qu'il reçoit des autres agents. Les stratégies de résolution de conflits peuvent être communes locales aux agents.

Dans ce type de négociation le protocole de négociation à étapes multiples est destiné à la détection et la résolution, dans un environnement distribué, des conflits entre les agents.

III.3.4.3. Le réseau contractuel : le réseau contractuel est une technique d'allocation de tâches dédiée à la résolution distribuée de problèmes (FER 89) (KHO 94). Il s'agit d'un ensemble d'agents qui peuvent passer des contrats publics.

IV. LES PLATES-FORMES MULTI-AGENTS [Web15] :

Il existe actuellement de nombreuses plates-formes multi-agents qu'on peut classer en plusieurs catégories.

- Les plates-formes pour agents mobiles (Voyager, Odissey, Aglet, etc.) qui fournissent la mobilité à des agents.
- Les plates-formes pour agents cognitifs (AgentBuilder, etc.) dans lesquels on trouve les plates-formes FIPA compliant (Jade, FIPA-OS, etc.) ou KQML compliant (JAT, JAT-Lite, etc.).

- Les plates-formes pour agents collaboratifs (Zeus, JAFMAS, KAoS, JAFIMA, etc.).
- Les plates-formes pour la simulation multi-agents (Cormas, Swarm, etc.)

Toutes ces plates-formes sont dédiées soit à un certain type d'agents (agents à base de règles pour AgentBuilder ou agents collaboratifs pour Zeus par exemple) soit à un certain type d'applications (simulation, mobilité). Il n'existe pas à notre connaissance de plate-forme générique qui permette de réaliser des applications dans tous ces domaines.

Dans les sections suivantes, nous allons présenter en détail quatre plates-formes multi-agents qui sont : AgentBuilder, MadKit, Jade et Zeus.

- **AgentBuilder :**

  AgentBuilder est un environnement de développement complet. Une modélisation orientée-objets avec OMT constitue la base de la conception des systèmes à laquelle on ajoute une partie «ontologie». L'élaboration des comportements des agents se fait à partir du modèle BDI et du langage AGENT-0. KQML est utilisé comme langage de communication entre les agents. L'exécution du système se fait à partir du moteur d'exécution d'AgentBuilder. Par contre, on peut créer des fichiers « .class » et les exécuter sur une JVM standard. AgentBuilder est un outil complexe qui demande des efforts d'apprentissage importants et de bonnes connaissances dans le domaine des systèmes multi-agents pour utiliser de façon performante. Il est limité au niveau de l'extensibilité, de déploiement et de la réutilisabilité.

- **MadKit:**

  MadKit est un environnement basé sur la méthodologie *Aalaadin* ou *AGR (Agent/Groupe/Rôle)*. L'outil fournit un éditeur permettant le déploiement et la gestion des SMA .la gestion faite via cet éditeur offre plusieurs possibilités intéressantes. L'outil offre aussi un utilitaire pour effectuer des simulations.

- **Jade:**

  Jade est un outil qui répond aux normes FIPA97. Aucune méthodologie n'est spécifiée pour le développement. Jade fournit des classes qui implémentent « JESS » pour la définition du comportement des agents. L'outil possède trois modules principaux (nécessaire aux normes FIPA). Le DF « Director Facilitor » fournit un service de pages jaunes à la plate-forme. Le ACC « Agent Communication Channel » gère la communication entre les agents. Le AMS « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et utilisation du système. Les agents communiquent par le langage FIPA ACL. Un éditeur est disponible pour l'enregistrement et la gestion des agents. Aucune autre interface n'est disponible pour le développement ou l'implémentation. A cause de cette lacune, l'implémentation demande beaucoup d'efforts. Il nécessite une bonne connaissance des classes et des différents services offerts.

- **Zeus:**

Zeus est un environnement complet qui utilise une méthodologie appelée « Role Modeling » pour le développement de systèmes collaboratifs. Les agents possèdent trois couches. La première couche est celle de la définition où l'agent est vu comme une entité autonome capable de raisonner en terme de ses croyances, ses ressources et ses préférences. La seconde couche est celle de l'organisation. Dans celle-ci, on décide des modes de communications entre les agents, protocoles, coordination et autres mécanismes d'interaction. La troisième couche est la couche de coordination, chaque agent est considéré comme une entité sociale c'est-à-dire en terme ses compétences de négociation et de coordination. L'outil est un des plus complets. Les différentes étapes de développement se font à l'intérieur de plusieurs éditeurs : Ontologie, description des tâches, organisation, définition des agents, coordination, faits et variables ainsi que les contraintes. Le développement de SMA avec Zeus est cependant conditionnel à l'utilisation de l'approche « Role Modeling ». L'outil est assez complexe et sa maîtrise nécessite beaucoup de temps.

Rapport entre ontologie et SMA :

Nous allons voir maintenant la combinaison entre les SMA et les ontologies qui pourraient s'avérer très prometteuse.

Les agents d'un SMA doivent communiquer entre eux et se comprendre mutuellement, le vocabulaire employé dans les messages envoyé entre eux, peut être défini par une ontologie partagée par les agents du système.

Une ontologie pourrait éventuellement fournir un vocabulaire structuré d'un domaine servant de support à l'expression des requêtes. Par ailleurs, elle permet l'intégration par exemple de différentes sources d'information accessible donnant ainsi une vue homogène des données manipulées par le système.

À l'aide d'une ontologie on pourra décrire : la notion d'agent, les relations entre les agents, les modes de communications et surtout la description des compétences d'un agent c'est-a-dire la méta-description des services.

CONCLUSION :

Dans les systèmes multi-agents la connaissance et le contrôle sont distribués entre les agents qui coopèrent à la réalisation d'un même but.

Les systèmes multi-agents offrent beaucoup d'avantages : la distribution des traitements et des connaissances, l'adaptabilité, la réduction du coût et de la complexité du problème, la facilité de développement de maintenance, les traitements parallèle, le gain en efficacité et en temps de traitement.

Le concept multi-agents est un thème de recherche en cours d'exploration qui fait intervenir plusieurs domaines de recherche, tels que : les systèmes répartis, la biologie, l'intelligence artificielle, la psychologie cognitive, la sociologie, ...

CHAPITRE IV

CONCEPTION DU SYSTEME

CONCEPTION DU SYSTEME

INTRODUCTION

La Recherche d'Information (RI) peut être définie comme une activité dont la finalité est de localiser et de délivrer des granules documentaires à un utilisateur en fonction de son besoin en information.

Les outils disponibles de recherche d'information (RI) sur le web souffrent d'une approche généraliste ne prenant pas en compte les caractéristiques de l'utilisateur. Cette approche généralisée confrontée à une prolifération massive de ressources d'informations hétérogènes limite la qualité des résultats retournés par le système. C'est pourquoi notre projet a pour but d'essayer de palier à cette lacune, en construisant une ontologie médicale qui sera la base sémantique de la recherche, celle-ci est basée sur une architecture multi agents que nous réaliserons.

Cette ontologie sera exploitée par un système de recherche intitulé M.S.E afin d'essayer d'améliorer la pertinence de la recherche en diminuant le bruit et le silence.

Ci dessous le schéma général de notre système :

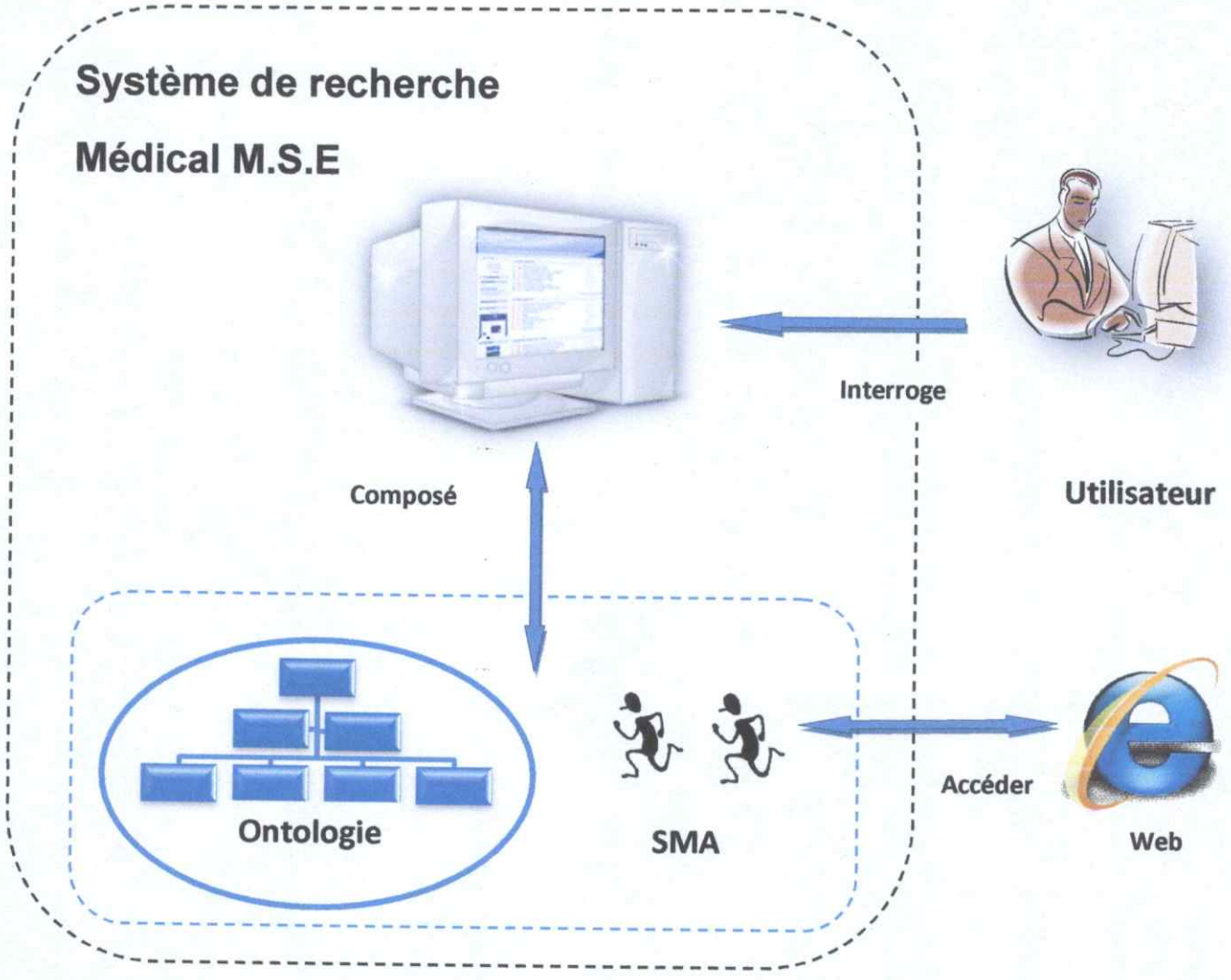


Figure IV.1 : schéma général du système

Notre système de recherche se compose de deux éléments à savoir l'ontologie, le système multi-agents.

Ontologie : L'objectif de notre ontologie est de modéliser un ensemble de connaissances dans un domaine donné.

Dans notre cas, le domaine traité est la médecine particulièrement l'ophtalmologie et la cancérologie.

Pour plus d'information à propos d'ontologie médicale se référer à l'ANNEXE G.

Nous avons construit une ontologie à base de contenu et d'usages médicaux, celle-ci traites les concepts anatomiques de l'ophtalmologie et quelques usages médicaux que nous avons définit, exemple : traitement, symptôme, pathologie…

Système multi agents : est un système distribué composé d'un ensemble d'agents, chaque agent à des informations ou des capacités de résolution de problèmes limités (ainsi chaque agent a un point de vu partiel). [4]

Notre système multi agent se compose de deux agents chacun d'eux ayant un rôle précis et qui travaillent en collaboration.

Les agents de notre système sont :

- « Agent ontologie »,
- « Agent interface ».

La conception de notre système se devise en deux parties, la 1ere est consacrée à la conception de l'ontologie, et la seconde partie a la conception de notre système globale en étalant la collaboration entre les agents.

PARTIE 1 : Conception de l'ontologie :

De nombreuses méthodologies pour la construction d'ontologie ont été proposées par la communauté de l'ingénierie des connaissances et de l'intelligence artificielle. Démontrer la meilleure méthodologie reste une tache relativement difficile néanmoins, l'étude comparative présentée par Fernandez Lopez dans (Fernandez-Lopez 99) permet de voir qu'une méthodologie peut se distinguer en termes d'autonomie face au domaine d'application et de couverture du cycle de vie d'une ontologie.

Dans notre cas, nous avons choisi METHONTOLOGY (*cf. ANNEXE F*) pour la construction de notre ontologie intitulée « OntoMed » vu que cette méthodologie propose une approche couvrant tout le cycle de vie de l'ontologie, elle est supportée par plusieurs outils.

Elle est aussi utilisée pour la construction d'ontologie à partir de zéro, aussi bien que pour la réutilisation ou la réingénierie d'ontologie existantes (Gomez-Perez et al 02)

I. Description de la méthodologie METHONTOLOGY [14] :

Cette méthodologie a été développée au Laboratoire d'Intelligence Artificielle de l'Université polytechnique de Madrid. Elle vise la construction d'ontologies au "niveau connaissance" et possède ses racines dans les activités principales identifiées par le procédé de développement de logiciel d'IEEE et dans d'autres méthodologies d'ingénierie des connaissances, elle repose sur :

- un processus de développement d'ontologies comportant des activités de gestion de projet (planification, assurance qualité), des activités orientées développement (spécification, conceptualisation, formalisation, implémentation) et des activités de support (Acquisition, intégration, évaluation, documentation).

METHONTOLOGY propose de commencer avec l'activité de planification, afin d'identifier l'ensemble des tâches à réaliser pendant le processus de développement de l'ontologie , leur organisation, et le temps et les ressources nécessaires , une fois les tâches planifiées , le développement d'ontologie qui sera réalisé suivant les quatre étapes cités dans l'activité orienté développement , commence en même temps que les tâches de gestion de projet et les taches de support.

Pendant tout le cycle de vie de l'ontologie, les tâches de gestion de projet et de support seront réalisées en parallèle aux activités de développement de l'ontologie.

METHONTOLOGY a été proposé pour la construction d'ontologie par FIPA (the Foundation for Intelligent Physical Agents), ce qui favorise l'interopérabilité à travers des applications basées sur agent.

Ci-dessous est présenté le cycle de vie de METHONTOLOGY :

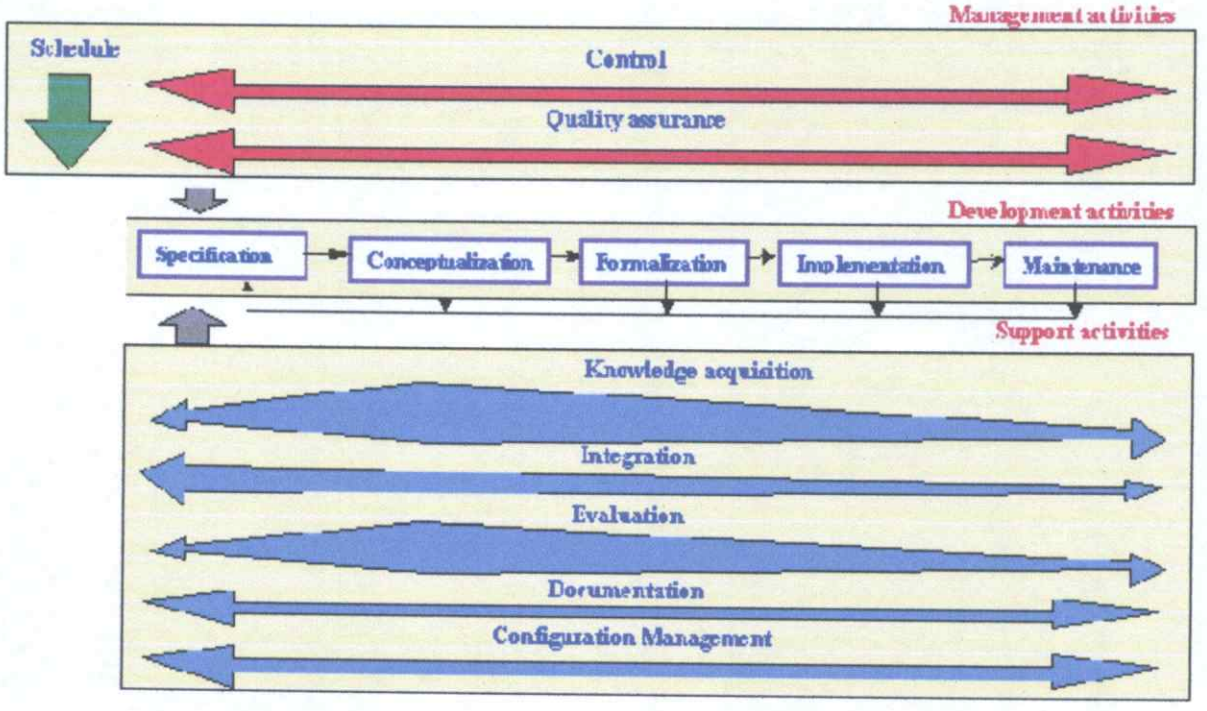


Figure IV.2 : Cycle de vie de METHONTOLOGY

II. Construction de l'ontologie

Notre projet consiste en la construction d'une ontologie à base de contenu et d'usage médical. Cette ontologie contient les deux notions suivantes :

Contenu : englobe les informations générales d'un domaine médical, dans notre cas ce serait des concepts anatomiques, cette notion de contenu permet à l'utilisateur d'élargir sa recherche, il aura plus de liberté dans la formulation de sa requête.

Usage : celui-ci à pour but mieux cerner la recherche de l'utilisateur, de connaitre son but, en lui proposant quelques usages médicaux les plus fréquents.

Ainsi en combinant cette notion de contenu et d'usage, nous essayerons de raffiner au mieux les résultats de la recherche.

On a préféré augmenter la portée des spécialités médicales pour mieux mettre en valeur la notion d'usage, ainsi nous aurions pu nous focaliser sur une seule spécialité, mais la notion d'usage n'aurait pas eu d'impact. L'augmentation a aussi été faite dans le but d'obtenir une ontologie plus riche.

Nous allons montrer l'importance de cette notion par des exemples :

Si on construisait une ontologie médicale exemple de cancérologie, qui contiendrait tout les concepts concernant ce domaine, lorsqu'une recherche est effectuée, sans

préciser l'usage de cette recherche, la pertinence des résultats n'est pas très approuvable.

Exemple : Rechercher « cancer du poumon »

Le résultat de cette recherche sera tout ce qui concerne le cancer du poumon partant de sa définition à son traitement, et peut être d'autres résultats.

L'utilisateur est immergé dans un grand nombre de résultats, il sera contraint à parcourir tout les résultats, une tâche fastidieuse à effectuer.

Pour y remédier, on inclut la notion d'usage à notre ontologie d'une part, qui comme définit au part avant fournira une aide à l'utilisateur et ceci en lui proposant quelques usages médicaux que nous définissons dans notre ontologie, ainsi la recherche sera plus raffinée.

Si on reformule notre recherche « cancer du poumon » en indiquant l'usage parmi ceux proposés (définition, symptôme, traitement ...), les résultats de la recherche concerneront uniquement l'usage sélectionné, et du coup il y aurait une diminution des résultats non pertinents.

Voici comment se présente la recherche dans l'ontologie :

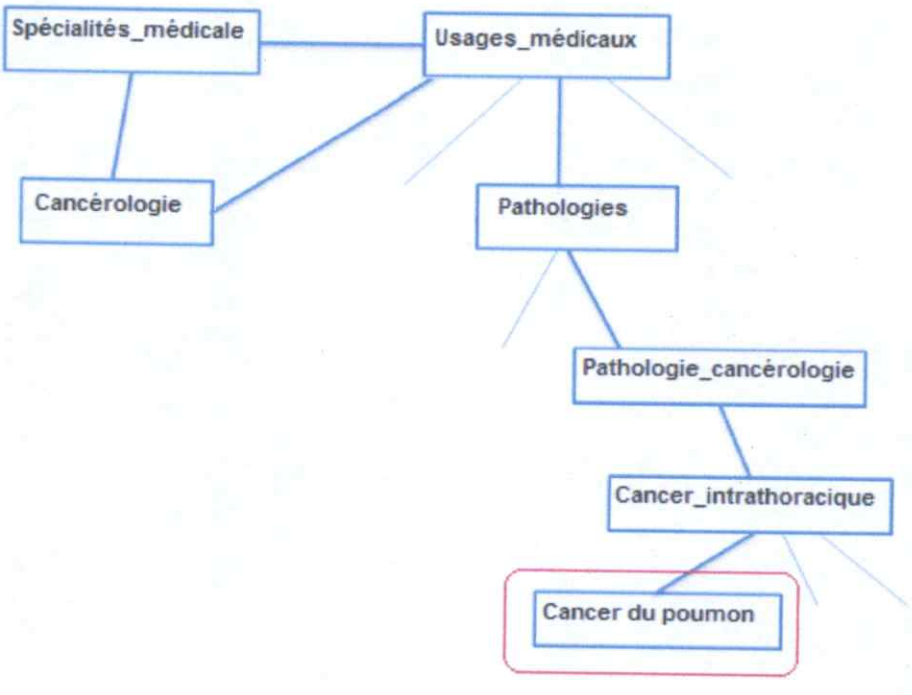


Figure IV.3 : Rechercher « cancer du poumon »

II.1. Activité de gestion de projet

II.1.1. La gestion de projet

Comme tout autre projet, la construction d'ontologie fait appel à de grandes équipes multidisciplinaires afin d'assurer un certain niveau de savoir faire et d'acceptation. Ainsi, la conduite de projet doit faire l'objet d'une grande attention pour gérer et coordonner les efforts de chaque membre de l'équipe.

La gestion de notre projet s'est faite implicitement, sans aucune attention particulière, vu que notre équipe ne dénombre que deux personnes, et le manque considérable d'experts dans le domaine de l'ingénierie ontologique ; ce qui nous a poussés à ne prendre en compte que l'activité de planification.

II.1.2. Planification

La construction d'ontologie est un projet qui doit être planifié et qui doit suivre un plan de travail bien précis. Ce qui nous a amené à identifier les tâches à réaliser et le temps nécessaire pour chaque tache. Les principales tâches de notre projet sont :

- choisir le domaine à modéliser par l'ontologie.
- collecter toute la documentation disponible du domaine.
- chercher une assistance auprès des experts du domaine.
- travailler en collaboration avec les experts du domaine.
- construire l'ontologie.
- valider l'ontologie.

La durée de réalisation de chaque tâche est plus ou moins importante. Néanmoins, nous avons estimé que le temps passé à la collecte d'information et de document, ainsi que la construction d'ontologie prendra plus de temps que les autres taches.

II.2. Activités orientées développement

II.2.1. Spécification

Il s'agit, dans cette étape, de constituer un document qui contient toutes les informations sur les raisons de construction de l'ontologie, ses utilisations prévues et ses utilisateurs potentiels.

Nous avons opté pour la construction d'une ontologie médicale, qui traite deux spécialités médicales : ophtalmologie, cancérologie. Ces dernières ont été choisies

sans aucune raison particulière. Nous avons jugé que ces spécialités relèvent les maladies les plus courantes à notre époque.

Nous avons également fixé le degré de formalisation et de granularité de l'ontologie.

- Le degré de formalisation du langage est formel puisque l'ontologie est employée par un outil logiciel dans notre cas (moteur de recherche) et aussi des agents intelligents, donc la sémantique de l'ontologie est beaucoup plus précise.

- La granularité est fine puisque l'ontologie devra capturer le plus grand nombre de connaissances.

L'ontologie contient tous les concepts d'usage et de contenu relatifs aux spécialités citées au par avant.

Nous avons respecté la terminologie francophone des concepts médicaux.

Pour la construction de l'ontologie, il a été question de faire recours à différentes sources d'information toutes intéressantes les unes que les autres, nous citons : les **ressources web**, les écrits (livres, articles, dictionnaire) ainsi que l'**aide précieuse des experts du domaine.**

Sur internet, nous avons recueillit des sites concernant les spécialités vu précédemment, nous citons quelque uns :

http://www.ophtalmologie.fr

http://www.centreoscarlambret.fr/frameset.asp

http://www.univ-st-etienne.fr/saintoph/finit/dcem4.htm

Pour les écrits nous citons quelque livre d'ophtalmologie « Diagnostique en ophtalmologie auteur : ANDRE VERGEZ » et « Guide pratique aux urgences en ophtalmologie par : Bruno Lumbroso » ;

Des séances de travail ont été organisées avec des experts du domaine de chaque spécialité exerçant dans des cabinets privé, polyclinique.

Finalement l'ontologie est destinée à être utilisée dans un système de recherche d'information à partir de ressources web basées sur une architecture multi-agents.

Les utilisateurs potentiels de ce système pourraient être familiers au domaine médical plus précisément dans les spécialités citées. Nous pouvons les définir comme étant des médecins ou étudiants , patients ou autre personne intéressée par le domaine médical.

Ci-dessous le document contenant les principales informations de l'ontologie.

Domaine : Médical

Date : 26 Mars 2007

Développée par : Hakima Mellah, Assia Abderrahim et Ilhem Nouas

Niveau de formalité : Formelle

Portée : spécialités: cancérologie, ophtalmologie

Liste des concepts : - usage : symptômes, traitements, pathologies, ...

-Contenu : Concepts anatomiques de l'ophtalmologie

Objectif : construire une ontologie opérationnelle conforme aux principes de l'ingénierie ontologique

Utilisation prévue : l'ontologie est destinée à être utilisée par un système de recherche d'information à partir de ressources web basé sur une architecture multi-agents.

Utilisateurs potentiels : étudiants, médecins, professeur,.....

Source des connaissances :

1- Site internet :

http://www.ophtalmologie.fr

http://www.centreoscarlambret.fr/frameset.asp

http://www.univ-st-etienne.fr/saintoph/finit/dcem4.htm

2- interview avec les experts du domaine.

3- livres et articles.

II.2.2. Conceptualisation

La conceptualisation consiste à la structuration et l'organisation des données collectées lors de l'étape d'acquisition des connaissances dans le but de créer une première représentation du domaine ou le premier modèle conceptuel.

Plus précisément, une vue informelle du domaine est organisée et structurée en utilisant un ensemble de représentations intermédiaires que l'expert du domaine et le développeur de l'ontologie peuvent comprendre.

Ainsi METHONTOLOGY propose d'utiliser ces représentations intermédiaires basées sur des notations graphiques et tabulaires, pour obtenir un modèle conceptuel.

La conceptualisation se base sur les étapes illustrées dans la figure suivante **[15]:**

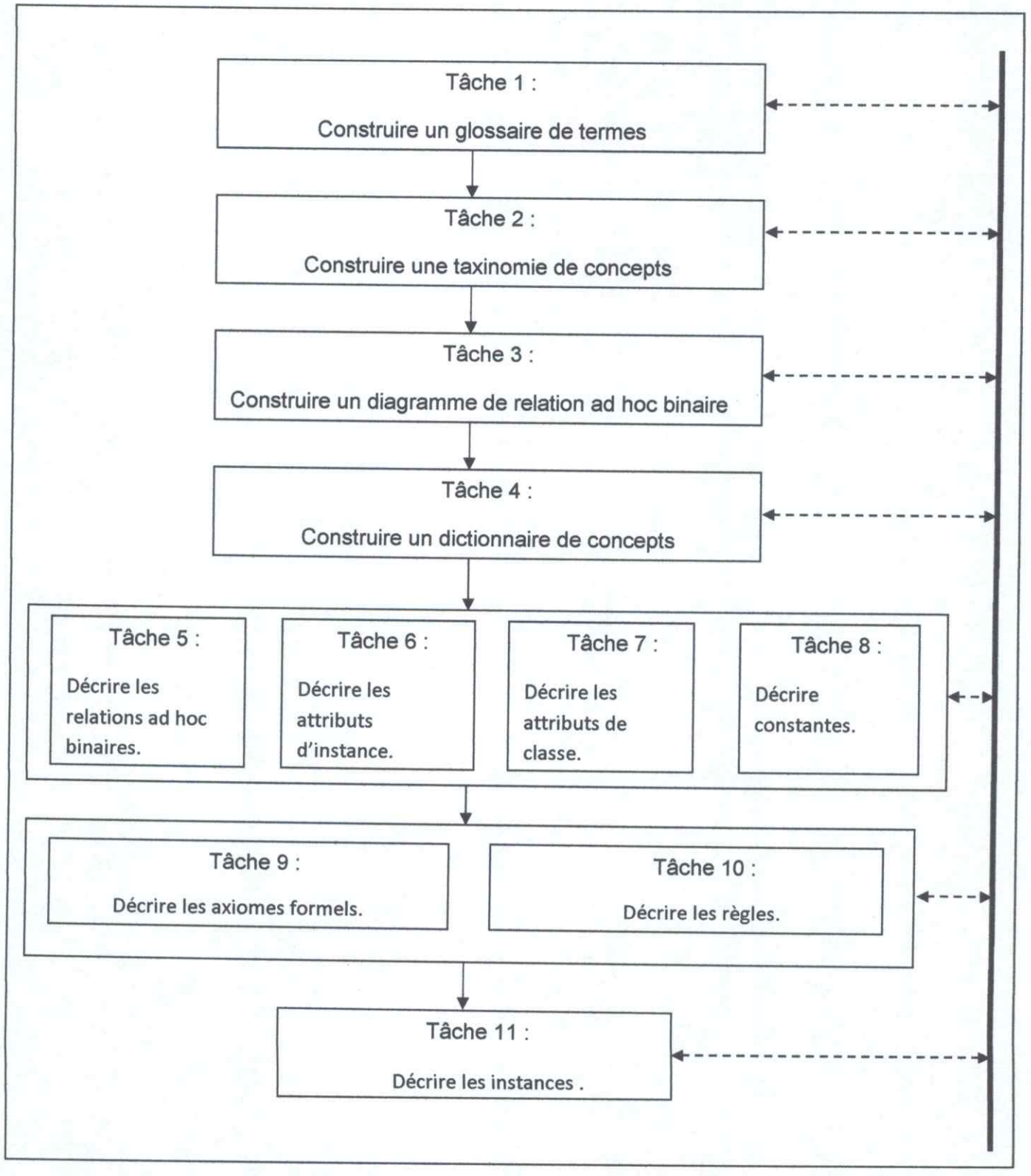


Figure IV.4: Tâches de l'activité de la conceptualisation dans METHONTOLOGY

II.2.2.1.Construction d'un glossaire de termes

Le glossaire a été construit à partir de sources de connaissances acquises, dont on a extrait le maximum de termes sans se préoccuper de leur structuration.
On aura comme résultat un tableau qui contiendra tous les termes relatifs à chaque spécialité et qui seront par la suite raffinés auprès des experts du domaine.
Le tableau ci-dessous représente un extrait du glossaire.

Tableau IV.1 : Extrait du glossaire

Nom	Synonyme	Acronyme	Description	Type
carcinome épidermoïde				Classe
carcinome basocellulaire				Classe
adénocarcinome				Classe
syndrome lymphoprolifératif	lymphome			Classe
leucémie				Classe
syndrome myéloprolifératif				Classe
myélome				Classe
mésothéliome				Classe
séminome				Classe
mélanome				Classe
gliome malin				Classe
sarcome				Classe
ostéosarcome				Classe
chondrosarcome				Classe
angiosarcome				Classe
Tumeurs du système nerveux				Classe
Tumeur cérébrale primitives				Classe
Métastases cérébrales				Classe
Tumeurs ORL				Classe
Cancer de la gorge	cancer ORL			Classe
Cancer du poumon				Classe
Segment antérieur				Classe
Kératocône				Classe
Conjonctivite				Classe
Kératite				Classe
Uvéite antérieure				Classe
Cataracte				Classe
l'irritation de l'œil				Classe
l'œil rouge				Classe
cligne de l'œil				Classe
abaisse l'acuité visuelle				Classe
Dégénérescence maculaire liée à l'âge				Classe
Rétinopathie diabétique				Classe
Trou maculaire				Classe
Maladie de Best				Classe

Orgelet				Classe
glaucome				Classe
Rétinopathie pigmentaire				Classe
myopie				Classe
astigmatisme				Classe

II.2.2.2. Construire une taxonomie de concept

Il s'agit de la hiérarchisation, sous forme d'une taxonomie, les concepts recensés dans le glossaire.

METHONTOLOGY propose d'employer les quatre relations taxonomiques définies dans « *the Frame ontology* » et « *the OKBC Ontology* » :

-SubClass-Of
-Disjoint-Decomposition
-Exhaustive-Decomposition
-Partition

SubClass-Of : Un concept C1 est une sous classe (SubClass-Of) d'un concept C2, si et seulement si, toutes les instances de C1 sont aussi des instances de C2 , prenons par exemple dans notre cas : le concept «Cancerologie » est une sous classe du concept « spécialités_médicales » .

Voici un exemple de relation SubClass-of de notre ontologie :

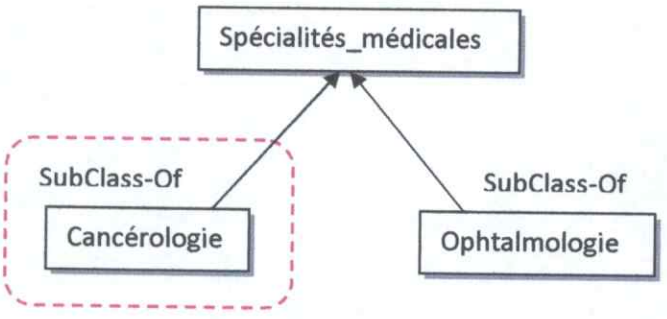


Figure IV.5 : Relation de subClass-of dans notre ontologie

Disjoint-Decomposition : une décomposition disjointe d'un concept C, est un ensemble de sous classes de C, qui n'ont pas d'instances en commun. Prenons par exemple, dans notre cas : « Definition_cancérologie » et Definition_ophtalmologie » sont des sous classes disjointes du concept « Définition », car il n'existe pas une définition qui peut être une définition concernant la cancérologie et l'ophtalmologie en même temps. Voici un exemple de concept disjoint :

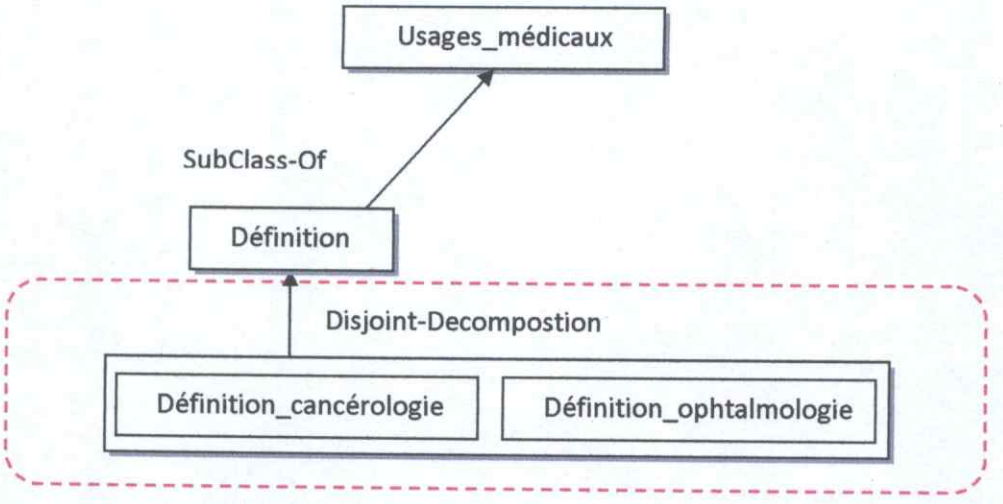


Figure IV.6: Relation Disjoint-Decomposition dans notre ontologie

Exhaustive-Decomposition : une décomposition exhaustive d'un concept C est un ensemble de sous classes de C qui peuvent avoir des instances en commun.

Dans notre cas, les concepts « cancer_du_côlon », « cancer_de_estomac », « cancer_du_foie », « cancer_du_pancréas », « cancer_du_rectum », « cancer_du_oesophage », forment une décomposition exhaustive du concept « cancer_digestifs », car tout cancer digestifs est forcement au moins un de ces concepts.

Voici un exemple de la décomposition exhaustive du concept « cancer_digestifs » :

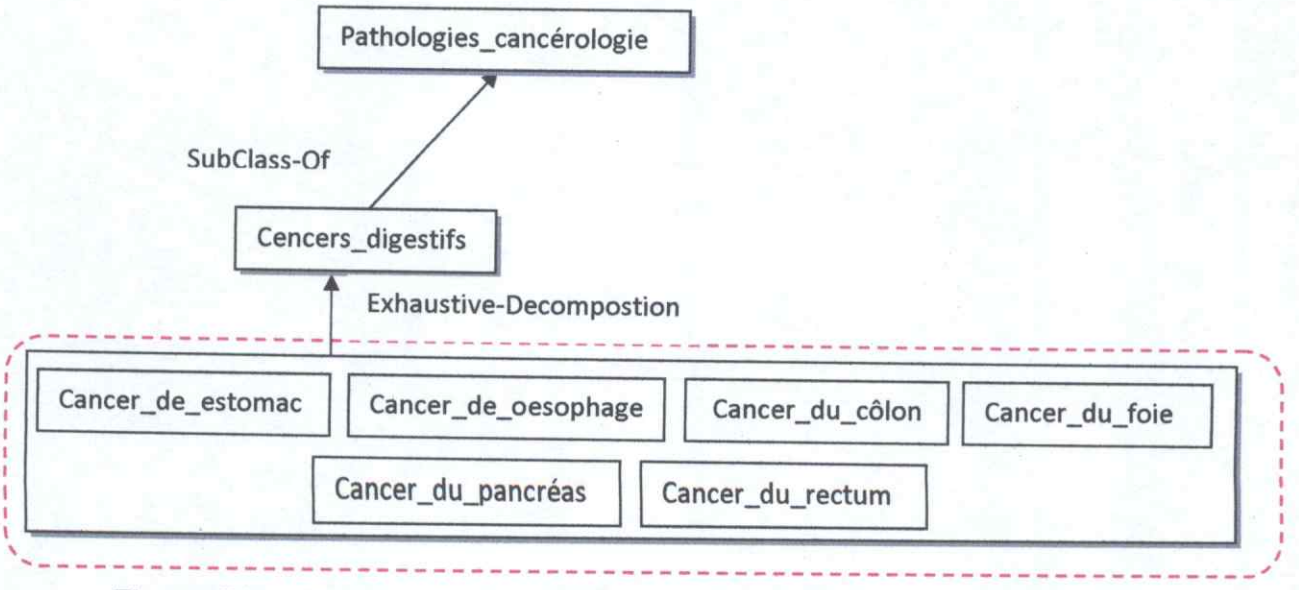


Figure IV.7 : Relation Exhaustive-Decomposition dans notre ontologie

Partition : une partition d'un concept C est un ensemble de sous classes de C, qui n'ont pas d'instance en commun et qui couvrent C (exemple : les concepts « pair » et « impair » forment une partition du concept « entier naturel », car tout entier naturel est, ou bien, pair ou impair).

Dans notre cas il n'existe pas de partition de concept.

Après avoir énuméré les termes du domaine sous forme d'une liste, nous avons adopté un procédé de développement **de haut en bas (TOP-LEVEL)** qui commence par une définition des concepts les plus généraux du domaine et se poursuit par la spécialisation des concepts. **[13]**

Par exemple nous pouvons commencer à créer des classes pour les concepts généraux dans notre cas « Spécialités_médicales et Usage_médicaux », puis nous spécialiserons la classe « Spécialités_médicales » en créant quelques unes de ses sous-classes : Cancérologie, Ophtalmologie. Nous pouvons en outre catégoriser la classe pathologie cancer, par exemple, en Cancer_intra_thoracique, Cancers_digestifs, Cancers_gynécologiques et ainsi de suite.
Dans la figure IV.8 est présentée la spécialisation de « spécialités_ médicales »

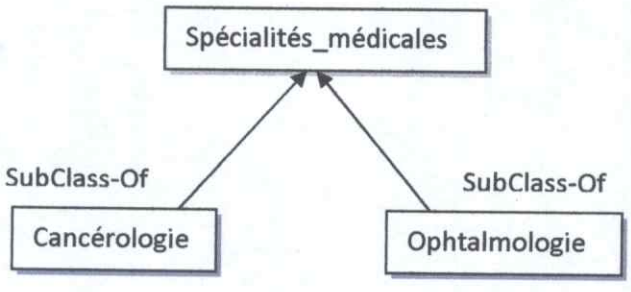


Figure IV.8 : Exemple de spécialisation de « Spécialités_médicales »

L'utilisation de notion de « contenu » dans notre ontologie, qui comme définit au par avant, élargira la recherche de l'utilisateur et lui laissera une certaine liberté dans sa recherche. Compte tenu du temps insuffisant pour le développement de toute la notion de contenu concernant les deux spécialités médicales et compte tenu aussi de la complexité du domaine à modéliser (le domaine médical), nous avons fusionné a notre ontologie, une partie d'une ontologie d'ophtalmologie déjà existante qui traite les concepts anatomiques de l'ophtalmologie et ce afin de traiter l'aspect « contenu » de notre ontologie.

II.2.2.3.Construire un diagramme de relation ad hoc binaires :

Ce digramme consiste à expliciter toutes les relations pouvant exister reliant les concepts d'une ou plusieurs ontologies, qui devraient être relié par au moins une relation.
Dans notre cas, on a relié une partie de l'ontologie d'ophtalmologie à la notre, via la relation *SubClass-Of*, entre le concept « Spécialités_médicales » et la racine de l'ontologie ophtalmologie, qui est représentée par le concept « Ophtalmologie », ceci afin de représenter la notion de contenu qui sera définie par les concepts anatomiques.

Les relations les plus importantes dans une ontologie médicale décrivant l'anatomie d'un organe précis est « composé- composant », il peut exister d'autre relation pouvant lier les concepts du domaine médical.

La figure ci-dessous, présente un extrait du diagramme des relations ad hoc binaire de notre ontologie, avec les relations :
Pathologie_avoir_symptômes, Spécialiés_avoir_usages et leurs inverse :
Usage_avoir_spécialités , Symptôme_de_pathologie.

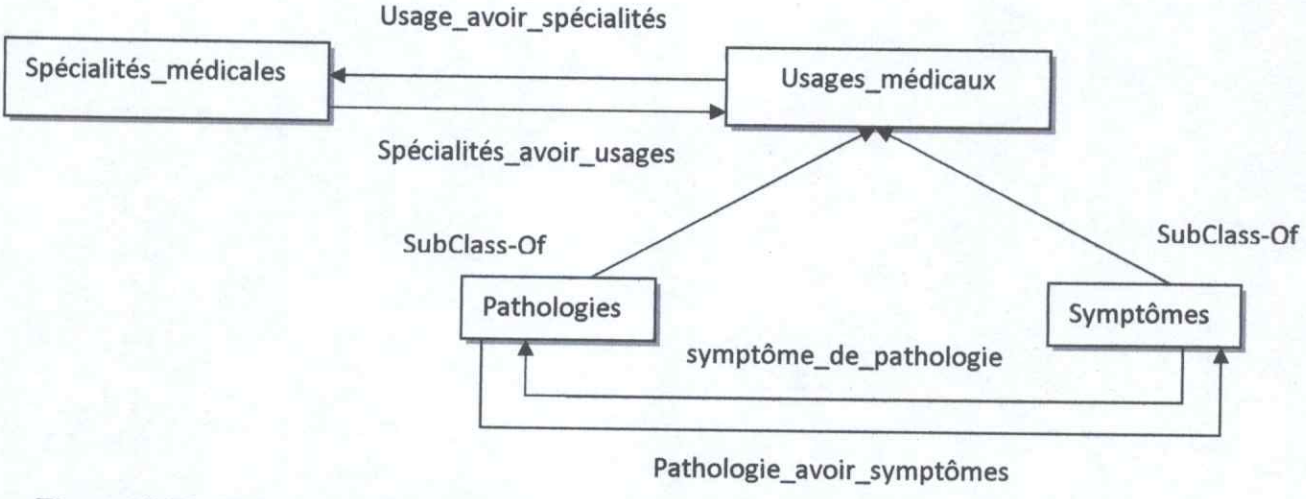


Figure IV.9: Extrait du diagramme de relation ad hoc binaire de notre ontologie

II.2.2.4.Construire le dictionnaire de concepts :

Le dictionnaire de concepts représente les concepts du domaine étudié, présenter les relations, les attributs d'instance, de classe.

Nous avons spécifié une partie du dictionnaire par le tableau ci-dessous :

Tableau IV.2: Extrait du dictionnaire de concepts

NOM	ATTRIBUTS D'INSTANCE	ATTRIBUTS DE CLASSE	RELATION
Usages_médicaux	(vide)	Attributs propres à la classe+ Synonymes	Usages_avoir_spécialités
Examen_clinique	(vide)	Attributs propres à la classe+ Synonymes	Examen_pathologie
Pathologies	(vide)	Attributs propres à la classe+ Synonymes	Pathologie_avoir_définition Pathologie_examinée_par Pathologie_avoir_traitement Pathologie_avoir_symptôme
Symptômes	(vide)	Attributs propres à la classe+ Synonymes	symptôme_de_pathologie
Traitement	(vide)	Attributs propres à la classe+ Synonymes	Traitement_de_pathologie
Spécialités_médicales	(vide)	Attributs propres à la classe+ Synonymes	Spécialité_avoir_usages Spécialité_avoir_définition

II.2.2.5.Décrire les relations ad hoc binaires en détail :

Cette étape consiste à expliciter l'ensemble de relations recensées dans le dictionnaire des concepts et ceci en le représentant dans un tableau qui contient le nom de la relation, ses concepts sources et cibles, sa cardinalité ainsi que sa relation inverse.

Tableau IV.3 : Extrait du tableau de relations binaires

NOM RELATION	CONCEPTS SOURCES	CARDINALITE	CONCEPT CIBLE	RELATION INVERSE
Spécialité_avoir_usages	Spécialités	N	Usages	Usages_avoir_spécialités
Actualité_de_spécialité	Actualités_médicales	N	Spécialités_médicales	Spécialité_avoir_actualités
Pathologie_avoir_définition	Pathologie	1	Définition	Définition_de_pathologie
Pathologie_examinée_par	Pathologie	N	Examen_Clinique	Examen_pathologie
symptôme_de_pathologie	Symptôme	N	Pathologie	pathologie_avoir_symptôme
Traitement_de_pathologie	Traitement	N	Pathologie	Pathologie_avoir_traitement

II.2.2.6.Décrire les attributs d'instance en détail :

Dans notre ontologie, on n'a pas défini les attributs d'instance, donc cette étape ne compte pas.

II.2.2.7.Décrire les attributs de classe en détail :

Nous avons déjà sélectionné des classes à partir de la liste des termes que nous avons créés dans l'Étape 1, la plupart des termes restants ont de fortes chances d'être des propriétés de ces classes.
Pour chaque propriété dans la liste nous devons déterminer la classe qu'elle décrit. Ces propriétés deviennent des attributs rattachés aux classes. En général, certains types de propriétés d'objets peuvent devenir des attributs dans une ontologie.
Ainsi dans notre ontologie nous allons décrire les attributs des classes dans le tableau ci-dessous qui contiendra des informations relatives aux attributs de classes, comme le nom de l'attribut, son concept source, le type de sa valeur, sa cardinalité et sa valeur.

Tableau IV.4 : Extrait du tableau attribut de classes

NOM	CONCEPTS SOURCES	TYPE DE LA VALEUR	CARDINALITE	VALEUR
Avoir_synonymes	Cancérologie	Texte	(1,1)	oncologie
Avoir_synonymes	pathologies	Texte	(1,1)	Maladies
Avoir_synonymes	un_état_de_fatigue_générale	Texte	N	une_faiblesse un_état_de_faiblesse_généralisé
Avoir_synonymes	Infection oculaire	Texte	(1,1)	infection de l'œil

Avoir_synonymes	uvéite antérieure	Texte	N	Iritis iridocyclite
Avoir_synonymes	scotome central	Texte	(1,1)	zone aveugle
Avoir_synonymes	Mouche volante	Texte	N	Myodésopsies Corps flottants

II.2.2.8.Décrire les Constantes :

Dans notre ontologie, il n'y a pas de constantes définies donc cette étape ne compte pas.

II.2.2.9.Décrire les axiomes formels :

Cette étape consiste à décrire les axiomes formel utilisés dans l'ontologie, pour chaque axiome définit, METHONTOLOGY propose d'inclure dans un tableau les informations suivante : Nom de l'axiome, sa description en langage naturel, l'expression logique qui décrit formellement l'axiome en utilisant la logique de premier ordre.

Tableau IV.5 : Extrait du tableau axiomes formels

NOM AXIOME	DESCRIPTION	EXPRESSION LOGIQUE	CONCEPT REFERE	RELATION REFERE	VALEUR
	Toutes les **pathologies de cancérologie** ont pour taitements seulement Traittement_chirurgicale_cancérologie ou Traitement_medicale_cancérologie	∀ Pathologie_avoir_traitement only (Traitement_chirurgicale_cancérologie or Traitement_medicale_cancérologie)	Pathologies_cancérologie	Pathologie_avoir_traitement	
	Tout les **Symptômes ophtalmologie** ont pour pathologies seulement les pathologies de l'ophtalmologie	∀ symptôme_de_pathologie only Pathologies_ophtalmologie	symptômes_ophtalmologie	symptôme_de_pathologie	
	Les **pathologies de la cancérologie** peuvent avoir pour traitements au moins un Traitement_chirurgicale_cancérologie	∃ (Pathologie_avoir_traitement some traitement_chirurgicale_cancérologie)	Pathologies_cancérologie	Pathologie_avoir_traitement	

II.2.2.10.Décrire les règles :
METHONTOLOGY propose d'indiquer des expressions de règle en utilisant le format *if <conditions> then <consequent>*, alors que dans notre ontologie il n'existe pas de règle utilisant ce format, du coup cette étape ne compte pas.

II.2.2.11.Décrire les instances :
Une fois le modèle conceptuel de l'ontologie établit, l'expert ontologique doit définir les instances appropriés qui apparaissent dans le dictionnaire des concepts (glossaire).Dans notre ontologie nous n'avons pas définit d'instances.

II.2.3. Formalisation

Une fois le modèle conceptuel construit, la méthodologie propose de le transformer en un modèle formel qui sera implémenté par la suite dans un langage de représentation des connaissances.

Dans ce processus (formalisation), le modèle conceptuel passe graduellement du niveau des connaissances au niveau de l'implémentation et son degré de formalité augmente progressivement pour devenir finalement un modèle computable (qui peut être exécuté sur une machine).

La formalisation consiste donc en la représentation des connaissances dans un langage formel exemple (UML), les graphes conceptuels, les réseaux sémantiques.

Dans METHONTOLOGY, la formalisation du modèle conceptuel n'est pas obligatoire, car il existe actuellement des outils qui sont capable de traduire le modèle conceptuel dans n'importe quel langage, exemple dans notre cas l'éditeur d'ontologie Protégé permet le passage direct du modèle conceptuel au modèle computable sans générer de modèle formel.

L'étape de formalisation est donc implicite.

II.2.4. Implémentation

Cette étape consiste en la réalisation de l'ontologie, ainsi qu'à choisir les langages et outils pour la réalisation de l'ontologie.

Dans notre cas pour l'implémentation de cette dernière, nous avons choisi l'éditeur d'ontologie intitulé « Protégé », c'est un outil recommandé par les experts de l'ingénierie ontologique, c'est aussi une plate-forme extensible, grâce au système de Plug-ins qui permet de gérer des contenus multimédias, requêter, évaluer et fusionner des ontologies, un exemple de plug-in : Jambalaya qui permet de visionner l'ontologie sous forme de graphe et de changer l'emplacement de ses racines.

II.3.Activités de support

II.3.1.Acquisition des connaissances :

Cette phase consiste à démontrer l'approche suivie dans l'acquisition des connaissances concernant le domaine médicale, dans notre cas : Ophtalmologie et Cancérologie.
Et pour ce faire, nous avons utilisés des techniques issues de l'ingénierie des connaissances ex : interviews formel, informel, analyse de texte...

La première étape consistait à la recherche sur internet des documents, des sites web relatif au domaine médical « ophtalmologie et cancérologie » afin d'y extraire les termes médicaux approprié au domaine.
Notre recherche c'est basés sur la définition de toutes les pathologies concernant l'ophtalmologie et cancérologie, puis nous avons cherché tous les symptômes, traitement, examen clinique de chaque pathologie.

Cette étape est l'étape la plus longue environ (4 mois) et la plus pénible à réaliser, puisqu'il ya eu une recherche des documents médicaux, des termes médicaux, puis une analyse de texte (détection des symptômes, examen clinique ...).

La deuxième étape était d'essayer d'organiser les résultats de la recherche afin de pouvoir établir un dialogue avec les experts du domaine.

La troisième étape était d'organiser des interviews avec les experts du domaine, ces premières entrevues étaient informelles, non structurée, les experts parlaient de leur domaine d'une façon générale, puis dans les interviews qui suivaient, les experts nous fournissaient les informations désirées ainsi que la correction de notre recherche.

II.3.2.Intégration :

Dans l'étape de conceptualisation nous avons définit deux notion dans la conception de notre ontologie, la notion d'usage et la notion de contenu, cette dernière est représentée par les concepts anatomique de l'ophtalmologie que nous avons extrait de l'ontologie « ophta » déjà existante, à l'aide du Plug-in PROMPT intégré dans protégé.
PROMPT est un algorithme de fusion et d'alignement d'ontologies qui est pensé pour être semi-automatisé, il exécute quelques tâches automatiquement et dirige l'utilisateur dans l'exécution d'autres tâches pour lesquelles son intervention est exigée.

Le plug-in dispose de plusieurs fonctionnalités : comparaison entre les ontologies, comparaison entre les différentes versions d'une même ontologie, extraction d'une portion à partir d'une autre ontologie, le mapping (mise en correspondance) de deux ontologies.

Dans notre cas nous avons utilisé extraction de portion d'une autre ontologie et l'intégrer à la notre.

Le processus d'extraction demande à l'utilisateur de choisir l'ontologie dont seront extrait les concepts, instance et slot (propriété) voulus.

Ainsi quand l'utilisateur aura sélectionné tout ce qu'il veut extraire, ces derniers seront intégrés à l'ontologie. Dans notre cas on a extrait les concepts anatomiques d'une ontologie d'ophtalmologie.

II.3.3. Evaluation :

Vu que les méthodes d'évaluation d'ontologie comme ONTOLINGUA et ONTOCLEAN utilisent des notions philosophiques tels que : la rigidité, l'unité , l'identité , l'existence , et compte tenu de notre manque de maitrise concernant ces notions , nous avons utilisé un outil d'évaluation performant , le moteur de raisonnement « RACER PRO » , cet outil est très performant pour détecter les inconsistances des concepts de l'ontologie et la classification de taxonomie , notamment en inférant de nouvelles classes , instances grâce aux restrictions logique appliqués sur les concepts , il peut également corriger les erreurs d'inconsistance.
Nous avons évolué dans un premier temps la première taxonomie construite, puis après la fusion de la portion d'ontologie de l'ophtalmologie nous avons réévalué notre ontologie et corriger quelques inconsistances.

II.3.4.Documentation :

Cette partie consiste à fournir un document de support de l'ontologie afin de laisser la trace de construction de l'ontologie pour faciliter le suivie et la réutilisation de l'ontologie.
L'outil protégé génère une documentation de l'ontologie en fichier HTML qui contient tout les concepts, relations et autres informations propres à l'ontologie.

Nous obtenons ainsi suite à cette conception, notre ontologie intitulée « OntoMed » et qui est prête à être utilisée dans un Système de recherche d'information qui sera développé par la suite.

Partie 2 : Conception du système M.S.E

I. INTRODUCTION

Dans cette partie du chapitre, nous utilisons UML pour décrire la conception de notre système globale, en incorporant la conception de notre système multi-agents pour établir la communication entre agents,

UML (Unified Modeling Language) traduite "**langage de modélisation objet unifié**") est né de la fusion des trois méthodes qui ont le plus influencé la modélisation objet au milieu des années 90 : OMT, Booch et OOSE.
Issu "du terrain" et fruit d'un travail d'experts reconnus, UML est le résultat d'un large consensus. De très nombreux acteurs industriels renommés ont adopté UML et participent à son développement. En l'espace d'une poignée d'années seulement, UML est devenu un standard incontournable **[Web16].** Parmi les avantages d'UML :

- Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation de solutions.
- L'aspect formel de sa notation, limite les ambiguïtés et les incompréhensions.
- Son indépendance par rapport aux langages de programmation, aux domaines d'applications et aux processus, en font un langage universel.

Nous avons développé notre système en suivant le processus de développement logiciel UP (Unified Process), qui regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.
Caractéristiques essentielles du processus unifié :

- Le processus unifié est à base de composants,
- Le processus unifié utilise le langage UML (ensemble d'outils et de diagramme),
- Le processus unifié est piloté par les cas d'utilisation, Centré sur l'architecture,
- Itératif et incrémental **[16].**

Cycle de vie du processus unifié :

Le processus unifié répète un certain nombre de fois une série de cycles.
Tout cycle se conclut par la livraison d'une version du produit aux clients et s'articule en 4 phases : création, élaboration, construction et transition, chacune d'entre elles se subdivisant à son tour en itérations. **[Web17]**

Le présent chapitre se divise en deux parties principales : la première décrit la présentation de notre application (composants de l'application et l'approche multi-agents) et la deuxième partie présente la conception détaillée de notre système intitulé M.S.E.

II. PRESENTATION DE L'APPLICATION

Parmi les applications web existantes les moteurs de recherche, dont le but est de retourner des informations à l'utilisateur, mais reste que ces informations ne sont pas assez pertinentes.

Notre application a pour but d'essayer de répondre aux besoins de l'utilisateur dans le domaine médical en lui renvoyant les meilleurs résultats correspondant à sa requête, et pour mieux raffiner les résultats, notre application offre la possibilité de spécifier l'usage de l'information recherchée, et de ce fait mieux cibler les pages web.

La figure ci-dessus illustre le schéma fonctionnel de notre système de recherche d'information médicale.

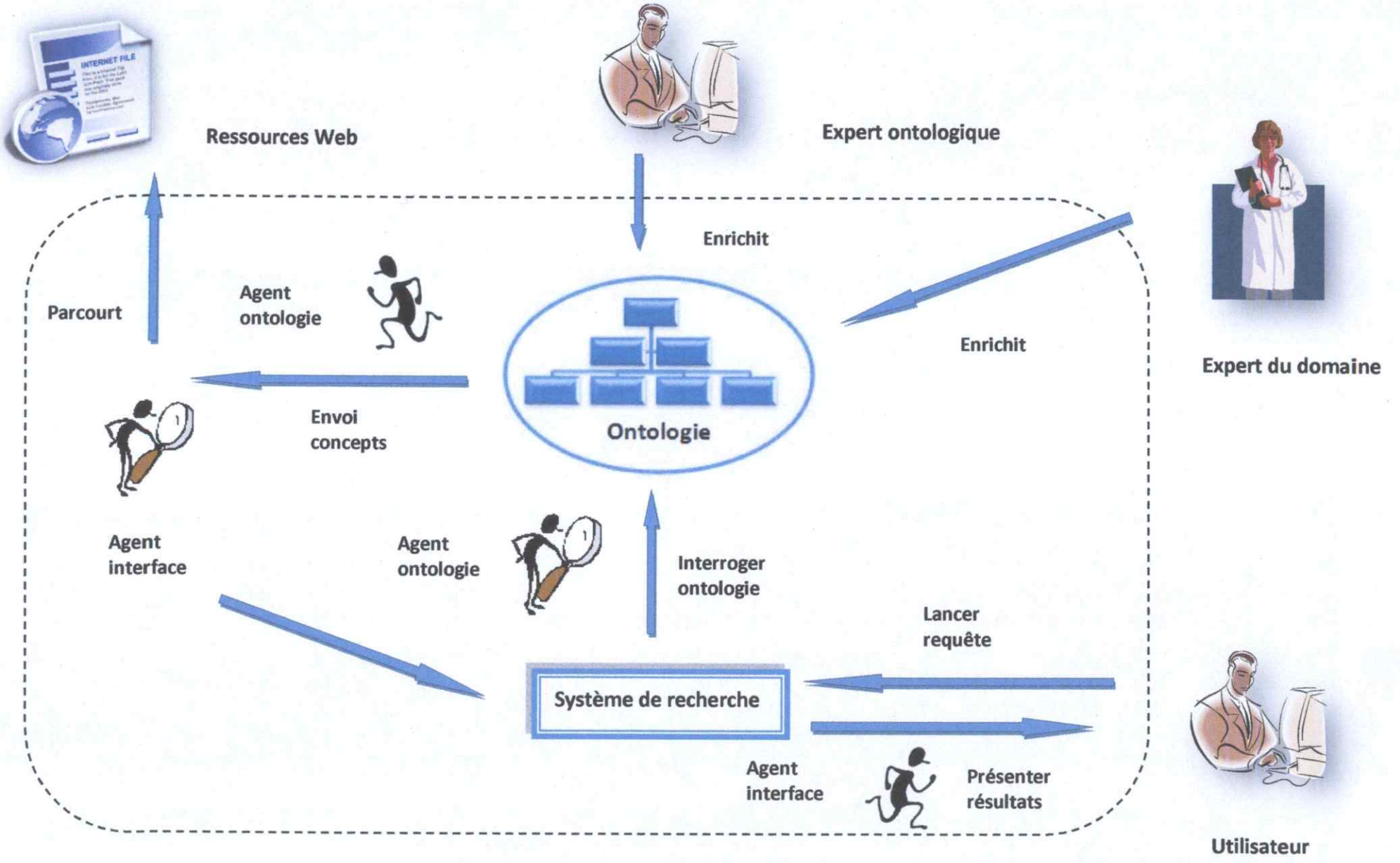


Figure IV.10 : Schéma fonctionnel du système

II.1 Approche multi-agents

Notre système multi agent est composé de deux agents qui coopèrent entre eux via des envoies de messages de type Fipa ACL message.

L'approche est basée sur les agents suivants :

- **Agent interface** :
  - Son rôle est de traiter la requête introduite par l'utilisateur : ce traitement consiste à :
    - éliminer les accents,
    - éliminer les mots vides,
    - éliminer les caractères spéciaux,
    - normaliser les termes de cette requête (féminin → masculin, Pluriel → singulier).
  - Envoie la requête traitée à l'agent ontologie.
  - Reçoit les résultats de la recherche de l'agent ontologie.
  - Affiche les résultats à l'utilisateur après une recherche dans le web à travers le Web service de Google.

Traitement de la requête

Le traitement de la requête consiste à :

- ***Eliminer les accents*** : dans cette étape les caractères accentués (é, è, î, ï, à, ù, etc.) sont remplacés par leur équivalent sans accent (e, i, u,.. etc.).
- ***Eliminer les mots vides*** : les mots vides sont des mots porteurs de peu de sens comme les articles, les conjonctions de coordination, les adverbes, les formes conjuguées des auxiliaires...Etc.

Pour détecter les mots vides, nous utilisons un dictionnaire des mots vides, qui contient la majorité des mots vides.

Exemple du contenu d'un dictionnaire de mots vides :

Les	est	des	une
Pour	dans	par	que
Vous	sur	qui	pas
Avec	son	cette	plus
Etre	sont	peut	comme
Aussi	sommes	nous	
Vous	leurs	leur	.....

Aussi tout mot dont la taille est inférieure ou égale à 2 caractères sera considéré comme mot vide.

EXEMPLE :
« Le cancer du poumon est le nom générique de diverses tumeurs malignes des poumons », après élimination des mots vide (le, du, est, le, de, des), nous obtenons
« Cancer poumon nom générique diverses tumeurs malignes poumons »

- ***Eliminer les caractères spéciaux :***
Les caractères spéciaux sont nombreux nous citons quelques uns (, ; : ! § % $ £ ¤ « # , ...)

Normalisation des termes de la requête : La normalisation ou lemmatisation d'une requête, consiste à transformer chaque terme de cette requête de sa forme fléchie en sa forme canonique, en transformant au singulier tous les mots qui sont au pluriel et à transformer au masculin tous les mots qui sont au féminin, ceci est rendu possible par l'utilisation de règles de normalisation qui sont données dans les tableaux suivants :

Règles de normalisation :
- *Du pluriel au singulier*

Tableau IV.6 : Les règles de normalisation du pluriel au singulier

REGLE	EXEMPLE
[radical] **s** → [radical]	Lois → Loi Recherches → Recherche
[radical] **aux** → [radical] **al**	Chev**aux** → Chev**al**
[radical] **eaux** → [radical] **eau**	Réseaux → Réseau
[radical] **eux** → [radical] **eu**	**Jeux** → **Jeu**

- *Du féminin au masculin*

Tableau IV.7 : Les règles de normalisations du féminin au masculin

REGLE	EXEMPLE
[radical] **trice** → [radical] **teur**	lec**trice** → lec**teur**
[radical] **ande** → [radical] **and**	march**ande** → march**and**
[radical] **onde** → [radical] **ond**	r**onde** → r**ond**
[radical] **aude** → [radical] **aud**	ch**aude** → ch**aud**
[radical] **arde** → [radical] **ard**	mouch**arde** →mouch**ard**
[radical] **ale** → [radical] **al**	Roy**ale** → roy**al**
[radical] **elle** → [radical] **el**	R**éelle** → r**éel**

[radical] **ine** → [radical] **in**	Sangu**ine** → sangu**in**
[radical] **onne** → [radical] **on**	B**onne** → b**on**
[radical] **enne** → [radical] **en**	Algéri**enne** → algéri**en**
[radical] **une** → [radical] **un**	Br**une** → br**un**
[radical] **ère** → [radical] **er**	Infirmi**ère** → infirmi**er**
[radical] **ise** → [radical] **is**	Marqu**ise** → marqu**is**
[radical] **esse** → [radical] **e**	Hôt**esse** → hôt**e**
[radical] **euse** → [radical] **eur**	Chant**euse** → chant**eur**
[radical] **ate** → [radical] **at**	Adéqu**ate** → adéqu**at**
[radical] **cte** → [radical] **ct**	Dire**cte** → dire**ct**
[radical] **ite** → [radical] **it**	Pet**ite** → pet**it**
[radical] **nte** → [radical] **nt**	Fréqu**ente** → fréqu**ent**
[radical] **atte** → [radical] **at**	Ch**atte** → ch**at**
[radical] **ive** → [radical] **if**	Sport**ive** → sport**if**
[radical] **ée** → [radical] **é**	Chass**ée** → chass**é**
[radical] **ie** → [radical] **i**	Apprenti**e** → apprenti
[radical] **ue** → [radical] **u**	Vend**ue** → vend**u**
Exception : [radical] **que** → [radical] **que** [radical] **gue** → [radical] **gue**	Informati**que** → informati**que** Dro**gue** → dro**gue**

- **Agent ontologie** :
  - Chargement du contenu de l'ontologie :
    Une ontologie au niveau implémentation, est un fichier contenant du texte formaté par des balises. Un agent, ou n'importe quel autre logiciel, ne pourra exploiter son contenu sémantique à moins de posséder une vue (schéma) représentative du contenu ontologique. Le chargement du contenu de l'ontologie consiste à créer un schéma exploitable par l'agent qui contient tous les aspects sémantique de l'ontologie.

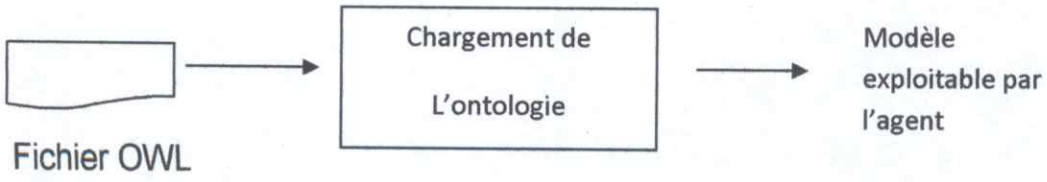


Figure IV.11 : Chargement du contenu de l'ontologie

- Réception de la requête (traitée et lemmatisée) et application de l'algorithme d'extraction des concepts pertinents :
  Une fois la requête purifiée, l'agent ontologie devra procéder à l'application de l'algorithme de désambiguïsation (*cf. ANNEXE D*) définit dans **[18]**. Le résultat de cet algorithme, i-e les concepts pertinents et leur information, sera renvoyé à l'agent interface pour la recherche dans le web.

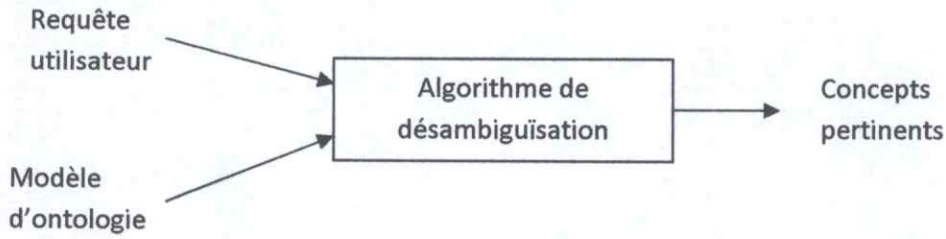


Figure IV.12 : Exécution de l'algorithme de désambigüisation

Mécanisme de désambigüisation : Il arrive qu'un mot clé puisse correspondre à plusieurs concepts de l'ontologie. Dans ce cas, les concepts sont considérés comme « ambigus » car, par ce fait, un même mot clé a plusieurs sens dans la requête.

Afin d'éliminer cette ambigüité et de n'associer qu'un seul concept à chaque mot clé, Khan, a proposé un algorithme *(cf. Annexe D)* basé sur la corrélation (rapport ou relation) entre les mots clés de la requête et les concepts ambigus.

III. ETUDE CONCEPTUELLE

III.1. Expression des besoins

Dans un processus de modélisation d'un système, la détermination précise des besoins des utilisateurs de système est la première étape car une bonne informatisation passe de plus en plus par une vision intégrante, non parcellaire qui s'optimise lorsqu'elle dépasse le point de vue d'une seule catégorie d'utilisateur. **[17]**
Le web c'est l'une des grandes sources d'information, avec les moyens de recherche existant on ne trouve pas facilement ce qu'on recherche, les besoins de l'utilisateur se présentent comme suit :

- Avoir des résultats plus précis.
- Avoir plus de sémantique dans la recherche.

III.2. Analyse :

L'intérêt de la phase d'analyse est de définir une approche conceptuelle du système. La phase d'analyse est totalement détachée des problèmes d'implémentation (langage, SGBD, matériels), c'est la définition de l'analyse : modéliser sans contraintes d'implémentations.
Nous ne présentons pas de diagramme à ce niveau, vu que notre application est purement informatique (implémentation), donc cette phase est implicite.

III.3. Conception :

III.3.1. Les cas d'utilisation

Les diagrammes de cas d'utilisation représentent les cas d'utilisation, les acteurs et les relations entre les cas d'utilisation et les acteurs. Un cas d'utilisation est une manière de décrire comment utiliser le système. C'est l'image d'une fonctionnalité du système, déclenchée par un acteur. **[17]**

- **Les acteurs**

Un acteur est une entité externe qui interagit avec le système (opérateur, centre distant, autre système...). En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin. Les acteurs de notre système sont définit comme suit :

L'acteur principal :

Utilisateur **:** Le système a été conçu pour lui, l'utilisateur peut rechercher des informations médicales en spécifiant l'usage pour raffiner les résultats.

Acteurs secondaires :

Expert ontologique **:** Enrichit l'ontologie.
Expert du domaine **:** Contribue à l'enrichissement de l'ontologie en fournissant une aide concernant le domaine étudié.
Web service de Google : c'est un système externe qui est utilisée pour la recherche dans le web.

III.3.2. Le Diagramme des cas d'utilisations

Pour que l'utilisateur effectue une recherche il doit :

- faire entrer sa requête via l'interface graphique
- Une liste de choix est affichée
- L'utilisateur choisi parmi les usages (usage relatif a son besoin)
- L'utilisateur valide sa requête après avoir choisi ou non un usage.

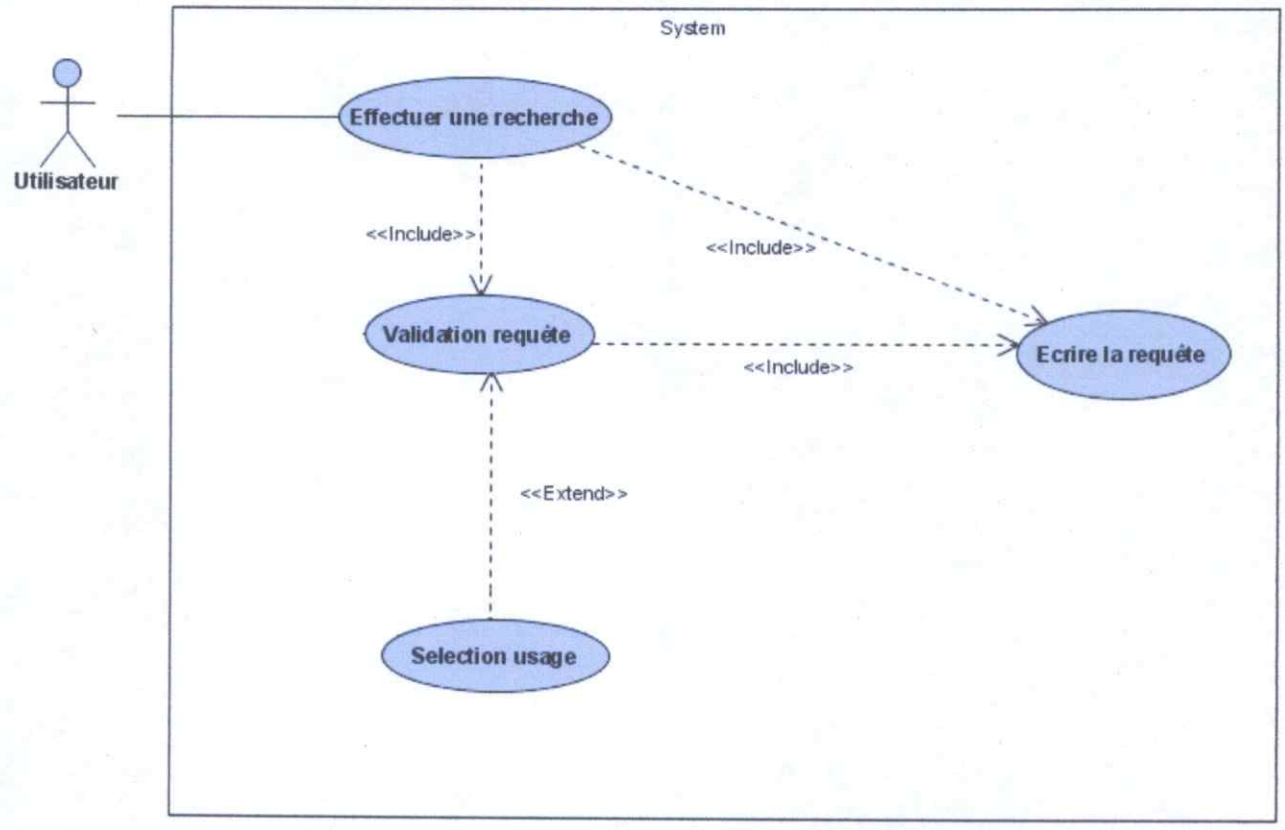


Figure IV.13 : Diagramme de cas d'utilisation

III.3.3. diagramme de séquence

Les diagrammes de séquence nous permettent de montrer les interactions entre objets, ils nous permettent de bien schématiser les scénarios des cas d'utilisation.

La représentation se concentre sur la séquence des interactions selon un point de vue temporel. Ils sont en général, plus aptes à modéliser les aspects dynamiques des systèmes temps réel et des scénarios complexes mettant en ouvre peu d'objets. **[17].**

- **Scenario de recherche d'information :**

1 : l'utilisateur lance sa recherche en écrivant sa requête dans un champ de saisie (en choisissant ou non l'usage).

2 : l'agent interface se charge de traiter la requête en la normalisant.

3 : l'agent interface envoit un message contenant la requête normalisée à l'agent ontologie.

4 : l'agent ontologie charge le contenu de l'ontologie

5 : l'agent ontologie reçoit les termes de la requête normalisée,

6 : parcourt les concepts de l'ontologie essayant de trouver les concepts appropriés.

7 : exécute l'algorithme de désambigüisation.

8: une fois les concepts trouvés, et désambigüisés l'agent ontologie renvoi ces concepts à l'agent interface.

9 : l'agent interface reçoit ces concepts.

10 : une fois les concepts reçus, l'agent interface présente ces résultats au web service de Google.

11 : Le web service de Google effectue la recherche dans le web.

12 : Affiche les résultats à l'utilisateur.

La figure suivante illustre le diagramme de séquence du système M.S.E dans le cas où les concepts sont trouvés:

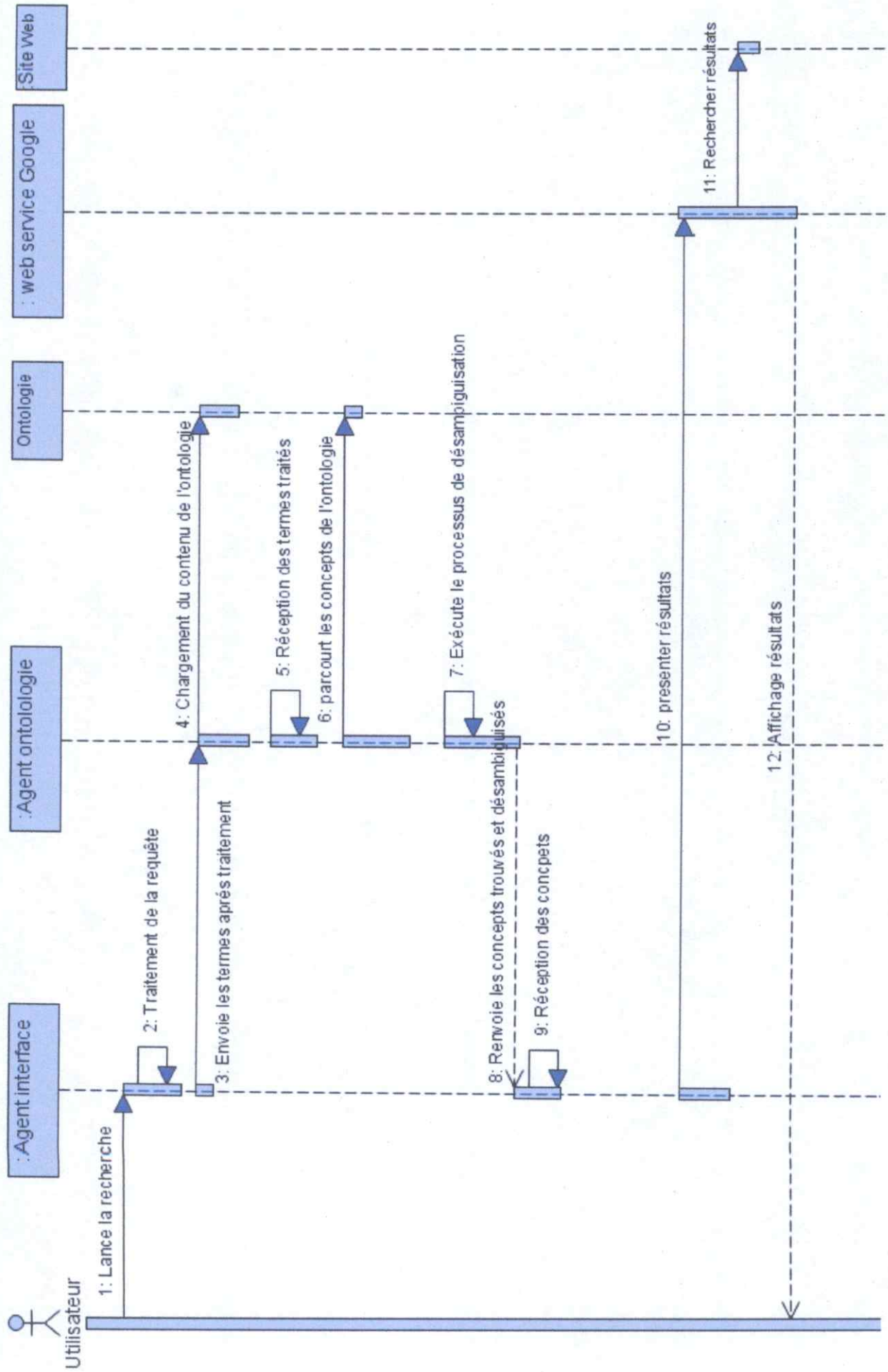

:Site Web
: web service Google
Ontologie
:Agent ontologie
: Agent interface
Utilisateur
1: Lance la recherche
2: Traitement de la requête
3: Envoie les termes aprés traitement
4: Chargement du contenu de l'ontologie
5: Réception des termes traités
6: parcourt les concepts de l'ontologie
7: Exécute le processus de désambiguisation
8: Renvoie les concepts trouvés et désambiguisés
9: Réception des concpets
10: presenter résultats
11: Rechercher résultats
12: Affichage résultats

Figure IV.14 : Diagramme de séquence de la recherche d'information

- **Scénario d'exception (concepts non trouvés):**

1 : l'utilisateur lance sa recherche en écrivant sa requête dans un champ de saisie.

2 : l'agent interface se charge de traiter la requête en la normalisant.

3 : l'agent interface envoie un message à l'agent ontologie lui renvoyant la requête normalisée.

4 : l'agent ontologie charge le contenu de l'ontologie.

5 : l'agent ontologie reçoit les termes de la requête normalisée.

6 : parcourt les concepts de l'ontologie essayant de trouver les concepts appropriés.

7 : exécute l'algorithme de désambigüisation.

8 :l'agent ontologie ne trouve pas les concepts relatifs aux termes de la requête, envoie un message à l'agent interface lui indiquant « concept non trouvé ».

7 : l'agent interface reçoit le message, et affiche à son tour un message d'erreur dans l'interface graphique indiquant qu'il n'ya pas de résultats.

La figure suivante illustre le diagramme de séquence du système M.S.E dans le cas ou les concepts ne sont pas trouvés:

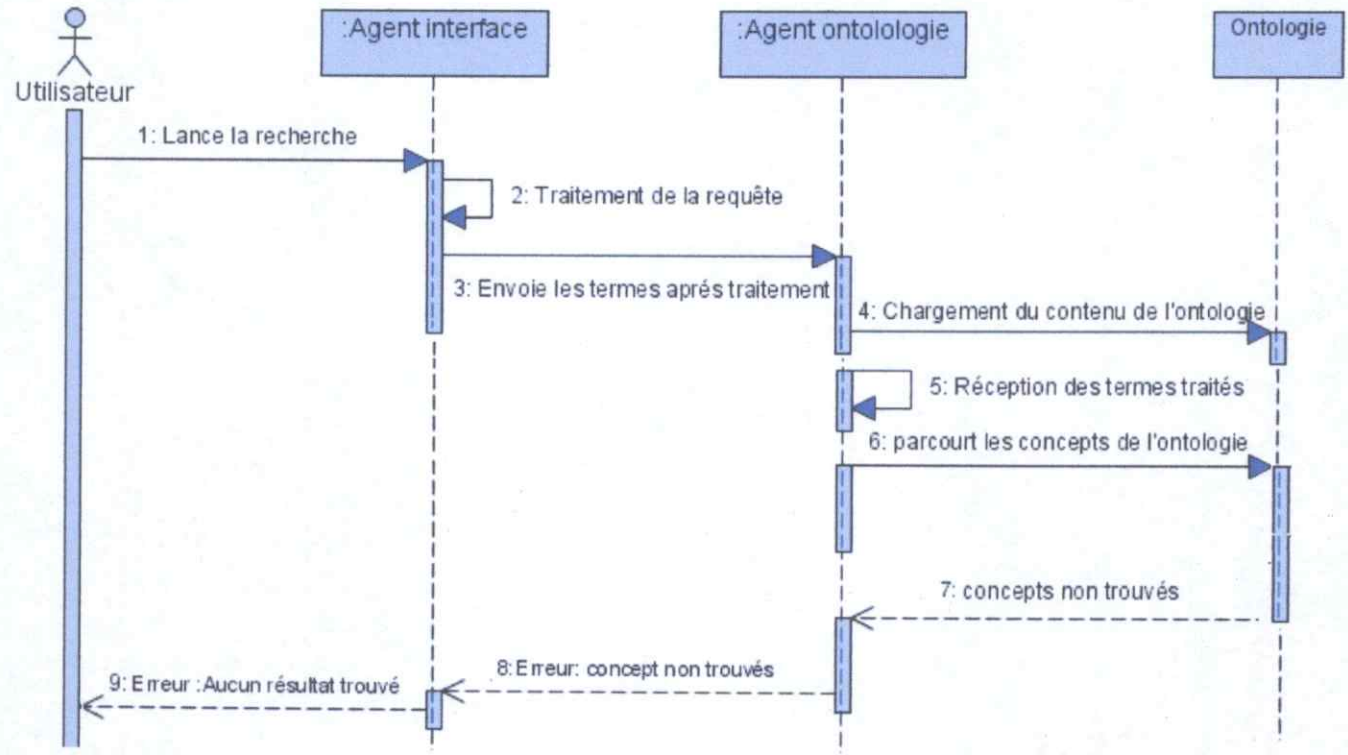


Figure IV.15 : Diagramme de séquence d'exception

III.3.4. Diagramme de collaboration du système M.S.E

Les diagrammes de collaboration (tout comme les diagrammes de séquence) représentent une vue dynamique du système. Ils montrent également les interactions entre objet à travers la représentation d'envoi de messages. L'ajout d'une dimension temporelle requiert la définition de numéros de séquence pour les messages.
Une collaboration définit les éléments qui sont utiles pour l'obtention d'un objectif particulier en spécifiant le rôle de ces éléments dans un contexte de la collaboration **[17].**
Le premier message de l'interaction est envoyé par l'acteur, dans notre cas l'utilisateur représenté par le symbole graphique des acteurs du modèle des cas d'utilisation.

Le diagramme de collaboration ci-dessous représente l'interaction des agents du système.

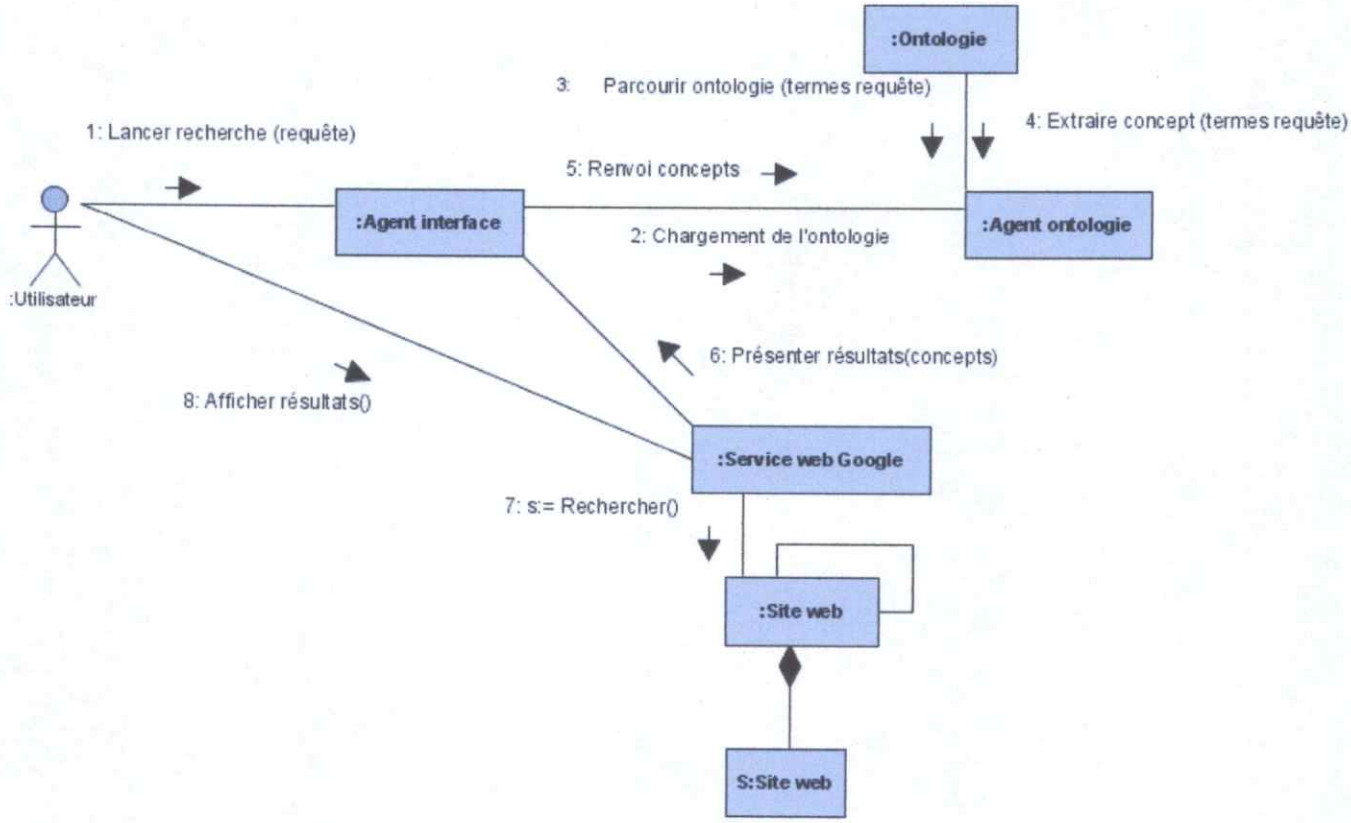


Figure IV.16: Diagramme de collaboration du système M.S.E

➢ **Description des rôles**

- L'opération lancer recherche () : consiste en la formulation de la requête, puis le choix ou non de l'usage parmi les usages proposés, et pour finir la validation de la requête qui se concrétise par l'envoi de la requête à l'agent interface.
- Chargement ontologie ():Le chargement du contenu de l'ontologie consiste à créer un schéma exploitable par l'agent qui contient tous les aspects sémantique de l'ontologie.
- Parcourir l'ontologie () : cette opération se traduit par l'exploration de l'ontologie à la recherche des concepts correspondants aux termes de la requête.
- Extraire les concepts () : après avoir trouvé les concepts, l'opération « extraire » consiste à récupérer ces concepts.
- Renvoi concepts () : l'agent ontologie renvoi les concepts à l'agent interface.
- Présenter résultats () : l'agent interface ayant récupéré les concepts, il les présente au service web de Google afin d'effectuer la recherche dans le web.
- Rechercher () : cette opération consiste à parcourir le web, à la recherche des pages web via le service web de Google.
- Afficher résultats () : cette dernière opération consiste à afficher les résultats à l'utilisateur.

III.3.5. Diagramme d'état transition

Les diagrammes d'états-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.

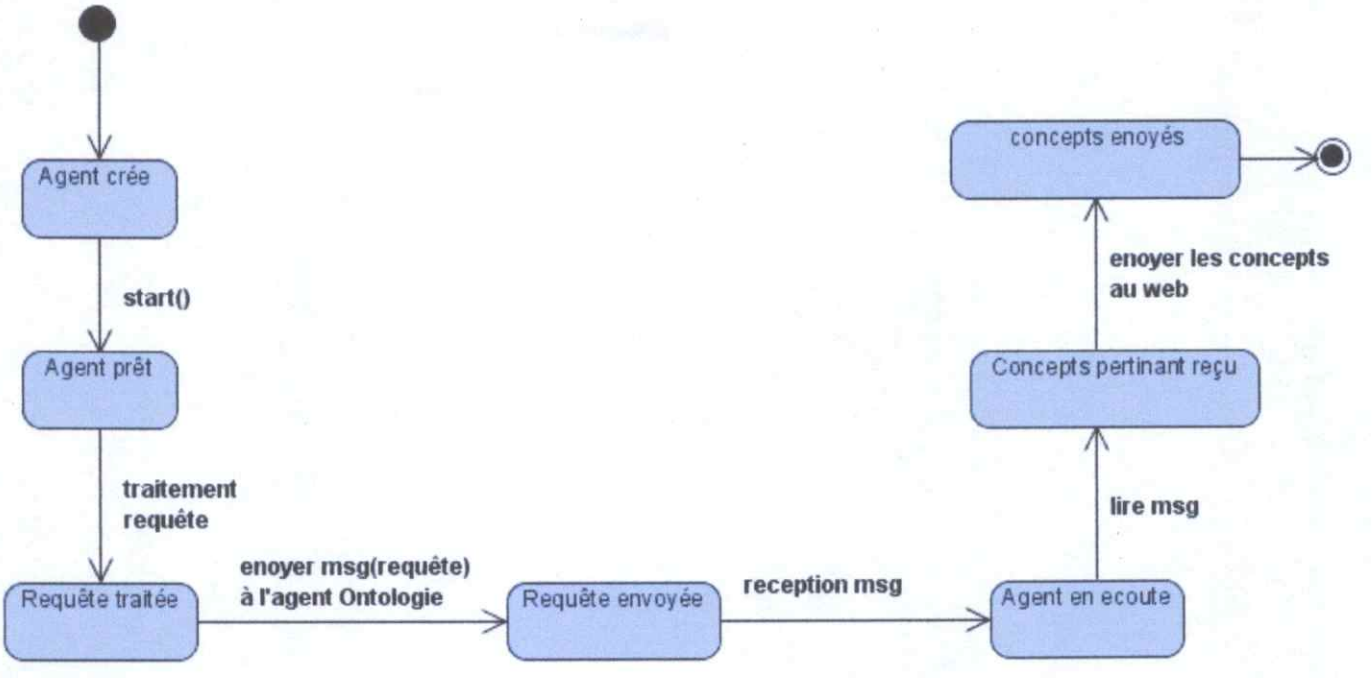


Figure IV.17 : Diagramme transition d'état de l'agent Interface

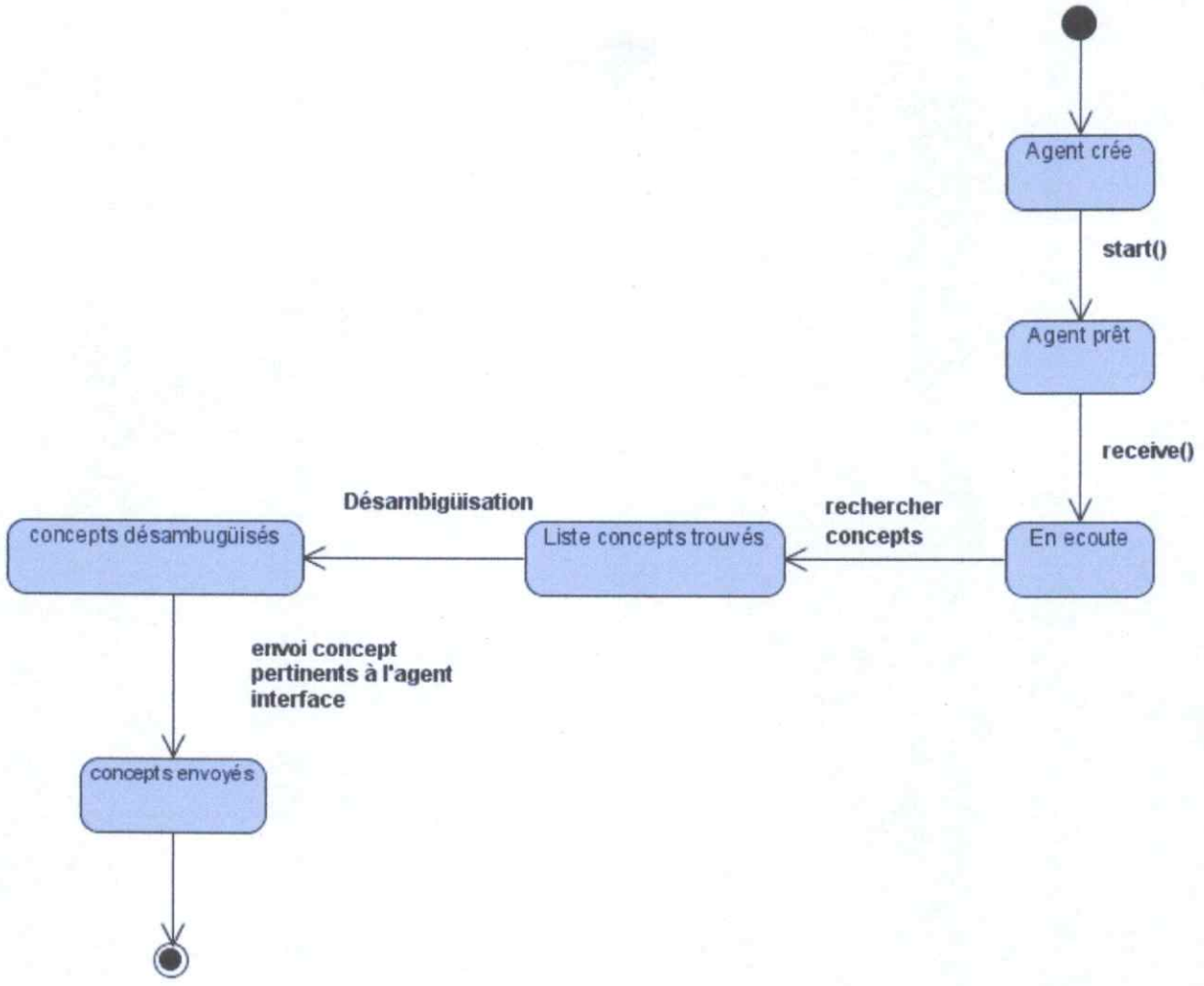


Figure IV.18: Diagramme transition d'état de l'agent Ontologie

III.3.6. Diagramme de déploiement :

Le diagramme de déploiement montre la disposition physique des différents matériels – les nœuds- qui rentrent dans la composition du système.

Il est représenté par un graphe composé de nœuds interconnectés par des liens de communication.

Ci-dessous est présenté le diagramme de déploiement de notre système.

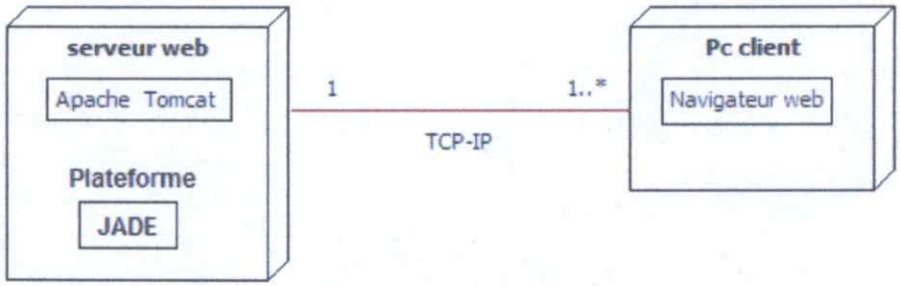


Figure IV.19 : Diagramme de déploiement du système

III.3.7. Diagramme de classe du système

Le diagramme de classe exprime de manière générale la structure statique d'un Système, en termes de classe et de relations entre ces classes. Le diagramme de classes suivant représente les éléments qui doivent être implémentés dans notre système :

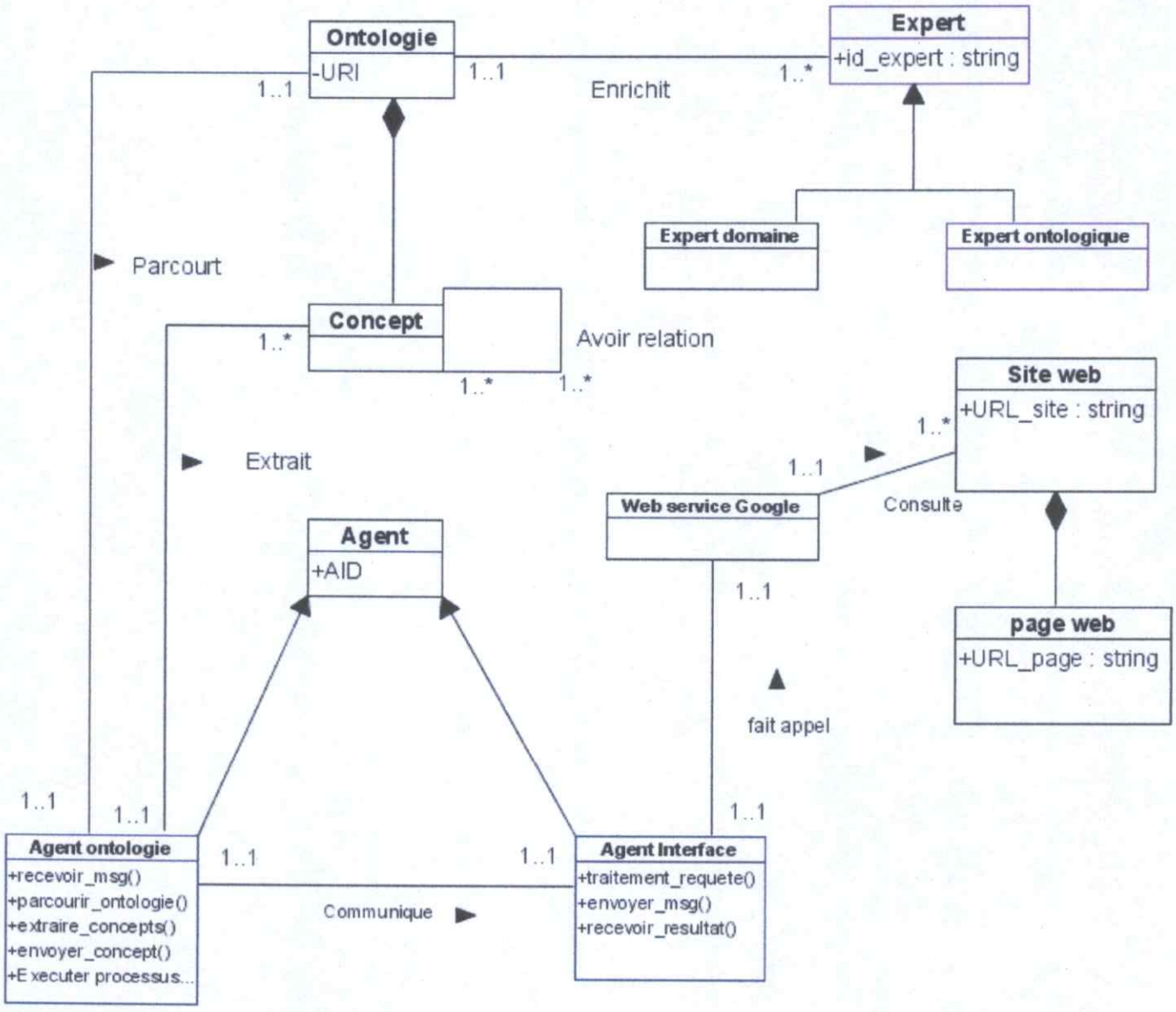


Figure IV.20: Diagramme de classe du système M.S.E

CHAPITRE IV

IMPLEMENTATION ET TEST

IMPLEMENTATION & TEST

I.INTRODUCTION

Apres avoir compléter l'étude conceptuelle du système de recherche et de l'ontologie, nous présentons dans ce qui suit la réalisation du système de recherche selon les spécifications dans le chapitre conception (Chapitre IV)
Notre système, comporte trois agents et une ontologie médicale qui est la fusion de deux ontologies, il propose à l'utilisateur une recherche en langage naturel en utilisant la notion d'usage.

Ce chapitre se divise en deux parties :

- Partie 1 : La réalisation de l'ontologie médicale intitulée « OntoMed » (Ontologie médicale).
- Parie 2 : La mise en œuvre du système de recherche intitulé « M.S.E » (Medical Search Engine).

II. ARCHITECTURE DE DEPLOIEMENT

Nous avons utilisé pour notre application l'architecture trois tier ou Architecture à trois niveaux.

II.1. Caractéristiques de l'architecture à trois niveaux

A cause des problèmes de l'architecture à deux niveaux (maintenance), un autre niveau a été rajouté. Ainsi une application est divisée en trois niveaux logiques :

- **Niveau 1 (User interface) :** associé au client qui de fait est dit "léger" dans la mesure où il n'assume aucune fonction de traitement à la différence du modèle 2-tiers; c'est donc la partie présentation de l'application.

  Ce niveau (couche) correspond à la partie de l'application visible et interactive avec les utilisateurs ; qui est notre interface client (page JSP) exploitée par le navigateur Web exemple : Internet Explorer.

- **Niveau 2 (Buisness logic) :** lié au serveur, qui dans de nombreux cas est un serveur Web muni d'extensions applicatives, ce niveau s'occupe donc du traitement de l'information.

  Ce niveau appelé aussi, couche métier, correspond à la partie fonctionnelle de l'application, dans notre cas, c'est le serveur Web Appache tomcat muni d'extensions applicatives qui sont la plateforme jade et l'ontologie.

  La plate forme jade, contient touts les agents responsables du fonctionnement de l'application, qui sont eux, responsables de l'exploitation de l'ontologie.

- **Niveau 3 (Data access)** : La partie accès et stockage des données.

Ce dernier niveau qui est l'accès aux données, nous utilisons le web service de google qui accède à la source de données : le Web, pour notre application, puis restitue les résultats aux clients.

Voici un schéma de l'architecture que nous avons employé :

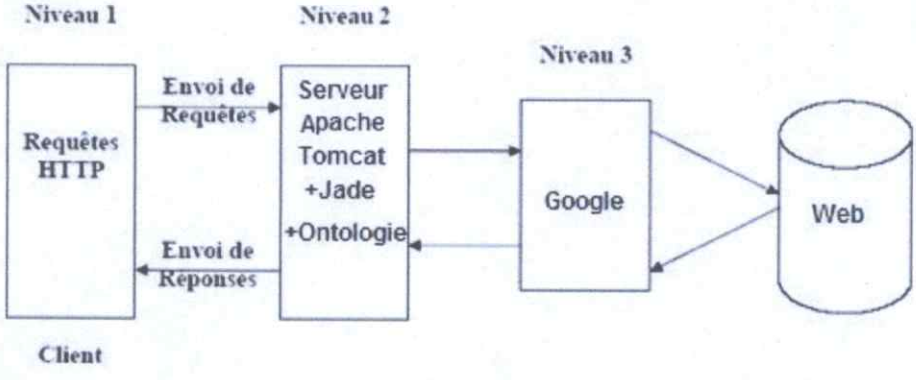


Figure V.1 : Architecture trois tiers du système.

III.ENVIRONNEMENT DE DEVELOPPEMENT

La figure suivante résume nos choix en termes d'environnement de développement.

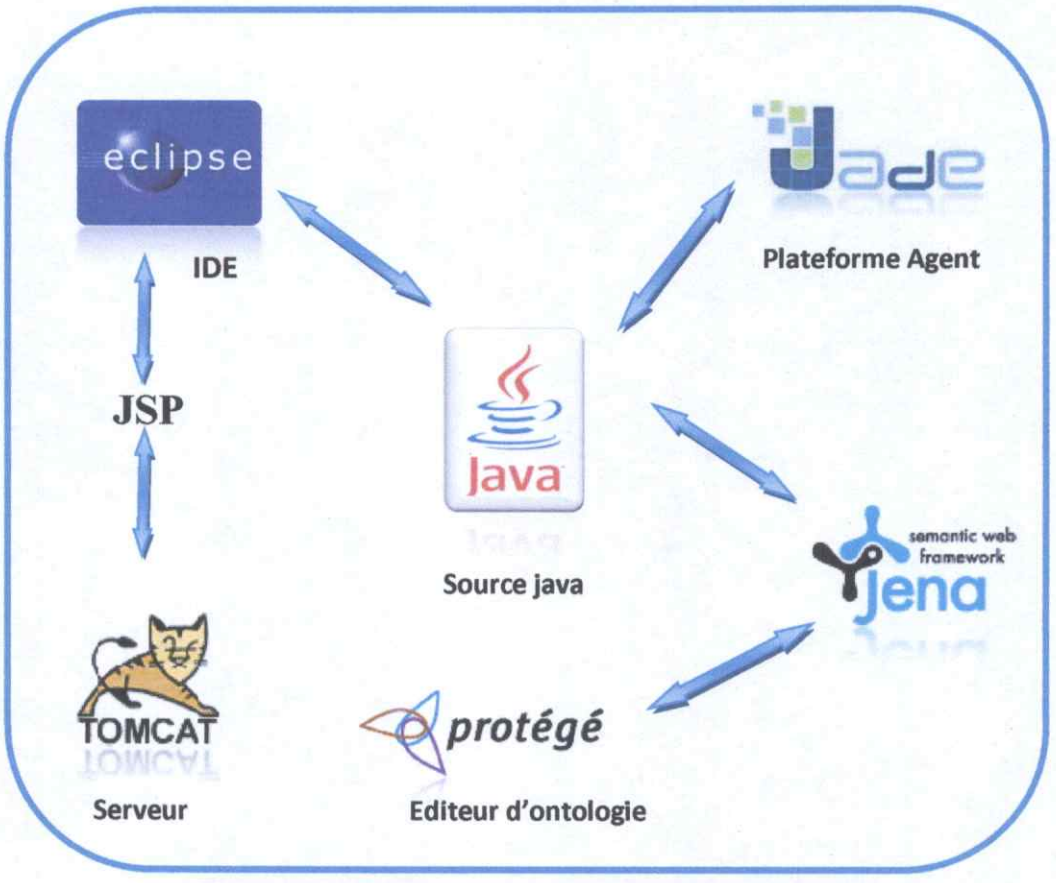


Figure V.2 : Environnement de développement

III.1. Choix des langages et outils de développement

➢ **Choix de la plateforme de développement des agents :**

JADE est un environnement de développement de système multi-agents basé sur java, il permet, grâce à un système de conteneur, de faire vivre des agents sur des réseaux locaux ou sur internet. Cet outil a été crée par TILAB anciennement CESLT (Centro Studi E Laboratori Telecomunicazioni) et a été normalisé par FIPA en 2000 et 2001. (*cf. Annexe A*)

Plusieurs applications dans le domaine de la recherche d'information et de l'ingénierie ontologique ont été développées avec JADE, ses caractéristiques héritent de java, la qualité de ces agents l'ont rendu une préférence dans le domaine du développement d'agents.

➢ **Choix de l'éditeur d'ontologies**

Afin d'implémenter notre ontologie, nous avons opté pour l'éditeur d'ontologies **Protégé** (*cf. Annexe B*) pour les raisons suivantes :

- Protégé est un éditeur open-source ;

- Protégé permet de fusionner, d'importer et d'exporter des ontologies dans les différents langages d'implémentation d'ontologie (RDFS, DAML+OIL, OWL, etc.).
- Protégé possède une riche bibliothèque de plugins dont le « Jambalaya » qui permet la visualisation graphique de l'ontologie.

Pour le langage d'implémentation de l'ontologie, on a choisi OWL qui est le langage standard adopté par le W3C pour la représentation des ontologies.

Grace à l'API de **Jena** 2.5.3, nous avons pu exploiter l'ontologie dans Jade ; Jena offre des méthodes pour la manipulation de l'ontologie. C'est est un kit Java pour le Web Sémantique, issu d'un projet open-source de « HP Labs Semantic Web Program ». Nous avons choisi Jena car :

- Il fournit des capacités de raisonnement sur ces méthodes ;
- Jena est un outil puissant pour le Web Sémantique et sa communauté **[Web18]** ;
- Il dispose d'API extensibles **[Web18]**.

➢ Choix du langage de programmation

Le langage répondant au mieux à la mise en œuvre de notre conception en termes de programmation orientée objets, de distribution, de réutilisation est le langage Java.

Java est un langage de programmation orienté objet et un environnement d'exécution, développé par Sun Microsystems. Il fut présenté officiellement en 1995. Java était à la base un langage pour Internet.

Le fer de lance de Java est « **Write once, run everywhere** », ou « Ecrire une fois, utiliser partout » qui illustre l'objectif de Java, il est possible d'utiliser le même code pour Windows 95/98/NT, UNIX, Macintosh, etc.

Java sera utilisé pour la réalisation de nos **pages web** qui permettront à l'utilisateur de manipuler notre système, la technologie Java manipulée sera donc la **JSP** (Java Server Page), qui génère des pages web dynamiques. Cette technologie mélange la puissance de Java coté serveur et la facilité de mise en page d'HTML coté client. Les JSP sont converties en Servlets par le moteur de Servlets, ils ont l'avantage de séparer le code de la présentation

Côté serveur nous avons utilisé **Apache Tomcat** qui est un conteneur de servlet. Issu du projet Jakarta, Tomcat est désormais un projet principal de la fondation Apache. Il implémente les spécifications des servlets et des JSP de Sun Microsystems.

Tomcat est un serveur Web qui supporte servlet et JSP. C'est le compilateur Jasper qui compile les pages JSP pour en faire des servlet. Le moteur de servlet Tomcat est souvent employé en combinaison avec un serveur Web Apache ou d'autres serveurs Web.

Le serveur web est alors en charge de toute la partie statique du site web, alors que Tomcat gère la partie dynamique (requêtes sur les servlets et les JSP).

Tomcat a été écrit en langage Java, il peut donc s'exécuter via la JVM (machine virtuelle java) sur n'importe quel système d'exploitation.

➢ Choix de l'Environnement de développement

Eclipse IDE est un environnement de programmation Open Source, développé par IBM. Disponible pour les plateformes Windows, Mac OSX, Linux, Solaris... il permet le développement d'application mettant en œuvre n'importe quel langage de programmation: Java, C++, PHP, .NET ...
La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plug-in (en conformité avec la norme OSGi) **[Web19]**.

Voila quelques avantages d'Eclipse **[Web20]:**

- Une technologie avancée
  - Des composants graphiques évolués
  - Une plate-forme à l'architecture ouverte
  - Des fonctionnalités annexes sous forme de plugins
  - Un mécanisme de mise-à-jour à distance
- Une technologie pragmatique
  - Performance : composants graphiques natifs
  - Simplicité : API relativement simple à manipuler
  - Bonne intégration avec l'OS
- Une technologie populaire
  - Open source
  - Soutien d'une large communauté de développeurs

PARTIE 1 : REALISATION DE L'ONTOLOGIE

Suite à la conception de notre ontologie, nous allons passer à la réalisation de celle-ci en utilisant l'éditeur d'ontologie Protégé 3.2.1, qui est un outil puissant de construction d'ontologie complexe.
Protégé permet donc de créer des ontologies, il offre un éditeur de classes (Figure V.3) ainsi qu'un éditeur de relations (Figure V.4) (*cf. Annexe B*).

Nous avons donc créé toutes les classes (concepts) relatives au domaine médical (cancérologie et ophtalmologie), dans la figure V.3 est représenté un concept (usages_médicaux) qui lui est attribué un commentaire.

La figure V.4 est associée à une propriété (usage_avoir_spécialités) étalant le rang et le domaine approprié ainsi que la relation inverse.

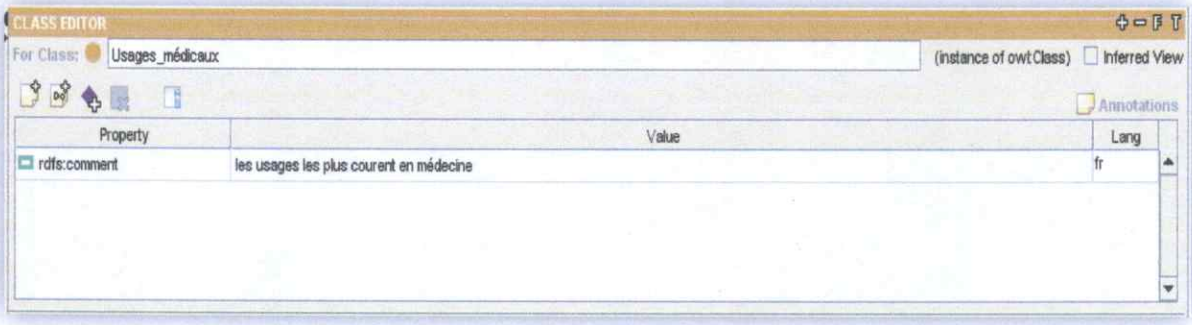


Figure V.3 : Editeur de classes

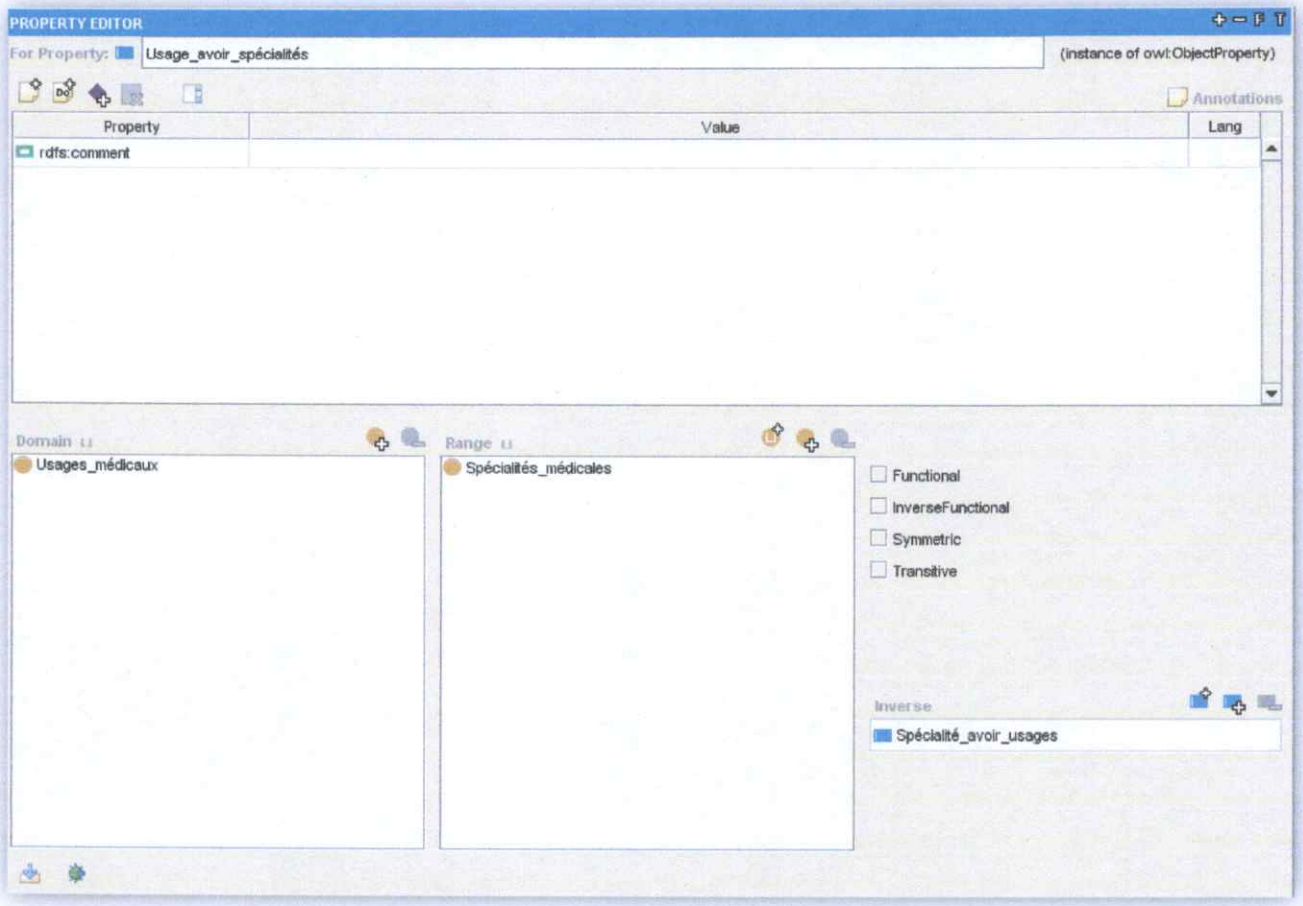


Figure V.4 : Editeur de relations

Pour le langage de représentation des connaissances, nous avons choisi OWL (*Ontology Web language)*, ce langage est actuellement le standard du W3C pour la construction d'ontologies.

Puisque le langage utilisé est OWL, Protégé OWL prend en charge toute opération de OWL à savoir la création de restrictions (Figure V.5), la création de classes disjointes (Figure V.6), annotation, etc.

La figure V.5 présente quelques restrictions construites concernant le concept Pathologies.

Figure V.5 : Editeur de restrictions

Dans la figure V.6 ci dessous, est présenté quelques relations disjointes de quelques concepts de notre ontologie.

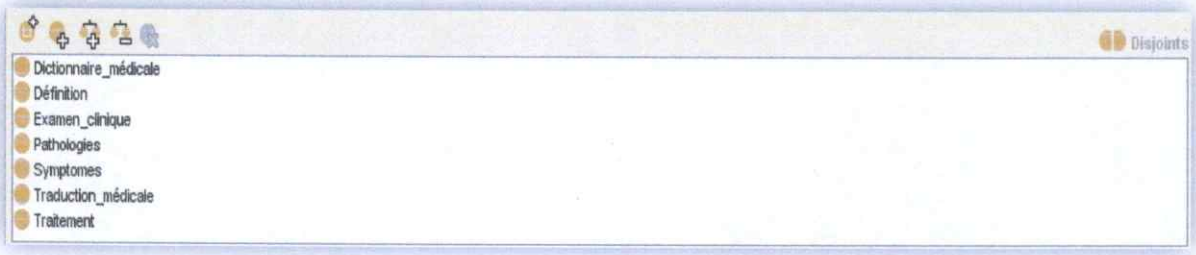


Figure V.6 : Editeur de relations disjointes

Pour la fusion de l'ontologie d'ophtalmologie, nous avons utilisé « PROMPT » (voir partie conception) et le raisonnement ou la consistance de l'ontologie, le raisonneur « RACER PRO ».

Un des avantage de Protégé OWL, est l'intégration des Plug-in qui permettent de gérer des contenus multimédias, requêter, évaluer et fusionner des ontologies, un exemple de plug-in intéressant : Jambalaya qui permet de visionner l'ontologie sous forme de graphe.

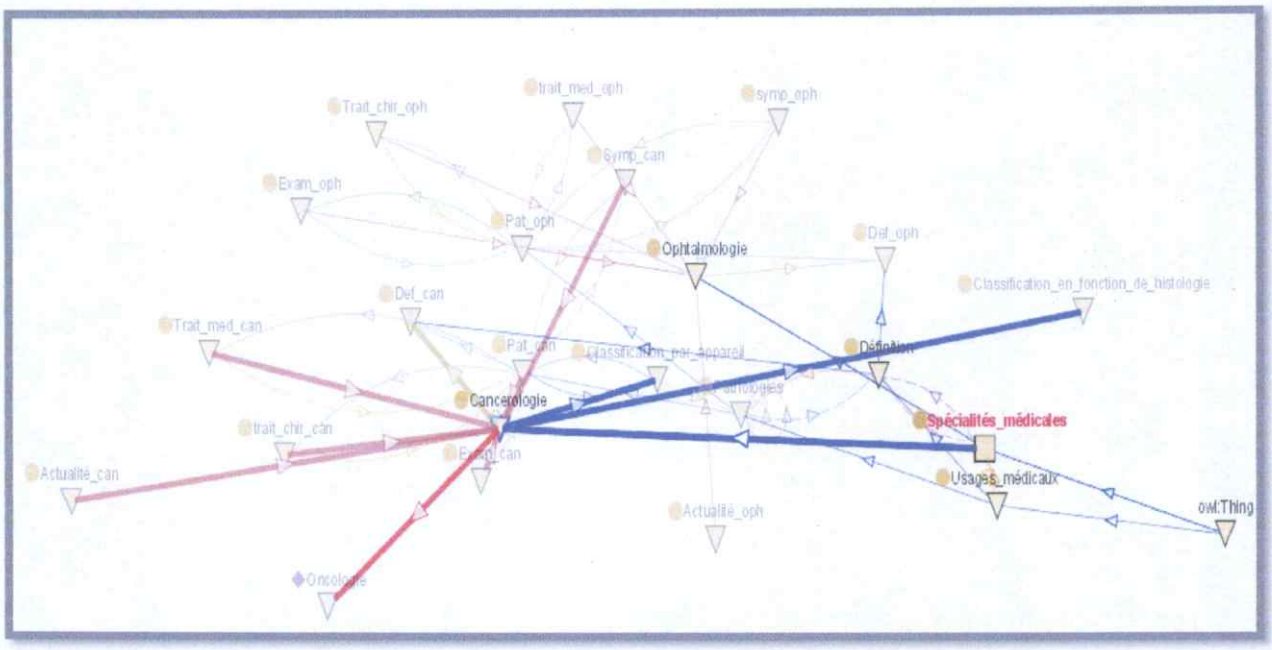


Figure V.8 : Extrait d'une partie de l'ontologie OntoMed

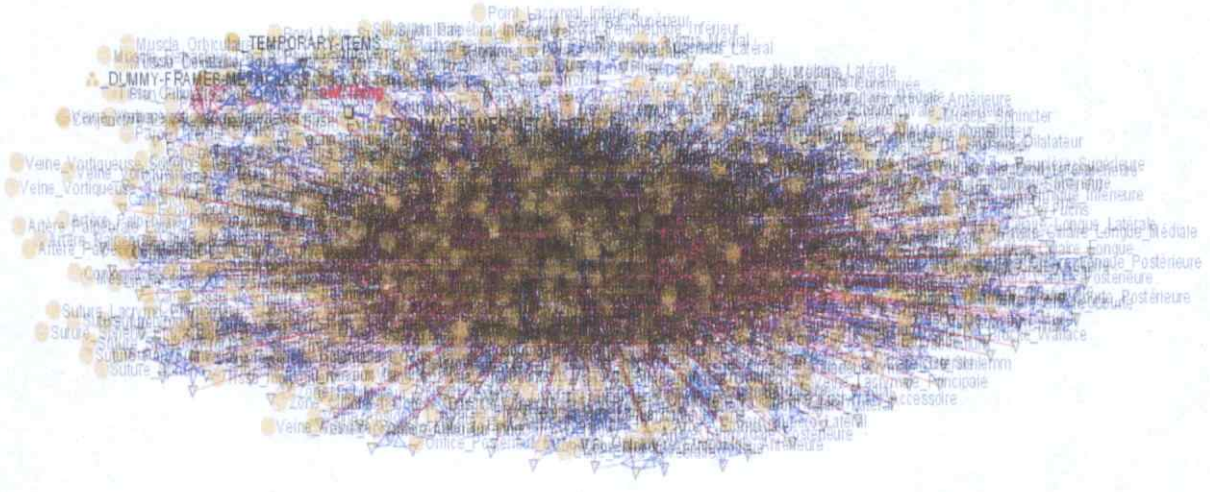
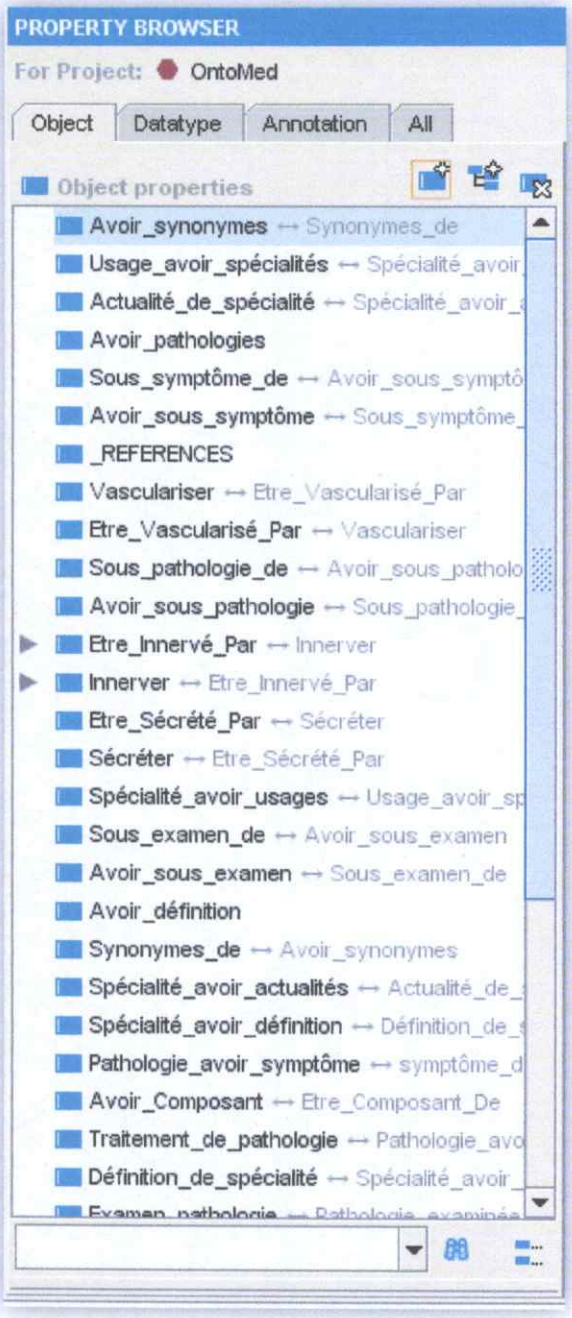


Figure V.9 : L'ontologie OntoMed vue par le plug-in Jambalaya de Protégé

Nous obtenons ainsi suite à cette réalisation, notre ontologie intitulée « OntoMed » et qui est prête à être utilisée dans un Système de recherche d'information qui sera développé par la suite.



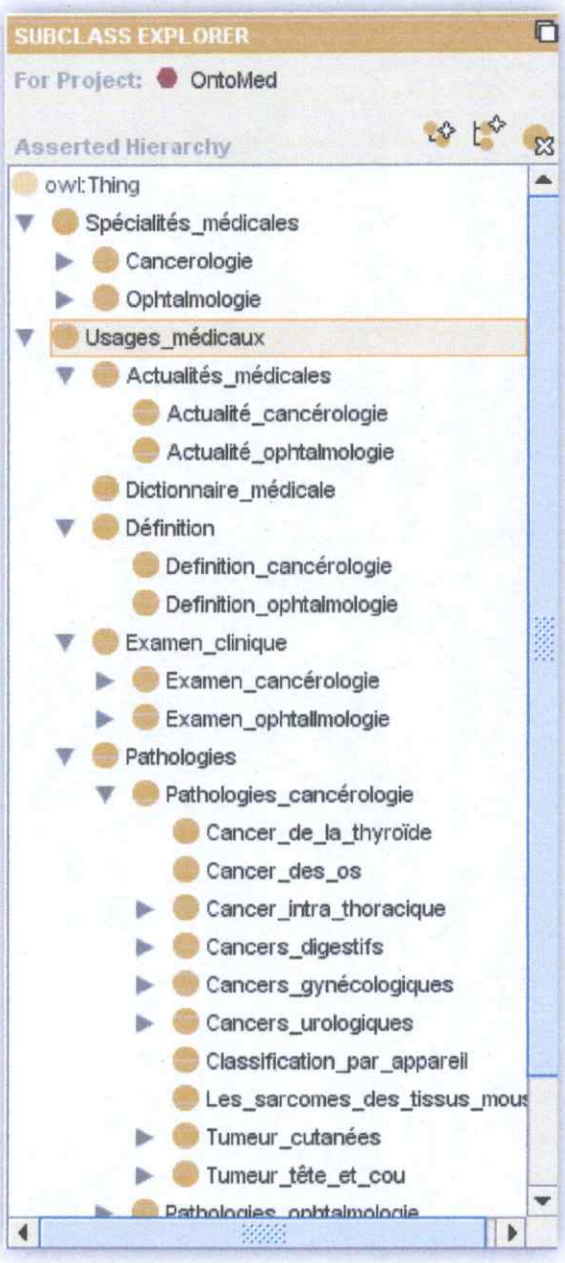


Figure V.10: Extrait des relations de l'ontologie

Figure V.11: Extrait de l'ontologie

Ci-dessus un extrait de l'ontologie « OntoMed » et des relations entre les classes.

PARTIE 2 : REALISATION DU SYSTEME M.S.E

Après avoir réalisé l'ontologie avec l'éditeur Protégé, et afin d'exploiter celle ci, nous passons à la mise en œuvre de notre système de recherche médical intitulé « M.S.E » (Medical Search Engine).

I. FONCTIONNEMENT DU SYSTEME M.S.E

Le schéma illustré par la figure V.12 ci dessous, décrit le processus de recherche dans M.S.E.

L'accès à notre système peut être effectué en tapant l'URL à travers un navigateur Internet **[1]**. Une requête http **[2]** est alors envoyée au serveur web (Apache Tomcat) qui héberge les pages JSP **[3]**.

La page JSP étant chargée, l'utilisateur fait entrer sa requête, débute après la communication entre les agents en se basant sur FIPA ACL **[4]** , l'agent interface est crée dans la plateforme JADE, il se charge d'envoyer la requête de l'utilisateur après son traitement à l'agent ontologie qui va exploiter cette dernière **[5]** , extraire les concepts associés aux mots clés introduits dans la requête ,puis renvoyer les concepts trouvés a l'agent interface qui va les présenter à travers le web service de Google qui s'occupera de la recherche dans le Web**[6],** ainsi que l'affichage de ces derniers à l'utilisateur dans le navigateur.

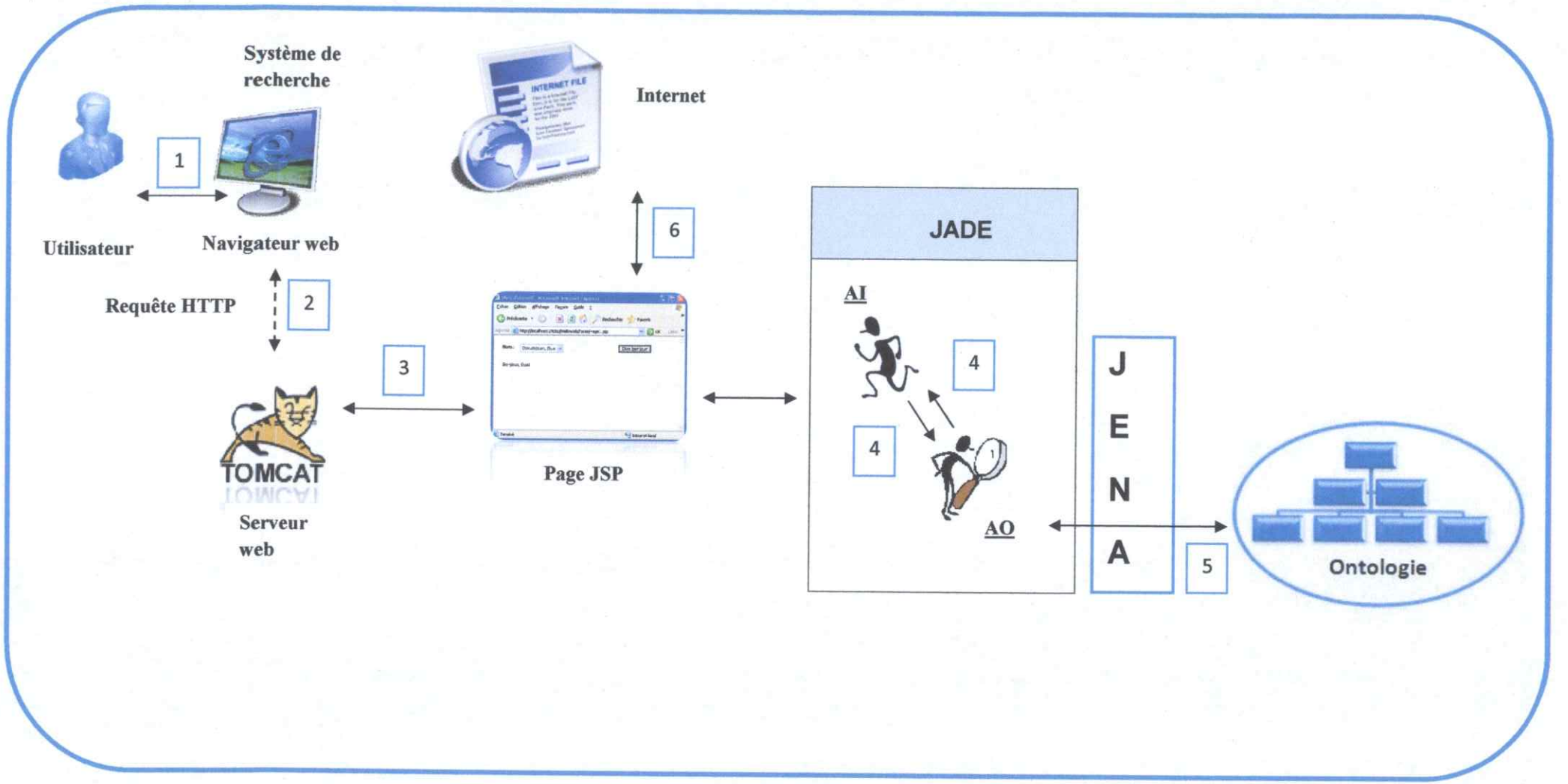


Figure V.12 : Architecture du Système M.S.E

II.RÉALISATION DU SYSTÈME MULTI-AGENT

II.1.Les agents de M.S.E

Grace à la plate-forme Jade, nos agents ont pu être réalisés, ces derniers sont une extension de la classe Agent qui fait parti de l'import « jade.core.Agent ».

Sous forme de classe Java, nos agents contiennent plusieurs méthodes les caractérisant, l'une des méthodes capitales dans un agent est la méthode « **setup()** » qui est obligatoire pour l'initialisation de l'agent.

Afin d'implémenter les comportements des agents, nous avons défini des objets de la classe Behaviour qui fait parti de l'import « jade.core.behaviours.Behaviour », chaque objet de ce type dispose d'une méthode « **action()** » qui constitue le traitement à effectuer par celui-ci, ainsi que d'une méthode « **done()** » qui vérifie si le traitement est terminé.

II.2. La communication entre Agents

II.2.1. Le langage de communication FIPA-ACL

L'organisation *Foundation for Intelligent Physical Agents* (FIPA) a été créée en 1996. Parmi ses préoccupations, une place importante concerne l'élaboration des spécifications du langage de communication entre agents FIPA-ACL.

FIPA-ACL possède 21 actes communicatifs, exprimés par des performatives, qui peuvent être groupées selon leur fonctionnalité de la façon suivante :

1. passage d'information : *inform**, *inform-if (macro act)*, *inform-ref (macro act)*, *confirm**, *disconfirm**
2. réquisition d'information : *query-if*, *query-ref*, *subscribe*
3. négociation : accept-proposal, *cfp*, *propose*, *reject-proposal*
4. distribution de tâches (ou exécution d'une action) : *request**, *request-when*, *request-whenever*, *agree*, *cancel*, *refuse*
5. manipulation des erreurs : *failure*, *not-understood*

Parmi la large librairie de performatifs proposée par ACL, nous n'avons eu à utiliser la performative suivante:

Request (Demande) : Communication par l'expéditeur d'une demande au destinataire d'effectuer une action.

Dans notre cas l'agent interface, utilise la performative Request, pour demander à l'agent ontologie d'exécuter des actions exemple (Chargement du modèle d'ontologie, exécution du processus de désambiguïsation, …).

II.3. Le fonctionnement des agents de M.S.E

Nous allons décrire à présent le fonctionnement des agents de notre système, pour le traitement d'une requête utilisateur.

Toute la communication entre agents est exécutée par messages FIPA ACL. Ces agents sont automatiquement créés et activés quand la plate-forme est activée.

Celle-ci est activée dans notre cas dans la page JSP.

II.3.1. Création des agents

Les agents du système sont crées et lancés, lorsque l'utilisateur lance une recherche.
La création des agents consiste en la création du conteneur, dans la plate forme JADE, dont lequel cohabiteront les agents ; puis en affectant à chaque agent un identifiant (AID) et finalement les lancer.
La figure suivante montre que le « container1 », contient les deux agents « ao » et « ia » qui sont respectivement les identifiants des agents « ontologie » et « interface »

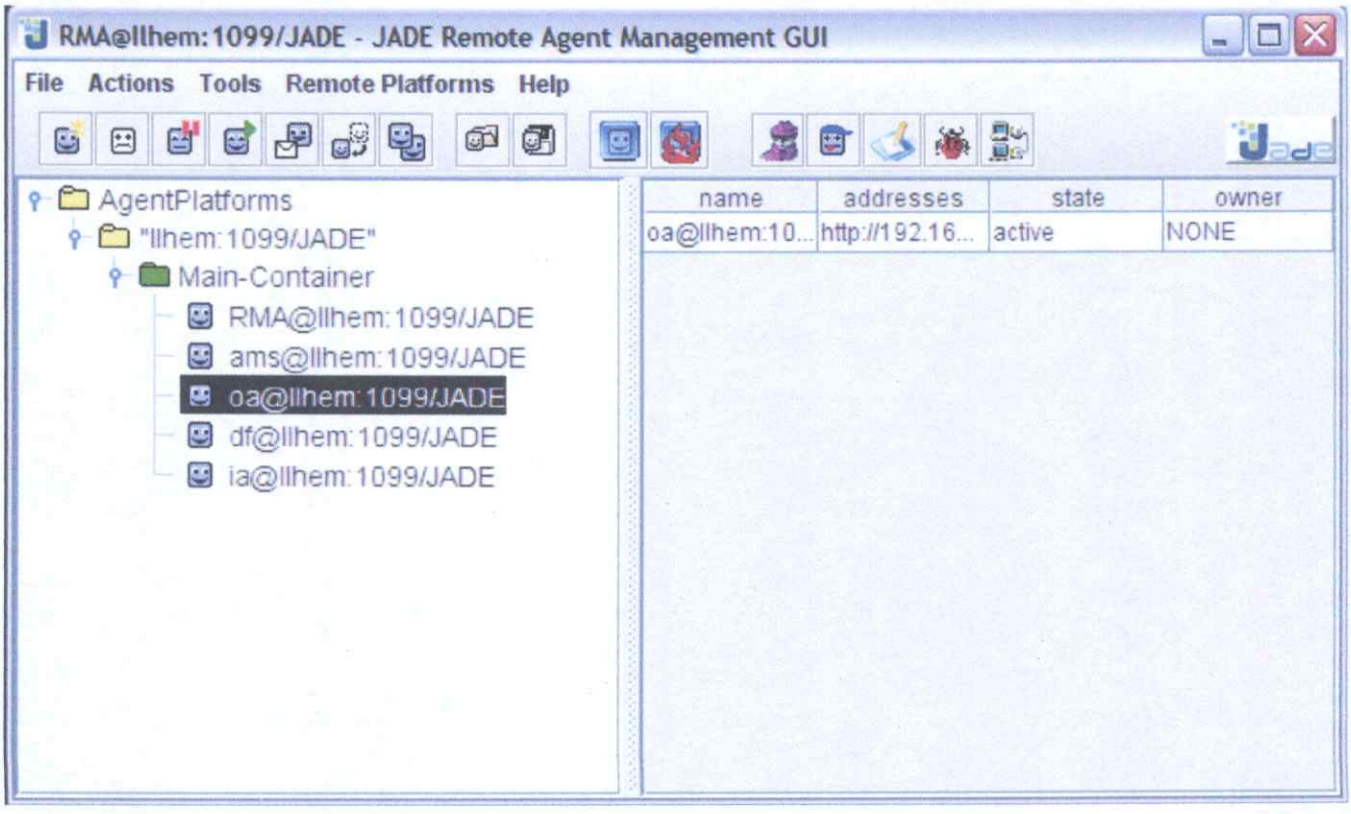


Figure V.13 : Création des agents dans Jade

II.3.2. Traitement de la requête

Après la création des agents, nous avons l'agent interface qui récupère la requête depuis la page JSP qui traite cette requête en effectuant des opérations sur celle-ci (élimination d'accent, des mots vides, caractères spéciaux ...)

II.3.3. les envois de messages

Après que l'agent ait traité la requête , commence la communication entre les agents. Les envois de massages sont de type FIPA-ACL , nous avons en 1er l'agent interface qui envoie un message, à l'agent ontologie qui était en attende ,contenant la requete traitée .
l'agent Interface envoi un message à l'agent ontologie de type request contenant la requete de l'utilisateur, ce dernier reçoi le message et effectue les operations adéquates puis renvoit le resultat a l'agent interface

III. REALISATION DE L'APPLICATION

L'application réalisée est une application web manipulant notre système Multi-agent, ce dernier est intégré dans l'application et fonctionne en arrière plan. Cela veut dire que lorsqu'un utilisateur demande le service offert par notre application, le système multi-agent prend en charge la demande de l'utilisateur sans que ce dernier sache qu'un système multi-agent lui offre ce service.

La figure V.14 montre l'interface de notre application ; à travers laquelle l'utilisateur interagit.

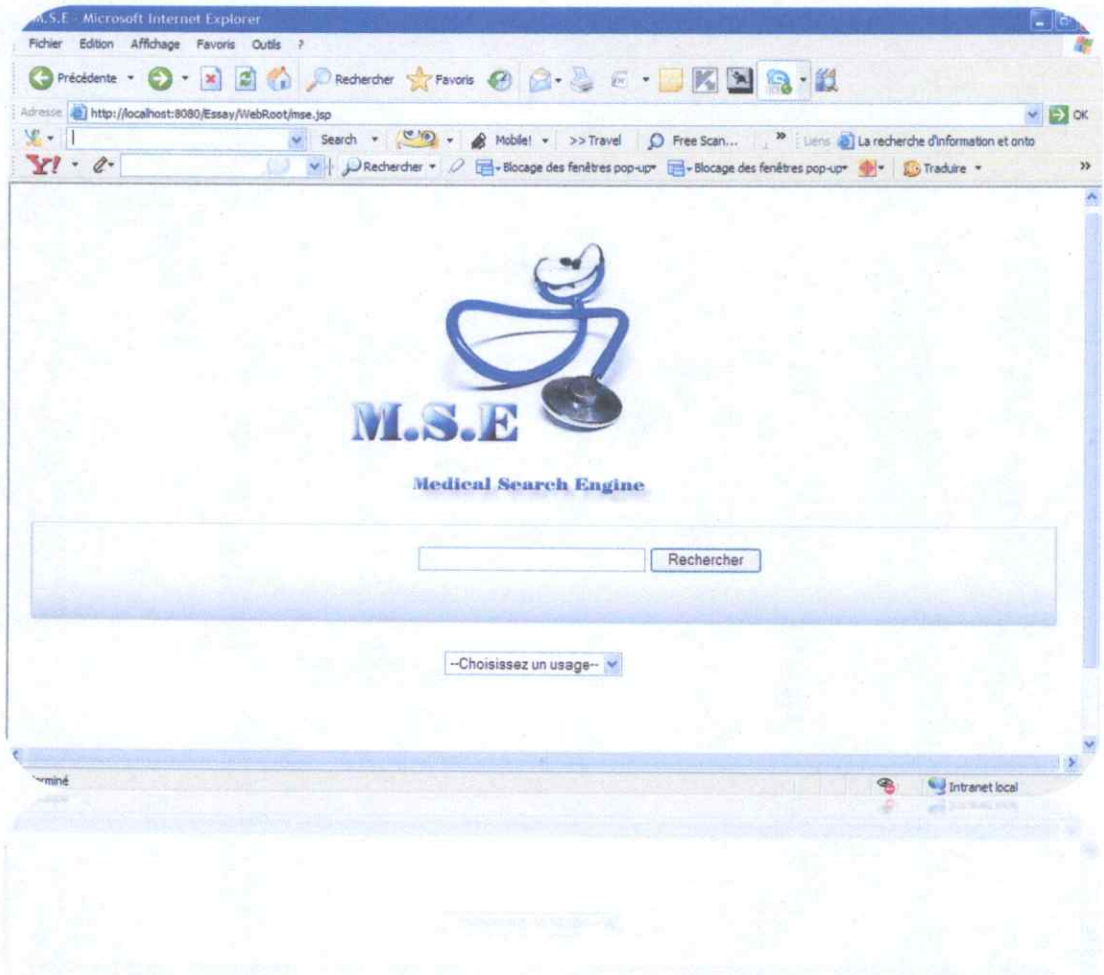


Figure V.14 : Interface de l'application

La figure V.15 représente les usages proposés et qui existent également dans notre ontologie.

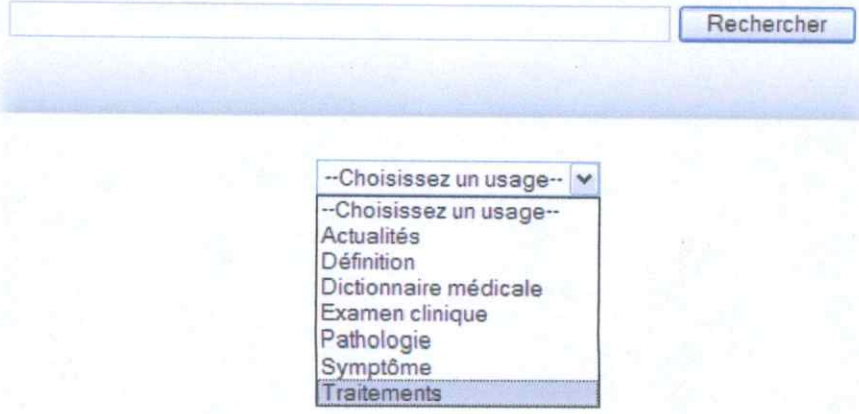


Figure V.15 : Liste des usages proposés

L'utilisateur a la possibilité d'effectuer une recherche directe sans spécifier l'usage. Pour cela il doit entrer sa requête et lancer sa recherche en cliquant sur le bouton « Rechercher », de ce fait le système multi-agent sera crée et prendra en charge la requête.
La requête sera traitée par l'agent interface et envoyée à l'agent ontologie pour effectuer la recherche dans l'ontologie.
L'agent ontologie cherche le concept (dans l'ontologie) qui sera le plus proche au niveau sémantique par rapport au mot clé saisit par l'utilisateur, ce concept sera envoyé à l'agent interface qui va le remettre au web service de Google pour effectuer la recherche dans le Web, et finalement les résultats seront affichés dans l'interface.

Exemple : la Figure V.16 représente les résultats d'une recherche effectuée (mouche volante), on remarque que les résultats retournés concernent aussi la MYODESOPSIES, puisque MYODESOPSIES est un synonyme de mouche volante dans l'ophtalmologie.

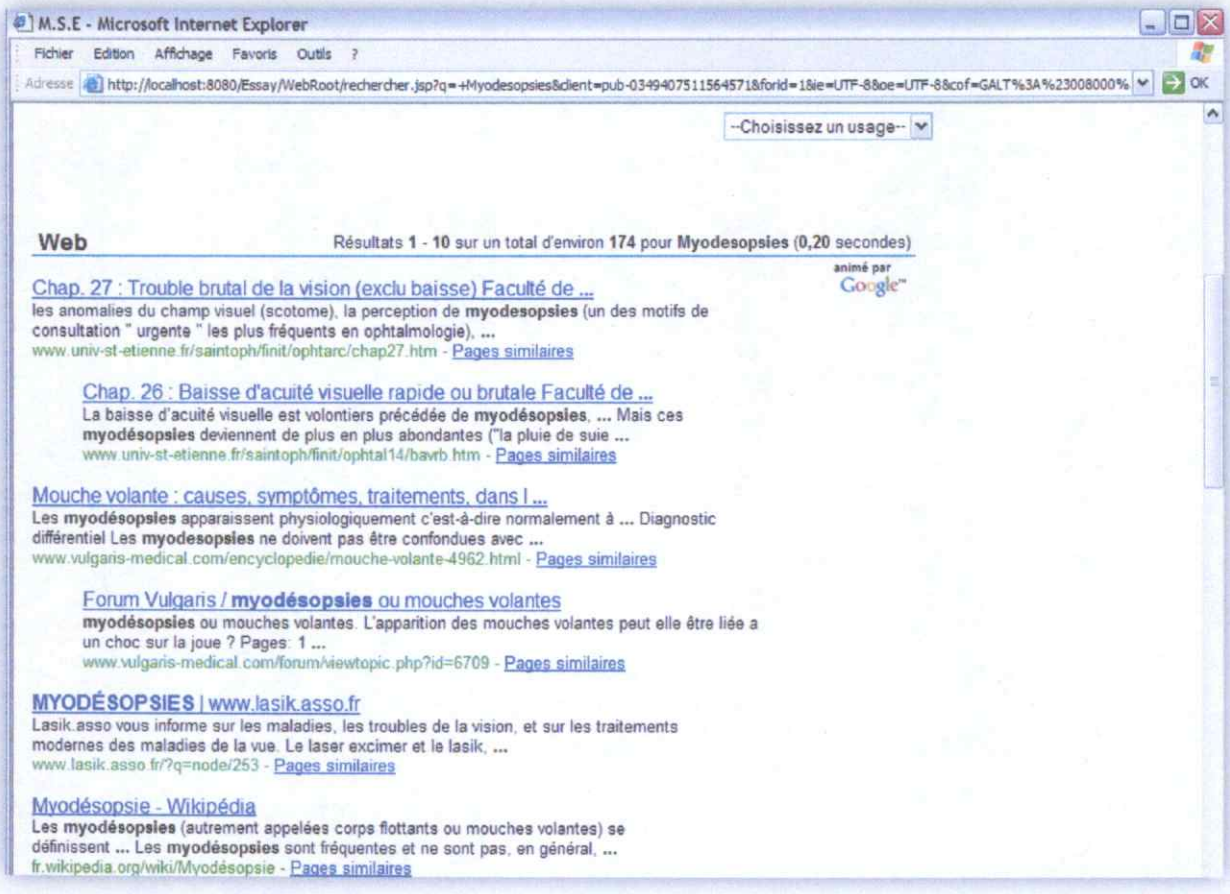


Figure V.16 : Résultat de recherche «mouche volante »

La 2eme possibilité, est que l'utilisateur saisisse sa requête et choisit un usage dans la liste déroulante, puis valide sa requête en cliquant sur le bouton « Rechercher », qui va lancer le système multi-agents, ce dernier va effectuer le même processus précédant en tenant compte de l'usage choisi.

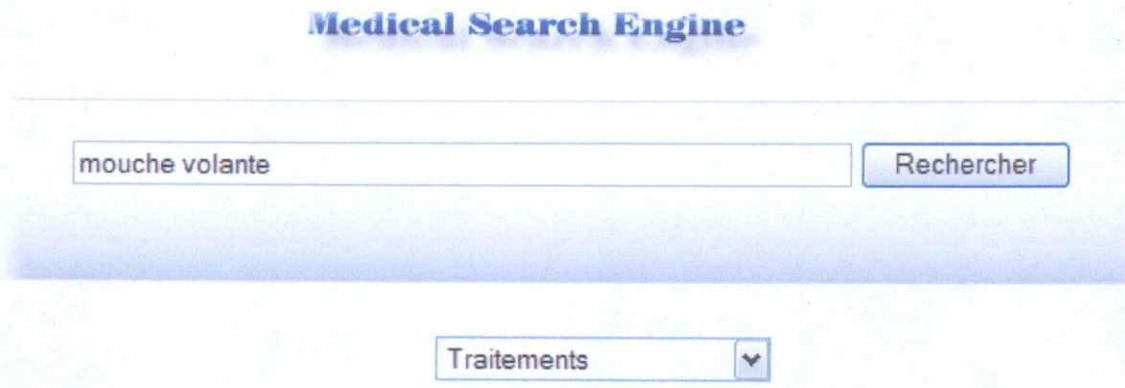


Figure V.17 : Saisie de la requête et choix de l'usage

La figure V.18 représente les résultats de la recherche « mouche volante » en tenant compte de l'usage choisi.

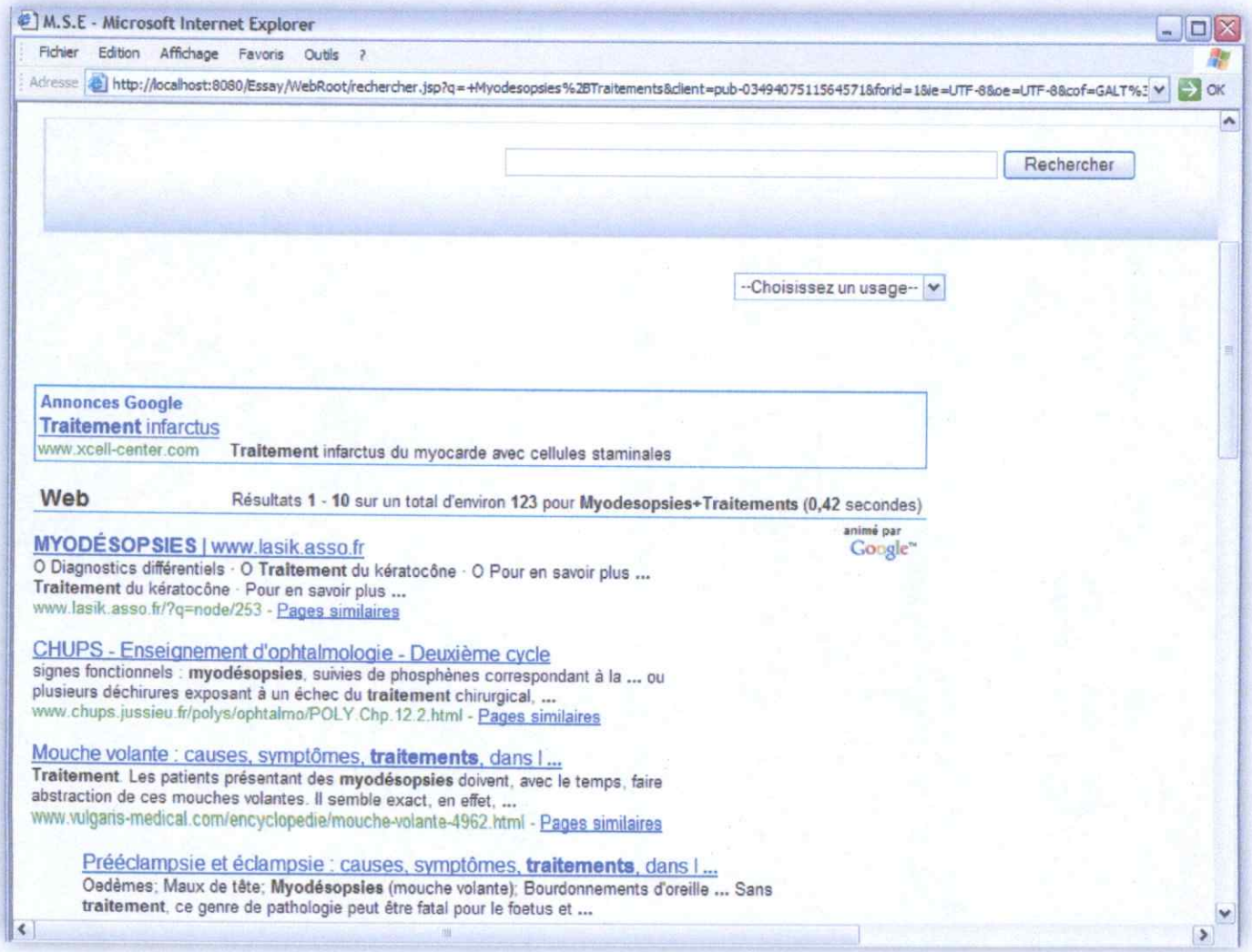


Figure V.18 : Résultat de la recherche « mouche volante avec traitement »

IV.TEST

IV.1. Logiciel de test utilisé

Le test de notre application a été effectué par le logiciel NeoLoad dans sa version 2.2

IV.2.NeoLoad [Web21]

NeoLoad est un outil de test de charge permettant de tester facilement et efficacement les applications web comme les intranets. Le test en charge permet de fiabiliser les applications web avant leur passage en production. Il permet d'étudier les statistiques sur le comportement du serveur et de montrer les erreurs et les problèmes de performance. **NeoLoad** simule des **utilisateurs virtuels** avec un comportement réaliste: ces utilisateurs virtuels sont capables de naviguer sur différentes pages, de remplir des formulaires avec des valeurs dynamiques.

IV.3.Technique de test

Le test s'effectue au départ en créant un profil d'utilisateur virtuel, en suite une population contenant différents type d'utilisateur est crée pour générer une charge réaliste sur l'application. A la fin un scénario de test est crée avec des paramètres d'exécution spécifiées.

Après le lancement du test, les statistiques peuvent être observées en temps réel, jusqu'à la création du résumé des résultats à la fin du test.

Notre scénario de test est: **« Lancer une recherche »**, avec une population d'utilisateurs virtuels comptée à 10 personnes (par défaut), et avec une durée de test estimée à 2 minutes et les résultats obtenus sont les suivants :

Synthèse

Synthèse des Résultats

Projet:	Recherche
Scénario:	scenario1
Date de début:	21 oct. 2007 21:55:44
Date de fin:	21 oct. 2007 21:57:44
Durée:	00:02:00
Injecteurs:	Localhost
Variation de charge:	La population **Population1** est constante avec **10** utilisateurs.
Description:	Lancer une recherche

Synthèse des Statistiques

Pages/s moyen	1	**Nb Pages total**	135
Temps de réponse moyen de page	1,88s	**Débit total**	0.99 Mo

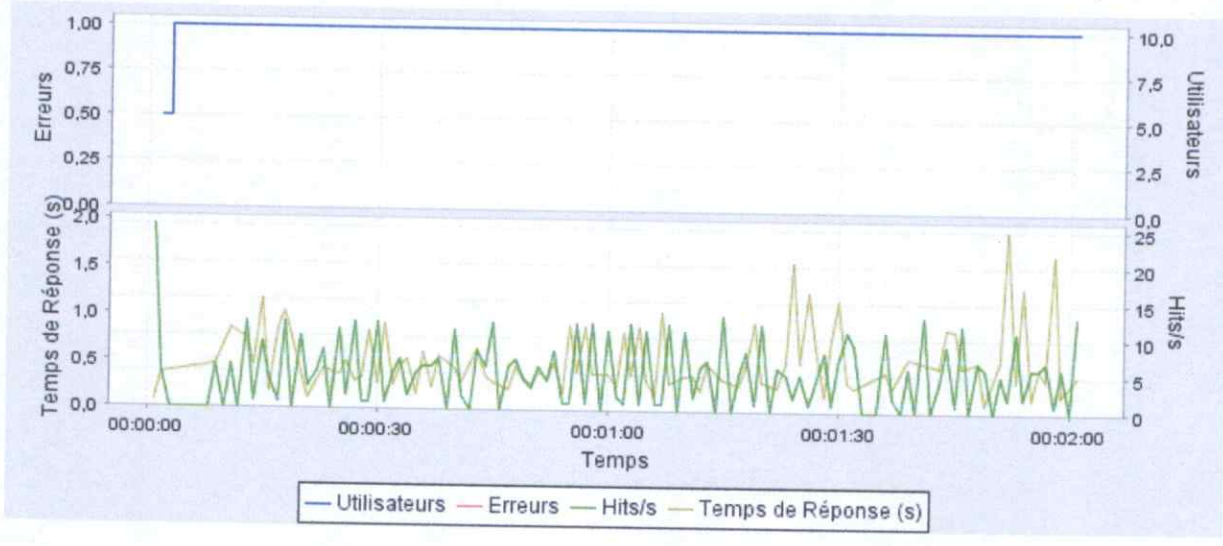


Figure V.19 : Synthèse des Statistiques

Graphes principaux

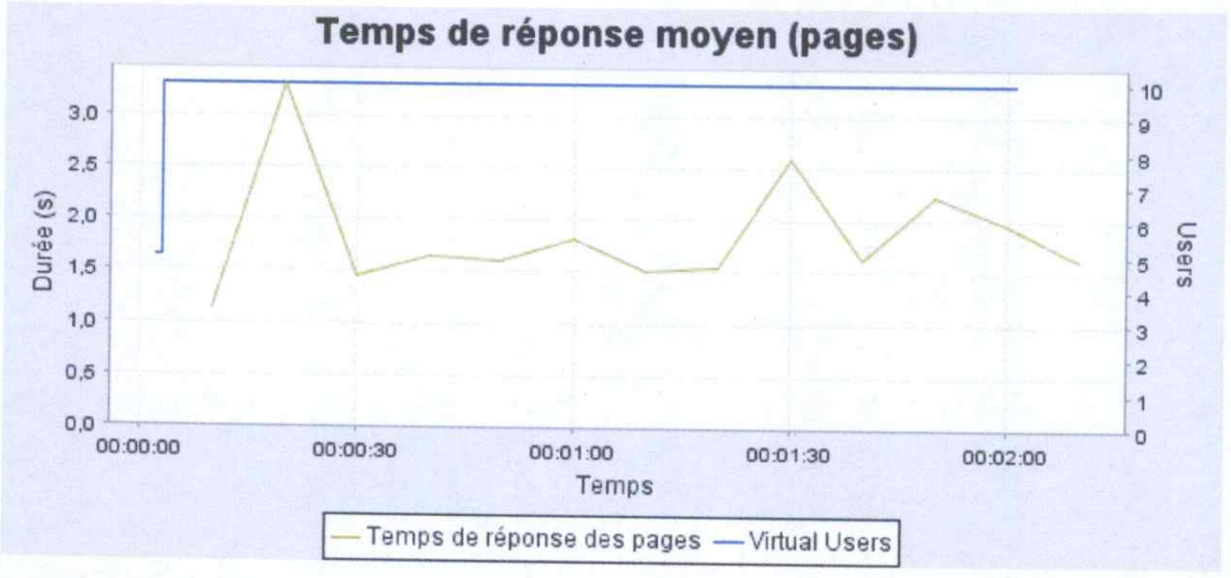


Figure V.20 : Temps de réponse moyen (pages)

Graph Min	Moyenne	Graph Max	Médiane	Moyenne 90%	Ecart Type
1,12	1,88	3,3	1,62	1,83	0,56

Notes

Affiche le temps de réponse moyen, en secondes, de toutes les pages pendant le test.

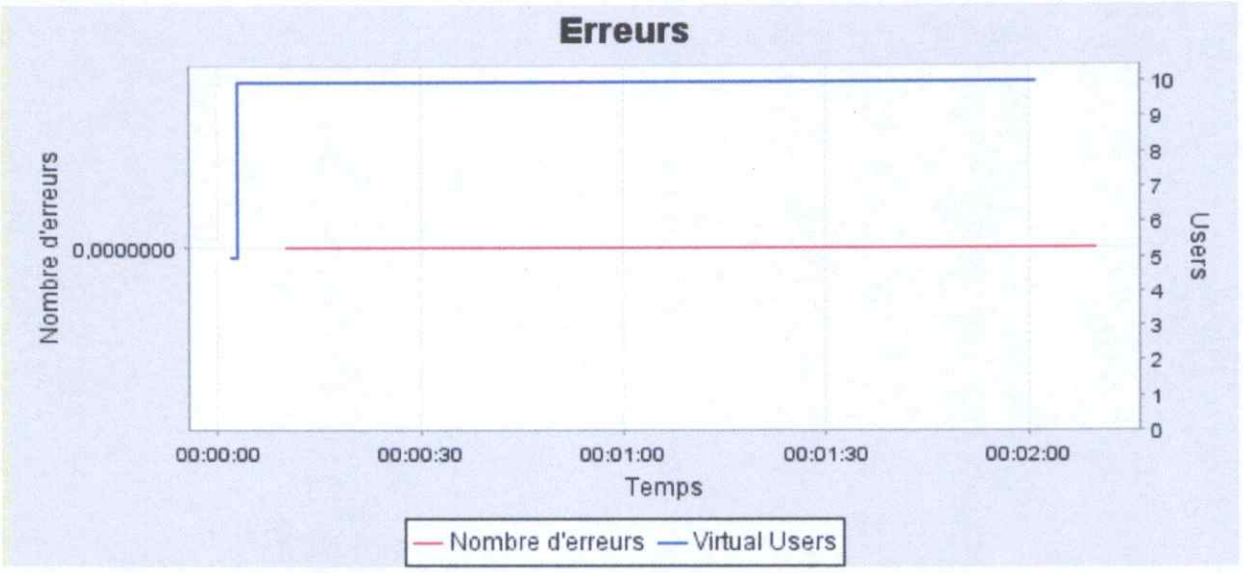


Figure V.21 : Erreurs

Graph Min	Moyenne	Graph Max	Médiane	Moyenne 90%	Ecart Type
0	0	0	0	0	0

Figure V.22 : Distribution du temps de réponse des pages

Notes

Affiche le pourcentage de pages qui ont été jouées avec un temps de réponse donné. Ce graphe permet de déterminer le pourcentage de pages qui vérifient un objectif de performance donné. Par exemple, le graphe peut montrer que 90% des pages ont un temps de réponse sous n secondes.

CONCLUSION ET PERSPECTIVES

Une ontologie définit les termes utilisés pour décrire et représenter un champ d'expertise. Les ontologies sont utilisées par les personnes, les bases de données, et les applications qui ont besoin de partager des informations relatives à un domaine bien spécifique, comme la médecine, la fabrication d'outils, l'immobilier, la réparation d'automobiles, la gestion de finances, etc. Les ontologies associent les concepts de base d'un domaine précis et les relations entre ces concepts, tout cela d'une manière compréhensible par les machines. Elles encodent la connaissance d'un domaine particulier ainsi que les connaissances qui recouvrent d'autres domaines, ce qui permet de rendre les connaissances réutilisables.

L'objectif de notre travail était d'essayer d'améliorer la pertinence de la recherche d'information, en diminuant le bruit et le silence, pour cela nous avons développé une ontologie médicale intitulée « OntoMed » qui manipule le contenu sémantique des ressources médicales, dans notre cas « Ophtalmologie » et « Cancérologie ».

Afin d'exploiter au mieux cette ontologie, nous avons développé un système de recherche baptisé « M.S.E » basé sur la technologie agent et ayant une ontologie comme base de connaissance.

Nous avons réalisé un prototype constitué de deux agents de ce système, ce produit est suffisamment efficace pour montrer l'apport des ontologies dans la recherche d'information.

Au cours de ce projet, nous avons acquis certaines connaissances dans l'ingénierie ontologique, et les systèmes multi agents, vu que ce sont deux technologies récentes, que ce soit dans le cadre des systèmes multi-agents et l'utilisation de la plateforme JADE, que dans celui portant sur les ontologies, dans les perspectives nous pouvons offrir les extensions suivantes :

-Le développement d'un agent web qui réalisera le processus d'indexation des pages web, basé sur l'ontologie.
-Etendre le système à l'utilisation de plusieurs ontologies de domaines.
-Réaliser un système multi agents plus robuste incluant la négociation entre agents, la planification,

-Développer un processus d'enrichissement des ontologies par l'utilisation de thésaurus et la participation des experts (Enrichissement coopératif des ontologies).

BIBLIOGRAPHIES

BIBLIOGRAPHIE RECHERCHE D'INFORMATION

[Web01]: http://www.annuaire-referencement.net/

[Web02]:
http://www.uhb.fr/urfist/Supports/FormaDoctor/FormDoctOutilsRech.htm#Avantages,%20inconvénients%20des%20annuaires

[Web03]:
http://www.bibliotheques.uqam.ca/InfoSphere/sciences_humaines/module5/moteurs.html

[Web04]:
http://www.medsyn.fr/perso/g.perrin/recherche/moteurs/avantagesETinconvenients.htm

[Web05]: http://www.asktibbs.com/php/rubrique.php3?id_rubrique=9

[Web06]: http://www.mpl.ird.fr/documentation/indexation/bruit.htm

[Web07]: http://urfist.univ-lyon1.fr/risi/45-probl.html

[Web08]:
http://www.uhb.fr/urfist/Supports/RechInfoInit/RechInfo3Problematique.html#3.1.2%20Typologies%20des%20recherches

[Web09]:
http://wcentre.tours.inra.fr/urbase/internet/documentation/recherche_information/AnnuaireOuMoteur.htm

[Web10]: http://fr.wikipedia.org/wiki/Recherche_d'information

[20] : Fraihat et Hasani « conception et réalisation d'un moteur de recherche basé sur les ontologies » Mémoire INI, 2004

BIBLIOGRAPHIE ONTOLOGIE POUR LA RI

[1]: Maxime Morneau, *Recherche d'information sémantique et extraction automatique d'ontologie du domaine*, 2006
« http://www.theses.ulaval.ca/2006/23828/23828.pdf »

[2] : Bulletin de l'afia : association française pour l'intelligence artificiel, avril 2003 N°54. « http://www.lalic.paris4.sorbonne.fr/stic/articles/websemantique.pdf»

[3] : doctorat Audrey BANEYX, thèse de doctorat : *construire une ontologie de la pneumologie, (février 2007).*
« http://tel.archives-ouvertes.fr/docs/00/17/62/24/PDF/MANUSCRIT_BANEYX.pdf »

[4] : Lylia ABROUK, thèse de doctorat : *annotation de documents par le contexte de citation basé sur une ontologie, novembre 2006.*
« http://www.lirmm.fr/~abrouk/These/these.pdf »

[5] : Valéry Psyché ,Olavo Mendes ,Jacqueline Bourdeau, Article de recherche, Apport de l'ingénierie ontologique aux environnements de formation à distance,2003 à : http://sticef.univ-lemans.fr/num/vol2003/psyche-06s/sticef_2003_psyche_06s.htm

[6] : Valéry Psyché, L'ontologie Apport en éducation *: Dans le cadre du cours DIC9340 Systèmes à Base de Connaissances, juin 2005.*

[7] : Yazid GRIM, Mémoire de fin d'étude INI *: « Conception d'une ontologie pour la recherche d'information à partir de sources d'informations distribuées et hétérogènes. »(20004/2005)*

[8] : FRÉDÉRIC FÜRST, Rapport de recherche *: l'ingénierie ontologique* Nantes: Institut de Recherche en Informatique de Nantes, *(octobre 2002).*
« http://lina.atlanstic.net/documents/RR_pdfs/RR-IRIN-0207.pdf »

[9] : Tony DUJARDIN, *De l'apport des ontologies pour la conception de systèmes multi-agents ouverts.*
« http://www2.lifl.fr/~dujardit/recherche/presentation-DEA.pdf. »

[10]: Delia Codruta ROGOZAN, *Proposition de recherche doctorale en informatique cognitive, Gestion de l'évolution d'une ontologie : méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions de l'ontologie.*
« http://www.dic.dinfo.uqam.ca/enseignement/rogozanprojet.pdf »

[11] : Amira TIFOU, Mémoire de fin d'étude INI : « Conception et réalisation d'un Système Multi-Agents pour la Recherche d'information, basée sur les Ontologies, dans des bases de données hétérogènes » 2004/2005

[Web11]: Ontologies informatique , 22/05/06
http://interstices.info/display.jsp?id=c_17672&part=1

[Web12]: http://www.greyc.ensicaen.fr/~chris/Cours_ws_cp_2005.htm

BIBLIOGRAPHIE SYSTEME MULTI-AGENT

[Web13]: http://www.vieartificielle.com/article/index.php?id=47

[Web14]: http://fr.wikipedia.org/wiki/Agent_(informatique)

[Web15] :
http://www.limsi.fr/~jps/enseignement/examsma/2005/1.plateformes_2/index.htm

[12] : A. MOUSSAOUI « LA PLANIFICATION DE PROCESSUS D'USINAGE DANS UN ATELIER INTELLIGENCE ARTIFICIELLE, LES SYSTÈMES MULTI-AGENTS : ÉTAT DE L'ART » U.S.T.H.B, 2000 (http://www.chez.com/produ/halim/)

BIBLIOGRAPHIE CONCEPTION DU SYSTEME

[13] : Natalya F. Noy et Deborah L. McGuinness, « Développement d'une ontologie 101 : Guide pour la création de votre première ontologie », Université de Stanford, Stanford, CA, 94305, 2002.
« http://www.bnf.fr/PAGES/infopro/normes/pdf/no-DevOnto.pdf »

[14]: Oscar Corcho1, Mariano Fernández-López, Asunción Gómez-Pérez, Angel López-Cima, « Building legal ontologies with METHONTOLOGY and
WebODE » , Facultad de Informática. Universidad Politécnica de Madrid
Campus de Montegancedo, s/n. 28660 Boadilla del Monte. Madrid. Spain
«http://www.cs.man.ac.uk/~ocorcho/documents/LawSemWeb2004_CorchoEtAl.pdf»

[15] : Gomez Pérez, Fernandez-Lopzez, corcho , « Ontological Engineering » ,Springer 2004.

[Web16]: http://laurent-piechocki.developpez.com/uml/tutoriel/Up/

[Web17]: http://fdigallo.online.fr/cours/uml.pdf

[16] : DI GALLO Frédéric, Méthodologie des systèmes d'information – UML, CNAM ANGOULEME 2000-2001

[17] : Pierre-Alain Muller, modélisation objet avec UML, EYROLLES, 2001.

[18]: L.Khan. « Ontology-based Information Selection ». Rapport de recherche.University of Southern California 2000.
«http://esc.utdallas.edu/publications/lkhan_def.pdf»

[19] : Jean-Pierre Briot et Yves Demazeau, « Principes et architecture des systèmes multi agents », Lavoisier 2001.

BIBLIOGRAPHIE IMPLEMENTATION ET TEST

[Web18]: M.HERLIN, Introduction à JENA, Présentation de cours, Département Informatique et de la recherche opérationnelle, Université de Montréal, canada 2007 www.iro.umontreal.ca/~lapalme/ift6281/Presentations/Jena.ppt

[Web19]: http://fr.wikipedia.org/wiki/Eclipse_(logiciel)

[Web20]:
http://wiki.improve.fr/wiki/_media/moni/eclipsercp.presentation.courte.ppt?id=moni%3Aarticles&cache=cache

[Web21] : www.neotys.fr

LES ANNEXES

ANNEXE A

LA PLATE-FORME JADE

1. Introduction

Le meilleur moyen pour construire un système multi-agent(SMA) est d'utiliser une plate-forme multi-agent. Une plate-forme multi-agent est un ensemble d'outils nécessaire à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test du SMA ainsi créé. Ces outils peuvent être sous la forme d'environnement de programmation (API) et d'applications permettant d'aider le développeur. Nous allons étudier dans cette partie la plate-forme JADE(Java Agent DEvelopment framework).

2. Bref description de JADE

JADE (Java Agent DEvelopement framework) est une plate-forme multi-agent créé par le laboratoire TILAB [TILAB] et décrite par Bellifemine et al. Dans [BEL 99][BEL 00]. JADE permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA [FIPA 00][FIPA 02]. Elle est implémentée en JAVA et fourni des classes qui implémentent « JESS » pour la définition du comportement des agents. JADE possède trois modules principaux (nécessaire aux normes FIPA).

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

Ces trois modules sont activés à chaque démarrage de la plate-forme.

3 .La norme FIPA

La FIPA (Foundation for Intelligent Physical Agents) est une organisation à but non lucratif fondée en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes. Par la combinaison d'actes de langages, de logique des prédicats et d'ontologies publiques, la FIPA cherche à offrir des moyens standardisés permettant d'interpréter les communications entre agents de manière à respecter leur sens initial, ce qui

est bien plus ambitieux que XML, qui ne standardise que la structure syntaxique des documents. Afin d'atteindre ce but, le FIPA émet des standards couvrant :

- Les applications (applications nomades, agent de voyage personnel, applications de diffusion audiovisuelles, gestion de réseaux, assistant personnel...) ;
- Les architectures abstraites, définissant d'une manière générale les architectures d'agents ;
- Les langages d'interaction (ACL), les langages de contenu (comme SL, CCL, KIF ou RDF) et les protocoles d'interaction ;
- La gestion des agents (nommage, cycle de vie, description, mobilité, configuration);
- Le transport des messages : représentation (textuelle, binaire ou XML) des messages ACL, transport (par IIOP, WAP ou HTTP) de ces messages.

Ces standards évoluent, et sont régulièrement mis à jour, ainsi que de nouveaux standards qui sont nouvellement proposés. Les standards qu'édicte la FIPA ne constituent pas vraiment une plate-forme de construction multi-agents. Ce n'est pas non plus l'objectif que s'est fixé la FIPA. Tout au plus, la FIPA normalise une plate-forme d'exécution standardisée dans un but d'interopérabilité. Ces normes s'appliquent donc pour la plupart en phase de déploiement. Elles n'abordent pas les phases d'analyse ni de conception. Elles peuvent cependant guider certains choix d'implémentation.

3.1 Architecture logiciel de la plate-forme JADE

JADE reprend donc l'architecture de l'Agent Management Reference Model proposé par FIPA. Les différents modules présentés dans la figure suivante sont présentés sous forme de services. Les services de base proposés sont le Directory Facilitator (DF) et l'Agent Management System (AMS). Il est possible de lui demander de tenir en plus le service de Message Transport Service (MTS) pour communiquer entre plusieurs plates-formes. Mais ce service sera chargé à la demande pour ne conserver par défaut que les fonctionnalités utiles à tout type d'utilisation.

L'agent est l'acteur fondamental de la plate-forme, un Agent Identifier (AID) identifie un agent de manière unique. Le DF est un composant qui fait office d'annuaire. C'est un service de « pages jaunes » qui permet de mettre en relation les agents avec leurs compétences. Un agent peut enregistrer ses compétences dans le DF ou interroger le DF pour connaître les compétences proposées par les autres agents. L'AMS est un autre composant important car il contrôle l'accès et l'utilisation de la plate-forme et maintient un répertoire contenant les adresses de transport des agents de la plate forme. Ce service est plus un service de type « pages blanches » qui effectue la correspondance entre l'agent et l'AID. Chaque agent doit s'enregistrer à un AMS pour avoir un AID. Il n'y a qu'un AMS par plate-forme. Le MTS est une méthode par défaut de communication entre agents de différentes plates-formes. Cela permet l'interconnexion entre systèmes hétérogènes ou tout au moins de système ne communicant pas de la même façon. L'Agent Platform (AP) constitue l'infrastructure physique sur laquelle se déploient les agents. Il contient le DF, l'AMS et le MTS. Lorsqu'on parle de AP, on inclut souvent le matériel électronique, l'OS, le software et les composants cités ci-dessus avec les agents. Enfin, l'Agent Identifier (AID) est un identifiant précis d'un agent. On lui donne plusieurs paramètres tels que l'adresse de transport, l'adresse de service de résolution de nom, ... Un exemple est : name@HAP (Home Agent Platform)

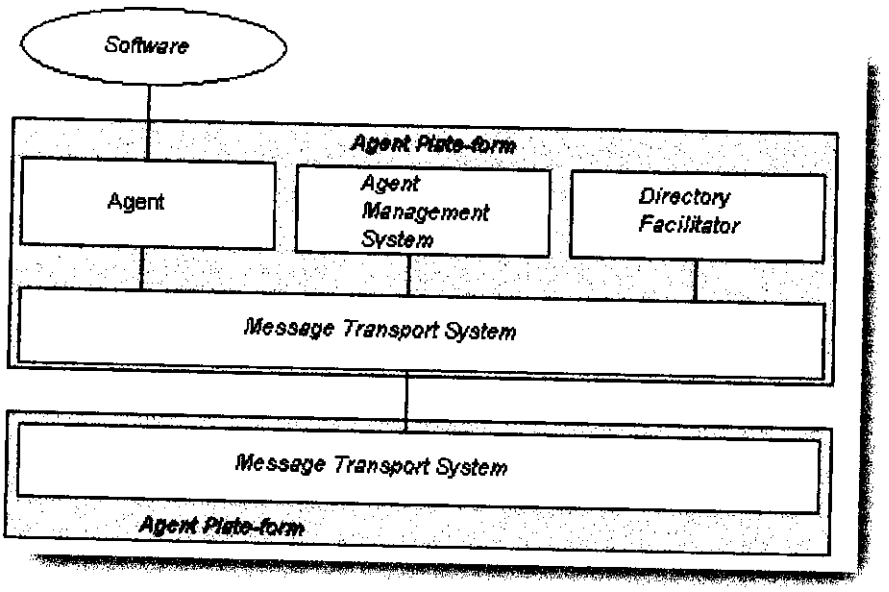


Figure 1 : Architecture logiciel de La plate-forme JADE

Dans la plate-forme JADE, deux méthodes sont fournies par la classe Agent afin d'obtenir l'identifiant de l'agent DF par défaut et celui de l'agent AMS : getDefaultDF() et respectivement getAMS(). Ces deux agents permettent de maintenir une liste des services et des adresses de tous les autres agents de la plate-forme. Le service DF propose quatre méthodes afin de pouvoir :

- Enregistrer un agent dans les pages jaunes (register).
- Supprimer un agent des pages jaunes (deregister).
- Modifier le nom d'un service fourni par un agent (modify).
- Rechercher un service (search).

Le service AMS s'utilise généralement de manière transparente (chaque agent créé est automatiquement enregistré auprès de l'AMS et se voit attribué une adresse unique). Ces deux services fournissent donc les annuaires qui permettent à n'importe quel agent de trouver un service ou un autre agent de la plate-forme.

3.2 Langage de communication de la plate-forme JADE

Le langage de Communication de la plate-forme JADE est FIPA-ACL(Agent Communication language). La classe ACLMessage représente les messages qui peuvent être échangés par les agents. La communication de messages se fait en mode asynchrone. Lorsqu'un agent souhaite envoyer un message, il doit créer un nouvel objet ACLMessage, compléter ces champs avec des valeurs appropriées et enfin appeler la méthode send(). Lorsqu'un agent souhaite recevoir un message, il doit employer la méthode receive() ou la méthode blockingReceive(). Un message ACL dispose obligatoirement des champs suivants :

Performative :	type de l'acte de communication
Sender :	expéditeur du message
Receiver :	destinataire du message

reply-to :	participant de la communication
content :	contenu du message
language :	description du contenu
encoding :	description du contenu
ontology :	description du contenu
protocol :	contrôle de la communication
conversation-id :	contrôle de la communication
reply-with :	contrôle de la communication
in-reply-to :	contrôle de la communication
reply-by :	contrôle de la communication

Tous les attributs de la classe ACLMessage peuvent être obtenus et modifiés par les méthodes set/get(). Le contenu des messages peut être aussi bien du texte que des objets car la sérialisation Java est supportée.

3.3 Les protocoles FIPA-Query et FIPA-Request

Dans le protocole FIPA-request, un agent sollicite un autre agent pour exécuter des actions et l'agent récepteur retourne soit une réponse favorable à l'exécution d'actions, soit une réponse défavorable expliquée par telle ou telle raison.

Supposons que l'agent ***i*** ait besoin de l'agent ***j*** pour exécuter l'action « action ».

L'agent ***i*** envoie « request » à l'agent ***j***. Si l'agent ***j*** accepte la requête, il retourne « agree ». Ensuite, quand ***j*** a fini d'exécuter « action », il en informe ***i*** en utilisant « inform ».

a. Si l'agent ***j*** accepte mais rencontre un problème durant le traitement de « action », il retourne « failure » et les raisons de l'échec.
b. Si l'agent ***j*** n'accepte pas la requête de l'agent ***i***, ***j*** retourne « refuse » et les raisons de ce refus.

Le protocole FIPA-Query signifie que l'agent émetteur sollicite l'agent récepteur pour exécuter un des types d'un performatif « inform », c'est-à-dire pour répondre à la demande. Supposons que l'agent ***i*** fasse une demande à l'agent ***j***.

a. L'agent ***i*** envoie un performatif « query » à l'agent ***j***. Si l'agent ***j*** peut répondre à la demande, il l'informe en utilisant le performatif « inform ».
b. Si l'agent ***j*** a essayé de répondre à la demande mais qu'il ne le peut pas, il retourne « failure » et les raisons de cette impossibilité.
c. Si l'agent ***j*** refuse de répondre à la demande, il retourne « refuse » et les raisons de ce refus.

3.4 Le protocole Contract-Net de JADE

Le Protocole Contract-Net (CNP) a été défini par Smith [SMI 80]. Il est un mécanisme de négociation par appel d'offre (ou Contract) entre deux types d'agents : l'agent gestionnaire et les agents contractants. L'agent gestionnaire, souhaitant sous-traiter une tâche qu'il doit accomplir, est l'initiateur du contrat. Chaque agent contractant est un agent auquel on propose ce contrat. Le fonctionnement du CNP à été décrit dans [SCD 92]. Ce protocole très souvent utilisé, a été normalisé par l'organisation FIPA et implémenté dans la plate-forme JADE.

4. Comportements des agents dans la plate-forme JADE

Un agent doit être capable de gérer plusieurs tâches de manière concurrente en réponse à différents évènements extérieurs. Afin de rendre efficace cette gestion chaque agent de JADE est composé d'un seul thread et chaque comportement qui le compose est en fait un objet de type Behaviour. Des agents multi-thread peuvent être créés mais il n'existe pour l'heure actuelle aucun support fournis par la plate-forme (excepté la synchronisation de la file des messages ACL).

Afin d'implémenter un comportement, le développeur doit définir un ou plusieurs objets de la classe Behaviour, les instancier et les ajouter à la file des tâches « ready »de l'agent. Il est à noter qu'il est possible d'ajouter des comportements et sous-comportements à un agent ailleurs que dans la méthode setup().

Tout objet de type Behaviour dispose d'une méthode action() (qui constitue le traitement à effectuer par celui-ci) ainsi que d'une méthode done() (qui vérifie si le traitement est terminé). Dans les détails, l'ordonnanceur exécute la méthode action() de chaque objet Behaviour présent dans la file des tâches de l'agent. Une fois cette méthode terminée, la méthode done() est invoquée. Si la tâche a été complétée alors l'objet Behaviour est retiré de la file.
L'ordonnanceur est non-préemptif et n'exécute qu'un seul comportement à la fois, on peut donc considérer la méthode action() comme étant atomique. Il est alors nécessaire de prendre certaines précautions lors de l'implémentation de cette dernière, à savoir éviter des boucles infinies ou des opérations trop longues. La façon la plus classique de programmer un comportement consiste à le décrire comme une machine à états finis. L'état courant de l'agent étant conservé dans des variables locales.
Enfin, il existe également quelques méthodes supplémentaires afin de gérer les objets Behaviour :

- reset() qui permet de réinitialiser le comportement;
- onStart() qui défini des opérations à effectuer avant d'exécuter la méthode action();
- onEnd() qui finalise l'exécution de l'objet Behaviour avant qu'il ne soit retiré de la liste des comportements de l'agent;

La plate-forme JADE fournit sous forme de classes un ensemble de comportements ainsi que des sous-comportements prêt à l'emploi. Elle peut les exécuter selon un schéma prédéfini, par exemple la classe SequentialBehaviour est supportée et exécute des sous-comportements de manière séquentielle. Toutes les classes prédéfinies dans JADE héritent de la classe Abstraite Behaviour. On peut les citer :

- Classe SimpleBehaviour (abstraite): modélise un comportement simple. Sa méthode reset() n'effectue aucune opération.
- Classe CompositeBehaviour (abstraite) : modélise un comportement composé. Les actions effectuées par cette classe sont définies dans les comportements enfants.
- Classe FSMBehaviour : Cette classe hérite de CompositeBehaviour et éxecute des comportements enfants suivant un automate à états finis défini par l'utilisateur. Lorsqu'un comportement enfant termine, sa valeur de fin retournée par la fonction onEnd() indique le prochain état à atteindre. Le comportement correspondant à cet état sera exécuté à la prochaine exécution de la classe. Elle se termine lorsque qu'un comportement associé à un état final à été exécuté.
- Classe SenderBehaviour : elle étend la classe OneShotBehaviour et encapsule une unité atomique qui effectue une opération d'envoie de message.
- Classe ReceiverBehaviour : Elle encapsule une unité atomique qui effectue une opération de réception de message. Ce comportement s'arrête dés qu'un message a été reçu. S'il n'y a pas de message dans la file d'attente de l'agent ou que le message ne correspond pas au MessageTemplate du constructeur de cette classe, alors il se met en attente.
- On peut aussi citer d'autres classes par exemples : Classe WakerBehaviour (abstraite), ParrallelBehaviour, SequentialBehaviour, CyclicBehaviour. (abstraite), OneShotBehaviour (abstraite).

5 .Outils de débogage de JADE

Pour supporter la tâche difficile du débogage des applications multi-agents, des outils ont été développés dans la plate-forme JADE. Chaque outil est empaqueté comme un agent, obéissant aux mêmes règles, aux mêmes possibilités de communication et aux mêmes cycles de vie d'un agent générique (agentification de service).

5.1 Agent RMA Remote Management Agent

Le RMA permet de contrôler le cycle de vie de la plate-forme et tous les agents la composant. L'architecture répartie de JADE permet le contrôle à distance d'une autre plate-forme. Plusieurs RMA peuvent être lancés sur la même plate-forme du moment qu'ils ont des noms distincts.

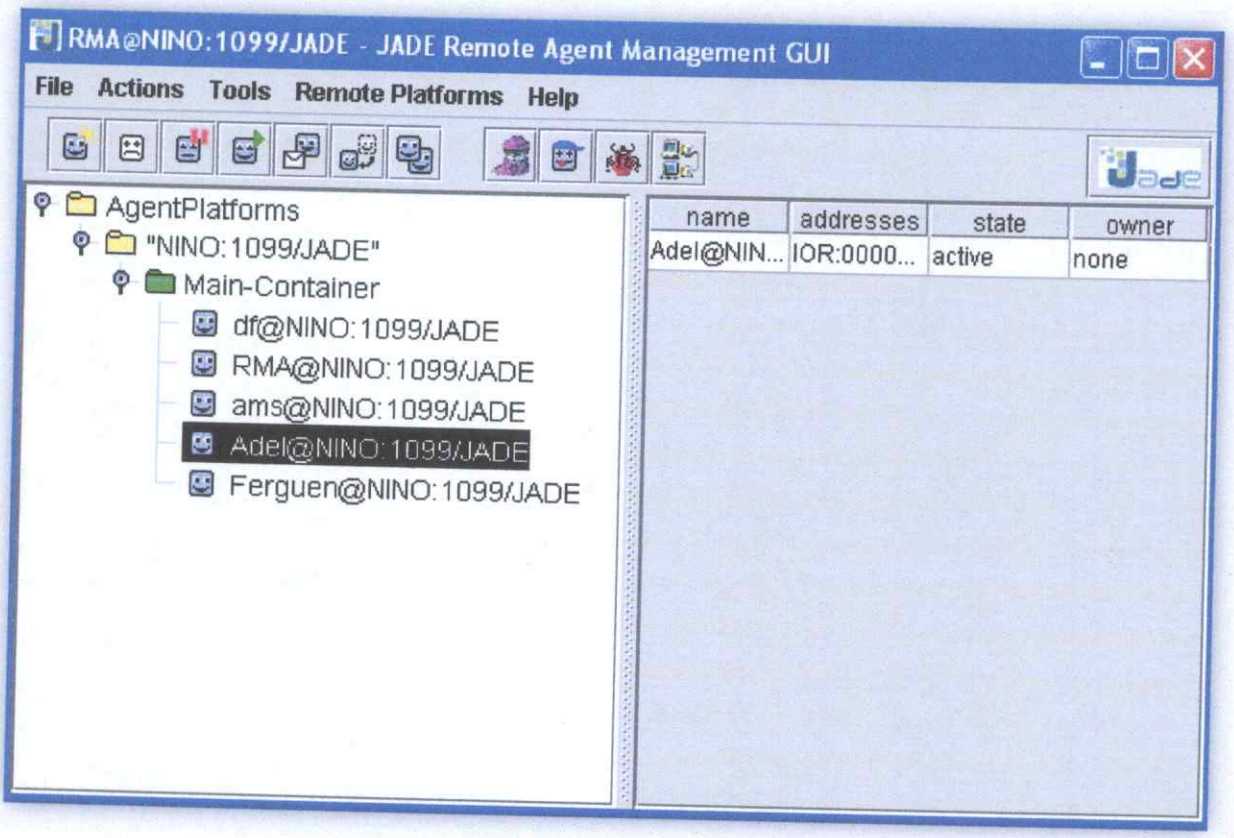


Figure 2 : L'interface de l'agent RMA

5.2 Agent Dammy

L'outil DummyAgent permet aux utilisateurs d'interagir avec les agents JADE d'une façon particulière. L'interface permet la composition et l'envoi de messages ACL et maintient une liste de messages ACL envoyés et reçus. Cette liste peut être examinée par l'utilisateur et chaque message peut être vu en détail ou même édité. Plus encore, le message peut être sauvegardé sur le disque et renvoyé plus tard.

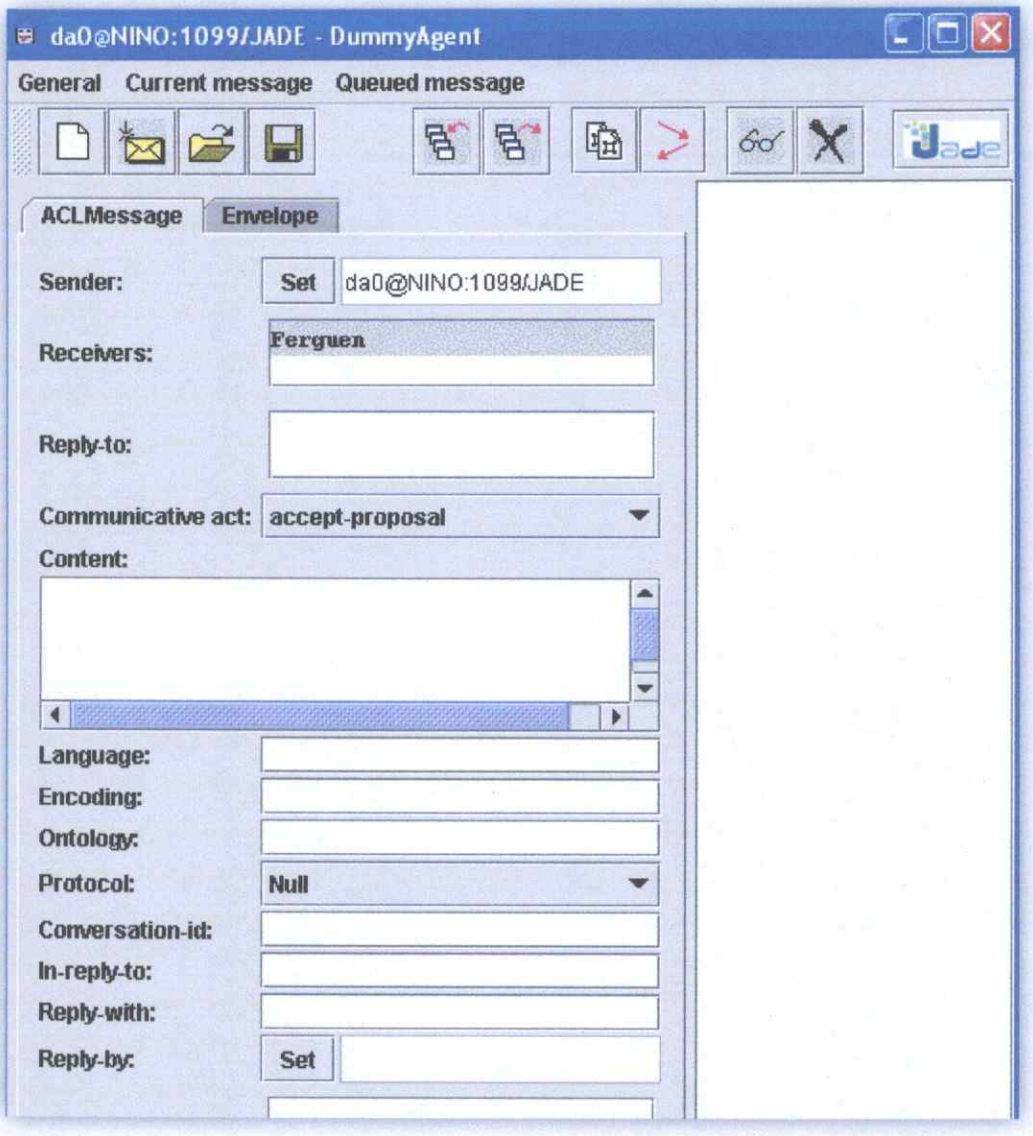


Figure 3 : L'interface de l'agent Dammy

5.3 Agent Direcory Facilitator

L'interface du DF peut être lancée à partir du menu du RMA .Cette action est en fait implantée par l'envoi d'un message ACL au DF lui demandant de charger son interface graphique. L'interface peut être juste vue sur l'hôte où la plate-forme est exécutée. En utilisant cette interface, l'utilisateur peut interagir avec le DF.

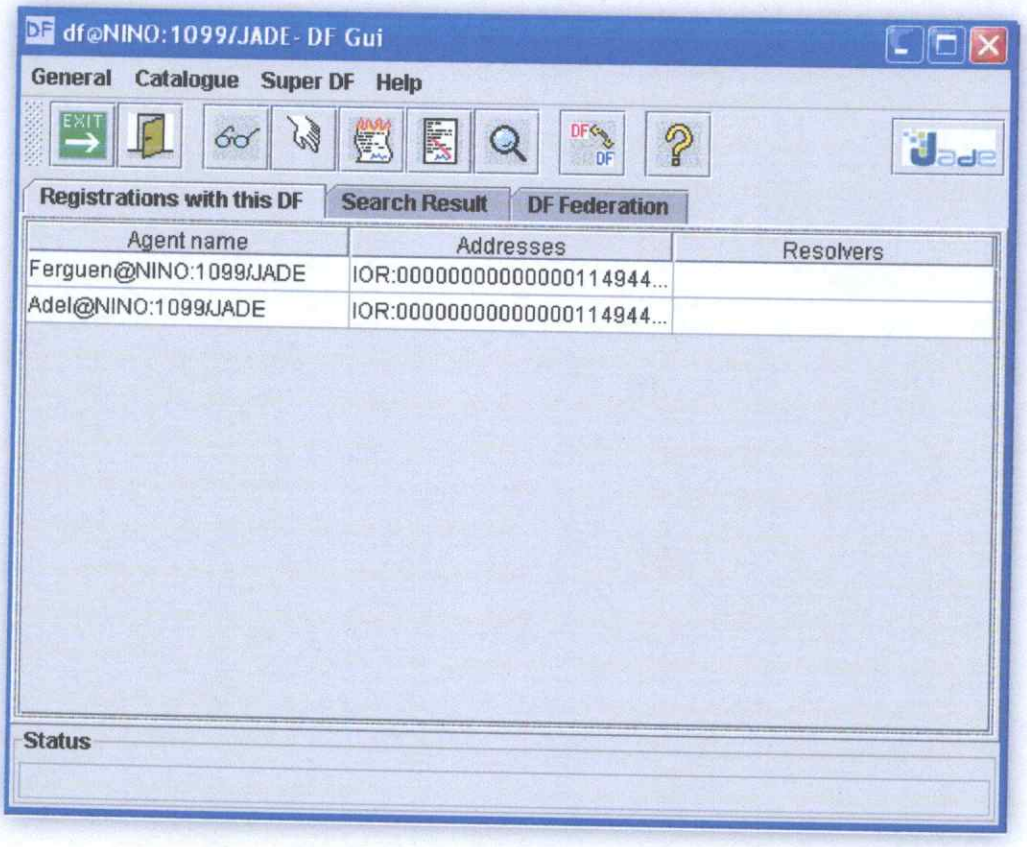


Figure 4 : L'interface de l'agent DF

5.4 Agent Sniffer

Quand un utilisateur décide d'épier un agent ou un groupe d'agents, il utilise un agent sniffer. Chaque message partant ou allant vers ce groupe est capté et affiché sur l'interface du sniffer. L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser plus tard. L'agent peut être lancé du menu du RMA ou de la ligne de commande suivante : Java jade.Boot sniffer:jade.tools.sniffer.sniffer

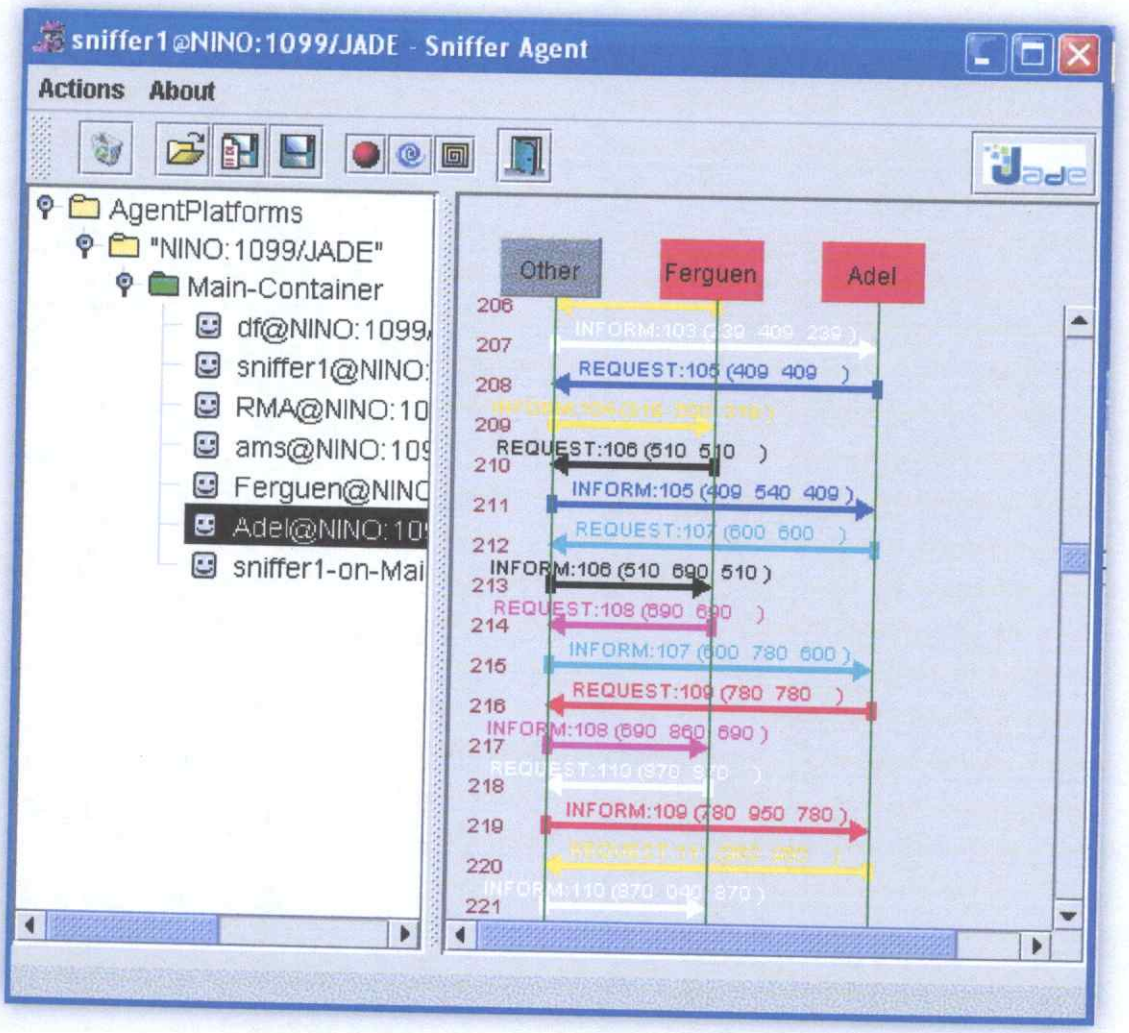


Figure 5 : L'interface de l'agent Sniffer

5.5 Agent Inspector

Cet agent permet de gérer et de contrôler le cycle de vie d'un agent s'exécutant et la file de ses messages envoyés et reçus.

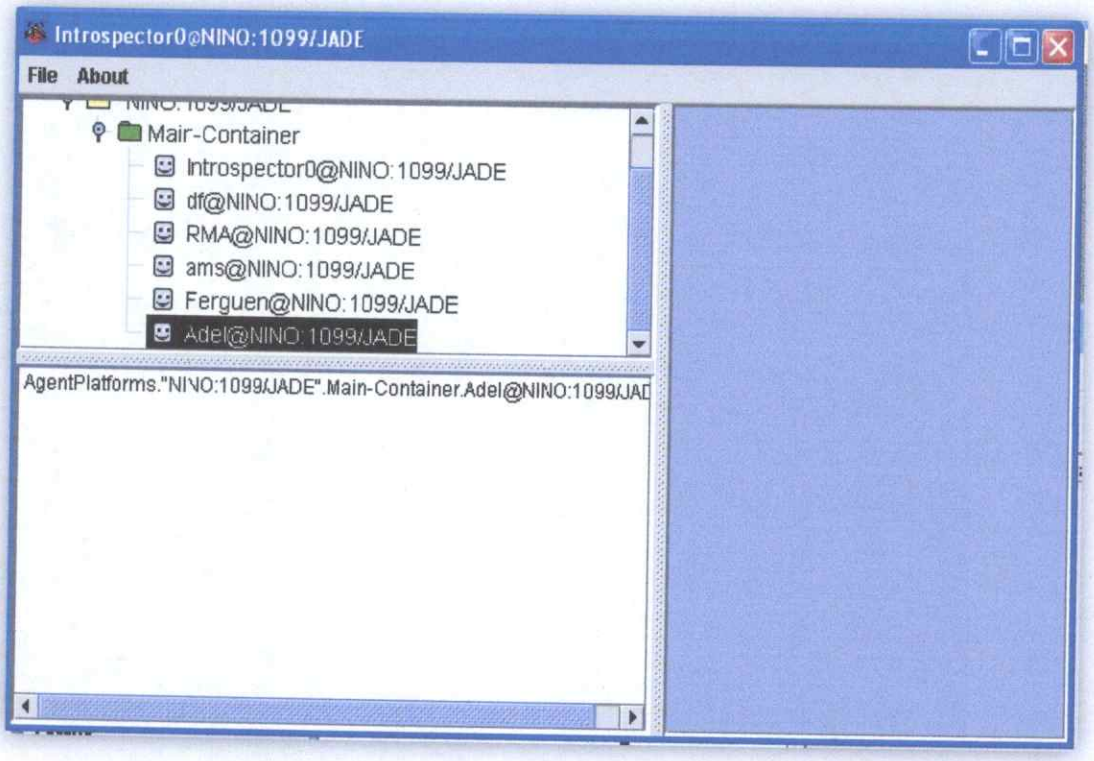


Figure 6 : L'interface de l'agent Inspector

ANNEXE B

Editeur PROTÉGÉ

Définition1 :

PROTÉGÉ est un environnement graphique de développement d'ontologies développé par le SMI de Université Stanford. Dans le modèle des connaissances de PROTEGE, les ontologies consistent en une hiérarchie de classes qui ont des attributs (slots), qui peuvent eux-mêmes avoir certaines propriétés (facets). L'édition des listes de ces trois types d'objets se fait par l'intermédiaire de l'interface graphique, sans avoir besoin d'exprimer ce que l'on a à spécifier dans un langage formel : il suffit juste de remplir les différents formulaires correspondant à ce que l'on veut spécifier.

Ce modèle autorise d'ailleurs une liberté de conception assez importante puisque le contenu des formulaires à remplir peut être modifié suivant les besoins via un système de métaclasses, qui constituent des sortes de « patrons » de connaissance. L'interface, très bien conçue, et l'architecture logicielle permettant l'insertion de plugins pouvant apporter de nouvelles fonctionnalités ont participé au succès de PROTÉGÉ.

Aujourd'hui, il regroupe une large communauté d'utilisateurs et bénéficie des toutes dernières avancées en matière de recherche ontologique : compatibilité OWL de référence, services inférentiels, gestion de bases de connaissances, visualisation d'ontologies, alignement et fusion, etc.

Définition2 :

PROTÉGÉ est un éditeur qui permet de construire une ontologie pour un domaine donné, de définir des formulaires d'entrée de données, et d'acquérir des données à l'aide de ces formulaires sous forme d'instances de cette ontologie. Protégé est également une librairie

Java qui peut être étendue pour créer de véritables applications à bases de connaissances en utilisant un moteur d'inférence pour raisonner et déduire de nouveaux faits par application de règles d'inférence aux instances de l'ontologie et à l'ontologie elle même (méta-raisonnement).

Les logiques de description permettent de définir les bases logiques des différents formalismes de représentation de la connaissance tant sur le plan de la représentation que sur le raisonnement. Dans les formalismes de représentation de la connaissance, il est souvent nécessaire de restreindre l'expressivité pour rendre certains types de raisonnement, tels que la classification automatique, faisable (« tractable »).

- *Comment utiliser le logiciel Protégé :*

- Lorsqu'on ouvre Protégé 3.2.1 l'écran suivant apparaît :

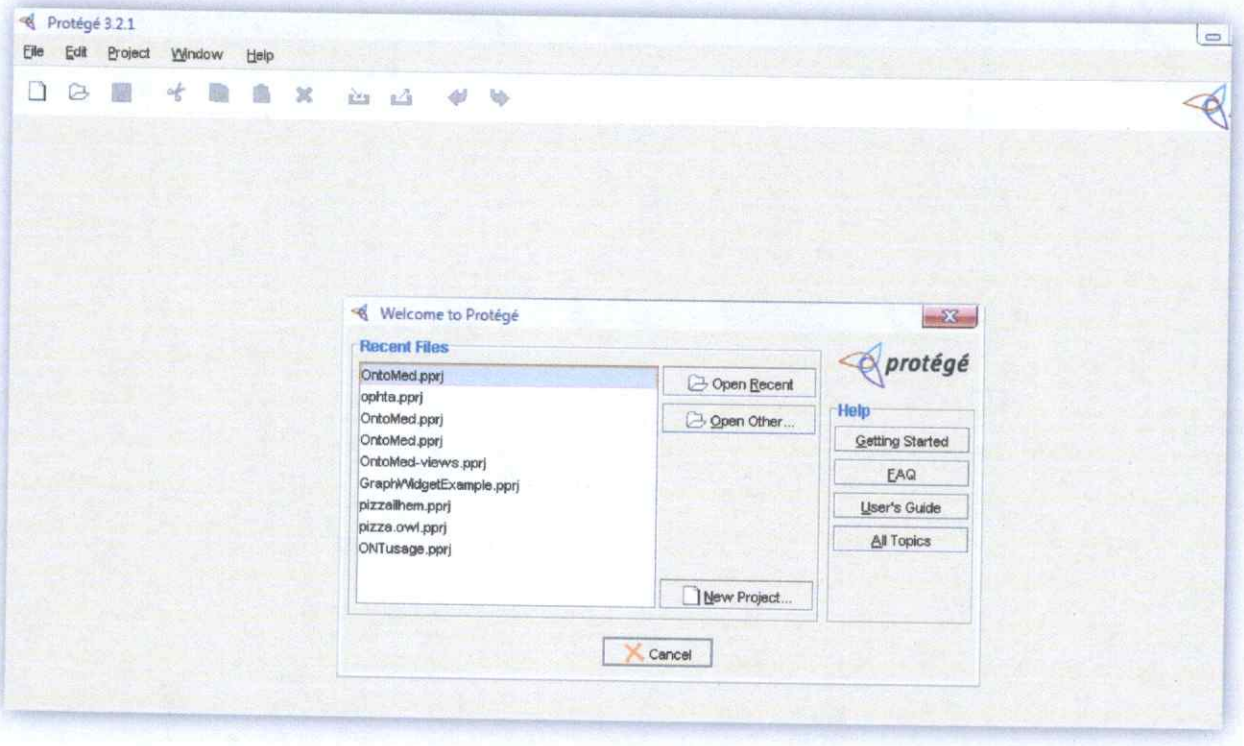


Fig1 : Welcome to protégé.

Vous avez trois choix :

- Ouvrir un projet existant en choisissant parmi « Recent Files ».
- Ouvrir un autre projet : il vous suffit de spécifier l'emplacement de l'ontologie puis de la sélectionner.

- Créer un nouveau projet : une fois que vous avez cliqué sur « New Project.. » la fenêtre suivante est affichée

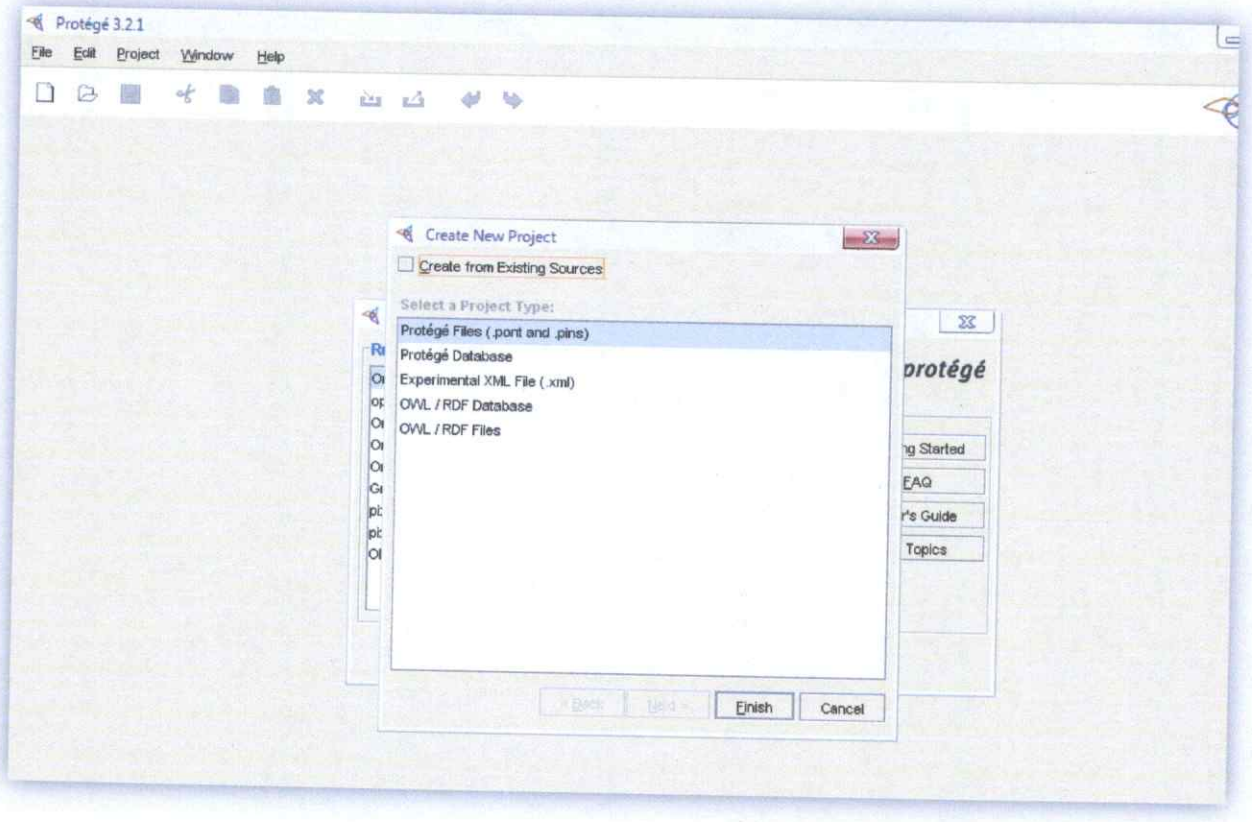


Fig2 : création d'un nouveau projet.

La création d'un nouveau projet ce fait par le choix du format (dans notre cas OWL/RDF Files), puis la validation par le bouton « Finish »

Spécifier le nom du projet en cliquant sur le bouton « Save Project » puis valider avec le bouton « OK »

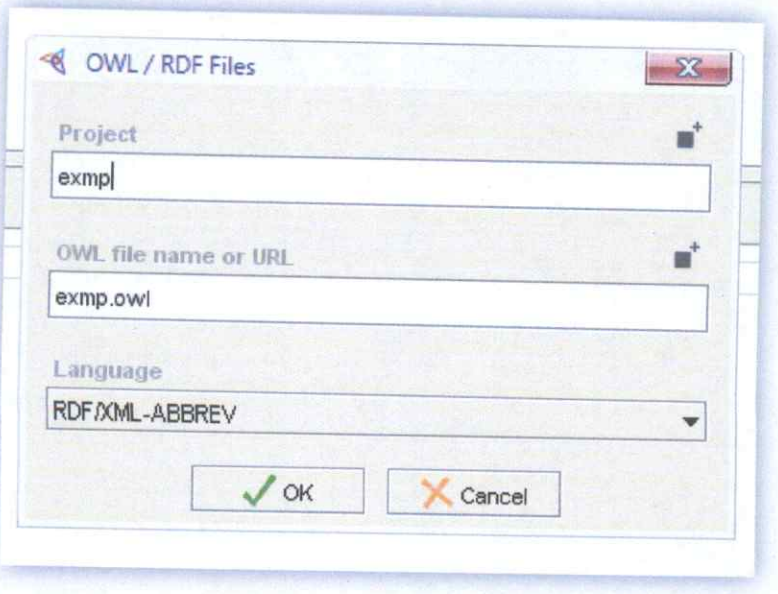


Fig3 : validation.

-*Modification de l'ontologie sous Protégé 3.2.1 :*

Editeur de class : cette onglette permet la création et la manipulation des classes OWL .il contient une partie « hiérarchie » qui permet la navigation dans l'ontologie et la modification de l'ordonnancement des concepts si nécessaire, et une autre partie « éditeur », qui permet la création et la modification des classes : nom classe, relation, restriction etc. :

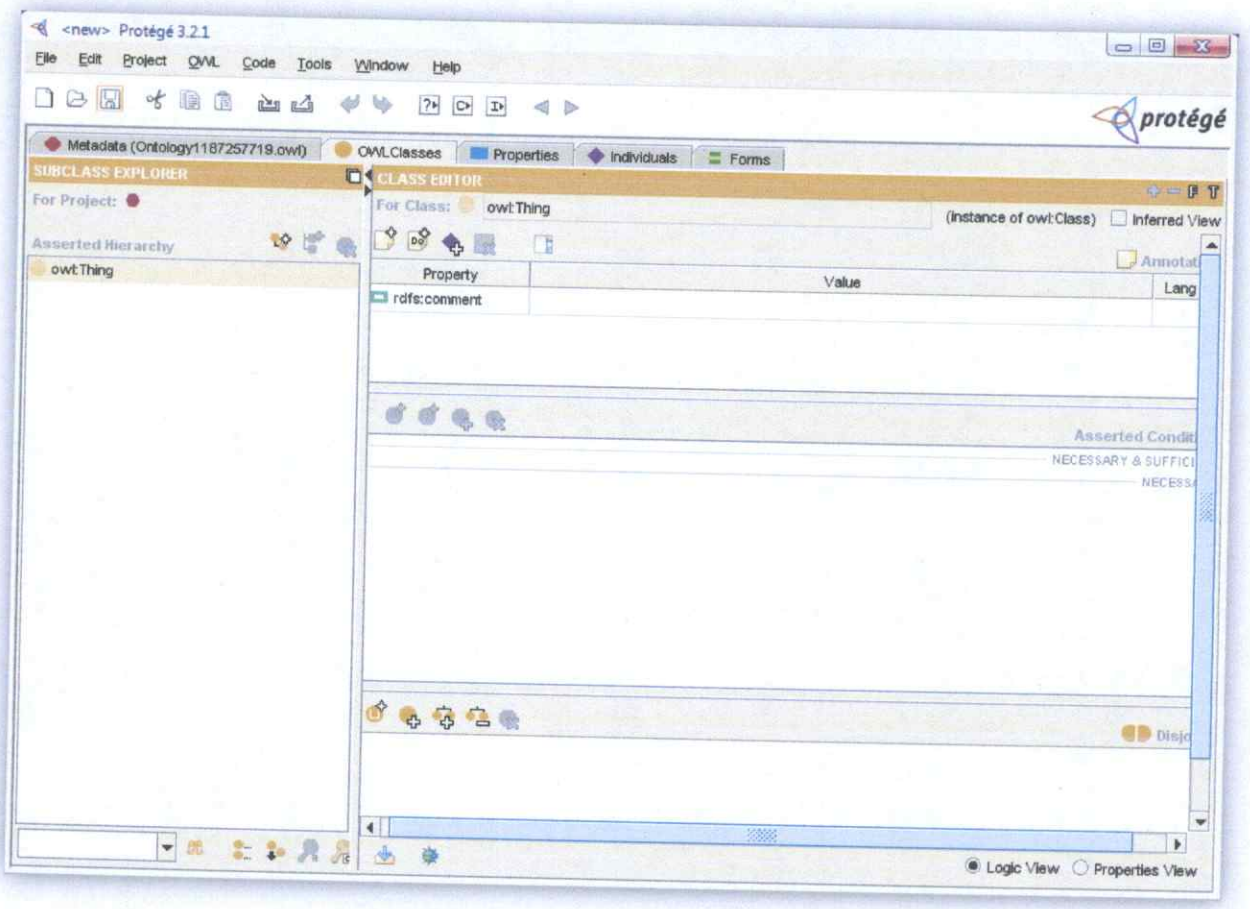


Fig4 : Editeur de classes.

L'ajout d'une classe :

Pour créer une classe, il suffit de se positionner dans la hiérarchie et de cliquer sur le bouton « créer une sous classe » ou sur « créer une classe adjacente » comme illustre la figure suivante :

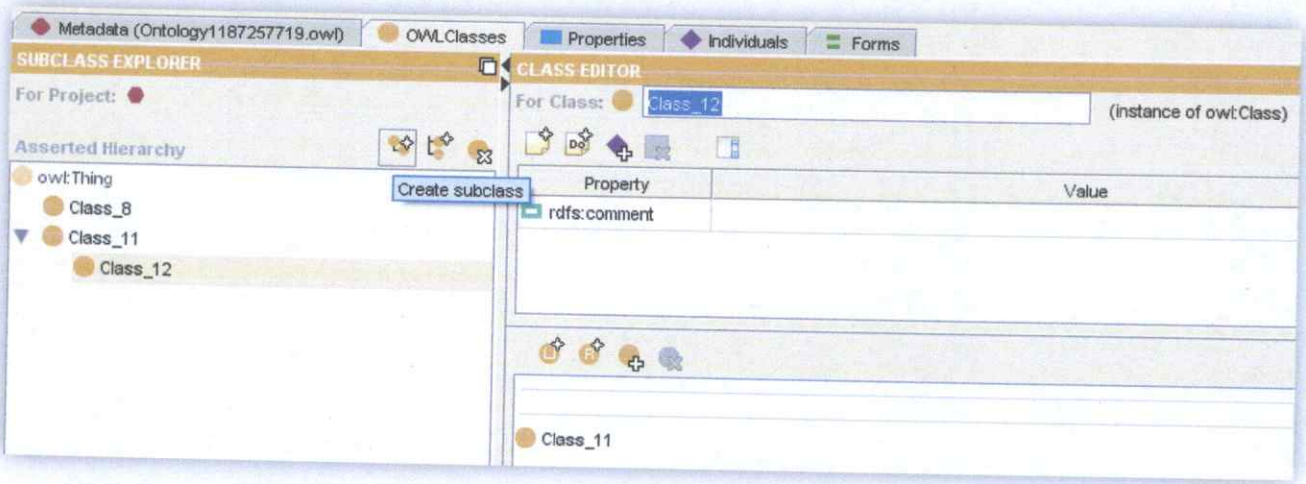


Fig5 : Création de classes.

Editeur de propriétés :

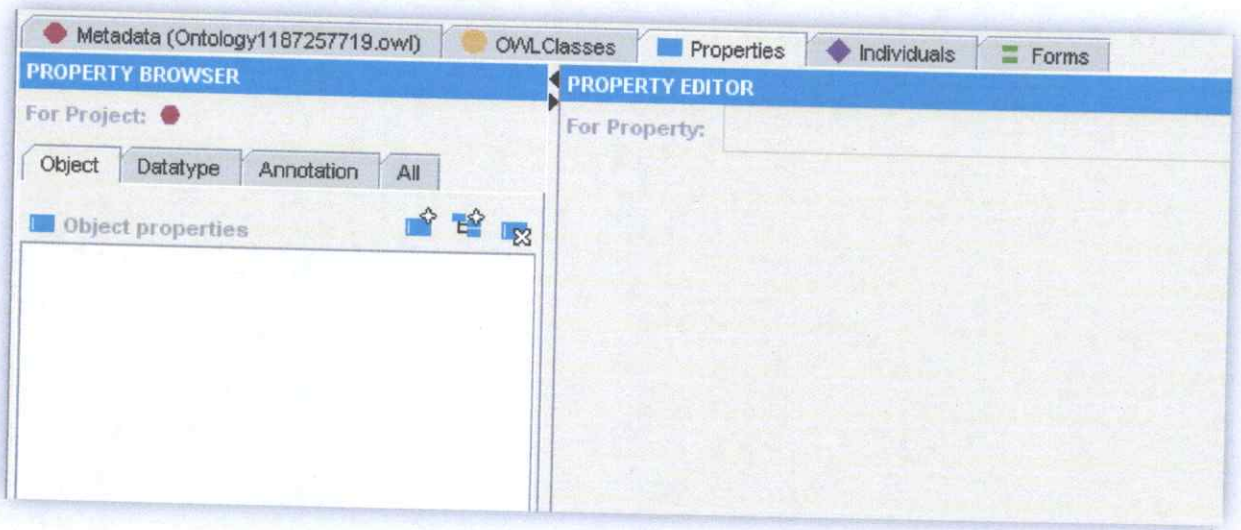


Fig6 : Editeur de propriétés.

Il existe deux types de propriétés dans OWL, ces types sont pris en charge par l'éditeur de propriétés de Protégé OWL :

DataType Property : ces relations peuvent être vues comme des attributs de classe ou d'instances. Une propriété de ce type lie une classe ou une instance a un type de donnée.

Object Property : ce type de propriété lie un domaine de classe à un autre, il peut être comparé aux relations entre individus dans MARISE.

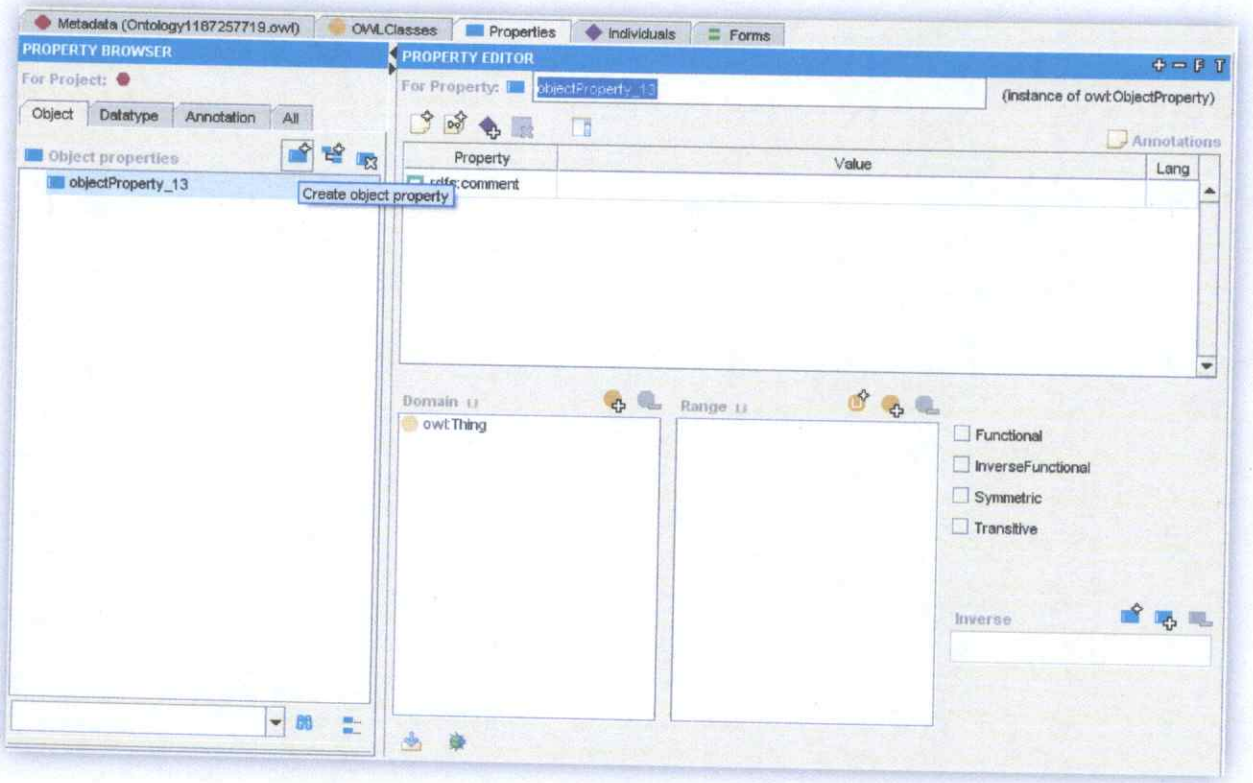


Fig7 : Création de propriétés.

Dans le cas de propriétés entre objets (classes), la création de la propriété se fait en donnant le nom, domaine, rang, et les attributs arithmétiques de la propriété (symétrie, transitivité, etc.).

Dans le cas d'attributs d'instances ou de classes, la définition de la propriété se fait par l'attribution d'un nom et le type de donnée associé, ce type de donnée comprend tous les formats pris en charge pas XML : date, string, integer, etc.

De même que pour les classes, il est possible de hiérarchiser les propriétés, a la seule condition qu'elles soient de même type.

Restriction sur les classes :

Une fois les classes et les propriétés définies, il est possible de créer des restrictions sur les classes en utilisant les propriétés.

Les restrictions servent à cerner la portée des classes de l'ontologie, elles sont exprimées dans Protégé par des expressions logiques dans l'éditeur de classes :

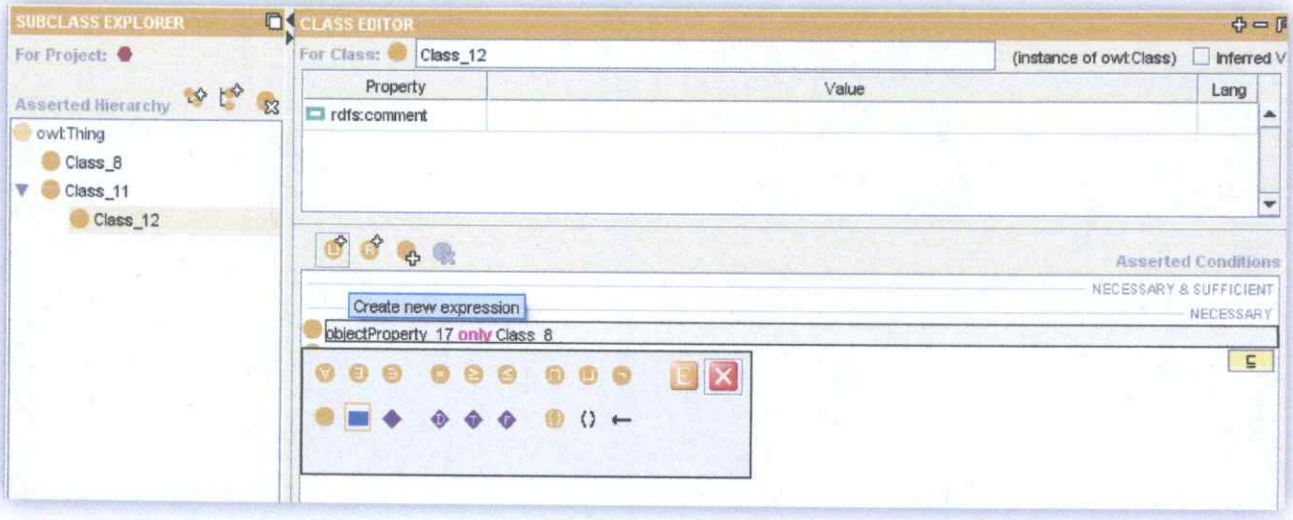


Fig8 : Création de restrictions.

Le tableau ci dessous représente les restrictions OWL, toutes ces restrictions portent sur des classes et des propriétés.

L'expression pour la classe Pathologie_cancerologie : ∃ Pathologie_avoir_traitement some Traitement_medical_cancérologie signifie que toutes pathologie cancérologique a au moins un traitement médical de cancérologie.

Pour la classe Symptôme cancérologie : l'expression ∀ symptôme_de_pathologie only Pathologie_cancérologie signifie que tout les symptômes cancérologie concernent seulement pathologie_cancérologie. Ce type de restriction est nommé *Closure axiome.*

OWL Element	Symbol	Key	Example	Meaning of example
allValuesFrom	∀	*	children ∀ Male	All children must be of type Male
someValuesFrom	∃	?	children ∃ Lawyer	At least one child must be of type Lawyer
hasValue	∋	$	rich ∋ true	The rich property must have the value true
cardinality	=	=	children = 3	There must be exactly 3 children
minCardinality	≥	>	children ≥ 3	There must be at least 3 children
maxCardinality	≤	<	children ≤ 3	There must be at most 3 children
complementOf	¬	!	¬ Parent	Anything that is not of type Parent
intersectionOf	⊓	&	Human ⊓ Male	All Humans that are Male
unionOf	⊔	|	Doctor ⊔ Lawyer	Anything that is either Doctor or Lawyer
enumeration	{...}	{ }	{male female}	The individuals male or female

Fig9 : Restrictions OWL dans protégé.

Les Requêtes :

Un des avantages de cet éditeur d'ontologie est la possibilité de faire des requêtes sur les données de l'ontologie. Pour cela, il faut aller sur l'onglet « Queries » :

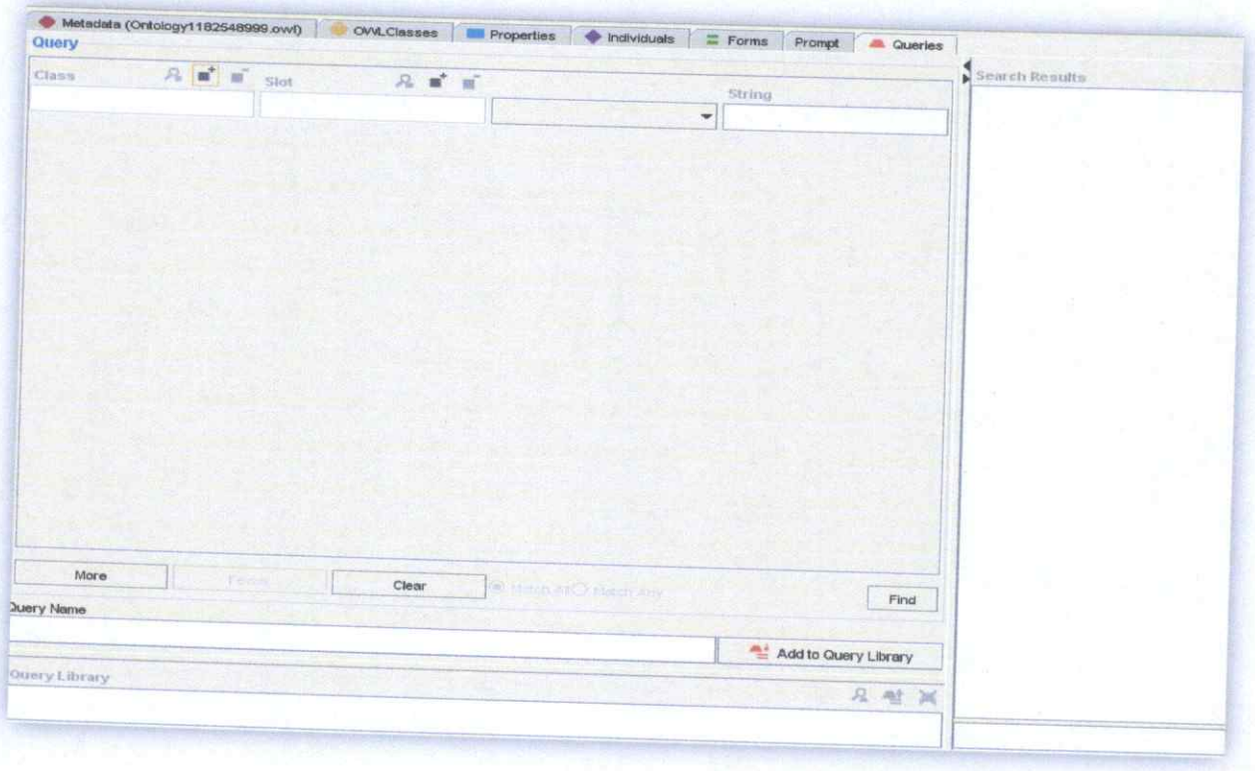


Fig10 : Onglet Queries dans protégé

Dans la zone de texte « Class », il faut indiquer la classe sur laquelle on veut effectuer la requête. Pour choisir la classe, il suffit de cliquer le « + » pour obtenir l'explorateur.
Une fois la classe choisie, il faut indiquer l'attribut sur lequel on effectue la requête. De la même manière, il suffit de cliquer sur le « + » pour obtenir la liste des attributs possibles concernant la classe sélectionnée précédemment.
Ensuite, nous avons une liste déroulante permettant de choisir le type de critère de la requête selon que l'attribut soit un numérique (« égal à », « plus petit que» ou « plus grand que ») ou du texte (« contenant », « ne contient pas », « égale à », « n'est pas égale à », « commençant par » et « fini par »).

Enfin la dernière zone de texte permet de saisir le texte ou le nombre qui correspond au critère de sélection de la requête.
Pour rajouter d'autres critères de sélection de la requête, il suffit de clique sur « More » en bas.
On a la possibilité d'enregistrer la requête. Il faut d'abord lui attribuer un nom puis de clique ensuite sur le bouton « Add to query library ». Cette option est intéressante pour les requêtes récursives, cela évite à l'utilisateur de ressaisir la même requête.

ANNEXE C

RI

Les principes d'interrogation [Web08]:

On décrit dans le tableau si dessous tout les operateurs utilisés dans la recherche par interrogation

Tableau I: Les opérations de recherche

Catégories d'opérateurs	Définition	Symboles	Types	Fonction	Exemples
Troncature (Wildcards, joker)	Substitution d'un symbole à des caractères	* + ?	Troncature à droite	Recherche sur tous les mots contenant la même racine ou le même préfixe	franco* : >> francophone, francophonie, francophobe
			Troncature à gauche	Recherche à partir d'un suffixe	*phobe : >> technophobe, agoraphobe, etc.
		? #	Masque ou troncature centrale	Remplace un ou plusieurs caractères dans un mot.	francopho?e > francophobe et francophone
Opérateurs booléens	Logique booléenne, permettant des relations entre les termes de la recherche	**ET AND** +	Opérateur d'intersection	Relie deux mots-clés se trouvant ensemble dans le même document	Pêche ET Bretagne : >> les documents traitant de la pêche en Bretagne
		OU OR	Opérateur d'union	Relie deux mots-clés dont l'un ou l'autre, ou les deux doivent se trouver dans le document	Pêche OU Bretagne : >> les documents traitants ou bien de pêche ou bien de Bretagne, ou des deux à la fois
		SAUF NOT	Opérateur d'exclusion	Relie deux mots-clés dont le premier doit être présent et le second absent	Pêche SAUF Bretagne : >> les

		-		dans un document	documents traitant de la pêche sauf en Bretagne
Opérateurs numériques	Permettent des recherches selon des critères quantitatifs	- égal : = - supérieur : >>br>- supérieur ou égal : >= - inférieur : < - inférieur ou égal : <= - intervalle entre deux dates : :		Recherche sur les dates, par exemple	□ = 1995 : en 1995 □ > 1995 : depuis 1995 □ >= 1995 : depuis ou en 1995 □ <= 1995 : avant ou en 1995 □ 1991:1995 : de 1991 à 1995
Opérateurs de proximité	Permettent des recherches sur le texte intégral selon la proximité des termes	**ADJ**	Opérateur d'adjacence	Recherche sur des termes adjacents, dans l'ordre donné	Fibre ADJ optique : >> texte contenant l'expression " fibre optique "
		nAV	Opérateur de distance	Recherche sur des termes séparés par une distance *n*	Ecole 1AV privée : >> école primaire privée, ou école technique privée
		NEAR **M (pour Mot) ou** **W (Word)**	Opérateur de présence	Recherche sur des termes présents dans le texte, quelle que soit leur distance	Fibre NEAR optique >> texte contenant les deux termes, mêmes séparés

Nous résumons les typologies de la recherche d'information dans le tableau ci-dessous :

Tableau I.1: typologie de la RI

Modes de recherche	*Principe, démarche intellectuelles*	*Type d'information concernée en priorité*	*Exemples d'outils*
Recherche par navigation arborescente	Figure de l'Arbre Démarche systématique, du général au particulier Recherche par menus successifs	Information structurée, organisée en plan de classement Information secondaire	Tables des matières des livres Classifications documentaires (CDU, Dewey) Annuaires thématiques du web (Yahoo, Nomade…) Page d'accueil d'un site web
Recherche par navigation hypertextuelle	Figure du Réseau Démarche associative, d'une notion à l'autre. Navigation dans un réseau de nœuds et de liens	Information non structurée Information primaire, brute	Renvois dans une encyclopédie Hypertextes sur CD-ROM Sites web
Recherche par requête sur les métadonnées du document	Principe de l'Index Démarche d'indexation de l'information Recherche par champs, logique booléenne	Information structurée en champs. Information secondaire	Index des livres Banques de données Catalogues de bibliothèques
Recherche par requête sur le texte intégral	Texte Démarche d'analyse linguistique Recherche contextuelle sur le contenu	Information non structurée Information brute, primaire	Outils de TALN (*Traitement Automatique du Langage Naturel*) Moteurs de recherche Outils linguistiques

Exemple d'annuaire, de moteur de recherche et de méta moteurs:

Exemple Annuaire :

Annuaires francophones :

Guide de Voilà : Voila Le Guide du web http://guide.voila.fr/
Lycos France: Lycos Annuaire de sites http://www.recherche.lycos.fr/annuaire/

Annuaires mondiaux

ANNEXE D

Algorithme de désambigüisation

1- Algorithme de désambigüisation de KHAN :

L'idée principale de cet algorithme est que l'union sémantique des mots clés d'une requête forme un « contexte » (khan00).

L'exemple utilisé par khan pour illustrer la notion de contexte est que les mots clés « base, batte et gant » peuvent avoir différents sens individuellement, mais leurs union sémantique converge vers un seul sens qui est la référence au baseball (Khan00)

Algorithme :
Le pseudo code de cet algorithme est comme suit :

- C1,C2, …..,Cn sont les concepts sélectionnés pour la requête ;
- Calcul du PScore de chaque concept :
  - PScore(C_i) = CScore i + $\sum \frac{Cscore\ k}{SD(Ci,Ck)}$ (i<k<n).
- Ranger les concepts et leurs scores dans un ordre descendant;
- Pour chaque mot clé associé à plusieurs concepts ambigus Ci,….,Cl ou les scores sont décroissant : Pi,…..Pl ;
- {
- Conserver Ci et éliminer les autres concepts ;
- }
- Conserver les concepts spécifiques et éliminer les concepts généraux

Description détaillée :

Pour désambiguïser via une ontologie, un certain nombre de définitions (formules) sont utilisées, comme le score d'un concept ou la distance sémantique entre concepts. Ceci a pour but de sélectionner les concepts à retenir lors de la désambiguïsation du sens des mots dans le texte.

Dans l'ontologie utilisée par Khan (sport), un concept est composé d'une liste complémentaire de synonymes : $Ci = (l_1, l2, ..., li, Ln)$. Les mots-clés dans le texte annoté sont appariés (groupés) avec chaque mot-clé d'un élément l_j du concept. Le calcul du score pour un élément l_j (*Score_element*) du concept C_i, est basé sur le nombre de mots-clés reconnus:

$$\text{Score\_element}_{i\,j=} \frac{nbre\ mots\ clé\ lj\ apparié\ (uni)}{nbre\ de\ mots\ clé\ de\ lj}$$

Le plus grand de ces scores est retenu comme score pour tout le concept:

$Score\_concept_i = \max Score\_element_{ij,}$ avec $1 \leq j \leq n$

Pour sélectionner la région de l'ontologie à laquelle le concept sera affecté, Khan calcule aussi un score pour les différentes régions possibles. Le score d'une région *R* est alors égal à la somme des *Score_concept* de tous les concepts sélectionnés. Notons que dans le cas où plusieurs concepts correspondent à un mot-clé (ambiguïté), la moyenne de leur *Score_concept* est retenue.

Puis, Khan définit la Distance Sémantique entre deux concepts C_i et C_j $SD(C_i, C_j)$ comme le plus court chemin entre les concepts C_i et C_j dans l'ontologie. Cette distance est égale à 1 si les concepts C_i et C_j sont directement reliés (comme c'est le cas dans "NBA" et "Los Angeles Lakers" dans l'ontologie représentée dans la Figure 1, et infinie si les concepts sont au même niveau de la hiérarchie et qu'il n'existe pas de chemin entre eux.

En outre lorsque deux concepts sont corrélés, leurs scores appelés *Propagated-score*, sont inversement proportionnels à leur position (distance) dans l'ontologie. Car plus la distance sémantique est grande, plus la corrélation entre les concepts diminue et vice versa.

Si le concept C_i est corrélé avec un ensemble de concepts (C_j, C_{j+1},..., Cn), la propagation du score de C_i est :

$$\text{PScore}(C_i) = \text{CScore i} + \sum \frac{Cscore\ k}{SD(Ci,Ck)} \quad (i<k<n).$$

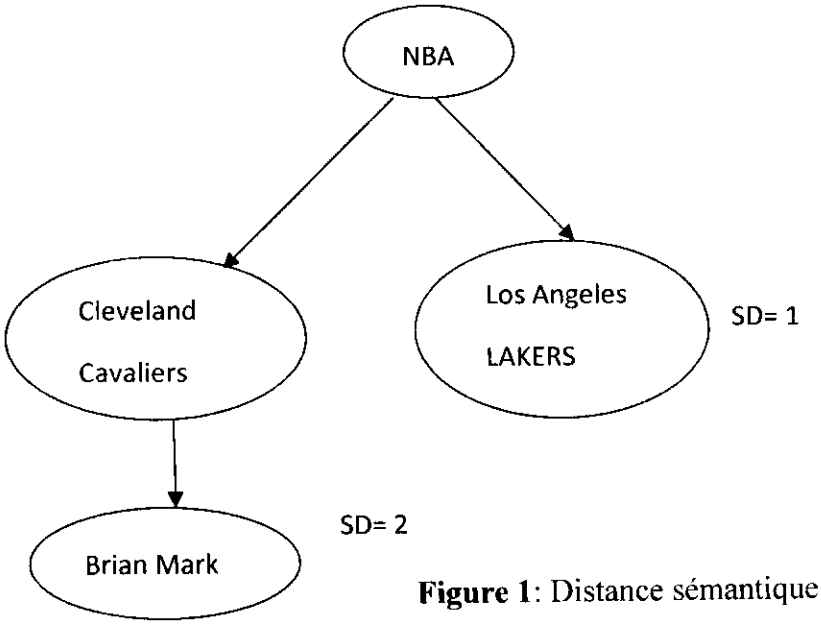


Figure 1: Distance sémantique

ANNEXE E

LA LIBRAIRIE JENA

I. Présentation de JENA

Jena est un outil de développement d'applications destinées au Web sémantique, c'est un projet open-source initié par les laboratoires Hewlett Packard (HP) en 2000. Jena peut être téléchargé sur le site www.SourceForge.com.
Jena est multi-plateforme car il a été entièrement développé avec Java. Cet outil, conforme aux spécifications RDF, fournit un environnement de développement pour les ontologies RDF, RDFS et OWL, ainsi qu'un moteur d'inférences pour le raisonnement sur les ontologies.
De plus, Jena inclut :

- Une API pour RDF ;
- Une API pour OWL ;
- Un modèle de stockage d'ontologies ;
- RDQL : langage permettant la formulation de requêtes sur les ontologies RDF et OWL.

II. L'API OWL de JENA 2

Ce n'est qu'à partir de cette version de Jena qu'a été incluse cette API, les versions précédentes furent créées avant l'apparition et la standardisation de ce langage.
Cette API permet de manipuler des contenus OWL par l'intermédiaire d'applications JAVA.
Le principe de fonctionnement de cette API est illustré dans la figure ci-dessous :
Ce schéma illustre la procédure de création de modèles d'ontologies dans Jena.

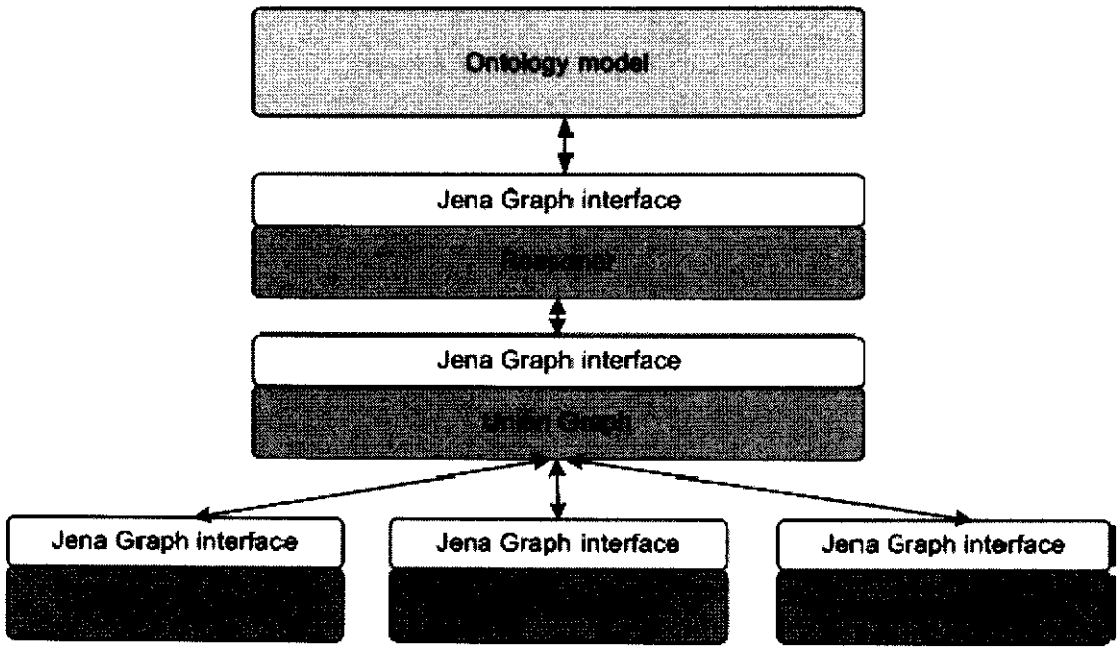


Figure 20 Structure interne d'une ontologie, incluant les imports [HP03]

II.1. Création d'un modèle d'ontologie

Un modèle d'ontologie est instancié à partir de la classe « OntModel », et est créé en utilisant la classe « ModelFactory » par la commande : **[HP03]**

```
OntModel m = ModelFactory.createOntologyModel ();
```

Cette commande permet de créer un modèle d'ontologie standard (ontologie OWL Full, raisonneur par défaut, et stockage normal).
Pour créer un modèle d'ontologie OWL, par exemple il suffit d'écrire : **[HP03]**

```
OntModel m = ModelFactory.createOntologyModel (ProfileRegistry.OWL_MEM);
```

Au-delà des spécificités du langage, Jena propose des raisonneurs afin d'approcher l'ontologie de la manière adéquate. Ces raisonneurs contiennent des règles et des inférences qui permettent aux programmeurs de manipuler les ontologies selon leurs besoins. Ces propriétés sont stockées dans la classe OntModelSpec.

OntModelSpec.OWL_MEM_RDFS_INF par exemple, associe à l'ontologie OWL un raisonneur basé uniquement sur RDFS.

II.2. Ajout d'une ontologie à un modèle

Le code ci-après permet d'inclure une ontologie (locale ou distante) à un modèle : **[HP03]**

```
OntModel m = ModelFactory.createOntologyModel ();
OntDocumentManager dm = m.getDocumentManager();
dm.addAltEntry ("http://www.xfront.com/owl/ontologies/camera/",
"file:" + JENA + "doc/user-manual/ontology/data/camera.owl");
m.read ( "http://www.xfront.com/owl/ontologies/camera/" );
```

La dernière ligne de code lit l'ontologie à partir de la source (fichier locale ou URL), cette source est contenue dans la classe DocumentManager.

II.3. Manipulation du contenu de l'ontologie

Les méthodes suivantes appartiennent à la classe OntClass et OntModel qui représente les classes de l'ontologie.

Méthode Description

listSubClasses (boolean) Renvoie dans un objet de type Iterator des objets OntClass contenant les sous-classes d'une classe. Le paramètre sert à restreindre la liste aux sous-classes directes.

listDeclaredProperties () Renvoie dans un objet Iterator des objets Property contenant les propriétés d'une classe.

isRestriction () Permet de savoir si la classe est une restriction.

listSuperClasses (boolean) Renvoi dans un objet Iterator des objets OntClass contenant les super-classes de la classe. Le paramètre sert à restreindre la liste aux super-classes directes.

Les restrictions peuvent être manipulées grâce aux méthodes de l'interface Restriction. Les principales méthodes sont :

Méthode Description

asHasValueRestriction () Permet de considérer une classe comme un restriction de type *HasValue*.

getOnProperty () Retourne un objet Property contenant la propriété concernée par la restriction.

asCardinalityRestriction () Permet de considérer une classe comme un restriction de type *Cardinality*.

ANNEXE F

METHONTOLOGY

4.1. Introduction :

METHONTOLOGY a été développée par le groupe d'ingénierie ontologique de l'UPM (Université Polytechnique de Madrid). Elle est basée sur la norme 1074-1995, recommandée par l'IEEE dans le développement des produits logiciels, et sur d'autres méthodologies de l'ingénierie des connaissances [Gomez Pérez et al 04].

METHONTOLOGY permet la construction d'ontologies au niveau des connaissances ; dans la méthodologie proposée par Uschold et King, par exemple, l'ontologie est codifiée directement dans un langage formel au niveau de l'implémentation (dernière étape du processus de développement), contrairement à METHONTOLOGY, où l'ontologie est exprimée dans les premières étapes du développement (dans l'étape de la conceptualisation).

FIPA *(Foundation for Intelligent Physical Agents]* recommande l'utilisation de METHONTOLOGY pour la construction d'ontologies.

4.2. Composants d'une ontologie selon METHONTOLOGY:

Comme toute méthodologie, METHONTOLGY possède une terminologie spécifique pour désigner les différents composants d'une ontologie :

Concept : entité physique ou morale possédant un sens déterminé.

Relation : association entre les concepts du domaine. Une relation qui lie deux concepts est appelée relation binaire. Toute relation binaire peut avoir une relation inverse qui lie les deux concepts dans la direction opposée.

Instance : élément ou individu d'un concept.

Constante : valeur numérique qui reste inchangée durant un certain temps.

Attribut : propriété d'un concept ou d'une instance. Deux type d'attributs sont distingués :

Attributs d'instance et attributs de classe. Les attributs d'instance prennent leurs valeurs aux nivaux de chaque instance, ils sont définis pour un concept et sont hérités par

ses sous-concepts et ses instances. Les attributs de classe prennent leurs valeurs aux nivaux du concept, là où ils sont définis, et ils ne sont pas hérités par les sous-concepts et les instances.

Axiome formel : expression logique toujours évaluée à vrai.

Règle : utilisée pour inférer des connaissances dans l'ontologie.

4.3.1. **Spécification :**

La spécification est l'activité qui précède le processus du développement, proprement dit, de l'ontologie. Il s'agit de constituer un document qui contient toutes les informations sur les raisons de construction de l'ontologie, ses utilisations prévues et ses utilisateurs potentiels.

Demain: Chemical
Date: May 15, 1996
Developed by Asunciôn Gômez-Pérez and Mariano Fernândez Lôpez
Purpose: Ontology about chemical substances to be used when information about chemical éléments is required in teaching, manufacturing, analysis, and so on.
Level of formality: Semiformal.
Scope: List of 103 éléments: lithium, sodium, chlorine, mercury, ...
List of concepts: élément, halogen. noble gas, semimetal, métal, third-transition
métal, ...
Information about at least thé following properties: atomic number, atomic weight, electronegativity, melting point, ...
Sources of knowledge:
(a) Three interviews with thé expert.
(b) The following books:
[Handbook, 84-85] Handbook of Chemistry and Physics, 65th éd., CRC Press Inc., 1984-1985.

Fig.4.1 : Document de la spécification de l'ontologie *CHEMICALS.*

4.3.2. Acquisition des connaissances :

Les activités de support sont des activités indépendantes du processus de développement de l'ontologie. Elles peuvent être effectuées parallèlement à n'importe quelle activité du processus.
L'acquisition des connaissances est considérée comme l'activité la plus importante des activités de support. Dans leur expérience de construction de l'ontologie *"CHEMICALS"* (ontologie des éléments chimiques), Gomez Pérez et ses collègues ont adopté une approche méthodique pour acquérir les connaissances nécessaires. D'abord, en cherchant des connaissances générales sur le domaine, une vue d'ensemble, grâce à des interviews non structurées avec les experts du domaine. Ensuite, en étudiant la documentation disponible sur le domaine (écris, pages Web, tables, graphes, images...). Enfin, lorsqu'un certain niveau de connaissances est atteint, les particularités du domaine sont visées en se basant sur des interviews structurées avec les experts.
D'autres activités de support sont disponibles, à savoir: l'intégration, l'évaluation, la documentation et la gestion des configurations. Chacune d'elles peut être appliquée indépendamment du processus de développement

4.3.3. Conceptualisation :

La conceptualisation est l'organisation et la structuration des connaissances collectées lors de l'activité d'acquisition des connaissances dans le but de créer une première représentation du domaine.

Une fois le modèle conceptuel construit, la méthodologie propose de le transformer en un modèle formel qui sera implémenté, par la suite, dans un langage de représentation des connaissances. Pendant ce processus, le modèle conceptuel passe graduellement du niveau des connaissances au niveau de fimplémentation, et son degré de formalité augmente progressivement pour devenir finalement un modèle computable.

Dans ce sens, Gomez Ferez et ses collègues ont adapté le modèle de Blum [Blum 96], du génie logiciel à l'ingénierie ontologique.

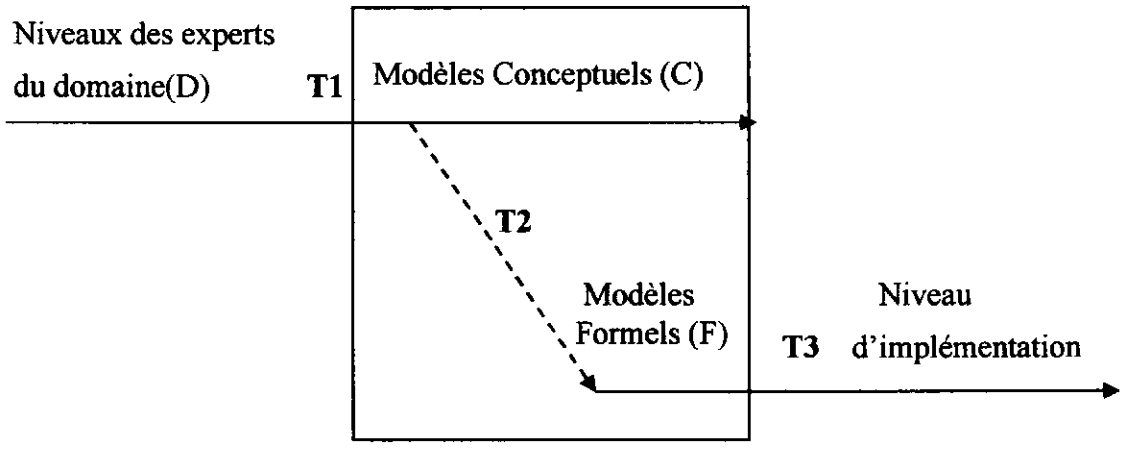


Figure 4.2 : Modèle de Blum dans le développement d'ontologies [Blum 96]

Contrairement à Tl et T 3, T2 est représentée par une ligne discontinue. Cela signifie, selon Gomez Ferez et ses collègues, que quelques connaissances du domaine peuvent disparaître lors de la transition du modèle conceptuel au modèle formel, si les composants utilisés pour construire le modèle conceptuel sont plus expressifs que ceux utilisés pour construire le modèle formel.

Pour cette raison, METHONTOLOGY propose d'utiliser des représentations intermédiaires basées sur des notations graphiques et tabulaires, aussi claires pour le développeur de l'ontologie que pour l'expert du domaine, afin de convertir une perception informelle du domaine en une spécification semi-informelle [Gomez Pérez 04]. Les représentations intermédiaires font correspondre la perception du domaine avec les formalismes et les langages d'implémentation de l'ontologie. Ainsi, les connaissances seront préservées lors du passage du modèle conceptuel au modèle formel

Après l'activité d'acquisition des connaissances, le développeur de l'ontologie dispose

d'une masse importante de connaissances non structurées qui doivent être organisées. Pour cela, MONTOLOGY propose une série de taches :

Tâche 1 : construire un glossaire de termes

La première tâche consiste à construire un glossaire qui contient tous les termes du domaine et leurs descriptions en langage naturel. Il est présenté sous forme d'un tableau qui contient entre autre : le nom du terme, ses synonymes, ses acronymes, sa description en langage naturel et son type (concept, relation, constante...).

Tâche 2 : construire une taxonomie de concepts

Il s'agit de hiérarchiser, sous forme d'une taxonomie, les concepts recensés dans le glossaire. Pour se faire, METHONTOLOGY propose l'utilisation des quatre relations taxonomiques définies dans *"thé Frame Ontology"* et *"thé OKBC Ontology" : SubClass-Of, Disjoint-Décomposition, Exhaustive-Decomposition, Partition.*

Un concept Cl est une sous classe *(SubClass-Of)* d'un concept C2, si et seulement si, toutes les instances de Cl sont aussi des instances de C2. Par exemple, le concept "être humain" est une sous classe du concept "être vivant".

Une décomposition disjointe *(Disjoint-Décomposition)* d'un concept C est un ensemble de sous classes de C, qui n'ont pas d'instances en commun et ne couvrent pas C. Par exemple, les concepts "Ministère" et "Entreprise" sont des sous classes disjointes du concept "Organisation", car il n'existe pas une organisation qui peut être un ministère et une entreprise en même temps, en plus, des organisations autres que ministère ou entreprise peuvent exister.

Une décomposition exhaustive *(Exaustive-Decomposition)* d'un concept C est un ensemble de sous classes de C, qui couvrent C et qui peuvent avoir des instances en commun. Par exemple, les concepts "Entreprise privée" et "Entreprise publique" forment une décomposition exhaustive du concept "Entreprise", car toute entreprise est forcément une sous classe d'au moins un des deux concepts.

Une partition d'un concept C est un ensemble de sous classes de C, qui n'ont pas d'instances en commun et qui couvrent C. Par exemple, les concepts "Mineur" et "Majeur" forment une partition du concept "Personne", car toute personne est, ou bien, mineure, ou bien, majeure.

Une fois construite, la (ou les) taxonomie doit être évaluée pour détecter d'éventuelles erreurs.

Tâche 3 : construire un diagramme de relation ad hoc binaires

La troisième tâche consiste à expliciter les relations existantes entre les différents concepts de la même (ou des différentes) taxonomie. En effet, le résultat de la tâche précédente peut se présenter sous forme d'une seule taxonomie, ou de plusieurs qui devraient être

reliées par au moins une relation.

A ce niveau, le développeur de l'ontologie doit s'assurer que les relations ne contiennent pas d'erreurs. Il doit vérifier que les ensembles de départ et d'arrivée des relations délimitent exactement et précisément les classes concernées par ces relations.

Tâche 4 : construire un dictionnaire de concepts

Un dictionnaire de concepts est constitué par tous les concepts du domaine, les relations les instances et les attributs de classes et d'instances. Les relations spécifiées pour chaque concept sont celles dont le domaine de départ est le concept en question.

Tâche 5 : décrire les relations ad hoc binaires en détail

Pour décrire en détail les relations figurant dans le dictionnaire de concepts, le développeur doit spécifier pour chacune : son nom, ses concepts source et cible, sa cardinalité, sa relation inverse et ses propriétés mathématiques.

Tâche 6 : décrire les attributs d'instance en détail

Les attributs d'instances sont décris en utilisant un tableau qui contient : le nom de l'attribut, le nom du concept où l'attribut est défini, le type de sa valeur, ses valeurs par défaut (si elles existent), la plage des valeurs, son unité de mesure, la formule ou la règle qui permet d'inférer les valeurs de cet attribut et les différents attributs et constantes utilisés dans les formules.

Tâche 7 : décrire les attributs de classe en détail

Les attributs de classe sont décrits de la même façon que précédemment.

Tâche 8 : décrire les constantes en détail

Les constantes sont décrites par : leurs noms, le type de leurs valeurs, leurs valeurs, leurs unités de mesure et les attributs qui peuvent être inférés en utilisant ces constantes.

Tâche 9 : décrire les axiomes formels

METHONTOLOGY propose de décrire les axiomes utilisés par l'ontologie en spécifiant : leurs noms, leurs descriptions en langage naturel, leurs expressions logiques (logique du premier ordre), les concepts, les attributs et les relations auxquels ils réfèrent et les variables utilisées.

Tâche 10 : décrire les règles

Les règles sont décrites exactement de la même façon que les relations.

Tâche 11 : décrire les instances

Cette tâche est facultative. Elle permet d'associer à chaque instance le nom de sa super classe et les noms et valeurs de ses attributs (s'ils existent).

4.3.4. Formalisation et implémentation :

Dans METHONTOLOGY, la formalisation du modèle conceptuel n'est pas obligatoire ; puisqu'il existe actuellement des outils qui sont capables de traduire le modèle conceptuel dans n'importe quel langage ou formalisme de représentation de connaissances.

ANNEXE G

Des ontologies pour la médecine

Plusieurs projets ont pour but de concevoir une ontologie pour la représentation des concepts médicaux. Nous discutons brièvement les deux principaux : GALEN et SNOMED RT.
Nous mentionnons aussi pour mémoire un projet apparenté, l'UMLS, et une contribution française, le projet MENELAS

Le projet GALEN : est la première initiative d'envergure à avoir eu pour objectif la construction d'une ontologie pour la médecine. Il s'agit d'une série de deux projets européens, GALEN (1992-1994) et GALEN-IN-USE (1996-1998).
La représentation employée est GRAIL, une variété de logique de description. Les concepts primitifs de l'ontologie de GALEN forment un arbre à quelques exceptions près. Chaque concept est accompagné d'une déclaration des relations qui doivent ou peuvent le lier à d'autres concepts. GALEN a pour but de faciliter la description d'informations cliniques, le codage et le transcodage dans des classifications diverses.
Le premier projet a mené à une hiérarchie contenant de l'ordre de 4000 concepts. La version actuelle en contient bien davantage. D'après la documentation du projet, elle couvre la moitié de ce que ses auteurs estiment qu'elle doit contenir, mais avec déjà une profondeur et une complexité plus grandes que dans les terminologies existantes ; et la couverture dans certains domaines comme les actes chirurgicaux est virtuellement complète. Un formalisme intermédiaire de dissections facilite l'entrée de concepts complexes, et un système de génération automatique d'expressions en langue naturelle permet de relire des concepts sous forme d'expressions en français, anglais ou allemand.

Le projet SNOMED RT est un remaniement de la nomenclature SNOMED visant à épauler ses termes par des descriptions dans un langage de représentation des connaissances (K-Rep, de la famille des logiques de description). Démarré vers 1996, il est mené aux États-Unis sous forme d'une collaboration entre le College of American Pathologists (soutien institutionnel traditionnel de la nomenclature SNOMED), la société Kaiser Permanente (Health Management Organization, système intégré privé d'assurance maladie et de réseau de soins) et la Mayo Clinic (grand réseau de soins).
L'idée est de distinguer une << terminologie de référence >> (RT) des terminologies qui peuvent être utiles pour des interfaces de saisie de données ou pour des systèmes de traitement automatique des langues.

En cela, ce projet est très proche des principes sous-tendant GALEN. La principale différence d'approche est que SNOMED RT part d'une terminologie existante, la nomenclature SNOMED III. La première version de cette << terminologie de référence >> est attendue pour 1999.

Le projet UMLS (Unified Medical Language System) :

N'est pas à proprement parler un projet de constitution d'ontologie. Il fait plutôt partie de la famille des terminologies, mais y tient une place particulière. Il s'agit de l'union raisonnée de plus de quarante terminologies biomédicales dont le thésaurus MeSH (y compris sa traduction en français et dans plusieurs autres langues), la Classification internationale des maladies et la nomenclature SNOMED III. Cette union est appelée Métathésaurus, et contenait, en 1998, 476 313 concepts et 1 051 901 termes, synonymes et autres variantes lexicales. La version 1999 contiendra plus de 625 000 concepts, et outre celle du MeSH, la traduction française de plusieurs autres terminologies internationales : Classification internationale des soins primaires (ICPC) et WHOART (WHO Adverse Drug Reaction Terminology).
L'UMLS comprend aussi un réseau sémantique de 134 concepts (types sémantiques) et 54 relations (version 1999), sorte d'ontologie embryonnaire et très générale du domaine bio-médical. Dans la mesure où chaque concept du métathésaurus possède un ou plusieurs pères dans la hiérarchie de concepts du réseau sémantique, on pourrait considérer que l'ensemble formé du métathésaurus et du réseau sémantique constitue une ontologie. Dans les faits, cet ensemble ne possède pas les propriétés formelles permettant de s'en servir ainsi dans un langage de représentation des connaissances. L'UMLS n'en demeure pas moins une ressource précieuse pour la recherche documentaire.

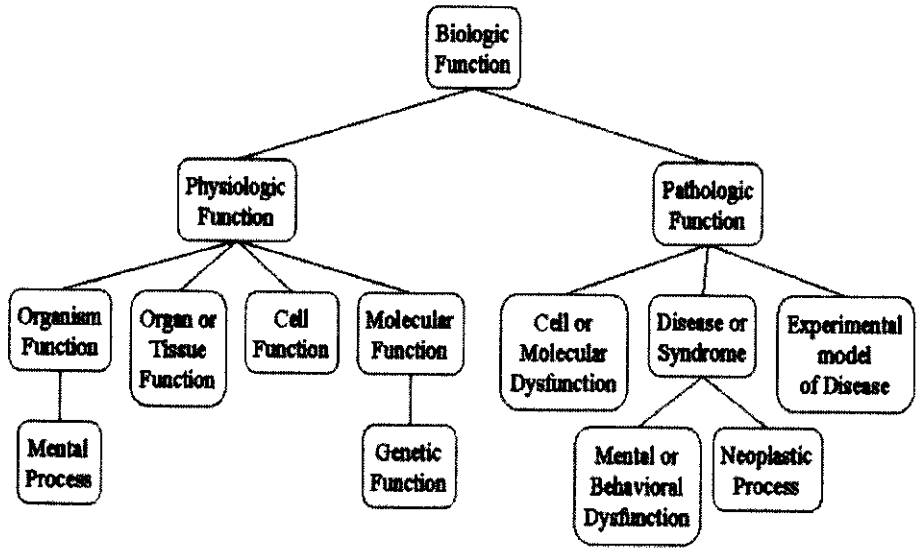


Figure 1: PARTIE DU SEMANTIC NETWORD de l'ontologie UMLS

Le projet européen MENELAS (1992-1995) s'est intéressé à la construction d'une représentation formelle (dans le formalisme des Graphes conceptuels) par analyse de comptes rendus d'hospitalisation rédigés en texte libre. L'objectif était entre autres de produire automatiquement des codes CIM pour les comptes rendus analysés et plus généralement de répondre à des questions concernant les informations décrites dans ces textes .Au moment où MENELAS a démarré, aucune ontologie médicale n'était disponible ; le projet a donc construit sa propre ontologie, qui contenait 1800 concepts et 300 relations à la fin du projet.
Les principes de structuration de cette ontologie se sont révélés proches de ceux employés dans le projet GALEN, mené en parallèle.

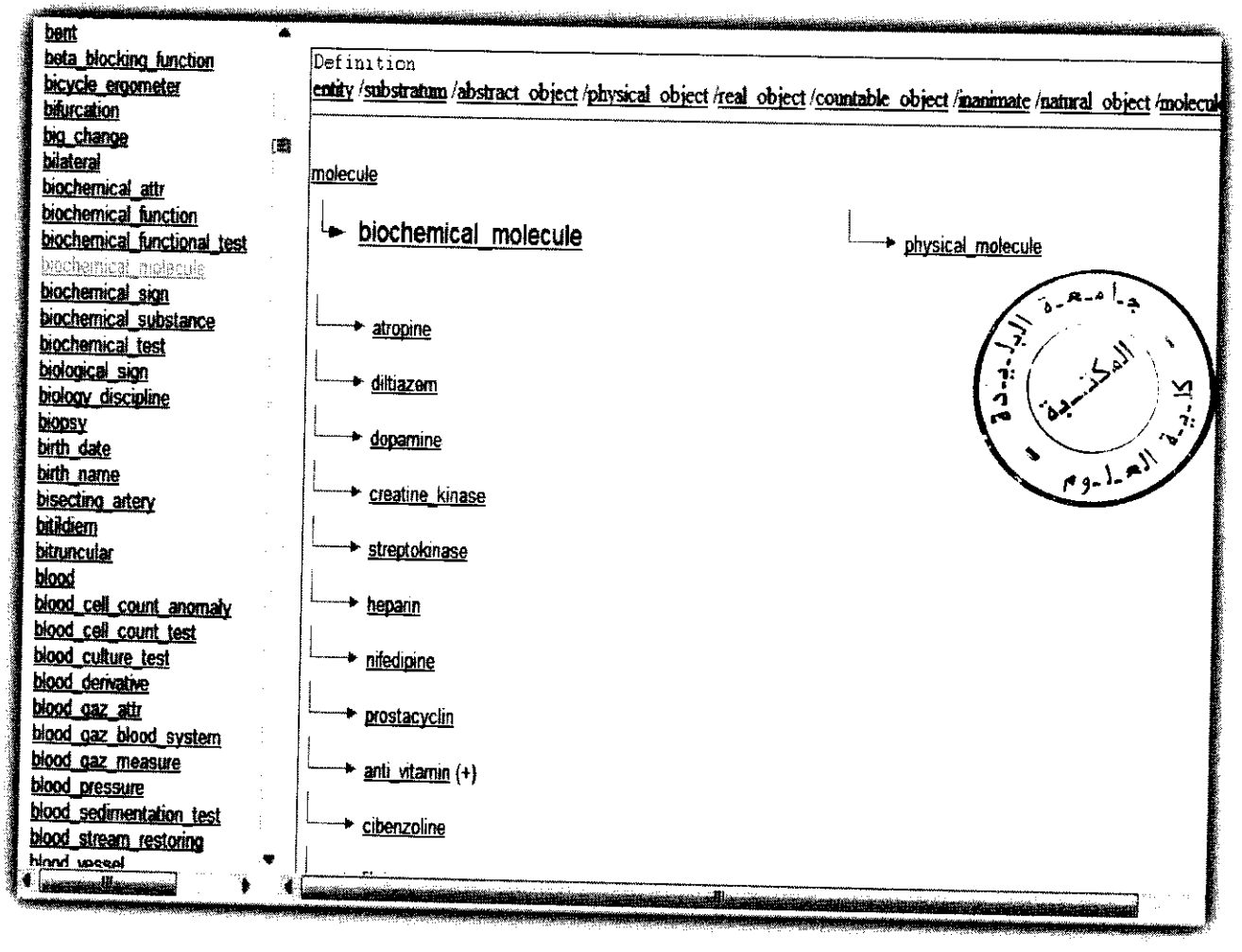


Figure2 : l'ontologie MANELAS