université Paris Ouest Nanterre La Défense

THÈSE

présentée à

l'Université Paris Ouest Nanterre La Défense
Ecole doctorale Connaissance, langage et modélisation (ED139)

pour obtenir le grade de
DOCTEUR

discipline
Sciences du langage
spécialité
Traitement automatique du langage

Titre

Élaboration d'ontologies médicales pour une approche multi-agents d'aide à la décision clinique

Présentée et soutenue publiquement
par

Ying Shen

Le 20 mars 2015

Devant le jury composé de :

Armelle Jacquet-Andrieu (*Directrice*), Université Paris ouest
Joël Colloc (*Directeur*), Université du Havre
Danh Thành Do-Hurinville (*Rapporteur*), INALCO
Jocelyne Faÿn (*Rapporteur*), INSERM
Béatrice Galinon-Mélénec (*Examinateur*), Université du Havre
Jean-Pierre Thierry (*Examinateur*), Université Paris Descartes
Nadia Belrhomari (*Examinateur*), Université Paris Est/ Val- de-Marne

REMERCIEMENT

Cette thèse n'aurait pas été possible sans les précieux conseils de mes professeurs tout au long de mes recherches doctorales. Par la présente, je tiens à remercier ma directrice de thèse, Madame Armelle Jacquet-Andrieu, dont l'assistance, l'orientation et le soutien, du début à la fin de cette investigation linguistique et informatique, m'a permis de développer mes recherches.

Je remercie le Professeur Joël Colloc, directeur de ma thèse, pour avoir encadré mon travail, et pour son aide précieuse, sa patience et ses encouragements inestimables durant ce travail.

C'est un plaisir de remercier Madame Jocelyne Faÿn, Directeur de recherche à l'INSERM (Institut national de la santé et de la recherche médicale) ainsi que Monsieur Danh Thành Do-Hurinville, Maître de conférences, habilité à diriger des recherches (HDR), à l'INALCO (*Institut national des langues orientales*), pour avoir accepté d'être rapporteur, pour ce manuscrit, et pour l'intérêt qu'il a manifesté à l'égard de ce travail de thèse.

Je suis reconnaissante au gouvernement chinois, qui m'a attribué une bourse d'études, pour mes travaux doctoraux.

Je tiens à remercier mes collègues et coéquipiers dont le contact humain était stimulant. J'exprime en particulier ma reconnaissance aux Docteurs Hadi Sabaayon, Mohamed Fliti, Catalina Santana, et Marine Damiani.

Je tiens à remercier mes parents, mon fiancé Guillaume Larsen, et tous mes amis pour l'environnement aimant, dont ils m'entourent, et pour leur soutien constant, au fil des jours.

RESUME

La combinaison du traitement sémantique des connaissances (*Semantic Processing of Knowledge*) et de la modélisation des étapes de raisonnement (*Modeling Steps of Reasoning*), utilisés dans le domaine clinique, offrent des possibilités intéressantes, nécessaires aussi, pour l'élaboration des ontologies médicales, utiles à l'exercice de cette profession.

Dans ce cadre, l'interrogation de banques de données médicales multiples, comme MEDLINE, PubMed… constitue un outil précieux mais insuffisant car elle ne permet pas d'acquérir des connaissances facilement utilisables dans une démarche clinique. En effet, l'abondance de citations inappropriées constitue du bruit et nécessite un tri fastidieux, incompatible avec une pratique efficace de la médecine.

Dans un processus itératif, l'objectif est de construire, de façon aussi automatisée possible, des bases de connaissances médicales réutilisables, fondées sur des ontologies et, dans cette thèse, nous développons une série d'outils d'acquisition de connaissances qui combinent des opérateurs d'analyse linguistique et de modélisation de la clinique, fondés sur une typologie des connaissances mises en œuvre, et sur une implémentation des différents modes de raisonnement employés. La connaissance ne se résume pas à des informations issues de bases de données ; elle s'organise grâce à des opérateurs cognitifs de raisonnement qui permettent de la rendre opérationnelle dans le contexte intéressant le praticien.

Un système multi-agents d'aide à la décision clinique (SMAAD) permettra la coopération et l'intégration des différents modules entrant dans l'élaboration d'une ontologie médicale et les sources de données sont les banques médicales, comme MEDLINE, et de citations extraites par PubMed ; les concepts et le vocabulaire proviennent de l'*Unified Medical Language System* (UMLS).

Concernant le champ des bases de connaissances produites, la recherche concerne l'ensemble de la démarche clinique : le diagnostic, le pronostic, le traitement, le suivi thérapeutique de différentes pathologies, dans un domaine médical donné.

Différentes approches et travaux sont recensés, dans l'état de question, et divers paradigmes sont explorés : 1) l'*Evidence Base Medicine* (une médecine fondée sur des indices). Un indice peut se définir comme un signe lié à son mode de mise en œuvre ; 2) Le raisonnement à partir de cas (RàPC) de fonde sur l'analogie de situations

cliniques déjà rencontrées ; 3. Différentes approches sémantiques permettent d'implémenter les ontologies.

Sur l'ensemble, nous avons travaillé les aspects logiques liés aux opérateurs cognitifs de raisonnement utilisés et nous avons organisé la coopération et l'intégration des connaissances exploitées durant les différentes étapes du processus clinique (diagnostic, pronostic, traitement, suivi thérapeutique). Cette intégration s'appuie sur un SMAAD : système multi-agent d'aide à la décision.

TABLE DE MATIERES

Liste des figures

Liste des tableaux

Liste des figures d'annexe

INTRODUCTION GENERALE

1. Contexte de la thèse

Cette thèse décrit la conception, l'implémentation et l'utilisation potentielle d'un système de réalisation de génération semi-automatique d'ontologies à partir de textes médicaux pour répondre aux interrogations des utilisateurs d'un système à base de connaissances. Cette étude organise la coopération et l'intégration des connaissances utilisées lors des différentes étapes cliniques: diagnostic, pronostic, traitement et suivi thérapeutique, pour fournir une aide à la décision aux personnels de santé.

Une ontologie peut être construite de trois manières différentes : manuelle, semi-automatique ou automatique. Dans cette thèse, nous présentons un prototype de système pour construire une ontologie de manière semi-automatique à partir de textes non structurés et exploitable par un Système Multi-Agents d'Aide à la Décision clinique (SMAAD).

Un SMAAD est conçu pour répondre à des besoins variés: gestion des modules, répondre aux requêtes, extraire des informations, et effectuer des recherches de façon conviviale.

Il gère l'échange d'informations (les requêtes, l'annotation et les ontologies nécessaires) entre l'ingénieur de connaissances, les utilisateurs et le système.

- Il s'agit d'une base de diffusion qui stabilise les ontologies existantes pour répondre aux requêtes de l'extraction de termes et d'associations.
- Il peut extraire les données demandées, formater les données reçues, envoyer ou recevoir des messages normalisés.
- Il structure le réseau et permet la navigation dans la base de connaissances avec les relations (hiérarchiques de correspondance, ou des chevauchements) entre les modules. Les méthodes sont conçues pour différents agents et modules raccordés à ce système.

Les recherches des linguistes (cf. chap. 2, ce volume) visent à identifier et à classer les modèles émergents dans la construction du sens de configurations nominales. La mise à disposition et le partage de nouveaux savoirs nécessitent d'affiner la méta-connaissance des concepts mis en œuvre lors de la description des objets. À partir du texte on peut déterminer les objets et concepts. L'analyse de texte est le point clé pour les recherches de la terminologie. Il est donc important de savoir quelles sont

les informations utiles que nous pouvons obtenir pour effectuer le travail terminologique sur la base conceptuelle. Dans une optique d'acquisition de connaissances à partir de terminologies, il s'agit de repérer les dénominations (objets), leurs caractéristiques (attributs qualifiants), les instances et les relations.

Dans ce contexte, l'approche générale d'acquisition de connaissances textuelles tel que l'approche *bottom-up* est utilisée pour définir les types d'objets à partir des objets, et unir les classes d'objets à des classes plus générales que l'on appelle des superclasses. L'approche *Top-down* permet de vérifier la hiérarchie des classes et sous-classes, qui vise à l'axiomatisation du domaine concerné à partir des objets, types d'objets et associations repérées et leurs interactions. La taxonomie est proposée entre les objets et types d'objets. Notre analyse linguistique permet :

- l'accès à l'information du système multi-agents par les utilisateurs finals à travers des requêtes en langage naturel. Il concerne la structure sémantique du texte de la recherche, de codage et de la gestion fonctionnelle des données linguistiques.
- l'accès à des classifications des activités et des produits à travers des textes descriptifs écrits en langage naturel

Pour le développement des ontologies (cf. chap. 4), nous utilisons les langages de programmation (e.g. le langage Java, le programme en langage OWL, le langage de modélisation orienté objet UML, les logiques de description (LD), le programme par prototypage etc.) et modèles (e.g. Le Raisonnement à partir de Cas) qui traitent et présentent de la construction du sens. Le «Module d'Interprétation» du SMAAD est un contrôleur qui contient tous nos algorithmes et programmes pour l'analyse et l'extraction des termes et des associations de requête. Nos contributions comprennent:

- La conception de la syntaxe et de la sémantique du langage de requête
- La méthode utilisée pour déterminer les réponses aux requêtes

Nous proposons les programmes et algorithmes : « L'analyseur morphologique », «Synonyme », « POS Disambiguation », « Algorithme d'alignement », « Compteur de mots », « Relations des mots remarquables » et « Règles d'analyse syntaxique ».

Le chapitre 5 introduit un processus de représentation linguistique à partir de savoirs extraits de texte. Les éléments importants des textes sont filtrés pour extraire les substantifs (objets), les adjectifs (attributs ou propriétés des objets) et les verbes et

adverbes (actions ou associations). Cette approche utilise des savoirs extraits du corpus et les associations remarquables pour construire automatiquement des structures qui décrivent et classent ces savoirs. Une phase d'extraction de savoirs du corpus, réalisée par l'analyse lexicale et l'analyse syntaxique conditionne cette opération.. Les étapes suivantes permettent de construire une ontologie de manière semi-automatique :

i. L'extraction termes à partir d'un corpus de textes, et expliquer les termes utilisés pour décrire le déroulement clinique habituel
ii. L'extraction des associations à partir d'un corpus de textes.
iii. La construction d'ontologies avec plusieurs hiérarchies conceptuelles superposées
iv. Instancier l'ontologie
v. Extension de la hiérarchie à travers de la découverte supervisée de concepts, de la génération de nouveaux savoirs (nouvelles requêtes disponibles), et de la création des prototypes de mots.
vi. Indexer l'ontologie, et établir les liens d'interactions entre le Dictionnaire Ontologique et le Module de Communication

Le développement de l'ontologie peut se décomposer selon les étapes suivantes: définition, implémentation, intégration et documentation.

Ce chapitre, enfin, introduit la création automatique de pôles de comportements dans le SMAAD. La génération de nouveaux savoirs nécessite d'affiner les processus auto-descriptifs des objets, de définir des règles pour la génération des nouveaux objets, et de créer un dictionnaire d'erreurs. Des outils de contrôle et de traçage évaluent les résultats construits, les représentations, les héritages qui seront utilisés par l'agent superviseur.

Dans ce qui suit, nous commencerons par présenter la structure d'un type d'agent ontologique qui provient de l'évolution de la notion d'Objet Application au concept d'agent puis à l'intégration dans un système multi-agents. Nous décrivons l'architecture multi-agents au travers de la définition et du rôle des différents modules modélisés sous la forme de types d'agents, d'expliciter les entrées et les sorties, la méthode et les interactions entre l'agent ontologique et les autres agents.

Un type d'agent ontologique utilise des connaissances du domaine et de l'analyse sémantique, en vue de la conception d'un système d'interrogation robuste. L'agent ontologique est utilisé pour étendre la hiérarchie du domaine, ainsi que l'interprétation

des requêtes des utilisateurs. Les détails des exemples feront l'objet du chapitre 6, pour illustrer le fonctionnement du SMAAD.

Chaque module du SMAAD fonctionne de manière autonome, mais coopère pour répondre à un besoin spécifique. Un de nos objectifs pour satisfaire une requête est la fourniture d'un accès rapide et exact à l'information souhaitée et à son environnement.

L'opérateur de SMAAD peut accéder à toutes les ontologies existantes dans le système, il peut définir une nouvelle ontologie construite par notre système.

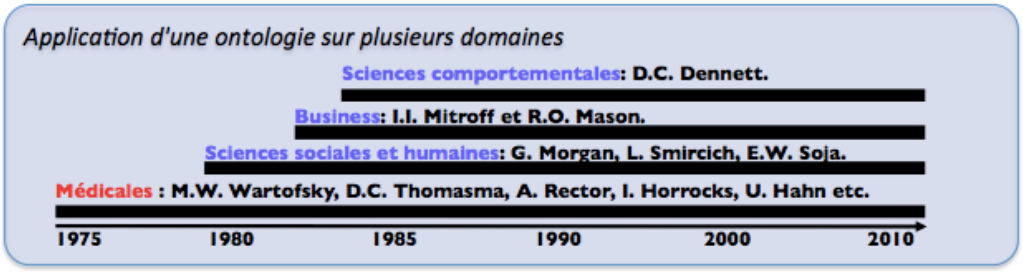
2. Problématique

Les banques de données médicales bibliographiques existantes constituent des outils précieux mais elles se révèlent insuffisants car elles ne fournissent pas des données et des connaissances facilement utilisables dans le contexte de décisions cliniques.

Il existe un certain nombre de verrous scientifiques :

En France, l'utilisateur du système final et le concepteur sont souvent la même personne car ils préfèrent des systèmes bien adaptés aux situations et méthodes de travail locales. Dans l'exercice médical, les connaissances médicales sont souvent tellement spécialisées qu'il n'existe pas de débouchés économiques pour les systèmes d'aide à la décision développés. Ainsi, souvent seule une copie du logiciel est produite et utilisée. De plus, la complexité du domaine médical rend coûteuse l'élaboration d'un système d'aide à la décision efficace. En outre, il est important de considérer que les corpus de connaissances diffèrent selon qu'il s'agisse de traiter l'étape de diagnostic, de pronostic, de traitement ou de suivi thérapeutique. Toutefois, ces différentes étapes cliniques sont fortement imbriquées (Colloc J. 2000).

Dans ce cadre, l'interrogation de banques de données médicales multiples comme MEDLINE, Pub Med… constitue un outil indispensable mais insuffisant, car elle ne permet pas d'acquérir des connaissances facilement utilisables dans une démarche clinique. En effet, l'abondance de citations inappropriées constitue du bruit et nécessite un tri long et fastidieux incompatible avec l'exercice d'une médecine performante. Par exemple, le MEDLINE répond les questions de recherches au format papier, thèse, annotation, site-web etc. Les médecins n'ont pas assez des temps pour regarder tous les documents. Ils ont besoin des suggestions diagnostiques et thérapeutiques dès que possible. Donc les interrogations de banques de données médicales existants ne satisfirent pas les demandes de médecins.



Date	Auteur	Publication à propos de Système Multi-Agents
1995	Ferber	Les systèmes multi-agents
1999	Weiss	Multiagent Systems.A Modern Approach to Distributed Artificial Intelligence
2000	Brooks et al.	An Introduction to Congregating in Multiagent Systems
2001	Colloc et Bouzidi	Architecture d'un système d'aide à la décision : intégration de plusieurs approches
2003	Colloc et Sybord	Un type d'agent cognitif pour l'intégration de plusieurs modèles de connaissance
2007	Colloc et al.	A Clinical Metaknowledge Model
2008	Colloc et Léry	Un métamodèle d'aide à la décision en éthique médicale
2012	Shen, Jacquet-Andrieu et Colloc	Un système multi-agents d'aide à la décision Clinique fondé sur des ontologies

L'application d'ontologies dans le domaine médical a commencé en 1975. Le laboratoire du CNRS UMR 7114, MoDyCo (Modèles, Dynamiques Corpus), de l'université Paris Ouest est une structure de recherche en linguistique, tandis que le laboratoire de l'université du Havre CIRTAI/IDEES, UMR 6266 du CNRS met l'accent sur la recherche de systèmes multi-agents dans le domaine médical, ici. Notre recherche combine des connaissances en linguistique, l'ingénierie informatique pour la réalisation d'ontologies, la médecine, et les systèmes multi-agents pour la coopération de bases de connaissances hétérogènes.

Le système multi-agents proposé dans le cadre de notre équipe, répond aux requêtes concernant des cas similaires et permet donc un enrichissement des bases connaissances. Les suggestions fournies par le système fournissent une aide aux médecins, il constitue une mémoire des cas rencontrés qui s'appuie sur les dossiers médicaux informatisés de patients ayant des pathologies similaires au cas en cours d'examen.

3. Apports d'ontologie SMAAD

Ce projet de recherche, fondé sur le principe de l'acquisition de connaissances médicales est prévu pour fournir une base de connaissances spécifiques, afin de répondre à un large ensemble de questions diagnostiques et thérapeutiques, réutilisables et fondées sur des ontologies médicales.

Le bénéfice s'adresse directement aux médecins et, par voie de conséquence aux patients placés au centre du système. Ces contributions constituent la base pour le développement d'un système de travail qui permet aux utilisateurs de soumettre des requêtes dans le domaine de la médecine, et sert de prototype pour l'extension de la base de connaissances.

Nous fournissons un cadre ontologique formalisé pour l'analyse de l'application des principes fondamentaux du modèle à des questions de la vie réelle dans le domaine de la médecine.

Ce modèle peut être réutilisé dans d'autres domaines de la décision clinique (DSS). Les applications de notre système et les méthodes associées vont soutenir le travail théorique dans la clarification et la modélisation des transformations ontogénétiques, physiologiques, pathologiques et évolutives, afin de soutenir un large éventail d'initiatives cliniques et scientifiques.

Les étapes à propos de l'exploitation et de l'application d'une ontologie sont présentées dans la figure ci-après. Tous les processus seront détaillés dans les chapitres suivants.

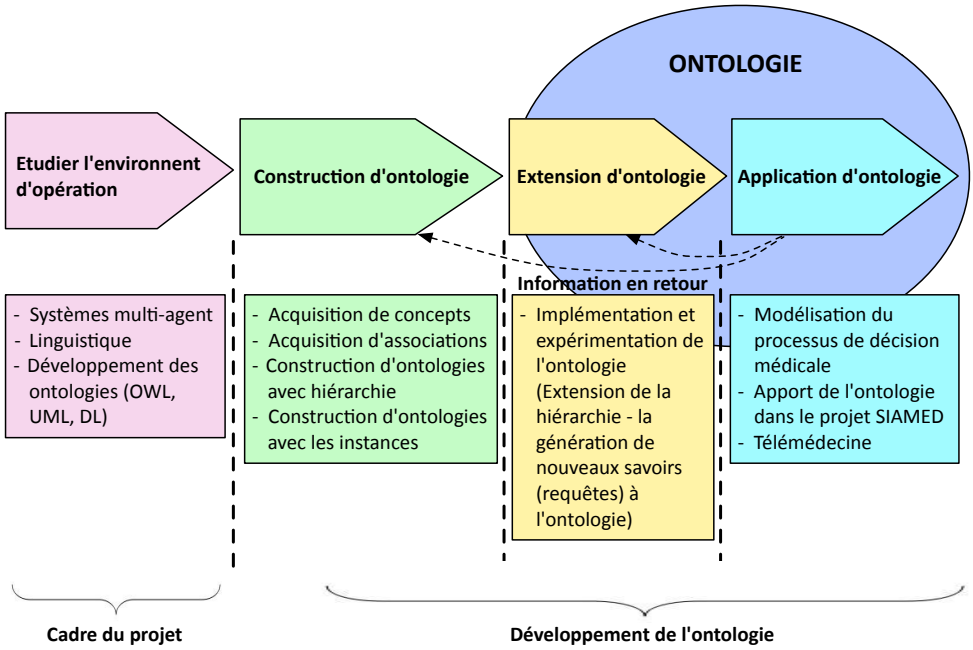


Figure 1 Etapes de construction d'une ontologie

ABREVIATION

Abréviation	Contenu
AL	Analyse Linguistique
FC	Format Conversion
CC	Connaissance sur ses propres capacités
CO	Compile Ontology
DO	« Dictionnaire ontologique »
DSS	Decision Support System
ETA	Extract Term and Association
ETO	Extract Text to Ontology
FC	Format Conversion
GUI	Graphical User Interface
LC	List Other Content
MC	« Module de Communication »
OWL	Ontology Web Language
UML	Unified Modelling Language
RàPC	Le raisonnement à partir de cas
SMAAD	Système multi-agents d'aide à la décision

Chapitre 1 SYSTEME MULTI-AGENTS

L'architecture multi-agent pour la prise en compte de l'ensemble du cycle de prise de décision clinique fournit un cadre pour la conception et l'implémentation de nos ontologies. Chaque agent spécialisé dans une étape clinique exploite une base de connaissances définissant les conduites à tenir qui correspondent à l'état de l'art associé à une base ontologique qui exprime les relations sémantiques entre les termes du domaine considéré. Un système multi-agents d'aide à la décision (SMAAD) permet l'intégration et la coopération des agents spécialisés dans différents domaines de connaissance (sémiologie, pharmacologie, cas cliniques, etc.). L'approche proposée repose sur la spécialisation d'agents cognitifs adaptés aux modèles de connaissances utilisés, aux étapes de la démarche clinique et aux ontologies. Les textes médicaux sont ciblés pour la présentation d'une maladie incluent le diagnostic, le pronostic, le traitement et le suivi-thérapeutique.

Un des éléments importants du système est de réaliser la génération automatique d'ontologies à partir de textes médicaux. Pour obtenir un logiciel de qualité, il faut en maîtriser des processus d'élaboration, la vie d'un logiciel est composée de différentes étapes et la succession de ces étapes forme le cycle de vie du logiciel. Le contrôle de la succession de ces différentes étapes est nécessaire.

1.1 Définition de systèmes multi-agents

Les systèmes multi-agents comprennent l'entité artificielle identifiable, située qui située dans un environnement où il détecte, réagit à et agit sur; l'aptitudes sociales d'interagir avec d'autres agents et humains; et l'autocontrôle de ses comportements.

Le système qui définit les systèmes multi-agents est une «Intelligence Artificielle Distribuée ». (Ferber, J. & Perrot, J. F. 1995) (Ferber, J. 1999) Le SMA utilise des agents cognitifs de grosse granularité, et fait partie des approches d'intégration de composants intelligents.

Les systèmes multi-agents fonctionnent également comme «La vie artificielle». Cette dernière exploite des agents réactifs et cherche à faire émerger des comportements intelligents sociaux sophistiqués à partir des interactions d'agents de petite taille et de structure frustre (Ferber, J., & Drogoul, A. 1992). Il faut donc comparer les caractéristiques des agents réactifs et des agents cognitifs pour la classification des systèmes multi-agents.

Les agents réactifs sont des agents nombreux avec une structure simple. Ils procurent l'émergence de comportements sociaux sophistiqués à travers des êtres déclenchés par les *stimulii* de l'environnement et être pourvus de capacités perceptuelles. Chaque agent est spécialisé et autonome mais ne dispose pas de capacités «intelligentes».

Les SMA réactifs permettent l'émergence de capacités complexes à l'aide d'agents simples, mais ils ne sont pas adaptés à fournir une réponse dans un temps raisonnable avec l'approche supervisée dans un SMAAD (système multi-agent d'aide à la décision).

Nos systèmes multi-agents sont des SMA cognitifs parce qu'ils peuvent intégrer des modèles de connaissances. Les agents cognitifs sont intentionnels et fixent leurs objectifs. Des capacités de prévision leur permettent de fixer des buts comme étant des états virtuels du système à atteindre. Ils construisent une représentation interne de l'environnement du système et de son état. Ils sont autonomes et disposent de moyens spécifiques pour modifier leur environnement (comme des robots). Les agents rationnels disposent d'heuristiques de décision et de connaissances réflexives: leurs propres capacités et limites. Ils peuvent connaître les capacités d'autres agents du système et déléguer des tâches à l'aide de processus de négociation.

Selon (Jennings, N.R. 2000), les agents sont des extensions des objets et des acteurs (Hewitt, C. et al. 1973) (Briot, J.P. 1989). Les capacités des agents cognitifs incluent la communication entre agents, le cohérence, coordination et coopération, de même que l'adaptation, la sélection et l'apprentissage.

Certaines capacités sont mises en évidence ci-dessous (Colloc, J. 2011),

- Les croyances: faits ou heuristiques réputées valides et applicables pour effectuer des tâches: états que l'agent cherche à atteindre.
- Les intentions: confrontation des connaissances réflexives de l'agent avec celles du système.
- L'autonomie de comportement et de décision: choix de participer ou non à une tâche.

Un échantillon de systèmes d'aide à la décision en Médecine de 2000 à 2014 est montré ci-après.

Nom du système	Année	Pays	Brève description	Références bibliographiques
DiagnosisOne	2003	États-Unis	DiagnosisOne dispose d'une plate-forme appelée SmartPath, qui comprend des éléments pour aider à la décision clinique, les ensembles de commandes, l'analyse, et l'enregistrement et la surveillance de la santé publique, construite sur un Microsoft, Oracle, et l'architecture de Red Hat. Il fournit des services de surveillance clinique en temps réel et d'aide à la décision clinique telles que la gestion des infections nosocomiales et le contrôle des infections, etc.	www.alereanalytics.com
DiagnosisPro	2008	États-Unis	*DiagnosisPro* : système expert médical qui offre des possibilités de diagnostic exhaustives pour 11 000 maladies et 30 000 résultats.	(Glasziou P.P. 2008)
DxMate	2013	Pologne	*DxMate* utilise des algorithmes d'intelligence artificielle pour soutenir le processus de diagnostic différentiel, en présentant des diagnostics alternatifs basés sur les symptômes et les résultats de laboratoire fournis par l'utilisateur.	www.dxmate.com
ESAGIL	2012	États-Unis	ESAGIL (*Medical Diagnosis Symptom Checker*) a pour objectif de trouver des correspondances entre les maladies et les symptômes, et de faciliter l'information sur le choix des médicaments appropriés, le choix des traitements et des centres de soins de santé à travers le monde.	http://esagil.org
GRIP	2003	Pays-Bas	Le programme *GRIP* (règlement de la glycémie pour les patients en soins intensifs) peut surveiller les patients ayant des valeurs extrêmes de glucose. Ce programme a également été appliqué avec succès pour surveiller les patients atteints du cancer des testicules non-seminome pour nouvelle b-HCG, AFP, et les valeurs de la LDH.	http://grip-glucose.sourceforge.net/
Micromedex solutions	2010	États-Unis	*Micromedex* est utilisé par les cliniciens pour informer et éduquer les décisions de soins, et obtenir l'accès en temps réel à la connaissance clinique fondée sur des preuves.	http://micromedex.com
Sharecare-PKC	2012	États-Unis	La *Dynamique plate-forme* de technologie sociale de *Sharecare* fonctionne avec un système de gestion des connaissances cliniques de la PKC et apporte la médecine fondée sur des preuves pour les patients et les médecins, et d'apporter des informations cliniquement pertinentes.	www.pkc.com
TheraDoc	2009	États-Unis	*TheraDoc* aide les organisations de soins de santé à améliorer leurs activités, en regroupant les données fragmentées de systèmes de source de l'organisation (LAB, pharmacie, microbiologie, ADT, etc.), ce qui permet aux cliniciens d'être alertés des risques potentiels, et de leur donner des connaissances et des conseils cliniques fondées sur des preuves.	www.theradoc.com
VisualDx	2011	États-Unis	*VisualDx* est un système d'aide à la décision clinique basé sur le Web, codé en Java. Il est conçu pour le diagnostic et le traitement des troubles dermatologiques. Il est développé par Images logiques et peut être utilisé dans les soins cliniques, pour développer des diagnostics différentiels.	www.visualdx.com
WebMD	2005	États-Unis	*WebMD* vise à offrir une expérience appropriée, et à remplir divers besoins différents (utilisation de sites-web de santé pour chercher les traitements adaptés, divers besoins de recherche en santé, les soutiens de la communauté, le e-commerce etc.) dans les moyens les plus appropriés possibles.	www.webmd.com
Wolters Kluwer	2008	États-Unis	*Wolters Kluwer Health* prend en charge la diffusion de	www.wolterskluwerheal

Health			l'information de santé *via* des formats interactifs avec ses bases de données bibliographiques et de référence, les logiciels d'information sur les médicaments, les outils de point sur des soins et des systèmes d'information sur le Web.	th.com
ZeroMD	2013	États-Unis	*ZeroMD* dédié aux professionnels de santé, avec des outils basés sur le Web, permet d'aider à l'amélioration du diagnostic médical, en utilisant le raisonnement à partir de cas et la collaboration de la communauté.	www.zeromd.com
Zynx Health	2004	États-Unis	*Zynx Health* comporte des plates-formes pour les soins hospitaliers et ambulatoires, il comprend plus de 500 règles cliniques de soutien à la décision et 1100 modèles de groupes d'ordonnances dans ses différents logiciels.	www.zynxhealth.com

Tableau 1 Échantillon de systèmes d'aide à la décision en Médecine de 2000 à 2014

Sur la base du SMA, l'analyse et les apports de l'approche SMAAD et TACG sont traités dans la suite de la thèse.

1.2 Architecture de système SMAAD et TACG

Le système construit dans cette thèse permet la génération semi-automatique d'ontologies est accompli dans l'architecture du Type d'Agent Clinique Général (TACG) et système multi-agents d'aide à la décision (SMAAD). Ces systèmes recherche permettent la conception de sociétés artificielles d'agents intelligents, et travaille pour l'intelligence artificielle distribuée.

Architecture d'un système multi-agent d'aide à la décision (SMAAD) clinique (Colloc J. & Sybord C. 2003)

Un système multi-agents d'aide à la décision (SMAAD) permet l'intégration et la coopération des agents spécialisés dans différents domaines de connaissances (sémiologie, pharmacologie, cas cliniques, etc.).

Le SMAAD instancie les types d'agents TAS, TACG, TAMC et TASDC au cours de deux étapes de spécialisation (figure 2). Durant la première spécialisation, les Types d'Agents Modèles de connaissances (TAMC) héritent des modules et fonctionnalités du TACG (figure 3 et 4) et sont dotés de fonctionnalités spécifiques d'un modèle des connaissances. Par exemple, un TAMC à base de règles aura accès à un moteur d'inférence. Un TAMC épidémiologique permettra de connaître l'incidence et la prévalence d'une pathologie. Un seul agent superviseur est instancié. Un AST est

un Agent spécialisé dans une tâche clinique diagnostique (Δ), pronostic (Π), traitement (Θ), et suivi-thérapeutique (SΘ).

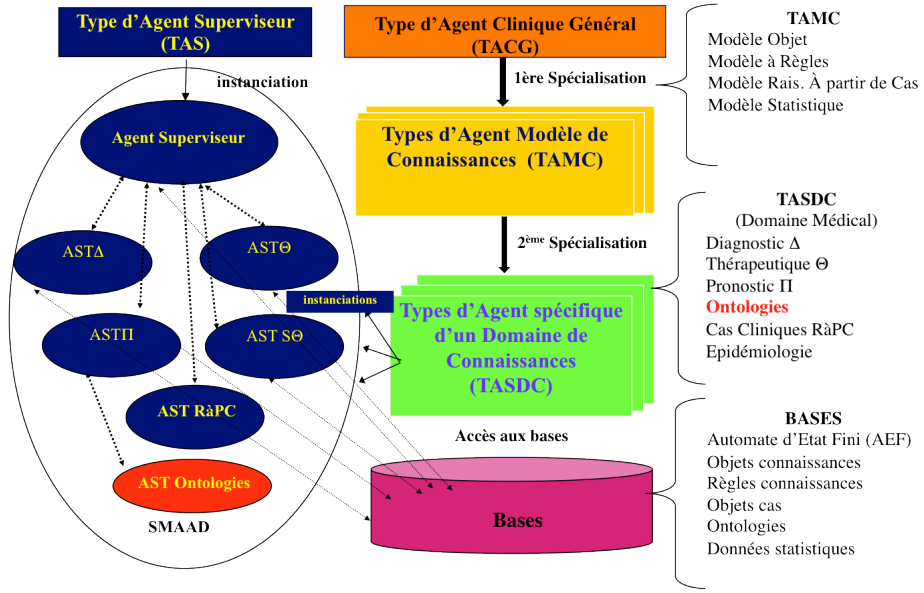


Figure 2 Spécialisation des agents impliqués dans un système multi-agent d'aide à la décision SMAAD (Colloc J. 2000)

La deuxième spécialisation produit des Types d'Agents spécifiques du Domaine de Connaissances (TASDC) et des étapes cliniques spécifiques Δ, Π, Θ, SΘ, par exemple : le ***diagnostic*** de maladies infectieuses, le ***pronostic*** de cette maladie selon sa sévérité et l'état général du patient, le ***traitement*** d'une maladie bactérienne par antibiothérapie. Ces agents utilisent des modèles de raisonnement et de connaissances hérités des TAMC (des règles de production, des distances entre objets connaissances, des objets cas cliniques, des statistiques etc.). Au cours de chaque étape clinique, les agents adressent des requêtes à des bases de connaissances spécifiques.

Dans cette approche hybride l'autonomie des agents spécifiques du domaine (TACD) est conservée : ils ont la capacité d'accepter ou refuser une tâche selon leur connaissance réflexive. Autre apport de l'approche SMAAD : c'est le superviseur qui permet de guider le processus de décision et d'obtenir une réponse pour l'utilisateur en un temps raisonnable (domaines où lorsque la décision arrive trop tard elle devient inutile dont la santé, gestion, crises etc.)

Architecture du Type d'Agent Clinique Général (TACG)

Les figures 3 et 4 décrivent le Type d'Agent Clinique Général (TACG) dont les modules sont fonctionnellement liés les uns aux autres et assurent la communication,

la négociation, la décision, l'évaluation du contexte (exprimé par le superviseur) et la gestion des tâches.

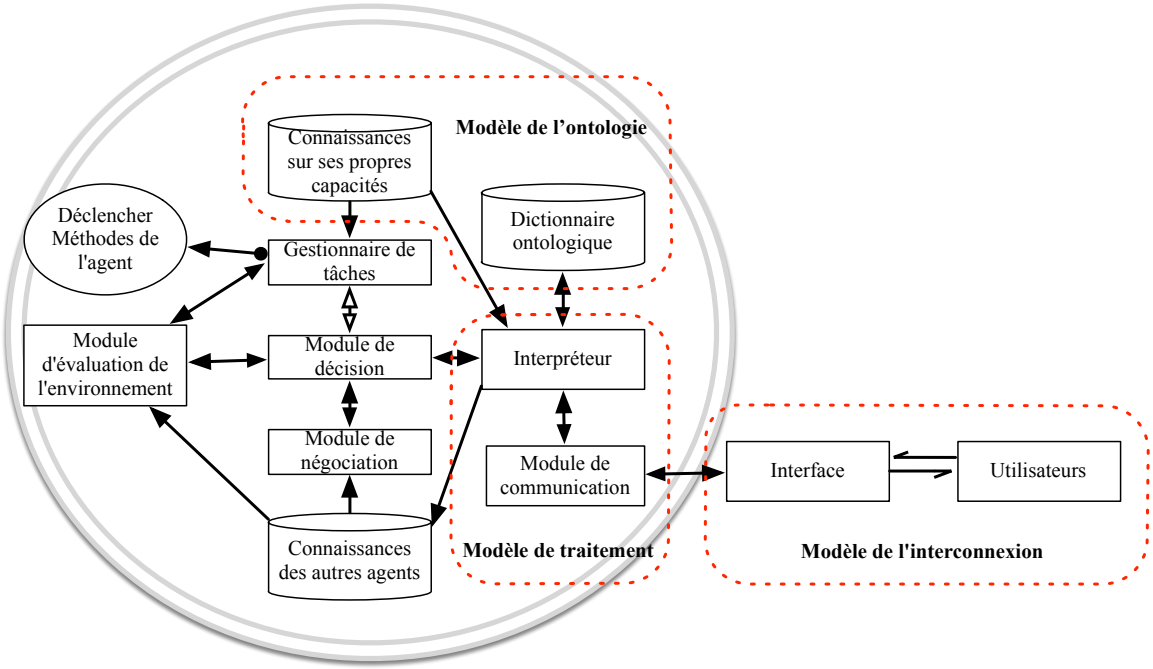


Figure 3 Architecture du Type d'Agent Clinique Général (TACG) tiré de l'article (Colloc J. & Sybord C. 2003)

Dans le système Type d'Agent Clinique Général (TACG), après avoir reçu les requêtes de utilisateurs finaux, le « **Module d'Interface** » transfère les requêtes au modèle de traitement et modèle de l'ontologie pour les analyses et les extractions. En fin de compte, il renvoie les réponses au format annotation, texte ou documents aux utilisateurs.

Le « **Module de Communication** » gère les communications entre les utilisateurs et les ontologies comme une boîte aux lettres.

Dans le « **Module Interpréteur** », le contrôleur de l'algorithme contient tous les algorithmes et les programmes pour l'analyse et l'extraction des termes et des associations. Il peut fournir des programmes et des algorithmes adaptés au système selon le réglage par défaut défini par l'ingénieur de connaissances. Programmes et algorithmes disponibles sont présentés ci-après et détaillés en Annexe.4 :

PROGRAMES

PACKAGE-1. Morphologie

Programme-1. Analyseur morphologique-Lemmatisation

PACKAGE-2. TaggedToken

Programme-2. Synonyme

Programme-3. POS Tagger

PACKAGE-3. Corpus

Programme-4. Algorithme de l'alignement

Programme-5. Dictionnaire

Programme-6. Hiérarchie

Programme-7. Instance

PACKAGE-4. ExtractRelationForWord

Programme-8. Les relations remarquables – RelationExample

Programme-9. Extraire des relations des mots cibles

PACKAGE-5. WordCounter

Programme-10. Compteur de mots

PACKAGE-6. BuildPathologieOntologie

Programme-11. La construction d'ontologie pathologique

Programme-12. Ajouter, supprimer ou modifier les concepts, relations, hiérarchies et instances d'ontologie

PACKAGE-7. OntologyGui

Programme-13. Visualisation d'ontologie à travers de GUI

ALGORITHME

Algorithme-1. La règle d'analyse syntaxique

Le module «**Connaissances sur ses propres capacités**» contient les connaissances existantes du système présenté ici. Après avoir analysé la requête, le « **Module Interpréteur**» consulte le module « **Connaissances sur ses propres capacités** » si la connaissance est disponible, les réponses seront «Oui» ou «Non». Avec le signal « Oui », le processus va continuer et la requête sera transférée au «**Dictionnaire Ontologique**» pour chercher les textes correspondants. Si « Non», le processus se termine et retourne le résultat vide.

Le « **Dictionnaire Ontologique**» contient tous les objets et concepts concernés : les dictionnaires existants, les ontologies avec les concepts, les synonymes et les associations dans la hiérarchie. Si les ontologies ou dictionnaires existants correspondent aux requêtes d'entrée, ils renvoient les textes correspondants au « **Module Interpréteur** », puis aux utilisateurs *via* l'interface.

Le « **Dictionnaire Ontologique** » et le « **Module Interpréteur** » expriment les interactions entre la terminologie liée aux domaines et les bases de connaissances cliniques.

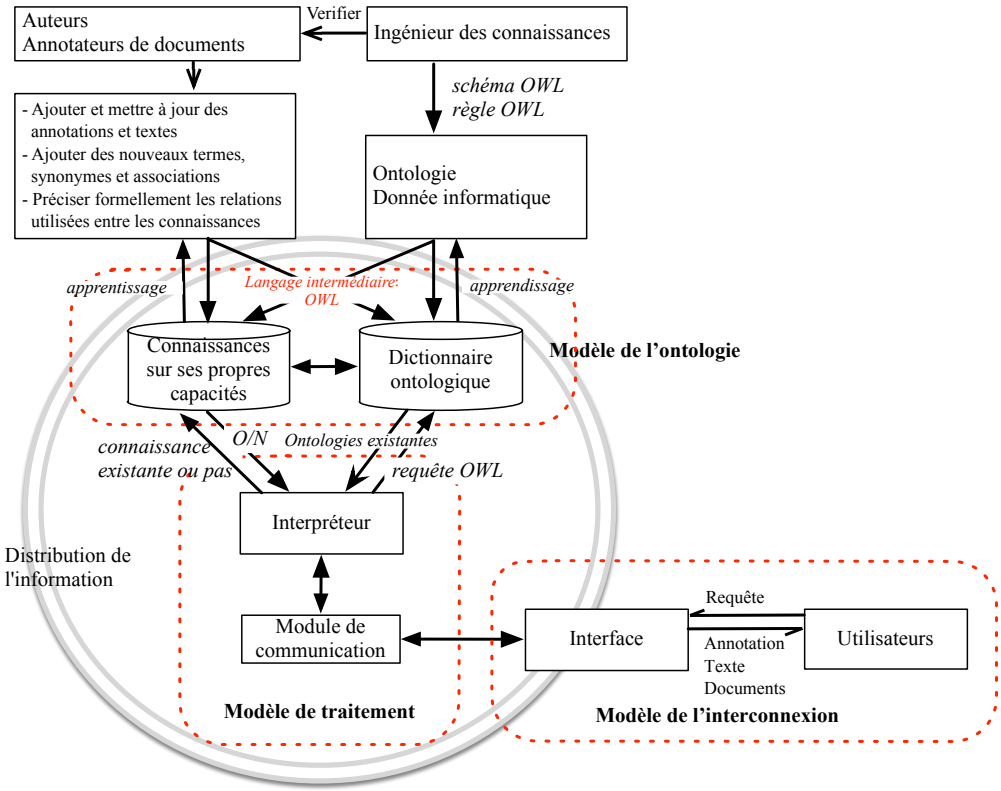


Figure 4 Modèles de la distribution de l'information (TACG)

1.2.1 Les rôles de sous modèles et leur interactions

La description de la conception ou de la conception architecturale du logiciel vise à diviser un système en un ensemble de sous-systèmes structurés, qui permettent de répondre aux besoins identifiés au cours du processus de spécification des exigences (IEEE Standard 1016-1998). Le processus de conception est composé de trois phases,

i. Décomposer le système en sous-systèmes principaux et identifier les liens entre les sous-systèmes,
ii. Définir un modèle de contrôle,
iii. Décomposer les sous-systèmes en modèles.

Afin d'intégrer plusieurs modèles de connaissances à l'aide d'un agent cognitif, il faut résoudre certains problèmes :

- Comment faire coopérer des modèles de connaissances hétérogènes dans un système d'aide à la décision.
- Comment obtenir d'un SMA une décision exploitable en un temps raisonnable ?

Une approche cognitive intégrée dans un SMA supervisé peut constituer une solution. Selon la notion de modèle de (Colloc J. et al. 2007), le modèle est une

représentation abstraite de la réalité qui exclut certains détails du monde réel. Il permet de réduire la complexité d'un phénomène en éliminant les détails qui n'influencent pas son comportement significatif, et reflète ce que le concepteur croit important pour la compréhension et la prédiction du phénomène modélisé, les limites du phénomène modélisé dépendent des objectifs du modèle.

Les systèmes d'information sont fondés sur une approche symbolique. Un modèle est un ensemble de concepts qui permet de faire correspondre à chaque objet du monde réel (ceux qui nous concernent) un symbole dans le système puis dans un ordinateur quand ce système est automatisé. Ce symbole selon sa complexité sera représenté à l'aide d'un système de codage (les données et les actions ou traitements sur ces données sont codés). L'informatique a ainsi exaucé le souhait de Platon et le rêve de Leibniz qui pensaient représenter l'ensemble du monde à l'aide de l'ensemble des entiers naturels N.

1.2.1.1 Modèle d'ontologie

Le module «Dictionnaire Ontologique» gère les archives locales au travers de recherches et extrait les références qui correspondent aux requêtes, informe les agents enregistres de l'arrivée des nouvelles connaissances, et mémorise de nouvelles connaissances.

Dans notre approche, les modèles sous-jacents aux langages de programmation (le programme en langage Java, les langages sémantiques comme XML et ses dérivés OWL, le langage de modélisation graphique UML, les Logiques de Description etc.) sont utilisés pour traiter et présenter la construction du sens. Le langage OWL permet de créer, interroger ou supprimer des composants d'ontologies tels que des classes, des propriétés et des instances. Le langage OWL permet également de stocker la structure de l'ontologie générée. Le prototype est capable de générer l'ontologie en utilisant la syntaxe OWL.

Dans l'ontologie construite, après avoir identifié les nouveaux concepts et associations, le prototype du système vérifie si ces nouveaux éléments sont déjà existants dans le module « Connaissances sur ses propres capacités ». Sinon, l'association, le concept et ses synonymes seront ajoutés à l'ontologie générée.

Les apports de l'ontologie comprennent :

- promouvoir la réutilisation de connaissances dans plusieurs applications ;

- faciliter la maintenance de logiciels grâce à une représentation explicite de l'ontologie sur laquelle ils sont basés ;
- rendre pérennes des connaissances ontologiques, dans une perspective de mémoire organisationnelle.

Ce module comprend le système de prototype : la description de la conception complète se trouve à l'annexe.1-3. Cette description de la conception conforme à la norme IEEE 1016-1998 (IEEE Standard 1016-1998)

Modèle de l'ontologie	
Sous modèle	Connaissance sur ses propres capacités / Dictionnaire ontologique
Responsabilités	Le Modèle de l'ontologie vise à construire l'ontologie en utilisant un ensemble des programmes, algorithmes et règles heuristiques pour organiser les termes et les associations extraits. - Description des termes d'entité ou individu - Clarifier les relations ou associations, les propriétés ou attributs - Chercher et vérifier les synonymes et instances de termes. - Ajouter les termes et associations extraites à l'ontologie existant. - Fournir les textes relatifs au «Module Interpréteur» si les ontologies ou dictionnaires existants correspondent avec les requêtes d'entrée.
Collaborateur	Modèle de Traitement (1) Modèle de l'interconnexion (2)
Méthodes externes	OPERATION-1~13 (qui sont présentés en Annexe.2)
Interactions	OWL syntaxe

Tableau 2 Le Modèle de l'ontologie

Module **Dictionnaire Ontologique**	
Super Modèle	Modèle d'ontologie
Responsabilités	Gérer les archives locales : - Rechercher et extraire les références qui correspondent aux requêtes - Informer les agents enregistres de l'arrivée des novelles connaissance - enregistres de nouvelles connaissances sur les documents de la mémoire
Collaborateur	Connaissance sur ses propres capacités (1) Interpréteur, Modèle de Traitement (1) Modèle de l'interconnexion (2)
Interfaces externes	Annotation requête de OWL
Compétence	Fixer a la base locale Exploiter la base locale Répondre par interroger les connaissances locales
Interactions	Requête - OWL

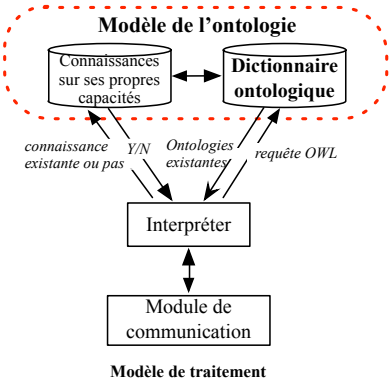


Tableau 3 Le Module «Dictionnaire ontologique »

1.2.1.2 Modèle de Traitement

Le « Modèle de Traitement » gère les communications entre les utilisateurs et les ontologies afin de fournir des programmes et des algorithmes adaptés au système

selon l'ordre de l'utilisateur ou le réglage par défaut défini par l'ingénieur de connaissances.

Modèle de Traitement	
Sous modèle	Module de communication / Interpréter
Responsabilités	- Gère les communications entre les utilisateurs et les ontologies - Fournir des programmes et les algorithmes adaptés au système selon l'ordre de l'utilisateur ou le réglage par défaut défini par l'ingénieur de connaissances - Assumer le changement de format de requête avec OWL. - Présenter la partie dynamique du Système informatique en termes de processus et opérations.
Collaborateur	Modèle de l'ontologie (1) Modèle de l'interconnexion (1)
Méthodes externes	OPERATION-1, 2, 14, 15 (qui sont présentés dans l'Annexe.2)
Interactions	OWL syntaxe et UML

Tableau 4 Le Modèle de Traitement

1.2.1.3 Modèle d'interconnexion

Le module « **Interface** » supervise les interactions entre les utilisateurs et le Type d'Agent Clinique Général (TACG). Ce module transfère les données aux autres modules du système, il surveille la distribution des tâches pour traiter les requêtes, il alloue une nouvelle annotation au dictionnaire ontologique, et notifie l'arrivée de nouvelles annotations pour déclencher les fonctions appropriées. Avec ces fonctions, le « **Modèle de l'interconnexion »** transfère les requêtes au « **Modèle de traitement** » et au « **Modèle de l'ontologie** », pour effectuer les analyses et les extractions de termes et d'associations. En fin de compte, il renvoie les réponses au format annotation, texte ou documents aux utilisateurs.

Le module «**Interface**» supervise les interactions entre les utilisateurs et le Type d'Agent Clinique Général (TACG). Ce module transfère les données aux autres modules de système, surveille la distribution des tâches pour traiter les requêtes, alloue une nouvelle annotation au dictionnaire ontologique, et notifie l'arrivée de nouvelles annotations pour déclencher les fonctions appropriées. Avec ces fonctions, le **modèle de l'interconnexion** transfère les requêtes au modèle de traitement et au modèle de l'ontologie pour effectuer les analyses et les extractions de termes et d'associations. En fin de compte, il renvoie les réponses au format annotation, texte ou documents aux utilisateurs.

Modèle de l'interconnexion	
Sous modèle	Interface / Utilisateur
Responsabilités	- Transfère les requêtes de utilisateurs finals au modèle de traitement et modèles de l'ontologie pour analyses et extraction. - Renvoie les réponses au format annotation, texte ou documents aux utilisateurs
Collaborateur	Modèle de Traitement (1) Modèle de l'ontologie (2)
Méthodes externes	OPERATION-1, 2, 15 (qui sont présentés dans l'Annexe.2)
Interactions	Requête – OWL – UML

Tableau 5 Le Modèle de l'interconnexion

Module **Interface**	
Super Modèle	Modèle de l'interconnexion
Responsabilités	Superviser les interactions : - Transférer de contact aux autres modules de système - Surveiller les distributions des tâches pour résoudre requête - Allouer une nouvelle annotation à l'archive - Notifier l'arrivée de nouvelles annotations pour déclencher les fonctions poussoirs
Collaborateur	Utilisateur, Module de communication (1) Interpréteur (2) Modèle de l'ontologie (3)
Interfaces externes	Annotation requête de OWL
Compétence	Gestion des requêtes et soumissions
Interactions	Requête - OWL

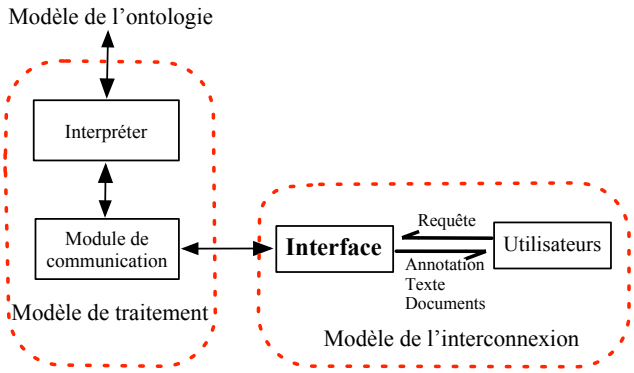


Tableau 6 Le Module Interface

1.2.1.4 Interaction entre Modèle de l'interconnexion et Modèle de traitement

On décrit la conception, l'implémentation et l'utilisation potentielle d'un système de génération automatique d'ontologies à partir de textes médicaux, pour répondre aux requêtes des utilisateurs du système d'exploitation de la base de connaissances.

Le Modèle de traitement est un « zoom » sur le Modèle de l'interconnexion. Dans les « **Modèle de l'interconnexion** », on représente les messages échangés entre modules. Dans les « **Modèle de traitement** », on représente comment un module de l'organisation réagit quand il reçoit ce message et quelle opération il effectue.

Les contributions informatiques pertinentes de ce travail comprennent :

- La conception de la syntaxe et de la sémantique du langage de requête,
- La méthode utilisée pour déterminer les réponses aux requêtes,
- L'interface utilisateur, pour présenter les résultats.

Les requêtes provenant de diverses sources numériques, y compris les experts du domaine, des articles évalués par les pairs et des livres de référence, ont été utilisées

pour tester le système et pour illustrer son utilisation potentielle dans les études de médecine.

1.2.1.5 Cohérence entre Modèle de traitement et Modèle de l'ontologie

De « **Modèle de l'ontologie** » à « **Modèle de traitement** », on doit s'assurer que toutes les données nécessaires sont représentées. Pour chaque élément du « Modèle de l'ontologie », on vérifie qu'il est bien utilisé dans au moins une opération, pour s'assurer que seules les données nécessaires sont représentées.

Le « **Modèle de l'ontologie** » doit vérifier que la connaissance correspondant à la requête entrée est disponible, puis transfert le signal « Oui » ou « Non » au « **Modèle de traitement** », pour chercher les textes correspondants, ou bien terminer et retourner un résultat vide.

Dans cette thèse, nous allons mettre l'accent sur les réalisations de « Modèle de l'ontologie » et sur la coopération avec les autres modèles.

1.2.2 Méthodes

Une inférence est un processus de raisonnement qui s'appuie sur des connaissances acquises et s'articule autour de règles fondamentales pour obtenir de nouvelles informations. L'Interface Homme-Machine fournit une visualisation de l'ontologie qui permet à l'utilisateur de comprendre le vocabulaire utilisé par le système informatique et de mieux formuler ses requêtes. (Kassel, G. 2002)

Tous les détails des méthodes entre les différents modules et leurs variables sont présentés en annexe.2. Un bref aperçu est présenté ci-dessous.

1.2.2.1 Extraction des termes et associations

Les méthodes travaillées dans le « **Modèle de l'ontologie** » permettent d'extraire des termes et des associations, à partir d'un corpus de textes et d'ajouter les termes, les associations et les synonymes au « **Dictionnaire ontologique** ». En outre, les méthodes sont responsables de l'enregistrement des termes et des associations en fichiers texte.

Méthodes	Fonction
ExtractTermsAndAssociation()	Extrait les termes et les associations des termes du corpus « **Dictionnaire ontologique** ».
getAttribute()	Obtenir des attributs de la classe cible dans la hiérarchie
getParentAttribute()	Obtenir des attributs de ses classes parentes dans la hiérarchie
getDomain()	Demander et obtenir le « domaine » de la propriété cible
getRange()	Demander et obtenir le « range » de la propriété ciblée
getDataType()	Obtenir le « type de données » pour analyser la propriété cible
AddClass()	Ajouter un terme au corpus de textes « **Dictionnaire ontologique** ».
AddProperty()	Ajouter une association entre deux termes de l'ontologie.
ExtractSynonyms()	Extraire ou ajouter des synonymes au « **Dictionnaire ontologique** ».
InputOwlRule()	Entrer les règles et les schémas d'OWL par l'utilisateur, l'auteur ou l'ingénieur de connaissances
CopyTextToDO()	Copier les textes sélectionnés dans le « **Dictionnaire ontologique** »

Tableau 7 Exemples de méthodes pour extraire des termes et associations

1.2.2.2 Construction de l'ontologie

Ce module fournit des méthodes pour ajouter des termes et des associations à l'ontologie générée en sélectionnant l'algorithme et l'heuristique adaptés.

Les fonctionnalités du module sont appelées par le module GUI (*Graphic User Interface*), afin de visualiser la construction d'une nouvelle ontologie. Cette interface graphique est créée pour permettre à l'utilisateur d'interagir avec les éléments précédents et d'attribuer des valeurs différentes aux paramètres, durant le processus de construction de l'ontologie.

Méthodes	Fonction
ConstructOntology()	Démarrer le processus de construction d'ontologies pour établir l'ontologie avec la liste des termes et associations appariés, les heuristiques sélectionnées et les schémas et règles OWL entrées. L'ontologie construite est enregistrée dans un fichier OWL.
UpdateOnto()	Initialiser tous les paramètres à mettre à jour dans l'ontologie existante

Tableau 8 Exemple des méthodes pour la construction d'ontologie

1.2.2.3 Interconnexion entre l'utilisateur et modules de système multi-agents

L'interface utilisateur doit être facile à utiliser, et doit aussi satisfaire toutes les exigences spécifiées dans cette thèse. L'utilisateur doit pouvoir d'utiliser toutes les fonctionnalités du système ontologique à l'aide de l'interface graphique.

Pour permettre aux utilisateurs de bâtir des requêtes, la méthode est écrite en OWL et elle fournit les fonctionnalités suivantes :

- Garder la trace des résultats de requêtes préalables, afin que l'utilisateur puisse y revenir ;
- Montrer le résultat obtenu en sortie sous de multiples formes, y compris du texte, un arbre, des graphiques et des références.

L'interface graphique (GUI) permet à l'utilisateur d'interagir avec les fonctions des modules mentionnés et de spécifier les valeurs de paramètres différentes.

Méthodes	**Fonction**
VisDO()	Permet de visualiser le contenu du corpus de texte (« **Dictionnaire ontologique** »).
VisTermExtAlgorithms()	Permet de visualiser les algorithmes du processus d'extraction des termes.
VisAssociationExtAlgorithms()	Permet de visualiser les algorithmes du processus d'extraction des associations.
WriteTerm()	Présente le terme(s) extrait par GUI.
WriteAssociation()	Présente l'association(s) extraite par GUI.
ListTermInDO()	Récupère les labels de tous les termes du « **Dictionnaire ontologique** » et il les écrit dans un nouveau fichier texte.
ListAssociationsInDO()	Récupère les labels de toutes les associations, ses domaines et range dans le « **Dictionnaire ontologique** », il les écrit dans un nouveau fichier texte.

Tableau 9 Exemple des méthodes pour l'interconnexion entre l'utilisateur et SMA

1.3 Requêtes

Une ontologie linguistique peut permettre de comprendre les requêtes de l'utilisateur formulées en langue naturelle (représentation du contenu). (Kassel, G. et al. 2000)

Les requêtes qui proviennent de diverses sources informatiques, y compris celles des experts du domaine, évaluées par les pairs, des articles et des ouvrages de référence, ont été utilisées pour tester le système et illustrer son utilisation potentielle dans les études de médecine.

Lors de la recherche d'information, la base de données impliquée inclut des références bibliographiques des articles (titre, auteur, journal, etc.). Les systèmes les

plus avancés peuvent commander une copie, ou obtenir un résumé ou la totalité de l'article, en temps réel.

Nous concevons le prototype d'un système d'interrogation en langage naturel d'un ensemble de documents. La hiérarchie de domaine est représentée dans la « **description logique** » (DL), offrant une efficacité et une tolérance aux pannes de données incomplètes ou erronées. Les mécanismes d'inférence logiques, fournis par la DL, sont utilisés pour étendre dynamiquement le modèle de domaine, et compléter les informations manquantes, identifiées par la requête de l'utilisateur. Le système utilise les fichiers d'index pour récupérer les documents. Les index peuvent être des mots clés, les termes, les structures syntaxiques ou sémantiques.

Le SMAAD permet l'accès aux classes de toute hiérarchie et de l'index. Son moteur de recherche accepte la requête de l'utilisateur, composée d'un ensemble de mots-clés, écrite dans un langage de commande ou en langage naturel.

Le module linguistique utilise des connaissances du domaine et de l'analyse sémantique, en vue de la conception d'un système d'interrogation robuste. Le module linguistique est utilisé pour étendre la hiérarchie de domaine et l'interprétation des requêtes des utilisateurs.

Une grammaire capable de détecter les ambiguïtés et analyse les requêtes, puis elles sont transformées en représentations sémantiques, afin d'aligner les contenus de l'index. La représentation sémantique de la requête pourrait être un ensemble de mots clés ou des représentations sémantiques plus complexes. Tous les détails de l'analyse linguistique figurent au chapitre 5.

Pour améliorer la précision de la réponse, le système utilise les liens sémantiques avec le corpus existant et des liens logiques entre les différents modules. Les réponses sont classées par l'ordre de fréquence et de pertinence.

Les constructions linguistiques et la connaissance du domaine requièrent un minimum d'efforts du concepteur humain, alors qu'il intègre des techniques de traitement du langage naturel. Le système pourrait être facilement porté par un autre domaine, en raison du maintien dynamique de la base de connaissances du domaine. Les nouveaux concepts inférés dans les nouveaux documents sont ajoutés et validés dans la hiérarchie existante. Cette méthode ne convient pas aux domaines non réservés, en raison de la taille limitée de l'ontologie supportée par le système.

1.3.1 Requêtes similarités, dissimilitudes

Ce paragraphe décrit comment rechercher les similarités et dissimilitudes entre requêtes, dans la perspective d'une approche analogique ou du Raisonnement à partir de cas) (RdeC).

Nous étudions ici une classe différente de requêtes destinées à trouver les similarités ou les dissimilitudes existant entre les objets d'une base de données et un objet cible donné. Une mesure de distance exprime la similitude entre deux objets. Certains moyens pour mesurer la distance sont utilisés dans ces applications.

La distance sémantique peut supporter de la reconnaissance d'objets car on dispose d'une métrique dans l'espace où sont placés les objets. Une distance métrique est une **fonction d** qui transforme une paire d'objets en des nombres non-négatifs (positifs ou nuls) satisfaisant les propriétés suivantes :

Soit des objets O1, O2, O3,

- $d(O_1, O2) >= 0$ et d(O1, O2) = 0 si et seulement si O1=O2 (définition de la non-négation).
- d(O1, O2) = d(O2, O1) (symétrie)
- d(O1,O2) <= d(O1,O3) + d(O3, O2) (inégalité triangulaire)

Tout le problème est de définir **d** selon l'application désirée.

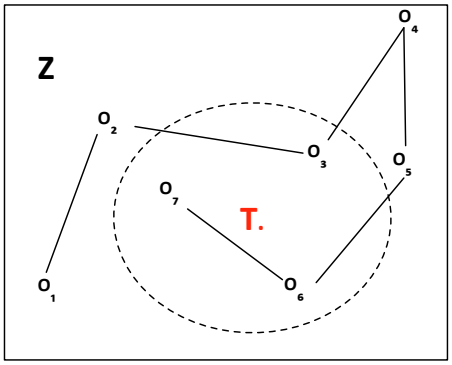


Figure 5 Calcul du poids de chaque chemin entre Oi et Oj

$$d: \quad Z \rightarrow IR^{+}$$
$$(O_i, O_j) \mapsto n_{ij}$$

$$\begin{cases} O_i \in Z \\ O_j \in Z \\ n_{ij} \in IR^{+} \end{cases}$$

$$Si: d(O_1,O_6) \le d(O_1,O_4) + d(O_4,O_5) + d(O_5,O_6)$$
$$\Rightarrow n_{1,6} \le n_{1,4} + n_{4,5} + n_{5,6}$$

La formule mathématique présentée ici montre que tous les objets appartiennent à l'espace Z. La fonction **d** est définie de Z dans R+. A un couple Oi, Oj, **d** associe un réel positif nij.

Nous utilisons deux phases pour évaluer une requête basée sur un ensemble de distances calculées préalablement.

- Tout d'abord, estimer les distances absentes et stocker les valeurs de ces distances, en incluant celles qui sont estimées et celles qui sont exactes dans un ensemble de distances approximatives (*ADM Approximate Distance Map*), si toutes les distances nécessaires ont préalablement été calculées, la phase TyQ1 est omise.
- Ensuite, on traite la requête basée sur l'ADM en filtrant le plus possible les objets qui ne peuvent probablement pas satisfaire la requête.

Méthode de calcul de l'ADM

L'utilisation d'un graphe pondéré est adaptée pour l'ensemble des distances approximatives (ADM *Approximate Distance Map*). Soit D une BD composée de n objets numérotés de 1 à n, et l'on appelle Oi l'objet numéro i. Pour calculer l'ADM, nous commençons par construire un graphe pondéré, non dirigé sur la base D, de manière à ce qu'un arc entre Oi et Oj soit évalué seulement si d(Oi, Oj) a été calculée. S'il existe un arc e le poids w(e) est la distance calculée. Nous définissons un chemin de Oi = Oi1, vers Oj = Oin, comme une séquence d'objets distincts Oi1, Oi2,..., Oin tels que {Oi1, Oi2}, {Oi2, Oi3}, ..., {Oin-1, Oin} sont des arcs du graphe, et le poids du chemin est la somme des poids de chaque arc constituant le chemin.

À propos de la méthode de calcul de ADM, en général, il est souhaitable d'obtenir une valeur, la plus grande possible pour cette borne. Soit P(i, j), l'ensemble contenant tous les chemins possibles de Oi à Oj. L'ADM[i,j] représente la borne maximale obtenue à l'aide de tous les chemins de P(i,j). Par inégalité triangulaire, ADM[i,j] = d(Oi, Oj), si les nœuds {Oi, Oj} ∈ P(i, j). En général, il est impossible d'énumérer tous les chemins de P(i,j), pour obtenir l'ADM[i,j] car leur nombre est exponentiel. En revanche, l'utilisation d'une technique de programme dynamique, similaire à l'algorithme de couverture transitive pour calculer l'ADM permet de réduire la tâche

de calcul. Pour en faciliter le calcul, on utilise une matrice supplémentaire MIN, où MIN[i,j] est le poids minimum de chaque chemin entre Oi et Oj. Il est clair que Min[i,j] = d(Oi,Oj) quand cette valeur est disponible.

$$D = \left(O_{ij}\right) \quad j = \{1,7\}$$
$$ADM = \left\{d\left(O_{i1},O_{i3}\right),d\left(O_{i1},O_{i4}\right),.....,d\left(O_{i2},O_{i3}\right),.....\right\}$$

Calcul

Pour rechercher les requêtes de similarités, on définit une cible T (Target) et une base de donnée D d'objets (Colloc J. 2011).

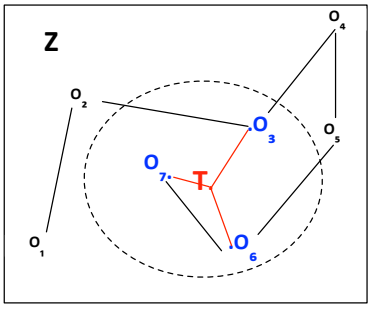


Figure 6 Trouver les objets de D les plus proches et le plus proche de la cible T

Les types de questions 1-3 recherchent les objets les plus proches et l'objet le plus proche de la cible T.

D = ensemble des objets Oij considérés dans l'espace Z

Z = espace de travail

ε désigne une valeur petite (par rapport aux distances inter-objets de l'ensemble D)

C(cm) est le cercle de rayon exprimé en cm.

d(T, Oid) exprime et égale la distance entre l'objet et la cible T.

Pour tous les objets appartenant à l'espace Z, on souhaite calculer la distance entre l'objet et la cible T, et trouver la somme minimale.

TyQ1 : trouver les k objets, tel que ces k objets de la base D soient **les plus proches** de la cible T.

$$C\left(c_m\right) \Rightarrow O_{i3},O_{i6},O_{i7}\left(k=3\right)$$

tel que $c_m \le \varepsilon \quad \left(d\left(TO_{i3}\right) \le \varepsilon, d\left(TO_{i6}\right) \le \varepsilon, d\left(TO_{i7}\right) \le \varepsilon\right)$

TyQ2 : Trouver l'objet de D **le plus proche** de la cible T (le plus similaire). NB : ce cas est le cas particulier de la requête 1 où k = 1.

TyQ3 : Trouver les objets de la base D qui sont suffisamment similaires à T c'est-à-dire qui sont à l'intérieur d'une distance choisie ε par rapport à l'objet cible T.

$$d\left(T,O_{id}\right)?$$

$$\text{on recherche } O_{id} \text{ tel que } \forall O_{ij} \in Z, d\left(T,O_{id}\right) = \min_j d\left(T,O_{ij}\right)$$

$$\text{par conséquent, } k = 1$$

Les types de questions 4-6 visent à chercher les objets les plus éloignés et l'objet le plus éloigné de la cible T.

ζ désigne une valeur grande (par rapport aux distances inter-objets de l'ensemble D)

Pour tous les objets appartenant à l'espace Z, on souhaite calculer la distance entre l'objet et la cible T, et trouver la somme maximale.

TyQ4 : Trouver les k objets de la base D qui sont **les plus éloignés** de la cible T.

$$O_{i1}, O_{i4}\left(k=2\right) \quad \text{i.e., il existe un cercle } C\left(c_M\right)\left(c_M \geq \zeta\right)$$

$$\text{tel que } d\left(T,O_{i1}\right) \geq \zeta, d\left(T,O_{i4}\right) \geq \zeta$$

TyQ5 : Trouver l'objet de la base D **le plus éloigné** de la cible T (c'est-à-dire le plus dissemblable).

$$d\left(T,O_{il}\right)?$$

$$\text{On recherche } O_{il}\text{, tel que } \forall O_{ij} \in Z, d\left(T,O_{il}\right) = \max_j d\left(T,O_{ij}\right)$$

$$\text{par conséquent, } k = 1$$

TyQ6 : Trouver les objets de D qui sont suffisamment dissemblables de l'objet cible T, c'est-à-dire situés au-delà d'une distance choisie ε, par rapport à l'objet cible T.

Les objets O_{il} suffisamment dissemblables à T vérifient $d\left(T,O_{il}\right) > \zeta$

Pour répondre à ces requêtes, un système de requête pourrait calculer les distances entre chaque objet de la base et la cible puis rechercher les objets désirés. Mais le problème majeur de cette approche est le coût de calcul, notamment lorsqu'il existe un

grand nombre de cibles à identifier, le calcul des distances est coûteux. Une solution est de pré-calculer des distances afin d'indexer les objets.

Pour présenter le lemme d'Inégalité triangulaire généralisée, nous supposons qu'il existe un chemin P de Oi à Oj, si ê est l'arc de poids maximum du chemin P allant de Oi à Oj. Alors :

- $d(O_i, O_j) \geq w(\hat{e}) - \sum_{e \in P - \{\hat{e}\}} w(e)$
- preuve : par induction sur le nombre d'objets dans P et l'application répétée de l'inégalité triangulaire.

Le lemme établit qu'il est possible d'obtenir une borne inférieure pour d(Oi,Oj) en appliquant l'inégalité triangulaire à un chemin entre Oi à Oj. Bien sûr, une telle borne est inutile si le terme de droite de l'inégalité est négatif ou nul.

1.4 Méthodes système multi-agents

1.4.1 Agent UML

L'Agent UML (Agent UML 2000) est une notation qui soutient le développement des systèmes axés sur les agents. Elle consiste à utiliser et étendre le langage de modélisation UML, afin de représenter les agents, leur comportement et les interactions entre eux. AUML ne se limite pas à l'utilisation d'UML. D'autres approches acceptables sont permises.

La notion d'agent est adaptable à de nombreux systèmes matériels et logiciels, mais il est particulièrement utile, dans des contextes assez complexes, de faire de simples notions d'objet pour décrire le système. Les agents partagent certaines caractéristiques communes (Wooldridge, M. & Jennings, N.R. 1995) :

- Identifier: identifie chaque agent dans un système multi-agents ;
- Rôle : définit le comportement d'un agent de la société ;
- Organisation : définit les relations entre les rôles ;
- Capacité : spécifie ce qu'un agent est capable de faire, dans quelles conditions ;
- Service : décrit les activités fournies de l'autre agent.

Aujourd'hui, OMG (*Special Interest Group*) et FIPA (*Modelling Technical Committee*) sont intéressés à AUML qui recommande des normes pour les

technologies de l'agent. D'autres méthodes similaires comprennent MESSAGE, Gaia, Tropos, Prometheus, MaSE, etc.

1.4.2 MaSE

Multi-agent Systems Engineering (MaSE) (DeLoach, S. A. & Hartrum, T. C. 1999) est une méthodologie et une langue, pour la conception de systèmes d'agents allant d'un système d'intégration de base de données hétérogène au système de virus-immune ordinateur ; MaSE est fondé sur la base biologie. Il utilise l'abstraction fournie par les systèmes multi-agents pour le développement de systèmes logiciels distribués intelligents.

La méthodologie MaSE est similaire à des méthodes de génie logiciel traditionnel, mais elle s'est spécialisée pour une utilisation dans le paradigme de l'agent distribué. MaSE utilise deux langues pour décrire les agents et systèmes multi-agents :

- L'*Agent Modeling Language* (AgML) est un langage basé sur le graphique qui décrit les types d'agents dans un système et leurs interfaces avec d'autres agents.
- L'*Agent Definition Language* (AgDL) est basé sur la logique et il est utilisé pour décrire complètement, le comportement interne de chaque agent individuel.

Même si les diagrammes MaSE peuvent ressembler à des diagrammes OMT ou UML, il dispose de fonctionnalités supplémentaires et modifie la sémantique orientée objet traditionnelles, pour capturer des notions et des comportements coopératifs. En outre, comparé avec l'utilisation des langages de définition orientés objet, spécifiée informelle et des méthodes telles que OMT ou UML, les langues AGML et AGDL peuvent fournir la synthèse de système plus directe. (DeLoach, S.A. & Madhukar K. 2005)

1.4.3 AALAADIN

Le modèle Aalaadin (Ferber, J. & Gutknecht, A. 1998) (vers une méthodologie organisationnelle de conception de SMA) présente un méta-modèle générique de systèmes multi-agents basés sur des concepts organisationnels tels que les groupes et les rôles. Aalaadin permet une description simple de schèmes de coordination et de négociation, à travers des systèmes multi-agents.

Le modèle Aalaadin comprend certaines étapes incluent la définition des architectures d'agents, la description d'organisations, l'identification de groupes et de rôles, l'utilisation de méthodologies (analyse, conception, réalisation...) et la spécification des structures organisationnelles etc.

Le modèle d'Aalaadin décompose l'analyse des structures collectives en deux niveaux :

- Le niveau descriptif correspond aux concepts centraux d'agent, de groupe et de rôle. C'est à ce niveau que se décrit une organisation réelle.
- Le niveau méthodologique définit l'ensemble des rôles possibles, spécifie les interactions et décrit les structures abstraites de groupe et d'organisation.

La conception et la réalisation de systèmes complexes rend nécessaire le développement de modèles permettant une réelle hétérogénéité tant au niveau des architectures d'agents que des mécanismes d'interaction et des applications supportées. Le méta-modèle Aalaadin permet effectivement de prendre en compte cette hétérogénéité à partir d'un point de vue organisationnel des SMA, fondé simplement sur les trois concepts fondamentaux d'agent, de groupe et de rôle. (Ferber, J. & Gutknecht, A. 1998)

1.4.4 Spécification des exigences et Description de la conception

Selon la littérature de génie logiciel (Sommerville, I. 2004), une spécification d'exigences de logiciel est une des étapes antérieures du procédé de développement de logiciels. Il vise à décrire les besoins des utilisateurs, la configuration système fonctionnels et non fonctionnels. Les spécifications des exigences permettent de faciliter le processus de développement, de même que le transfert des connaissances aux nouveaux utilisateurs (IEEE Standard 830-1998). Les exigences de logiciels utilisés dans cette thèse sont les suivants :

- Langage naturel : les exigences logicielles sont décrites en utilisant des phrases de la langue naturelle, tableaux et diagrammes.
- Langage naturel structuré : les exigences logicielles sont définies à l'aide de modèles définis comme des descriptions de cas d'utilisation en UML (*Unified Modelling Language*).
- Langage description de la conception : les exigences logicielles sont définies en utilisant un langage pseudo-programme comme le langage Java.

- Spécifications mathématiques : calculer la fréquence des occurrences mots

Les exigences du système prototype ont été décrites en utilisant la spécification du langage naturel, selon la norme IEEE 830-1998 (IEEE Standard 830-1998). La description complète du prototype de fonctionnement est montrée en annexe.2.

Chapitre 2 LINGUISTIQUE

Les linguistes sont concernés par la question des ontologies dans la mesure où les données dont on dispose pour élaborer les ontologies consistent en des expressions linguistiques de connaissances. L'ontologie régionale (non universelle) que l'on obtient est une spécification de signifiés normée.

Ce chapitre insiste d'abord sur l'étude linguistique de la construction du sens, les moyens de représentation des connaissances et l'approche générale d'acquisition de connaissances textuelles. Ces fondements théoriques vont nous aider à résoudre les problèmes non triviaux, lorsque notre module linguistique doit donner une représentation matérielle de certains faits de langue, en particulier quand il s'agit de construire un dispositif informatique qui résolve ou illustre les problèmes liés au traitement automatique de ces faits de langue.

2.1 Étude linguistique

L'association des savoirs aux mots, c'est-à-dire l'étude des régularités et des redondances, traite globalement les problèmes de langue.

Les modèles et les ontologies sont construits à partir des connaissances des experts du domaine considéré. Le cogniticien interroge l'expert et construit une structure composée, généralement, de deux types d'objets : des concepts et des relations (Sowa, J. F. 1983) (Sowa, J. F. 2000). Notre approche vise à faire émerger et représenter les comportements des mots, puis à affiner les représentations existantes, afin que mettre en évidence les corrélations multiples entre les mots, dans un flot continu de discours.

2.1.1 De la construction du sens

Selon la théorie de (Adam, J. M. 2011), le texte descriptif est analysé selon les critères suivants :

- Un texte est transformé dans une formule contenant une ou plusieurs interprétations ;
- Une formule est décrite par des termes (ou concepts) qui rendent compte du réseau sémantique du thésaurus. Les attributs précisent la vocation de termes dans la formule (et donc dans les textes descriptifs) ;
- Des liens de domaine associent les termes du thésaurus ; ils définissent la proximité et la généricité-spécificité ;

- Les termes du thésaurus sont associés à des vocables du dictionnaire. Les mots ont des propriétés telles que leurs emplois, leur champ sémantique (domaine), les liens de synonymie, etc.

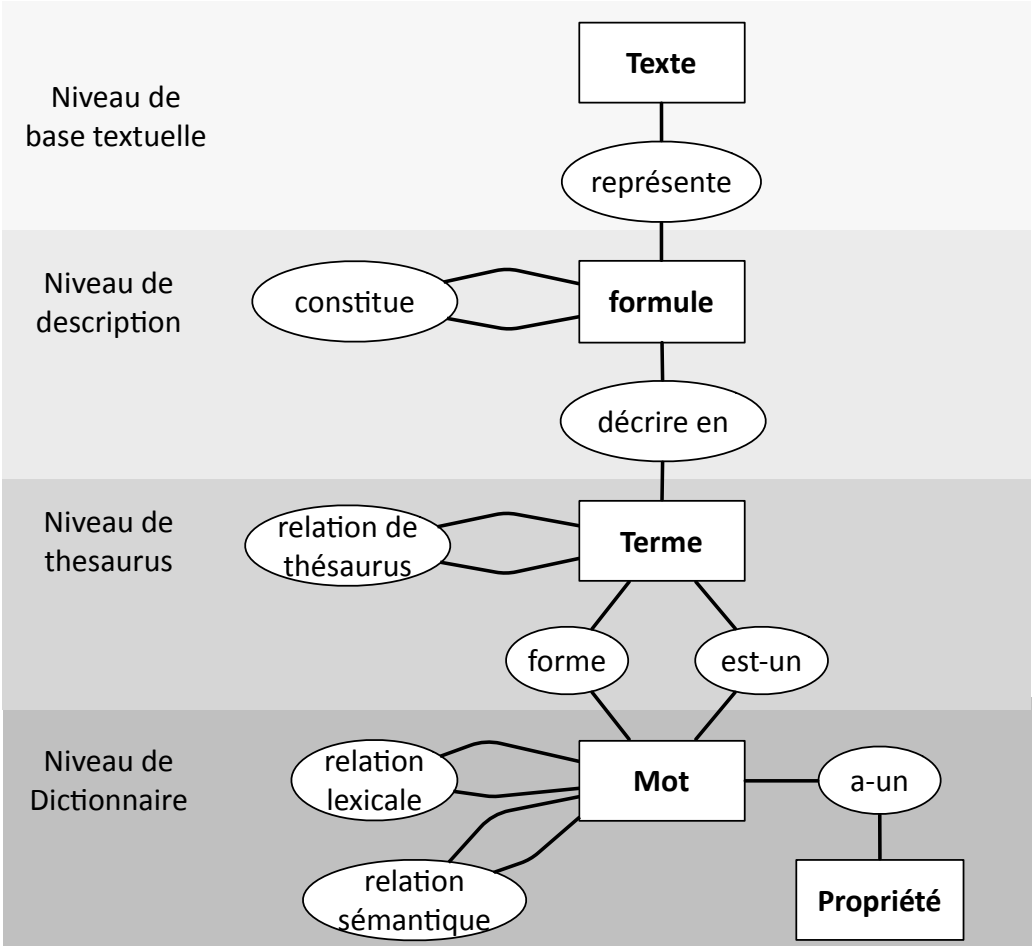


Figure 7 «*The breaking up of a text*» (Adam, J. M. 2011)

(Greengrass, E. 2001) distingue certains niveaux dans le traitement du langage naturel :

- Niveau phonologique : étudie la reconnaissance de la parole qui concerne l'analyse des sons. Cette recherche se situe hors de ce cadre de travail.
- Niveau morphologique : reconnaît les formes et variantes d'une occurrence-mot donnée. Il est utilisé pour générer des systèmes de lemmatisation. Il s'agit du premier processus de traitement du langage naturel avant le marquage des mots (*tagging*).
- Niveau lexical : reconnaît la structure et les significations du mot. Il sert à construire des listes « *stop words* », des thésaurus, et à détecter et marquer les mots avec leur type tels que nom propre, verbe, etc.

- Niveau syntaxique : analyse la structure des phrases. Il peut être utilisé, pour cartographier les formes passives et les formes actives, en participant à la normalisation de la forme de la phrase avant son interprétation sémantique.
- Niveau sémantique : interprète la signification des résultats de niveau inférieur. Il est utilisé pour désambiguïser et construire les structures conceptuelles. Pour faire correspondre du contexte local des occurrences de termes à un thésaurus, avec les sens et contextes différents, les instances peuvent fournir de l'aide à la désambiguïsation.
- Niveau discours : interprète la structure du document pour déterminer la structure des clauses, des phrases et des paragraphes qui déterminent le flux rhétorique d'un document et sa signification. Il peut être utilisé pour le document de récapitulation ou pour isoler la connaissance extraite.
- Niveau pragmatique : introduit les connaissances externes tels que le profil de l'utilisateur, le contexte, la connaissance commune, la connaissance du domaine, afin d'améliorer l'interprétation.

En outre, certaines approches peuvent être également utilisées pour analyser le texte, structurer les documents et explorer les connaissances du domaine. Les exemples d'approches principales sont les suivantes (Gandon, F. 2002) :

- Les citations bibliographiques : co-citations (cooccurrences de documents dans les citations d'un groupe de documents sur un sujet), le couplage bibliographique (deux documents citant le même troisième document traitant d'un sujet donné.)
- Lien hypertexte : les ancrages du lien permettent de localiser un ensemble de documents connexes ; de même, les étiquettes des liens nous permettent d'améliorer la description des sujets de ces documents.
- Indices structurels : ils servent à déclencher des termes tels que « légende », « conclusion », « exemple » pour localiser les informations importantes dans le texte, la source du document, etc.

2.1.2 Moyens de représentation des connaissances

Selon le définition de la connaissance de (Minsky, M. 1974) :

> La connaissance est constituée de données brutes perçues et mémorisées mais aussi des raisonnements mis en œuvre, pour établir des relations entre elles, construire des schémas qui seront comparés à d'autres, analogues, déjà rencontrés.

Il est donc obligatoire de disposer d'un système capable de représenter et mémoriser ces schémas. Ce sous-chapitre est consacré aux différents moyens de représenter des connaissances.

Terminologie

> Les terminologies sont des listes de termes d'un domaine ou sujet donné représentant les concepts ou notions les plus fréquemment utilisés ou les plus caractéristiques, cette liste étant ou non-structurée. (Lefèvre, P. 2000)

La terminologie est l'ensemble des termes, rigoureusement définis, spécifiques d'une science, d'une technique, d'un domaine particulier de l'activité humaine. C'est une discipline dont l'objet est l'étude théorique de la dénomination des objets ou des concepts utilisés dans tel ou tel domaine du savoir. La terminologie concerne également fonctionnement des unités spécifiques d'une langue de spécialité, les problèmes de traduction, de classement et de documentation. (Larousse 2009)

Classification

> Une classification est l'action de distribuer les mots par classes, ou par catégories. Plus précisément, une classification est la répartition systématique en classes, en catégories d'êtres, de choses, d'objets ou de notions ayant des caractères communs notamment afin d'en faciliter l'étude (Bourigault, D. et al. 2004)

La classification est de regrouper les classes ou les catégories de sous-classes ayant des significations proches, de façon à établir une hiérarchie. La classification internationale des maladies (CIM) et la Classification commune des actes médicaux (CCAM) sont des exemples de classifications hiérarchiques du domaine médical.

Nomenclature

> Une nomenclature est un ensemble de catégories, chacune étant associée à une définition explicite des éléments qu'elle contient, et de liens entre ces catégories du type « est une sorte de ». (DSM-III 1980)

En fait, la nomenclature est une sorte de classification méthodique. C'est une liste, un catalogue détaillé, où figurent l'ensemble des entrées choisies dans un dictionnaire ou dans un lexique. SNOMED (*Systematized Nomenclature of Medicine*) est un exemple de nomenclature du domaine médical.

Thésaurus

> Un thésaurus est un ensemble structuré de termes d'un vocabulaire, par exemple, les termes techniques utilisés en médecine, représentés de façon normalisée par des descripteurs ou des mots-clés (Foskett, D.J. 1997).

Le thesaurus est utilisé pour indexer les documents dans le domaine de la recherche d'information. Il s'agit d'un documentaire basé sur une structure hiérarchique de termes désignant des concepts. Il se présente sous la forme d'une liste alphabétique des termes organisés conceptuellement et reliés entre eux par des relations sémantiques : relations hiérarchiques, associatives et d'équivalence. Le thésaurus propose une définition des termes. Dans le domaine médical, l'un des thésaurus les plus connu est MeSH (*Medical Subject Heading*).

Taxinomie

La taxinomie (ou taxonomie) complète la systématique : science de l'organisation du classement des taxons et de leurs relations. La taxinomie est la science des lois de classification, aussi est-ce une forme de classification spécifique, suite d'éléments formant des listes qui concernent un domaine ou une science. La taxonomie propose une classification sous la forme d'un arbre (ou arborescence) dont les branches partagent une racine commune et dont chaque nœud est appelé « taxon ». Plus le rang du taxon est élevé, plus le degré de ressemblance entre les individus concernés est minimum, et inversement.

Ontologie

« *Une ontologie est la spécification d'une conceptualisation*. » (Gruber, T. R. 1993) Une ontologie est une description des concepts et des relations qui concernent un objet ou un ensemble d'objets. Elle permet la capitalisation, l'organisation, la représentation et le partage de connaissances d'un domaine particulier. L'objectif principal d'une ontologie est de partager et de réutiliser des connaissances propres à un domaine. Cette notion est développée en détail dans la section suivante.

2.2 Banque de données

Il existe différents types de bases de données, qui sont mis à jour pour ajouter et modifier des informations. Elles disposent de fonctions de recherche multicritères qui touchent tous les domaines de la science.

2.2.1 Banque de faits théoriques ou expérimentaux

La toxicologie, la pharmacologie, la médecine interne, la biologie, plus particulièrement la biologie moléculaire et la génétique, sont le sujet de nombreuses banques de données.

Les banques d'informations existantes incluent MESH, CADUCEUS-MYENO, SNOMED, CASNET-Réseau sémantique, etc. Elles permettent de consulter les propriétés d'une maladie ou de rechercher les médicaments qui la soignent : nom du malade, symptôme, pathologie, médicament, thérapie, posologie, etc.

2.2.2 Dossiers médicaux

Certaines banques de données médicales sont enrichies de dossiers standardisés de patients, car certains dossiers médicaux sont représentatifs du problème étudié, avec les données valides.

Par exemple, les dossiers du patient peuvent aider aux décisions cliniques sur l'épidémiologie d'une pathologie (cancer de l'estomac etc.), sur le suivi de patients atteints d'une maladie chronique (bronchite chronique, etc.), ou sur la pratique d'une procédure thérapeutique (antibiothérapie, chimiothérapie etc.).

2.2.3 Banque de connaissance

Les systèmes précédents d'aide à la décision clinique restent au niveau des faits. Mais les faits restent insuffisants pour établir une base de connaissances.

L'ajout des faits aux connaissances implique de modéliser le raisonnement et de fournir des bases de données et des capacités de raisonnement, propres à créer des connaissances.

Dans ce contexte, les systèmes d'aide à la décision décrit les maladies et les liens entre maladies, de même qu'entre entre les signes qui permettent la description d'une pathologie et l'aide à la décision diagnostique.

2.3 Module linguistique SMAAD

Le module linguistique SMAAD est conçu pour faciliter l'accès à la base de données du système multi-agents pour les utilisateurs non-avertis, et pour fournir une aide à la construction de l'ontologie en utilisant le langage et pas seulement des choix logiques exogènes. Le module linguistique inclut :

- **Une base de connaissances**

  Le « **Dictionnaire ontologique** » (DO) contient tous les objets et concepts concernés : les dictionnaires ou bibliographies existants (MESH, MEDLINE, SNOMED, CASNET-Réseau sémantique, CADUCEUS-MYENO etc.), les banque de données (banque de faits théoriques ou expérimentaux, dossiers médicaux, banque de connaissance etc.), et les ontologies construites avec les concepts, les associations et les instances dans les hiérarchies correspondantes.

- **Une base d'index**

  Le module « **Connaissances sur ses propres capacités** » contient les connaissances existantes de notre système. Ses fonctions comprennent le dictionnaire qui permet d'enrichir le thésaurus et d'indexer de nouvelles ontologies. Ce module peut analyser la construction d'ontologies pour corriger les index, les grammaires et même des textes descriptifs. En cas d'index excessif (*over-index*), ce module modifie ou corrige les index des ontologies déjà chargées.

- **Une plate-forme d'exploitation**

  Le « Module interpréteur » (MI), le contrôleur de l'algorithme contient tous nos algorithmes et programmes pour l'analyse et l'extraction des termes et des

associations. Il peut fournir des programmes et des algorithmes adaptés au système selon l'ordre de l'utilisateur ou le réglage par défaut défini par l'ingénieur de connaissances. Ce module est aussi responsable du passage d'une requête au format OWL, pour des recherches ultérieures.

Par exemple, la base de connaissances constitue le réseau lexical, syntaxe et sémantique du domaine concerné. Un analyseur morphologique reconnaît les formes de variantes d'un mot donné. Il simplifie les mots français pour générer des systèmes de lemmatisation ; c'est le prétraitement automatique (TAL) nécessaire à l'extraction de termes et d'associations entre la requête et le dictionnaire existant. Les phénomènes de synonymie, la généralité et la spécificité sont pris en compte. Grammaires de requête et d'indexation, ainsi que des algorithmes de recherche peuvent également être gérés, mais ils restent de la compétence du linguiste.

Une conséquence du module linguistique est de renforcer l'échange d'informations entre le système multi-agents, l'utilisateur final, les ontologistes et les ingénieurs de la connaissance. L'utilisateur final entre une demande ou requête, puis il attend la réponse du système. Les agents et modules du système coopèrent pour effectuer la recherche du résultat. Les ontologistes et les ingénieurs de la connaissance prêtent attention au fonctionnement du système. La réponse du système reflète la qualité de la base de connaissances, la base de l'index et la plate-forme d'exploitation.

Chapitre 3 ONTOLOGIES

Ce chapitre est centré sur l'ingénierie des ontologies. C'est la discipline de l'informatique qui concerne tout ce qui a trait à l'ontologie : méthodes et méthodologies, techniques, langages, outils de conception et d'implémentation des ontologies (Gómez-Perez, A. et al. 2003). C'est principalement ce que nous allons développer dans ce chapitre.

Nous insistons d'abord sur l'étude de la méta-connaissance qui manipule des entités abstraites générales (objets, classes, schémas, paramètre, etc.) indépendamment du contexte. La mise en place d'acquisitions et la restitution de connaissance, la mise à disposition et le partage de nouveaux savoirs requièrent d'affiner la méta-connaissance des concepts mis en œuvre, lors de la description des objets.

Ce chapitre présente le nécessaire génération semi-automatique d'ontologies, à partir de textes médicaux, puis les différentes approches et outils mis en place pour réaliser la génération automatique d'ontologies.

Ce travail repose sur une approche multi-agents de la décision clinique. Les ontologies cliniques sont développées grâce à un système multi-agents destiné à aider à la décision clinique. Le raisonnement à partir de cas (C) mémorise et restitue l'expérience de résolution des problèmes similaires. Un système multi-agents d'aide à la décision (SMAAD) (Colloc J. & Sybord C. 2003) permet l'intégration et la coopération des agents spécialisés dans différents domaines de connaissances (sémiologie, pharmacologie, cas cliniques, etc.). La coopération de bases de connaissances cliniques hétérogènes utilise les ontologies médicales. Un modèle médical est une représentation explicite du diagnostic, du pronostic, de la thérapie, du suivi thérapeutique, du comportement, des objectifs et des contraintes de processus médicaux. L'ontologie correspondante comprend les principales caractéristiques des entités modélisées et les formes de relations existant entre eux dans un vocabulaire, sans ambiguïté consensuelle.

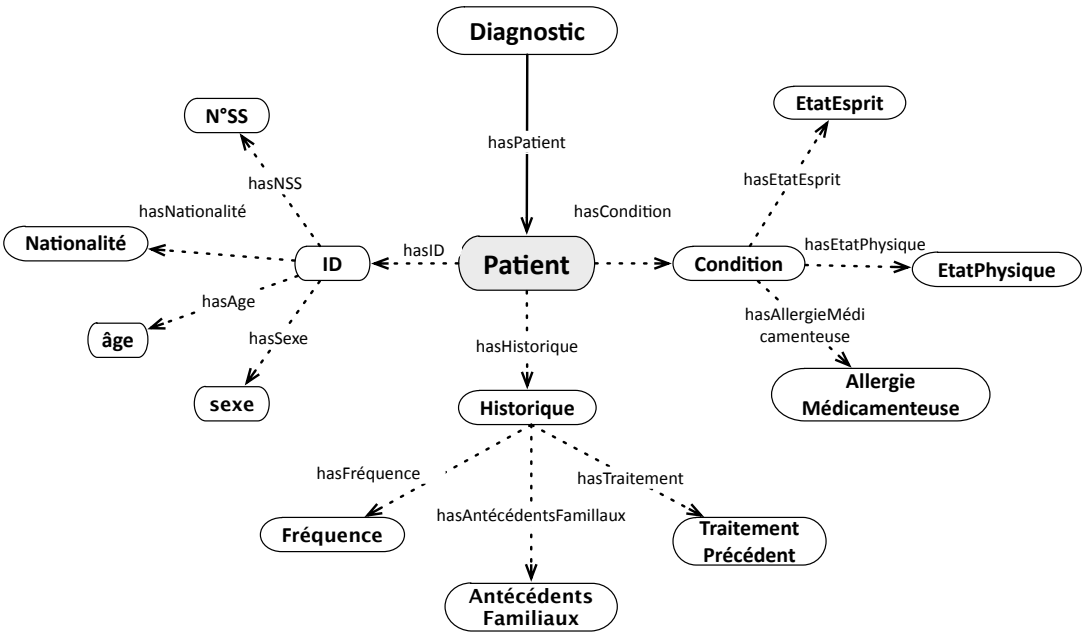


Figure 8 Echantillon de l'ontologie

3.1 Définitions du concept d'ontologie

Ontologie :

étymologie : ontos (l'existant) + logos (l'étude)

L'Ingénierie ontologique (IO) est la branche de l'Ingénierie des Connaissances qui exploite les principes de l'Ontologie (formelle) pour construire des ontologies (Guarino, N. et al. 1994).

Initialement, l'ontologie a été utilisée dans la philosophie d'enquête, pour la conception de la réalité, en utilisant les entités et les relations pour décrire les catégories de l'être dans la métaphysique. Aujourd'hui, les ontologies sont utilisées en informatique pour décrire la connaissance d'un domaine spécifique.

En 1993, Gruber écrivait « [qu']*une ontologie est une spécification explicite d'une conceptualisation* ». Cela signifie que l'ontologie est une spécification compréhensible par une machine dont les concepts sont explicitement définis et partagés dans une communauté donnée. Les ontologies fournissent un vocabulaire commun d'une région et, avec différents niveaux de granularité et non formalité, elles définissent la signification de termes et leurs relations. Elles sont généralement organisées en taxonomies avec des primitives de modélisation : classes, relations, fonctions, axiomes et instances. (Gruber, T. R. 1993)

Dans le domaine de l'intelligence artificielle, Neches et al. (Neches, R. et al. 1991) précisent :

> Une ontologie définit les termes de base et les relations qui composent le vocabulaire d'un domaine thématique, ainsi que les règles de combinaison des termes et les relations entre les définitions et les extensions du vocabulaire.

(Swartout, B. et al. 1996) définissent l'ontologie comme

> […] un ensemble structuré hiérarchiquement de concepts décrivant une connaissance d'un domaine spécifique qui peut être utilisé pour créer une base de connaissances. L'ontologie contient les concepts, une hiérarchie de subsomption et les relations arbitraires entre les concepts et les axiomes. Cet ensemble peut également contenir d'autres contraintes et fonctions.

Les ontologies sont à la sémantique, ce que la terre est à l'électronique : une base commune sur laquelle s'appuyer et une référence partagée à aligner. Les ontologies visent à capturer la connaissance d'un domaine d'une manière générique et à fournir une compréhension communément admise dans le domaine. Elles peuvent être réutilisés et partagées entre les applications et les groupes (Chandrasekaran, B. et al. 1999).

Les ontologies sont considérées comme un outil puissant pour lever des ambiguïtés, c'est l'un de leurs principaux rôles. L'ontologie est un domaine sémantique, un vocabulaire conceptuel consensuel, sur lequel on peut construire des descriptions et des actes de communication (Bachimont, B. 2000). Cela explique que, d'une part, les ontologies fournissent des ressources théoriques pour formuler et construire des connaissances explicites et, d'autre part, elles constituent un cadre commun que les différents acteurs peuvent mobiliser. L'ontologie peut représenter le sens des différents contenus échangés dans les systèmes d'information.

Les modèles d'ontologie proposés ici sont destinés à l'axiomatisation d'un domaine de connaissances. Ils sont fondés sur des graphes qui peuvent être implantés à l'aide de plusieurs formalismes comme la logique formelle, les modèles à règles de production, des langages sémantiques comme XML et ses dérivés OWL, des langages fonctionnels comme Objective Caml.

Comparée avec un ***thésaurus***, le contenu de l'ontologie est composé de taxinomies de concepts et de relations ; elle est décrite dans un langage de représentation des connaissances et exploitée par un système informatique. En revanche, les thésaurus sont composés de descripteurs et de mots-clés qu'un agent humain utilise pour indexer des documents ou comme vocabulaire contrôlé. Donc, dans les thésaurus, la sémantique des relations, surtout d'héritages, n'est pas parfaitement fixée. Dans un

domaine où il est nécessaire de faire encore plus d'inférences, les ontologies sont encore plus indispensables. (Charlet, J. 2003)

Comparer avec le ***corpus*** *de textes*, le contenu d'ontologie est plus organisé que l'ontologie qui est établie à partir de mots extraits du corpus, dans la hiérarchie.

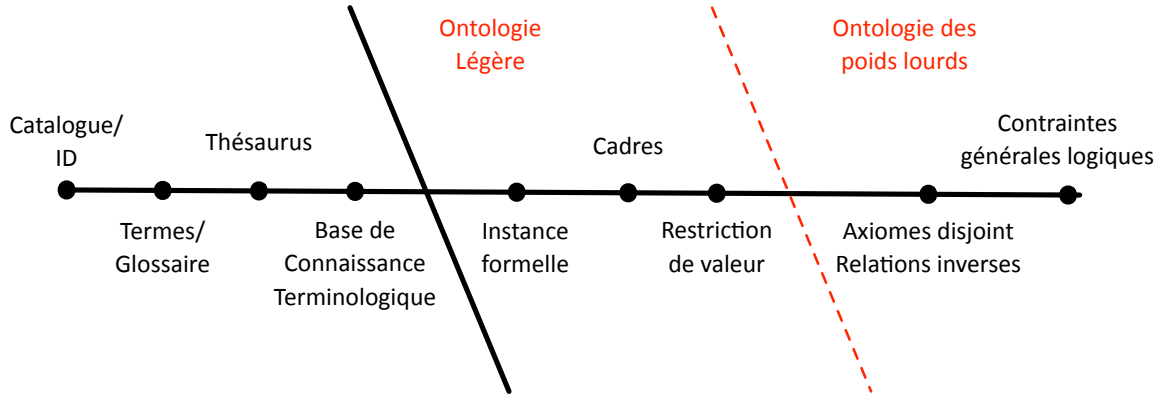


Figure 9 La relation entre l'ontologie, le corpus et les thésaurus

Composantes de l'ontologie

L'ontologie permet de capitaliser des connaissances d'un domaine particulier. La définition de ces connaissances se base sur les composants de l'ontologie : concepts, relations, propriétés, et instances.

	Composantes de la terminologie	Composantes de l'ontologie
Classe	**Terme** : Un mot substantif ou composé utilisé dans un contexte spécifique pour exprimer le sens (s).	**« Heading » de Classe** : Un « heading » balaie le contenu d'une entité et cherche à le nommer
	Type de terme : Un terme type est la représentation commune que l'on adopte pour des termes porteurs des mêmes caractéristiques	**« La note explicative de classe »** : Les notes explicatives (structurées, officielles ou non) précisent le contenu et définissent en partie les frontières d'inclusion et d'exclusion
Propriété	**Association** : lien entre plusieurs entités	**Propriétés** : caractéristiques associées a une entité type
	Type d'association : représentation d'un ensemble de relations qui possèdent les mêmes caractéristiques, lien entre plusieurs entités types.	**Valeur** : Valeur que prend une propriété (à l'intérieur du domaine) pour une entité particulière. Elle permet de décrire un ensemble de données relatives à un domaine défini, afin de les intégrer dans une Base de données.
	Donnée d'association **annotation d'association**	
Hiérarchie	**Niveaux** : un ou plusieurs	**Hiérarchie :** Dans une hiérarchie, l'opérateur est en mesure d'accéder aux relations élémentaires qui composent la hiérarchie alors, soit les éléments de la source (du niveau de la source) ou les éléments cibles (du niveau de la cible). Ainsi, à partir du domaine, niveau et domaine diffèrent dans les hiérarchies liées, un opérateur peut être consulté dans une entité de base pour répondre à la requête.
	Domaine : la zone de code, peut-être déconnectée	
	Variables : L'histoire de l'objet, des notes explicatives, les relations historiques et d'autres relations.	

Instances	**Instances** : ou encore appelées individus. Elles représentent les éléments qui peuplent l'ontologie.	Exemple : Classe = Virus Instance = les salmonelloses, les shigelloses, les campylobactérioses, le rotavirus, le clostridium, la entérotoxine A, la cytotoxine B etc.

Tableau 10 Composantes de l'ontologie et de la terminologie

Les composantes de l'ontologie peuvent être résumée en formats de formules mathématiques :

Structure d'une ontologie :
O = {C, R, H, I}

C : les concepts $C = \{C_1, C_2, C_3 \ldots C_n\}$

R : les relations $R \rightarrow C \times C\ (rel)$

rel définit des relations sémantiques non taxonomiques avec 2 fonctions associées :

- domaine : $R \rightarrow C$ avec $dom(R) = \Pi_1(rel\ (R))$
- range : $R \rightarrow C$ avec $range(R) = \Pi_2(rel\ (R))$ co-domaine

H : hiérarchie de concepts $H \subseteq C \times C$

I : un ensemble d'instances

- inst : $C \rightarrow I$ fonction d'instanciation de concept
- instr : $R \rightarrow I \times I$ fonction d'instanciation de relation

3.1.1 Type d'objet

Nous devons identifier et classer les modèles émergents dans la construction du sens de configurations nominales. Il s'agit d'étudier les problèmes de représentations nécessaires à la résolution de l'encodage ou du décodage de certains faits de langue. En outre, il est nécessaire d'analyser les mécanismes qui effectuent cette construction du sens, en confrontant les connaissances des domaines.

La mise en place d'acquisitions, la restitution de connaissances, la mise à disposition et le partage de nouveaux savoirs nécessitent d'affiner la méta-connaissance des concepts mis en œuvre, lors de la description des objets.

La méta-connaissance, c'est toute connaissance sur la connaissance. Elle décrit la nature et l'utilisation des connaissances disponibles dans le système, et elle fournit des définitions génériques de la connaissance indépendamment de tout domaine

d’application. En outre, elle manipule le plus souvent des entités abstraites générales (objets, classes, schémas, paramètres, etc.) indépendamment du contexte.

Le niveau méta consiste à représenter une représentation, c’est-à-dire à définir plus spécifiquement les concepts qui nous servent à décrire un modèle de représentation. Si l’on prend comme exemple le modèle sémantique qui consiste à relier des objets (les nœuds d’un graphe) par des relations (les arcs du graphe), il convient alors de définir ce que l’on entend par objet, par relation et, sans aucun doute, ce que l’on entend par classe ou type d’objet. Cette méta-connaissance, qui décrit les concepts, est réflexive car elle utilise le même langage que pour décrire les objets.

À partir du texte, on peut déterminer objets et concepts. Les objets du texte sont les instances des types d'objets. La classe Objet constitue la racine d’une hiérarchie de spécialisation. Les propriétés de l’objet sont transmises par héritage à toutes les classes de la hiérarchie, connectées par des arcs de spécialisation.

Dans de nombreuses applications informatiques basées sur les ontologies, pour les applications de traitement de l'information ou pour la gestion des connaissances, il est nécessaire de prendre en compte l'utilisation de la terminologie pour l'acquisition des connaissances de domaine. Les terminologies sont essentielles dans la communication spécialisée précise et efficace. Les terminologues jouent un rôle important dans toute activité professionnelle où les compétences discursives, le transfert et l'organisation des connaissances sont nécessaires. Ils décrivent les concepts pour ancrer les dénominations, dans un contexte donné, et construire un système théorique avec son vocabulaire, ses associations, ses hiérarchies, la définition et l'analyse de la terminologie.

L'analyse de texte est un point clé des recherches en terminologie. Il est donc important de savoir quelles sont les informations utiles sur le plan conceptuel. L'extraction des termes et des associations pose le problème des méthodes et des outils pour définir le système conceptuel, pour la définition et la gestion de la terminologie. Dans ce contexte, les ontologies, tel que définies par l'ingénierie des connaissances, offrent de nouvelles opportunités et de nouvelles orientations aux terminologues.

Si les corpus de spécialités constituent une source privilégiée d'information, il reste à définir des méthodes d'analyse qui tiennent compte des différences existantes entre les textes et la conceptualisation formelle. La terminologie permet d'y répondre, en distinguant les dimensions linguistique et conceptuelle. Elle établit des principes

théoriques et méthodologiques utiles pour la construction d'ontologies. (Roche, C. 2005)

Dans une optique d'acquisition de connaissances, à partir de termes, il s'agit de repérer les ***dénominations***, les ***instances*** et les ***relations***.

3.1.1.1 Dénominations

Les dénominations soutiennent une relation reconnue et préconstruite avec l'objet, par exemple le signe « infection » désigne l'objet infection de manière établie et arbitraire. Mais l'infection ne présente rien sur l'objet, elle le désigne simplement.

Dans cette recherche, nous considérons les dénominations comme des unités ontologiques. Globalement, on peut réunir différentes propriétés dans un signe dénotatif, sans avoir à prendre en considération les constituants des objets dénommés.

L'objet peut être analysé en constituants, pas sa dénomination. L'existence d'une dénomination montre que la communauté linguistique a trouvé utile d'associer tel objet de son expérience à un signe qu'elle a pérennisé en une dénomination. D'ailleurs, lorsque l'utilité sociale d'un objet cesse, le signe qui le nomme disparaît, sans que cela affecte nécessairement les signes nommant ses constituants (Frath, P. 2005).

Nous identifierons des entités nommées plus générales comme les entités (virus, outil, maladie, médicament, etc.), les personnes, les organisations, les emplacements et certains domaines exclusifs tels que : diagnostic, pronostic, traitement, suivi-thérapeutique etc. Nous mettrons l'accent sur la constitution de ces listes pour chaque type de terminologie.

Les noms de maladies, l'acte et l'anatomie sont fondés sur un domaine riche en terminologies : CIM-10, le MeSH et SNOMED (Côté R.A. 1979). À propos des noms de médicaments, nous utilisons les données du site médical Doctissimo. Pour mieux caractériser les symptômes et les maladies, on utilise le MeSH (*Medical Subject Headings*), thésaurus biomédical de référence. C'est est un outil d'indexation, de catalogage et d'interrogation des bases de données de la NLM (*National Library of Medicine*), notamment MEDLINE/PubMed.

3.1.1.2 Instances et Relations

Les instances sont des dénominations au sein d'un syntagme nominal. Du point de vue de l'acquisition des connaissances, ces instances contiennent le savoir socialement admis sur les dénominations.

Les relations représentent les liens sémantiques des grandes associations et elles structurent des objets. La relation s'exprime de trois façons : son nom, son intension et son extension. La présentation des relations est importante. La plupart des modèles sémantiques utilisent le concept de Relation qui nous vient de la théorie des ensembles. Ainsi une relation se caractérise par sa symétrie ou son antisymétrie, et par sa cardinalité (le nombre d'objets qu'elle relie et leurs types respectifs, du fait qu'elle est bijective, inductive, surjective, etc.).

D'une manière générale, l'acquisition automatique de connaissances, à partir de textes, consiste à en extraire les connaissances, et à les représenter d'une manière structurée (Frath, P. et al. 2000). Dans le cas de l'acquisition de connaissances à partir de textes, le texte peut naturellement être considéré comme contenant les connaissances d'un expert, et peut être utilisé par l'ingénieur des connaissances pour établir l'ontologie qui contient un ensemble de concepts, un domaine de relations structurées avec ses hiérarchies et leurs instances relatives.

Il existe trois approches pour identifier les termes de l'ontologie : *Top-down*, *Bottom-up* et *Middle-out*. Les deux premières sont le plus souvent utilisées pour acquérir les connaissances. L'approche *top-down* concerne la spécialisation de classes en sous-classes, en ajoutant des propriétés (attributs), et l'approche *Bottom-up* correspond à l'approche agrégative.

Le choix d'une approche et ses motivations sont étroitement liés au domaine d'intervention et au type de données manipulées. Différentes techniques peuvent être utiles à différentes étapes du processus. Une source donnée peut être exploitée dans une ou plusieurs des perspectives adoptées pour l'ingénierie de l'ontologie, en fonction de la nature de la source.

Pour construire la taxonomie des concepts, les approches de l'acquisition automatique de connaissances ont été étudiées dans la littérature :

Approche *top-down*

L'approche *Top-down* est bien adaptée à des domaines bien cernés ou à des textes stéréotypés. Elle se fonde sur l'idée que les connaissances sont constituées d'unités

discrètes, donc isolables, et reliées par un petit nombre de types de relations ayant un caractère général, par exemple : représenter les connaissances du sens commun. (Frath, P. et al. 2000)

L'approche *Top-down* fonctionne en collectent les concepts les plus génériques, puis en construisant une structure par spécialisation. L'ontologie est construite en déterminant et en spécialisant les « top concepts ». Cette approche est utilisée pour établir les ontologies SMAAD car elle permet la réutilisation des ontologies et la vérification de la hiérarchie des classes et sous-classes qui peut être très intéressante, pour maintenir la cohérence. On commence par poser des ensembles de concepts et de relations couvrant idéalement tout le domaine. L'analyse manuelle des textes permet ensuite de remplir la structure abstraite et de construire l'ontologie.

L'approche *Top-down* propose également une taxonomie. Selon l'encyclopédie Wikipédia : « *La taxinomie est la science qui a pour objet de décrire les organismes vivants et de les regrouper en entités appelées taxons afin de les identifier puis les nommer et enfin les classer.* » Les taxonomies concernent tous les domaines de recherche.

L'axiomatisation du domaine peut fonctionner sur les types d'objets, les classes, les associations, et leurs interactions, puis présenter les graphes visualisables entre leurs interactions avec l'aide de l'UML.

Certains programmes utilisent l'approche *Top-down*, par exemple, GAIA est une analyse *top-down* influencé par des approches orientées objet. (Wooldridge, M. et al. 2000) La méthodologie AAII (Kinny, D. et al. 1996) est une approche *top-down* pour les systèmes d'agents BDI (Rao, A. S. & Georgeff, M. P. 1995), basée sur des méthodologies orientées objet, renforcée avec certains concepts basés sur des agents.

Approche *Bottom-up*

Dans (Daille, B. 1994), on décrit une méthode qui diffère de l'approche *top-down* en ce qu'elle applique un traitement statistique aux candidats à être des termes, de manière à minimiser le bruit. Ces méthodes produisent des données plus assurées que celles issues de l'intuition du cogniticien, et semblent plus aptes à construire des ontologies de qualité.

On peut faire précéder l'approche *top-down* par une approche *bottom-up*, où l'on commence par faire l'inventaire des connaissances du texte, pour ensuite construire l'ontologie.

L'approche *Bottom-up*, qui est aussi utilisée pour établir les ontologies SMAAD, fonctionne en collectant les concepts les plus spécifiques, puis en construisant la structure par généralisation. L'ontologie est construite en déterminant et en généralisant les concepts du niveau taxonomique bas. Cette approche vise à fournir des ontologies adaptées et spécifiques aux concepts bien détaillées.

Dans ce cas, on détermine les objets et concepts à partir de textes pour collecter des données linguistiques susceptibles de livrer les connaissances qui y sont contenues. Il existe trois associations remarquables dans l'approche *Bottom-up* : les relations d'instanciation /anti-instanciation (a-un), généralisation/spécialisation (est-un), et composition /décomposition (est-partie-de).

L'association d'instanciation/anti-instanciation indique qu'une classe instancie un objet, et qu'un objet est une instance d'une classe. Prenons la classe « Médicament » comme exemple : elle possède des propriétés ou attributs « nom », « marque », « validité » qui sont des propriétés statiques, tandis que hasNom() est une propriété dynamique. Les propriétés sont unies à la classe « personne » par l'association (a-un), cela signifie qu'un médicament a un nom, une marque et une validité.

La généralisation/spécialisation est une association entre classe et sous-classe. Par exemple, Pénicilline G est un antibiotique et Antibiotique est une sous-classe de Médicament, dont Pénicilline G hérite de toutes les propriétés de Médicament mais en plus a des propriétés spécifiques comme « contreIndicationAntibiotique », « effectSecondaireAntibiotique », etc.

La composition/décomposition : capsule est un composant de médicaments sous forme de capsule. L'objet « capsule » a des propriétés ou attributs comme « couleur ». Pour créer un objet Médicament comme Amoxicilline, il faut instancier un objet capsule de la classe capsule et attribuer une valeur à la propriété ou attribut couleur = « blanc » et « rouge ».

Ces trois associations et les contraintes qu'elles génèrent sont indispensables à la définition des objets et à leur autonomie. Elles définissent les fondements de la structure des objets regroupés en ensembles d'objets associés à une classe regroupant leurs propriétés communes.

D'autres associations sont complémentaires : temporelles (précède, suit,…), causalité (produit, provoque, cause, évoque…), spatiales (est dessus, est dessous, est à proximité de…), sont également étudiées pour présenter les propriétés de classes.

Approche *Middle-out*

L'approche *Middle-out* sert à identifier les concepts centraux du domaine choisi. Après avoir identifié les concepts principaux, les méthodes de généralisation et de spécialisation sont utilisées pour compléter l'ontologie. Cette approche vise à encourager l'émergence de domaines thématiques et d'améliorer la modularité et la stabilité du résultat.

(Uschold, M. & Gruninger, M. 1996) pensent que l'approche *Middle-out* est la meilleure. Ils estiment que l'approche *Bottom-up*, d'un niveau très élevé, dans le détail, augmente les efforts d'ensemble et rend difficile le repérage des points communs entre les concepts connexes, et augmente le risque d'incohérences menant à retravailler. En outre, ils pensent que l'approche *Top-down* aboutit à un choix arbitraire des catégories de haut niveau, ce qui conduit à des modèles moins stables.

> Une approche *Middle-out*, en revanche, établit un équilibre en termes de niveau de détail. Le détail ne se pose que si nécessaire, en spécialisant les concepts de base, de sorte que certains efforts sont évités. En commençant par les concepts les plus importants, et par la définition des concepts de plus haut niveau, les catégories de niveau supérieur se posent naturellement et sont donc plus susceptibles d'être stables. Ceci conduit à moins de re-travail et moins d'effort global. (Uschold, M. & Gruninger, M. 1996)

Cependant, dans cette thèse, nous n'adoptons pas cette approche car, comme l'ont noté ces auteurs eux-mêmes, la notion de concept de base peut dépendre étroitement du contexte d'application. Pour établir une ontologie réutilisable et augmentable, cette approche conduit à des extractions de résultats inexactes.

3.1.1.3 Créer un type d'objet

Créer la classe

Sur la base du type d'objet, il faut établir une axiomatisation du domaine concerné, les classes et les associations d'objets détectées, de même que leurs interactions.

Une classe se composant d'un seul membre est appelée « individu ». Toute chose unique, à laquelle nous pouvons donner un nom pour la distinguer de toutes les autres choses, peut être considérée comme une classe à un seul membre. Pour créer une classe d'objets, tout d'abord, il faut observer les objets qui interviennent dans l'environnement que nous cherchons à représenter. Après cette étape de conception, il

faut étudier les caractéristiques de l'objet que l'on observe. Certaines questions se posent et l'on doit réfléchir avant définir les interactions. Par exemple, quelles données sont essentielles et quels types de données sont associés ? Quels comportements, quelles actions, quelles utilisations permettent de consulter, modifier les données qui caractérisent ces objets ? etc. Ensuite, il faut regrouper les objets ayant les mêmes propriétés ou caractéristiques (données, comportement, traitements, etc.) en un ensemble d'objets que l'on appelle type ou classe d'objets. Enfin, on définit les attributs et les traitements de cette classe.

L'approche *Top-down* permet de vérifier la hiérarchie des classes/sous-classes avec l'aide de la taxonomie des concepts ; elle est bien adaptée à des domaines bien cernés ou à des textes stéréotypés. Elle fonctionne en collectant les concepts les plus génériques, puis en construisant une structure par spécialisation. L'ontologie est construite en déterminant et en spécialisant les « *top concepts* », puis des ensembles de concepts et de relations couvrant idéalement tout le domaine sont posés. L'analyse manuelle des textes permet ensuite de remplir la structure abstraite et de construire l'ontologie.

L'approche *Top-down* propose ensuite une taxonomie (ou taxinomie) et la complète en organisant et en effectuant le classement systématique des taxons et leurs relations.

Comment créer une classe d'objet ? Tout d'abord, il faut observer les objets qui interviennent dans l'environnement que nous cherchons à représenter. Après cette étape de conception, il faut étudier les caractéristiques de l'objet que l'on observe.

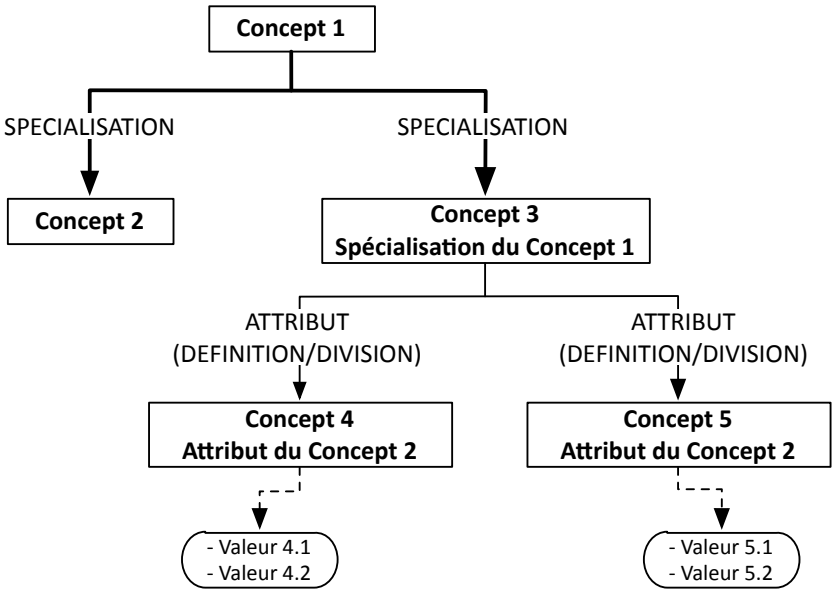


Figure 10 La subdivision de concepts pour créer une classe

Afin de diviser le concept pour créer la classe, l'approche « spécialisation » fonctionne pour la classification. Prendre la classe « médicament » par exemple :

> Le médicament est une substance ou préparation administrée en vue d'établir un diagnostic médical, de traiter ou de prévenir une maladie, ou de restaurer, corriger, modifier des fonctions organiques.

Pour créer la classe « antibiotique », à partir de la classe « médicament », la classe « antibiotique » hérite de toutes les propriétés de la classe « médicament » : attributs et traitements de « médicament » sont transférés du fait de la « **Spécialisation** » à « antibiotique ».

Autre approche « définition/division » vise à obtenir les attributs d'une sous classe. « *La division est une opération intellectuelle par laquelle une classe de choses peut être divisées en deux ou plusieurs autres classes plus petites.* ». (Whorf, B.L. 1956) Une classe obtenue par division est dite co-divisionnelle à toutes les classes obtenues par cette division et bien entendu à elle même. Par exemple, si la classe initiale est « médicament », on peut y distinguer « antibiotique ». La classe différente de « médicament » et « antibiotique » est nommée : « non-antibiotique ».

Le format général d'une classe d'objets et d'un objet est présenté ci-dessous. À travers la commutation de données entre la classe « médicament » et « objet médicament », « Attributs de la classe » liste toutes les propriétés génériques de la classe « médicament », en incluant la catégorie, groupe, nombre, genre, traits, type, notions en rapport, synonymes, description, et annotation. Les « sous-objets » montrent les noms de ses sous-classes tel que les « anesthésiants », « anti-diurétique » etc. Avec les connaissances données par « objet médicament », les « attributs des objets instances » et les « traitements des objets instance » coopèrent et instancient les objets. Par exemple, méthode *hasEffetSecondaireMédicament( )* aide à chercher l'effet secondaire de la classe « Pénicilline G », puis présente les résultats comme « allergie, anxiété, fièvre, essoufflement, hypertension, accélération du rythme cardiaque etc. »

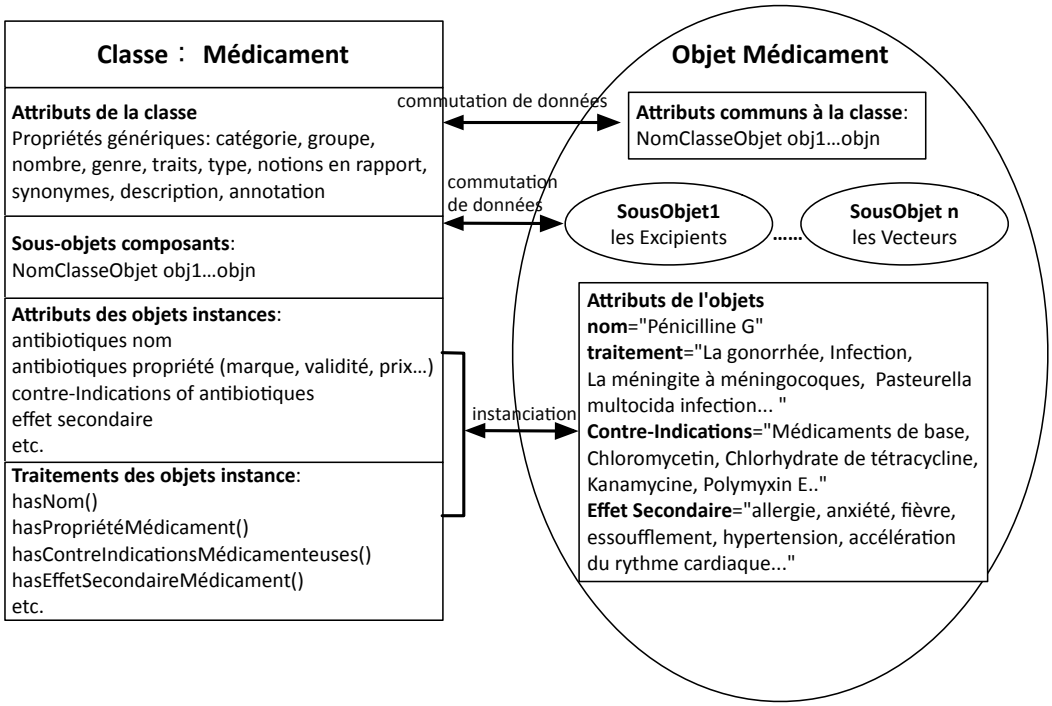


Figure 11 Format général d'une classe d'objets et d'un objet

3.2 Établir une ontologie

Le problème de l'application des langues naturelles est qu'elles manquent de formalisme et leur interprétation n'est pas collective. Un langage de modélisation approprié est formalisé et son utilisation et son sens doivent être sans ambiguïté. Les ontologies sont utilisées pour capturer le sens voulu et sans équivoque des primitives de modélisation. Une telle ontologie peut être utilisée pour créer des modèles explicites pour l'automatisation de la génération ou pour soutenir l'échange d'informations et de connaissances dans le domaine de la médecine. En outre, cette représentation devrait éliminer une grande partie du programme nécessaire pour répondre à de « simples » questions de bon sens (Gruninger, M. & Fox, M. S. 1995), en capturant les caractéristiques essentielles des entités existantes, dans un système, et leurs relations.

L'ontologie permet de promouvoir la réutilisation de connaissances dans plusieurs applications, elle facilite la maintenance de logiciels, grâce à une représentation explicite et elle rend pérennes des connaissances ontologiques, dans une perspective de mémoire organisationnelle.

Analyse de la connaissance du domaine

Elle est possible avec la spécification déclarative des termes. L'analyse formelle des termes est extrêmement précieuse lorsque l'on tente de réutiliser et d'étendre des ontologies existantes. (Noy, N. F. & McGuinness, D. L. 2001)

L'ontologie est utilisée également pour séparer les connaissances du domaine de la connaissance opérationnelle. Nous pouvons décrire une tâche de configuration d'un produit, à partir de ses composantes, conformément à une spécification requise, et mettre en œuvre un programme qui réalise cette configuration. (McGuinness, D.L. & Wright, J.R. 1998)

Aide à la conception et à l'utilisation des systèmes d'information

Les ontologies sont un apport pour l'ingénierie des systèmes d'information (Guarino, N. 1998) (Kassel, G. et al. 2000) :

Spécification et Acquisition de connaissances : une ontologie peut aider à l'analyse des besoins et à définir les spécifications d'un SI.

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

Interopérabilité : en jouant le rôle d'un format d'échange, l'ontologie permet de faire de coopérer à des systèmes d'information, basés sur des paradigmes de modélisation et des langages d'implantation différents.

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.

Réutilisation et Partage : une ontologie peut être, ou peut devenir (suite à une traduction) un composant réutilisable et/ou partagé par plusieurs logiciels.

Fiabilité et 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.

Aide à la communication entre agents humains et aussi entre ordinateurs

Le rôle des ontologies est d'améliorer la communication entre humains, mais aussi entre humains et ordinateurs et, finalement, entre ordinateurs (Van Bemmel & al., 1997). Par exemple, supposons que certains sites Web différents contiennent des informations médicales ou fournissent des services d'e-commerce médical. Si ces sites Web partagent et publient des termes qu'ils utilisent tous, issus de la même ontologie sous-jacente, alors les agents informatiques peuvent extraire et agréger l'information de ces différents sites Web. Les agents peuvent utiliser cette information agrégée pour répondre aux requêtes des utilisateurs ou comme données d'entrée à d'autres applications.

En somme, l'ontologie sert à favoriser la diffusion des informations et leur exploitation, et à promouvoir une nouvelle approche de conception des systèmes d'information (réutilisation de codes, interopérabilité des logiciels). Pour ces besoins de standardisation du vocabulaire, une terminologie ou une ontologie informelle peuvent suffire (O'Leary, D. E. 1998).

Réutilisation des connaissances de domaine

C'est l'un des points clés des applications de l'ontologie. Si nous construisons une grande ontologie, nous pouvons intégrer plusieurs ontologies existantes décrivant les parties d'un grand domaine. Nous pouvons également réutiliser une ontologie générale et l'étendre, pour décrire un domaine d'intérêt.

3.2.1 Ontologies médicales

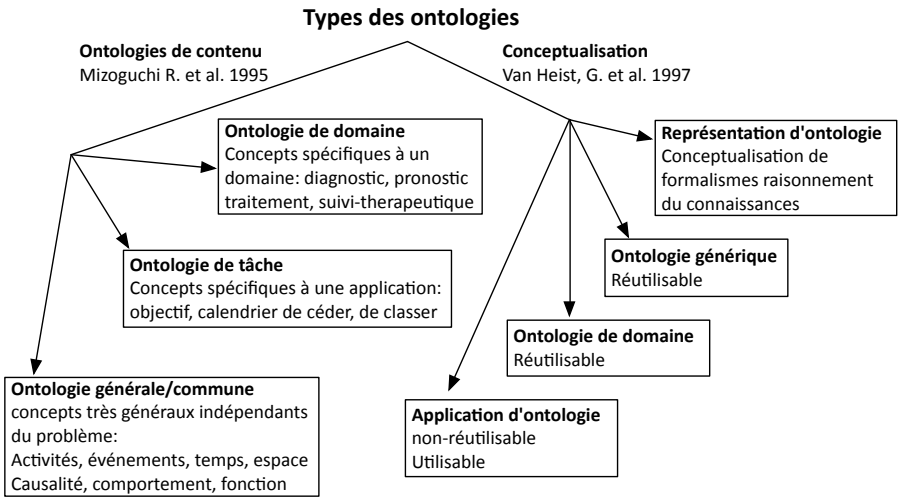


Figure 12 Types d'ontologies

Il existe différents types d'ontologies : ontologies de haut niveau, ontologies linguistiques et ontologies de domaine (ingénierie, médicales, business etc.).

Dans le domaine des ontologies médicales, l'application de ressources termino-ontologiques médicales est, principalement, l'enregistrement d'informations cliniques, l'Indexation contrôlée pour la recherche d'information et l'Indexation « médicale», pour la recherche d'information. Il y a plusieurs ressources terminologiques et ontologies existantes.

MÉNÉLAS

MÉNÉLAS (*an access system for medical records using natural language*) (Zweigenbaum, P. 1994), l'un des premiers projets ontologiques européens, propose une approche d'accès aux informations et aux connaissances médicales en différentes langues. Une ontologie dans le domaine des maladies coronariennes a été développée à partir de plusieurs sources incluant l'analyse des résumés d'articles et les interviews de spécialistes. Elle contient une hiérarchie de 1 500 concepts et 500 relations.

GALEN

GALEN (*General Architecture for Language, Encyclopedia and Nomenclature*) est un projet européen (1992-1999) (Rector, A.L. et al. 1993) qui décrit des concepts du domaine médical, dans le but de mettre en place un serveur de terminologie médicale partageable et réutilisable. Cette ontologie propose une représentation des concepts médicaux, indépendamment de l'application choisie, afin de fournir une base pour la création de terminologies en combinant les concepts. Le projet se fonde sur une ontologie CRM (*Common Reference Model*), développée en langage de logique descriptive, appelé GRAIL (*Galen Representation And Integration Language*).

ASBRU

ASBRU (*A task-specific, intention-based, and time-oriented language for representing skeletal plans*) (Miksch, S. et al. 1997) est utilisé pour représenter des protocoles cliniques. Il fournit une ontologie destinée à soutenir des tâches et à résoudre des problèmes, afin de représenter et d'annoter les directives cliniques sous une forme standardisée.

METHONTOLOGY

METHONTOLOGY (Fernández-López, M. et al. 1997) est inspirés des méthodes de l'Ingénierie des Connaissances. Elle repose sur un processus de développement d'ontologies tels que planification, assurance qualité, spécification, conceptualisation, formalisation et documentation. Le cycle de vie de METHONTOLOGY se fonde sur des prototypes évolutifs. Elle est supportée par les outils ODE, WebODE, OntoEdit, et Protégé. Tous ses concepts sont spécifiés et identifiés a travers de l'approche *Middle-out*.

TERMINAE

Terminae (Biebow, B. et al. 1999) propose la construction d'ontologies à partir de textes, en suivant quatre principales étapes : 1) constitution d'un corpus à partir d'une analyse des besoins de l'application visée, 2) étude linguistique pour identifier des termes et des relations lexicales, en utilisant des outils de traitement de la langue naturelle, 3) normalisation sémantique pour établir des concepts et des relations sémantiques, définis dans un langage semi-formel ; 4) formalisation et intégration des concepts au sein d'une base de connaissances formelle. Dans sa version actuelle, cet outil peut intégrer l'outil LEXTER, pour identifier des termes candidats (Bourigault, D. 1994).

CIM

Les maladies, l'acte et l'anatomie sont analysés grâce à un large champ terminologique : CIM-10, le *Medical Subject Headings* (MeSH) et SNOMED.

CIM (*Classification Internationale des Maladies*) est « *une classification médicale codifiée classifiant les maladies et une très vaste variété de signes, symptômes, lésions traumatiques, empoisonnements, circonstances sociales et causes externes de blessures ou de maladies.*» (CIM-10 1993)

MeSH

Le MeSH (*Medical Subject Headings*), thésaurus biomédical de référence, est un outil d'indexation, de catalogage et d'interrogation des bases de données de la NLM (*National Library of Medicine*), notamment MEDLINE/PubMed (Bodenreider, O. 2004). Le MeSH offre une organisation hiérarchique et associative et comprend jusqu'à neuf niveaux de profondeur. Nous n'entrons pas dans le détail de sa

description, le MeSH est un thesaurus très élaboré développé pour l'indexation et non pour les inférences (Charlet J. 2002).

SNOMED

SNOMED (*Systematized Nomenclature of Medicine*) (Côté R.A. 1993) est une nomenclature systématique de termes médicaux pour la médecine humaine et vétérinaire. Elle est interfacée avec CIM. SNOMED et permet d'indexer, stocker et récupérer des données médicales organisées de façon cohérente pour l'établissement d'outils d'analyse décisionnelle.

UMLS

Le UMLS (*Unified Medical Language System*) (Lindberg, D.A. et al. 1993) est un programme élaboré par la NLM (*National Library of Medecine de Bethesda*) ; il fournit un vocabulaire biomédical provenant de sources disparates telles que les terminologies cliniques, les sources de médicaments, des vocabulaires dans différentes langues, et des terminologies cliniques. Pour réaliser cet objectif, ce langage utilise un méta-thesaurus qui présente tous vocabulaire médical existant avec des millions de termes. Par ailleurs, ce langage fonctionne grâce à un réseau sémantique, fondé sur une hiérarchie de types sémantiques et une hiérarchie de relations ; il représente une classification de tous les concepts représentés dans le méta-thesaurus et les relations possibles entre eux.

En résumé, une ontologie peut définir et fournir une sémantique formelle pour l'information et permettre son exploitation par un ordinateur. Il peut aussi définir et fournir une sémantique interprétative d'un domaine du monde réel, fondée sur un consensus et permettant de lier le contenu exploitable par la machine à sa signification pour les humains.

3.2.2 Critères de conception des ontologies

Pour guider et évaluer les types de construction d'ontologies, nous avons besoin de critères objectifs, fondés sur le but de l'*artefact* résultant. (Gruber, T. R. 1993) propose une première série de critères de conception d'ontologies dont le but est le partage et l'interopérabilité des connaissances entre programmes fondés sur une conceptualisation partagée. Les principes sont les suivants :

- Clarté : l'ontologie doit communiquer efficacement le sens voulu des termes définis. Les définitions doivent être objectives.
- Cohérence : il faut éviter toute incohérence et ambiguïté dans les définitions. À tout le moins, les axiomes définissant doivent être logiquement cohérents.
- Exhaustivité : une ontologie doit être conçue pour anticiper les usages du vocabulaire commun. Elle doit offrir une base conceptuelle pertinente pour permettre une série de tâches prévues et la représentation doit être conçue de telle sorte qu'on puisse étendre et spécialiser l'ontologie.
- Biais d'encodage minimal : la conceptualisation doit être spécifiée au niveau des connaissances, sans dépendre d'un encodage de niveau symbolique particulier.
- Intervention ontologique minimale : une ontologie doit exiger un engagement ontologique minimal suffisant, pour soutenir les activités de partage des connaissances prévues.

3.2.3 Méthodes et méthodologies de la construction d'une ontologie

Le développement d'une ontologie est un processus complexe auquel plusieurs travaux ont été consacrés. Malgré la panoplie de méthodes proposées pour le développement d'une ontologie, aucun consensus n'existe sur sa construction, actuellement. Néanmoins, nous pouvons résumer la conception d'une ontologie en quatre étapes (Noy, N. F. & McGuinness, D. L. 2000) :

- Spécification et analyse des besoins : identification des objectifs, des besoins et du contexte.

  Le développement d'une ontologie commence par la définition du domaine et de sa portée, ce qui induit une réponse à certaines questions : quel est le domaine que l'ontologie va couvrir ? À quoi sert cette ontologie ? À quels types de questions les informations de l'ontologies doivent-elles répondre ? Qui va utiliser et maintenir l'ontologie ? Etc.
- Conception de l'ontologie : modéliser ses concepts, relations et axiomes.

  La conceptualisation consiste à identifier et à structurer les connaissances d'un domaine, à partir des sources d'informations. L'acquisition de ces connaissances peut s'appuyer à la fois sur l'analyse de documents et sur l'interview avec des experts du domaine. Une fois les concepts identifiés par leurs termes et leurs relations sémantiques, l'ontologie peut être décrite dans

un langage semi-formel (tables et graphes), à travers leurs propriétés, leurs instances connues et les relations qui les lient entre eux.

- Développement de l'ontologie : implémenter l'ontologie dans un langage formel.

  Une ontologie peut s'exprimer selon plusieurs degrés de formalisation allant des définitions les plus informelles, en langage naturel, aux expressions écrites en logique du premier ordre qui doivent respecter une sémantique et une syntaxe très strictes. Le degré de formalisation de l'ontologie va dépendre principalement des besoins. On peut considérer les quatre degrés cités plus haut. Il faut retenir que les ontologies doivent être compréhensibles par des humains et des ordinateurs, à la fois : interface homme/machine. Pour obtenir un bon équilibre entre la précision technique et compréhensibilité, il est important, pour chaque définition technique, de conserver une description informelle de la définition.

  Enfin, comme l'ontologie devra être exploitée par un ordinateur, il est nécessaire qu'elle soit calculable et, pour cela, il est nécessaire de l'implémenter dans un langage formel.

- Évaluation de l'ontologie : évaluer sa cohérence.

  Transcription de l'ontologie dans un langage formel et opérationnel de représentation de connaissances du domaine, et adaptée à une application particulière.

En général, les travaux de construction d'une ontologie sont issus d'expériences de développement d'ontologie. Ils peuvent être classés suivant différents critères. Le plus courant consiste à distinguer les méthodes manuelles des méthodes qui aident des outils pour extraire des connaissances à partir de corpus textuels.

Sachant que plusieurs méthodes de construction d'ontologies coexistent dans la littérature, ce sous-chapitre vise à fournir un aperçu des approches existantes et à montrer leurs similitudes et différences.

Méthodes d'acquisition de connaissances
Intelligence Artificielle : - **KOD** [Vogel C. 1989] - **KADS** [Wielinga B. J. et al. 1992] - **MACAO** [Aussenac-Gilles N. et al. 1994] - **MKSM** [Ermine J.L. et al. 1996] **MASK** [Ermine J.L. et al. 2001]
Méthodes sémantiques de systèmes d'information classiques sont déjà très puissantes pour extraire de la sémantique issue de textes :
Réseaux sémantiques : - **J.F. Le Ny** : réseaux sémantiques multiples, un parcours cognitif - **MERISE** : modèles entité-associations - **Chen** : modèle entité-association - **UML 2** : modèles orientés objet
Langages de communication entre agents
- **KQML – ACL** [Labrou Y. et Finin T. 1997]

Tableau 11 Méthodes et méthodologies de construction d'ontologies

Nous faisons référence aux méthodes et méthodologies de la construction d'une ontologie. Par exemple, l'approche KADS est la méthode la plus complète pour la définition des bases de connaissance. La direction de ses recherches est très générale et elle met l'accent sur la modélisation des étapes du processus : commercialisation, entreprise, etc.

La méthode KADS travaille sur des bases théoriques ; nos recherches mettent l'accent sur les différentes étapes cliniques : diagnostic, pronostic, traitement et suivi-thérapeutique. L'objet de l'étude et les méthodes de recherche sont diversifiés.

Méthodes d'acquisition de connaissances

KOD (Vogel, C. 1989)

La méthode KOD (*Knowledge Oriented Design*) a été développée en 1988 par l'anthropologue Claude Vogel, pour fournir un support à l'activité Intelligence Artificielle de CISI Ingénierie.

Cette méthode se fonde sur l'analyse systématique du texte et permet de générer un système à base de connaissance pour le traitement du corpus, visant à identifier les

concepts, les associations, les catégories et les caractéristiques présentées dans Génie cognitif de Vogel. (Vogel, C. 1988)

La méthode KOD utilise des techniques de l'ethnologie et de la linguistique pour construire un système à base de connaissance. KOD propose trois modèles : (Frécon, L. & Kazar, O. 2009) pratique, cognitif et informatique. Ils fonctionnent dans trois paradigmes : représentation, action et interprétation. La coopération de modèles et paradigmes vise à réaliser la modélisation de l'expertise.

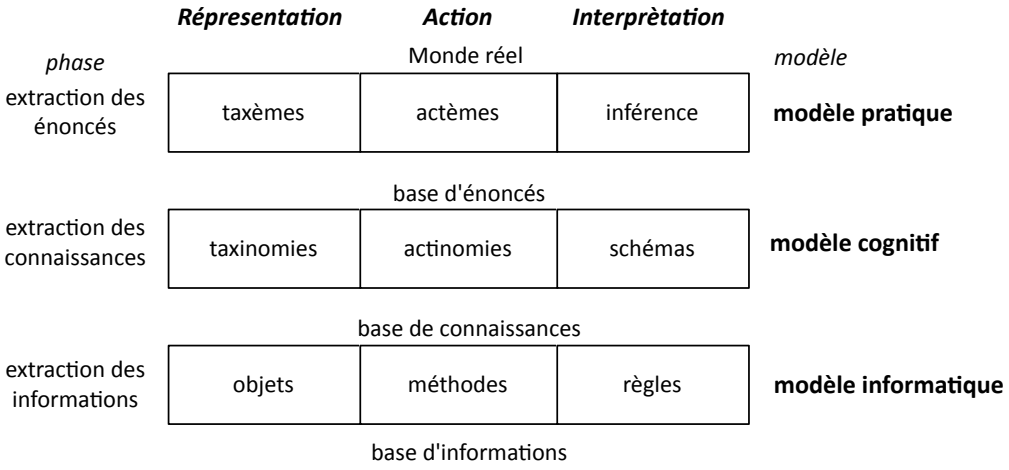


Figure 13 Schéma de l'approche KOD

KADS, CommonKADS (Wielinga, B. J. et al. 1992)

KADS (*Knowledge Based Systems Analysis and Design Support*) a été développé en 1989, dans le projet européen ESPRIT. KADS indique qu'une insuffisance des méthodes de développement dans les systèmes experts, en particulier dans la construction de bases de connaissances, par exemple, fait obstacle au transfert de compétences. Afin de résoudre ce problème, KADS propose deux approches pour soutenir correctement la production de bases de connaissances :

- Un cycle de vie répond aux contraintes techniques et économiques, afin de réaliser le contrôle du processus de production, l'assurance qualité du système, etc.
- Un ensemble de modèles structurent un système, afin de réaliser les tâches d'analyse et la transformation des connaissances d'expert en une forme exploitable par la machine.

Par conséquent, KADS et CommonKADS sont des méthodes très complètes et bien adaptées à la construction des bases de connaissances et au développement des systèmes sous-jacents. Les composantes de cette approche évolutive sont : (1) une méthodologie pour la gestion de projets d'ingénierie des connaissances ; (2) un atelier

de l'ingénierie des connaissances ; (3) une méthodologie pour effectuer le déclenchement de la connaissance.

MACAO (Aussenac-Gilles, N., & Matta, N. 1994)

KADS est une démarche descendante (*Top down*), tandis que MACAO est une démarche ascendante (*Botton up*) en quatre étapes, pour acquérir des connaissances avec MACAO :

I. Détermination des matériaux de base (figure 14)
   - Identification de l'expertise
   - Caractérisation de l'expertise à partir d'exemples

II. Construction du schéma du modèle conceptuel
   - Construction du modèle du domaine (connaissances statiques) et du modèle de raisonnement (méthodes de résolution) par catégorie

III. Construction du modèle complet

IV. Opérationnalisation et validation du modèle

La figure ci-après montre les détails de la détermination des matériaux de base au processus de KADS.

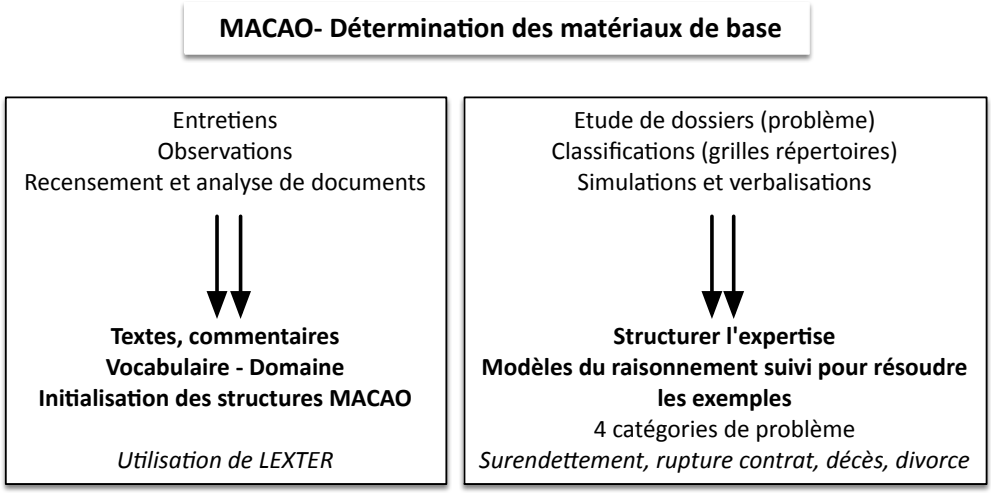


Figure 14 MACAO- Détermination des matériaux de base (Bourigault, D. et al. 2004)

MKSM et MASK (Ermine, J.L. et al. 1996) (Ermine, J.L. 2001)

MKSM présente une méthodologie d'analyse qui permet de contrôler de la complexité dans les projets de gestion des connaissances (Ermine, J.L. et al. 1996). Cette méthode permet de raffiner l'analyse et la modélisation de connaissances, afin

d’accéder à une lisibilité appropriée des connaissances à gérer, des projets possibles à implémenter et des critères de décision pertinents.

MASK (Ermine, J. L. 2001) est une méthode basée sur le programme MKSM. La méthodologie MASK est appliquée pour formaliser la connaissance. Évaluée par un grand nombre d'entreprises ou d'organisations de nature diverse en France et à l'étranger, elle est largement utilisée et diffusée. Certaines raisons de choisir MASK sont listées ici :

- Création, capitalisation et transfert de connaissance d’experts ;
- Diffusion de connaissances d’experts avec des outils diversifiés : livres, produits audio et vidéos, etc.
- Construction de corpus d'informations ou de documents ;
- Gestion et modélisation de connaissance des processus industriels et d'entreprise, pour améliorer la productivité et la compétitivité et l'innovation.

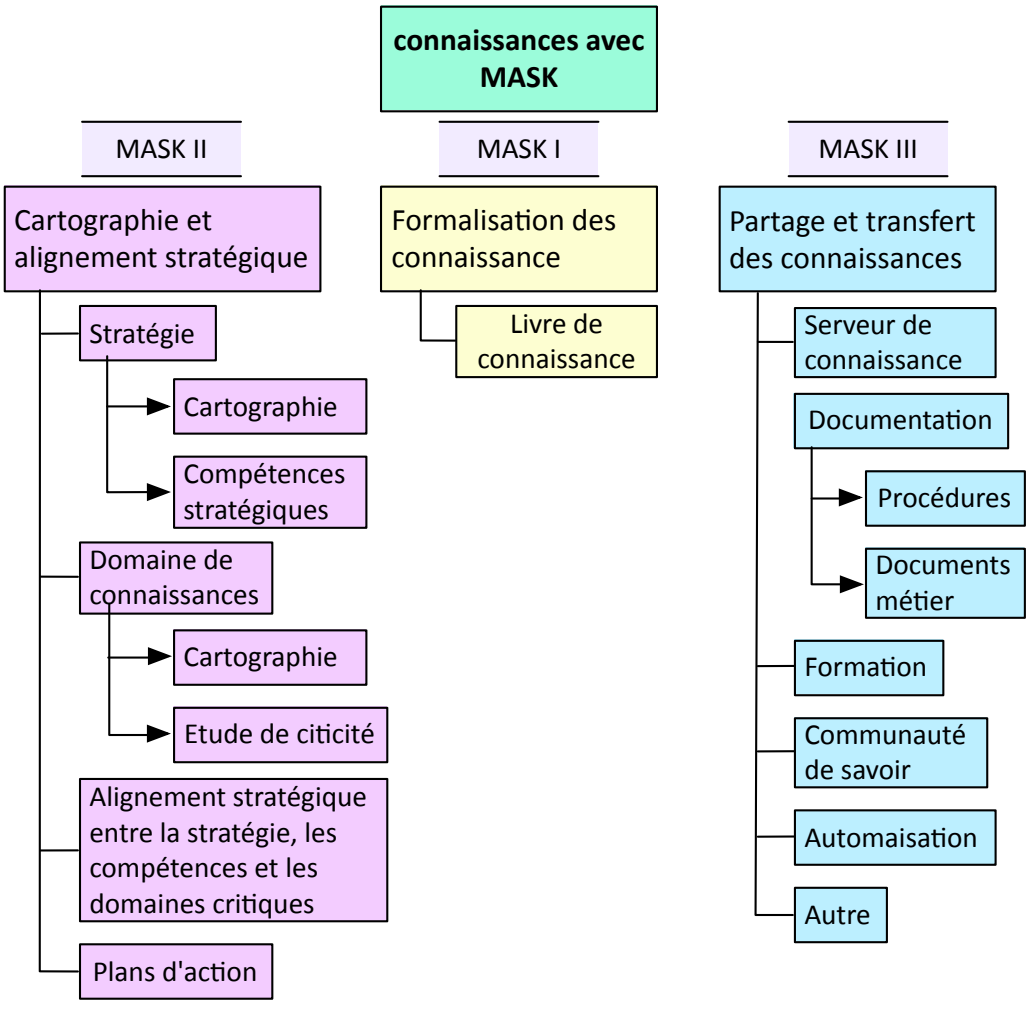


Figure 15 Connaissances avec MASK (Manning, C.D. & Schütze, H. 1999)

KQML (Labrou, Y. & Finin, T. 1997)

KQML (*Knowledge Query and Manipulation Language*) est un langage de haut niveau de communication entre agents cognitifs. Il se base sur la théorie des actes de langage, dans le but de permettre à l'échange de l'information. KQML est indépendant de la syntaxe du contenu et de l'ontologie des messages, du mécanisme de transport et du langage de codage des messages (Labrou, Y. & Finin, T. 1997).

Dans le langage KQML, les agents peuvent comprendre le vocabulaire d'un domaine donné à l'aide d'ontologies. L'ontologie peut spécifier et simplifier le domaine exploité.

Méthodes sémantiques

Le lexicologue, **J.F. Le Ny**, montre tout l'intérêt des réseaux sémantiques (Le Ny, J.F. 1975).

MERISE : Modèles entité-associations

La méthode Merise (Tardieu, H. et al. 1983) est une méthode d'analyse, de conception et de réalisation de systèmes d'informations informatisés. Merise est utilisée pour la gestion de projets pour les organisations. Merise comporte deux types d'associations :

- les CIF (*Contrainte d'intégrité fonctionnelle*) qui a une cardinalité min à 0 ou 1 et une cardinalité max à 1 ou n, avec la particularité d'être binaire. CIF n'a pas de propriétés.
- les CIM ont toutes leurs cardinalités max à n pour être N-aires. CIM peut avoir des propriétés.

L'association est un lien sémantique entre entités :

- 1 entité reliée à elle même : la relation est dite réflexive,
- 2 entités : la relation est dite binaire,
- 3 ou plus : parfois ternaire, voire de dimension supérieure.

Cette description sémantique s'enrichit de la notion de cardinalité. Le nombre minimum (0 ou 1) et maximum (1 ou n) peut participer à une association. Par exemple, un laboratoire est implanté dans un lieu (cardinalité. Min = 1) et un seul

(cardinalité. Max = 1), une université ; et une université peut faire l'objet soit d'aucune (cardinalité. Min = 0) aucune implantation de laboratoire, soit en avoir plusieurs (cardinalité. Max = n). On a donc les combinaisons (0, 1), (0, n), (1, 1) et (1, n).

Chen : Types de « modèles » (méta-modèles) dits « *Entity-Relationship* »

Peter Chen montre qu'un modèle entité-relation peut être utilisé comme base d'unification des différentes vues des données : le modèle de réseau, le modèle relationnel, et l'ensemble modèle de l'entité. (Chen, P. P. S. 1976) Il présente la classification suivante des différents modèles « entité-association ». (Chen, P. P. S. 1976) (Chen, P. P. S. 1981)

Dans l'étude d'un modèle de données, Chan identifie quatre niveaux de données logiques :

(1) Informations concernant les entités et les relations existant dans l'esprit humain.

(2) Structure de l'information : organisation de l'information dans laquelle les entités et les relations sont représentées par des données.

(3) Structure de données chemin-d'accès-indépendant : les structures de données ne sont pas impliquées dans des programmes de recherche, schémas d'indexation, etc.

(4) Structure de données chemin-d'accès-dépendant

L'ensemble du modèle est principalement concerné par les niveaux 1 et 2. Le modèle entité-relation est lié aux niveaux 1 et 2. Le modèle relationnel est principalement concerné par les niveaux 2 et 3. Le modèle de réseau est principalement concerné par le niveau 4.

Entité et Ensembles de l'entité

Les entités sont classées en différents ensembles d'entités : EMPLOYE, PROJECT, et DEPARTMENT. Un prédicat est associé à chaque ensemble d'entités pour tester si une entité lui appartient ; par exemple, si une entité est située dans l'ensemble des entités EMPLOYE, il a les propriétés communes avec les autres entités de cet ensemble. Notons que les ensembles de l'entité EMPLOYE ne peuvent être mutuellement disjoints. Par exemple, une entité qui appartient à l'entité MALEPERSON appartient aussi à l'ensemble des entités PERSONNE. Dans ce cas, MALEPERSON est un sous-ensemble de PERSONNE.

Relation et Rôle

Considérons des associations entre entités. Par exemple, « mariage » est une relation entre deux entités dans l'ensemble de l'entité PERSONNE. Le rôle d'une entité dans une relation est la fonction qu'il a dans la relation. « Husband » et « wife » (« Mari » et « Femme » sont des rôles.

Attribut et Valeur

Les informations relatives à une entité ou relation se fondent sur des observations et des mesures. Elles sont exprimées par un ensemble de paires attribut-valeur. « 4 », « bleu » et « Alice » sont des valeurs. Elles sont classées en différents ensembles de valeurs : « COLOR », « FIRST-NAME », et « LAST-NAME ». Une valeur d'une série de valeurs peut être équivalente à une autre valeur dans un ensemble de valeurs différentes.

Peter Chen utilise le principe du schéma, pour présenter et représenter les entités et les relations : diagramme entité-relation.

UML 2 : Modèles orientés objet

En informatique, UML (*Langage de modélisation unifié*) est un langage de modélisation graphique à base de pictogrammes (Booch, G. et al. 1998) (Arlow, J. & Neustadt, I. 2005). Il est utilisé en développement logiciel, et en conception orientée objet.

UML spécifie des ensembles, des relations dont on donne les propriétés et fonctions : injective, surjective, relation quelconque, etc. UML utilise des « cardinalités » (cf. 0-1 pour les fonctions partielles et 1-1 pour les fonctions totales) et il présente 6 cas de relations. Par exemple *date_de_naissance* est une fonction de l'ensemble PERSONNE, vers l'ensemble Dates, *date_de_mariage* qui une fonction partielle de l'ensemble PERSONNE vers l'ensemble Dates.

3.2.4 Comment les ontologies fonctionnent

> *"Ontologies are developed for the purpose of reusability and shareability of knowledge, and reusability is directly linked with generalisation, i.e., generic concepts are usually more reusable than specific ones. The desired scope of reusability is a very important decision that has to be taken before an ontology is designed. Although it is true that generic concepts are in general more reusable, the reuse of generic concepts for specific applications may involve, in certain cases, a big design effort to translate the generic concepts into specific ones. This effort has to be seriously considered during the design of an ontology, and compared with the effort of reusing, for example, an ontology built of specific concepts belonging to related applications."*
>
> Bernaras A. et al., 1996

L'ontologie fournit un mode de modélisation et de représentation pour réaliser la communication entre les humains (l'ontologie est une composante structurée de la mémoire qui permet des annotations de ressources d'information) et entre agents (e.g. message).

La représentation et la modélisation et comprennent le mode computationnel et déductif (par exemple : la distance et la projection), et le mode co-opérationnel (par exemple : les requêtes et le protocole d'allocation).

Pour acquérir des connaissances et établir une ontologie, il est nécessaire d'étudier la logique de conception complexe et de nombreuses questions ouvertes en pratique. Cette procédure a besoin d'une plate-forme intégrée, avec tous les outils de conception.

Nous pouvons considérer l'ontologie comme un objet vivant, car il soutient le cycle de vie et maintient les relations entre tous les objets construits dans ou avec l'ontologie.

L'ontologie fournit des suggestions, générées automatiquement et basées sur une requête de l'utilisateur, à travers la carte de corpus de texte ou à partir du corpus de texte, en regroupant les documents similaires.

Un système d'ontologies permet d'organiser, de placer et de visualiser la réponse à une requête, en extrayant les mots clés principaux des documents, puis en énumérant les documents contenant les concepts.

Chaque concept est relié à un ensemble de documents. Les documents sont attribués automatiquement aux concepts. Les affectations de documents peuvent être éditées manuellement par l'ingénieur des connaissances.

Les méthodes de découverte de concepts comprennent les méthodes non supervisées (suggestions fournies par le système) et les méthodes supervisées (l'apprentissage et le visualisation du concept). L'apprentissage du concept, qui aide à comprendre les concepts découverts, est réalisé par l'extraction de mots clés et la génération d'une liste de mots-clés caractéristiques d'un concept donné. En ce qui concerne la visualisation de concepts, nous devons créer une liste des documents à partir d'un concept donné, puis la présenter à l'utilisateur final.

Les utilisateurs peuvent interagir avec le système multi-agents en temps réel avec l'apprentissage automatique intégré et les méthodes de *Text Mining*. Les utilisateurs sélectionnent des suggestions appropriées et les ajoutent à l'ontologie.

Chapitre 4 DÉVELOPPEMENT DES ONTOLOGIES

Le travail de développement des ontologies est une aide à la construction ontologique automatique. Les approches pour la modélisation des ontologies incluent l'utilisation de modèles entité-association (*cf*. *supra*, chap. 3) et celle des langages de programmation tels que les Logiques de Description, le Programme en langage OWL et UML, etc.

UML et sa syntaxe OWL sont utilisées pour la matérialisation de la mémoire, tandis que le système multi-agents et les techniques d'apprentissages automatiques, mentionnés dans le premier chapitre sont adoptés pour l'exploitation de la mémoire.

Nous présentons également le langage Java et modèles (e.g. le Raisonnement à partir de Cas : RàPC) qui traitent et présentent la construction du sens. Les fonctions de langues de modélisations employées dans cette thèse sont présentées ci-dessous.

Modéliser la représentation de savoirs linguistiques (Java, OWL, UML, Description Logiques)

- Permet d'extraire des termes et des associations à partir d'un corpus de textes, grâce à l'utilisation de programmes et d'algorithmes d'analyse linguistique
- Modéliser les échanges d'éléments de connaissance entre les différents modules du SMAAD
- Fournir les connaissances correspondant aux requêtes, pour répondre à la requête d'utilisateur finale, ainsi que pour renouveler l'ontologie

Modéliser la représentation d'ontologies (Java, OWL, UML)

- Initialisation de l'ontologie
- Extension de la hiérarchie d'une ontologie à travers la génération de nouveaux savoirs
- Permettre la réutilisation des ontologies indépendantes

L'application des connaissances pour les experts médicaux (Java, OWL, GUI, XML, RàPC)

- Montrer la sortie sous de multiples formes, y compris du texte, des arborescences, des graphiques et des références.
- Interface d'utilisateur pour la visualisation pendant le processus de conception mis en œuvre

Après avoir défini les textes médicaux ciblés et traité les connaissances médicales, il est nécessaire d'écrire le code par énumération des concepts et des attributs, et de décrire les relations entre les concepts et les attributs du domaine médical. Nous réalisons tous les programmes avec l'aide de la bibliothèque Java de Stanford. Il existe déjà de nombreuses règles en langue française dans cette bibliothèque ou sur l'internet. On réorganise les règles en code Java.

Les modèles d'ontologie proposés sont destinés à l'axiomatisation d'un domaine de connaissances. Ils sont fondés sur des graphes qui peuvent être implantés à l'aide de plusieurs formalismes : logique formelle, modèles à règles de production, langages sémantiques comme XML et ses dérivés OWL, des langages fonctionnels comme Objective Caml. Il faut donc réfléchir à la chronologie étapes du cycle de vie et du développement d'une ontologie.

Figure 16 résume les principales langues de l'ontologie et leurs relations entre elles, sous la forme d'une pyramide (Corcho, O. et al. 2003). Tous ces langages sont basés sur la syntaxe XML : XOL (*Ontology Exchange Language*), RDF (*Resource Description Framework*), son extension RDF Schéma et, enfin, SHOE (*Simple HTML Ontology Extension*), avec sa version antérieure, basée sur HTML. Les langages OIL (*Ontology Inférence Layer*), DAML+OIL (*Darpa Agent Markup Language et Ontology Inference Layer*) et OWL qui au plus haut de la pyramide sont des extensions de RDF(S). OWL est le standard le plus utilisé car il offre une plus grande expressivité, surtout pour les restrictions dans la taxonomie des concepts. Il dispose de trois extensions : OWL LITE, OWL DL et OWL FULL.

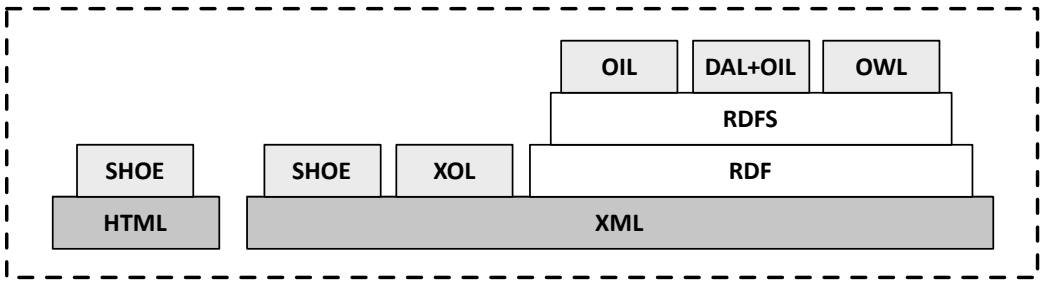


Figure 16 Pyramide des langages d'ontologies basés Web (Corcho, O. et al. 2003)

Modèle UML, fonctionne selon la conception orientée-objet avec son schéma UML, c'est un générateur de code qui peut traiter les codes XML, JAVA, C++ et OWL.

UML peut représenter les textes en classes d'objets et associations, tandis que XML, Java, C++ et OWL sont les langages informatiques pour l'implémentation de programme réel.

L'opérationnalisation dans un langage, par exemple, les graphes conceptuels ou les logiques de description (e.g. OWL), peuvent réaliser la représentation des connaissances. Le code OWL utilisé dans cette thèse est conçu pour présenter l'Ontologie Brute Spécialisée selon le domaine du texte analysé, encapsulé, ensuite, dans un agent Ontologie Spécialisée. Les fonctions du code OWL incluent la détermination des attributs et des nombres cardinaux, la déclaration d'opérations logiques sur la classe, etc. Dans les paragraphes suivants, nous présentons OWL, langage le plus utilisé.

Etapes successives	**Fonctionnalité**	**Langage**
Niveau Conceptuel	Conception du système Modélisation orientée-objet Représentation les textes en classes d'objets et associations.	UML
Niveau Logique	Choisir le modèle de représentation	
Niveau Physique	Implémentation	XML, OWL, Java, C++

Tableau 12 Niveaux et fonctionnalités de différents langages

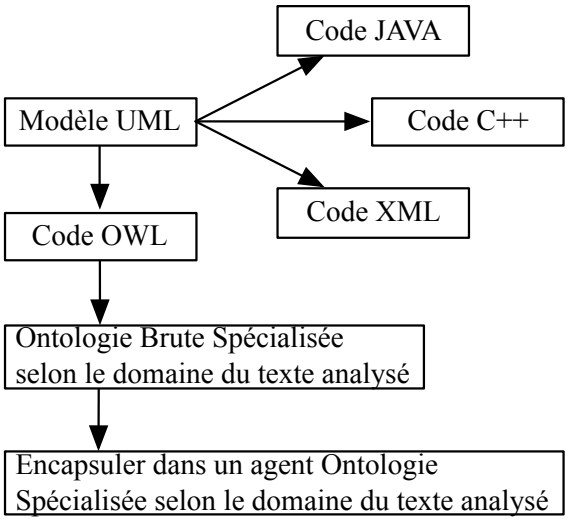


Figure 17 Opérationnalisation de Langages pour l'élaboration d'une ontologie

4.1 OWL (*Ontology Web Language*)

OWL est un langage de balisage sémantique pour publier et partager des ontologies sur le *World Wide Web*. C'est un langage fondé sur la syntaxe RDF/XML et héritier des travaux de DAML+OIL (Horrocks, I. 2002).

OWL (*Web Ontology Language*) et DAM + OIL : leur langue de fonde sur la logique de description qui permet de représenter les connaissances d'un domaine spécifique d'une manière bien structurée, avec une sémantique formelle basée sur la logique. L'idée principale de la langue basée sur la logique de description consiste à utiliser des ensembles de concepts et de relations binaires pour créer les expressions de concepts et relations complexes.

OWL, extension de RDF, comporte de nouvelles fonctions. Il étend les possibilités de RDF Schéma et permet de décrire des vocabulaires, des ontologies et les langages les plus usuels ; il introduit l'aspect sémantique qui manque à RDF manque : les outils qui comparent les propriétés et les classes (e.g. identité, équivalence, restriction, expression, etc.). OWL étendu à RDFS se résument comme suit :

Domaine	**Nouvelle fonction**	**Description**
Identité des données	OWL divise l'univers en deux parties disjointes. La première partie consiste en des valeurs qui appartiennent à des types de données XML Schema (remplace littéraux en RDF (S)). La deuxième partie se compose d'objets (individuel) considérés comme des membres des classes décrites (ressources en RDF (S)).	
Équivalence de classe	`owl:sameClassAs` `owl:equivalentTo`	Exprime l'équivalence entre classes
Equivalence de propriété	`owl:samePropertyAs` `owl:equivalentTo`	Exprime l'équivalence entre propriétés
Équivalence d'instance	`owl:sameIndividualAs`	Indique que deux objets sont les semblables
	`owl:differentIndividualFrom`	Indique que deux objets sont différents
Expression de la classe	`owl:oneOf`	Énumération
	`owl:intersectionOf`, `owl:unionOf`, `owl:complementOf`	Une propriété de restriction, ou une combinaison booléenne d'expressions de classes
Restrictions de propriétés	`owl:allValuesFrom`, `owl:hasValue`, `owl:someValuesFrom`	Les restrictions de valeurs
	`owl:cardinality`, `owl:maxCardinality`, `owl:minCardinality`	Les restrictions de cardinalités
Propriétés des propriété	`owl:inverseOf`	Indique la propriété inverse
	`owl:TransitiveProperty` `owl:SymmetricProperty`	Présente les propriétés algébriques
	`owl:FunctionalProperty` `owl:InverseFunctionalProperty`	Présente les propriétés uniques des valeurs

Tableau 13 OWL étendu RDFS avec ses nouvelles fonctions

Nom de la propriété	Domaine	Range
owl:intersectionOf	owl:Class	rdf:List
owl:unionOf	owl:Class	rdf:List
owl:complementOf	owl:Class	owl:Class
owl:oneOf	owl:Class	rdf:List
owl:onProperty	owl:Restriction	rdf:Property
owl:allValuesFrom	owl:Restriction	rdf:Class
owl:hasValue	owl:Restriction	*Not specified*
owl:someValuesFrom	owl:Restriction	rdf:Class
owl:minCardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite : {0,1} OWL DL/Full : {0,…,N}
owl:maxCardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite : {0,1} OWL DL/Full : {0,…,N}
owl:Cardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite : {0,1} OWL DL/Full : {0,…,N}
owl:inverseOf	owl:ObjectProperty	owl:ObjectProperty
owl:sameAs	owl:Thing	owl:Thing
owl:quivalentClass	owl:Class	owl:Class
owl:sameIndividualAs	owl:Thing	owl:Thing
owl:differentFrom	owl:Thing	owl:Thing
owl:disjointWith	owl:Class	owl:Class
owl:distinctMembres	owl:AllDifferent	rdf:List
owl:versonInfo	*not specified*	*not specified*
owl:priorVersion	owl:Ontology	owl:Ontology
owl:incompatibleWith	owl:Ontology	owl:Ontology
owl:backwardCompatibleWith	owl:Ontology	owl:Ontology
owl:imports	owl:Ontology	owl:Ontology

Tableau 14 Liste des propriétés de OWL

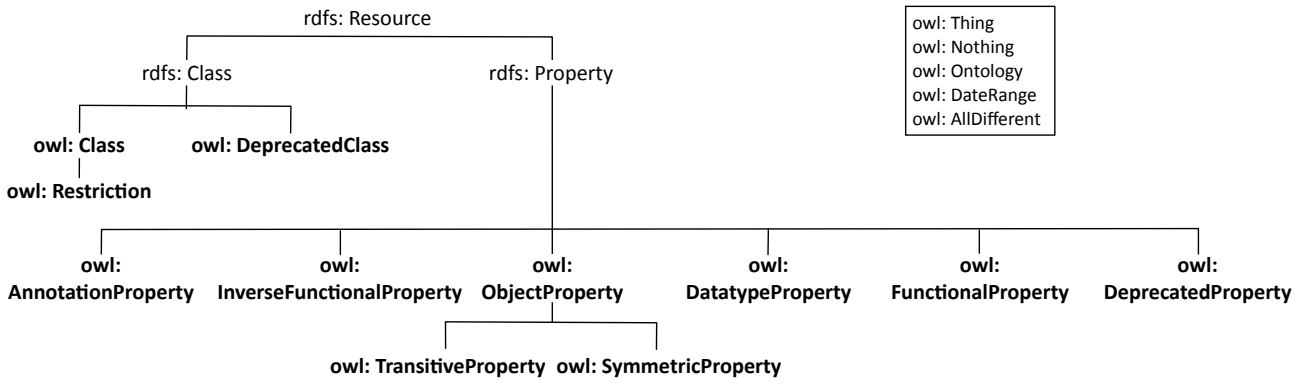


Figure 18 Taxonomie de classes de l'ontologie (Gómez-Perez, A. et al. 2003)

Le groupe de travail du W3C décompose OWL en trois sous-langages destinés à des communautés différentes d'utilisateurs. Ces trois sous-langages sont présentés ci-dessous selon leur expressivité et complexité croissante.

- OWL-Lite : C'est le moins expressif des sous-langages d'OWL. Il tend à saisir certaines des fonctionnalités les plus utilisées de OWL et de décrire une langue utile enrichit RDFS, dans le but d'ajouter des fonctionnalités importantes et soutenir les applications web.

- OWL-DL : Ce sous-langage fonctionne sur la logique de description (DL) et offre une expressivité maximale, en assurant la complétude et la décidabilité informatique. Tous les résultats sont calculables dans un temps raisonnable.

- OWL-Full : Ce dernier offre l'expression la plus complète, mais il n'existe pas d'implémentation complète de OWL-Full à ce jour. En outre, ses propriétés ne sont pas relatives à notre travail basé sur l'ontologie.

La Figure 19 montre les relations entre language RDF, RDFS, OWL Lite, OWL DL et OWL Full.

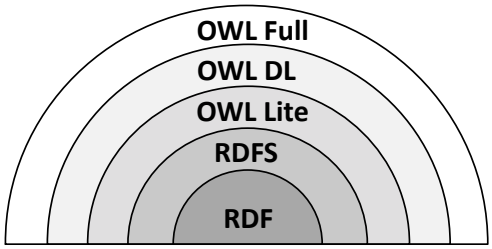


Figure 19 les relations entre language RDF, RDFS, OWL Lite, OWL DL et OWL Full

Les exemples de programme en OWL sont présentés à la suite, pour expliquer les fonctions de OWL ; ils incluent les annotations du modèle, le vocabulaire et l'ontologie.

```
1     <Modyco:Publication rdf:about="http://www.modyco.fr/publications.html">
2       <Modyco:Title> SDMC : Un système multi-agents d'aide à la décision clinique fondé
      sur des ontologies </Modyco:Title>
3       <Modyco:Author>
4         <Modyco:Person rdf:about="http://www.modyco.fr/annuaire/Membres/" />
5       </Modyco:Author>
6     </Modyco:Document>
```

Figure 20 Fonction de OWL: annoter le modèle

L'exemple de programme proposé ici sert à définir la région de ObjectProperty et les ObjectProperty de class. La classe « Bactérie » est une sous-classe de « Agent infectieux ». Il existe quatre propriétés de « Bactérie » (Figure 21). Son ObjectProperty "occuresIn" correspond à la "partie # affectée" (associations : est-situé). Les ObjectProperty "hasSymptômes", "isCausedBy", " bearerOf " sont en relation avec le région "# Symptômes" (Confirmer le symptôme-diagnostic), "# Causes" (associations : le causalité, provoque, cause,…) et "# Rôle lié à la santé" (Exposer l'âge, le sexe, l'identité – femme enceinte, cardiaque,... et d'autres informations personnelles du patient) respectivement.

```
1   <ow:l Class rd:f ID="Bactéries">
2   <rdfs: subClassOf rd:f resource="#Agent infectieux" />
3     <ow:l ObjectProperty rd:f ID="occuresIn">
4       <rdfs: domain rd:f resource="#Bactéries" />
5       <rdfs: range rd:f resource="#partie affectée" />
6     </ow:l ObjectProperty>
7     <ow:l ObjectProperty rd:f ID="hasSymptômes">
8       <rdfs: domain rd:f resource="#Bactéries" />
9       <rdfs: range rd:f resource="#Symptômes" />
10    </ow:l ObjectProperty>
11    <ow:l ObjectProperty rd:f ID="isCausedBy">
12      <rdfs: domain rd:f resource="#Bactéries" />
13      <rdfs: range rd:f resource="# Causes" />
14    </ow:l ObjectProperty>
15    <ow:l ObjectProperty rd:f ID=" bearerOf ">
16      <rdfs: domain rd:f resource="#Bactéries" />
17      <rdfs: range rd:f resource="# Rôle lié à la santé" />
18    </ow:l ObjectProperty>
19  </ow:l Class>
```

Figure 21 Fonction de OWL: annoter les classes de l'ontologie

Ce sous-chapitre présente seulement le détail de OWL concerné par notre recherche. Pour une explication plus complète du rôle et des composants de l'élément OWL, le lecteur peut se référer à la recommandation du W3C « *OWL Web Ontology Language Reference* » (Dean, M. et al. 2004).

4.2 UML (*Unified Modeling Language*)

En informatique, UML (*Unified Modeling Language*) ou Langage de modélisation unifié, est un langage de modélisation graphique à base de pictogrammes. Il est utilisé en développement logiciel, en conception orientée objet et pour les projets logiciels. (UML 2001)

Unified Modeling Language (UML) (Rumbaugh, J. et al. 1999) est une langue de l'*Object Management Group* (OMG) (Soley, R. M. 1992), avec son *Object Constraint Language* (OCL) (Warmer, J. & Kleppe, A. 2003) associé. La sémantique pour OCL,

qui comprend nécessairement une sémantique pour les diagrammes de classes, a été proposée par (Richters, M. & Gogolla, M. 1998) et (Hamie, A. et al. 1998).

La comparaison entre UML et OWL est détaillée dans l'article (IBM 2003). Les concepts similaires de UML et OWL sont présentés ci-après.

UML	OWL
Class	Class
Instance	Individual
Attribute, Binary association	Property
Subclass	Subclass
N-ary Association, Association class	Class, Property
Enumeration	oneOf
Navigable, Non-navigable	Domain, Range
Multiplicity	minCardinality, maxCardinality, inverseOf
Package	Ontology

Tableau 15 Concepts similaires dans UML et OWL

Il faut fournir l'architecture globale de l'application sous la forme des diagrammes UML : diagramme de classes, diagramme d'activité, diagramme de cas d'utilisation, diagrammes de séquences et, éventuellement, diagramme d'états-transitions, ce qui permet de mieux comprendre l'organisation des package Java et de vérifier que l'architecture de notre application est correcte ; application que nous présentons dans le cadre de cette thèse.

Il y a 13 diagrammes en UML 2.0, ces diagrammes sont dépendants hiérarchiquement et se complètent, afin de permettre la modélisation d'un projet tout au long de son cycle de vie.

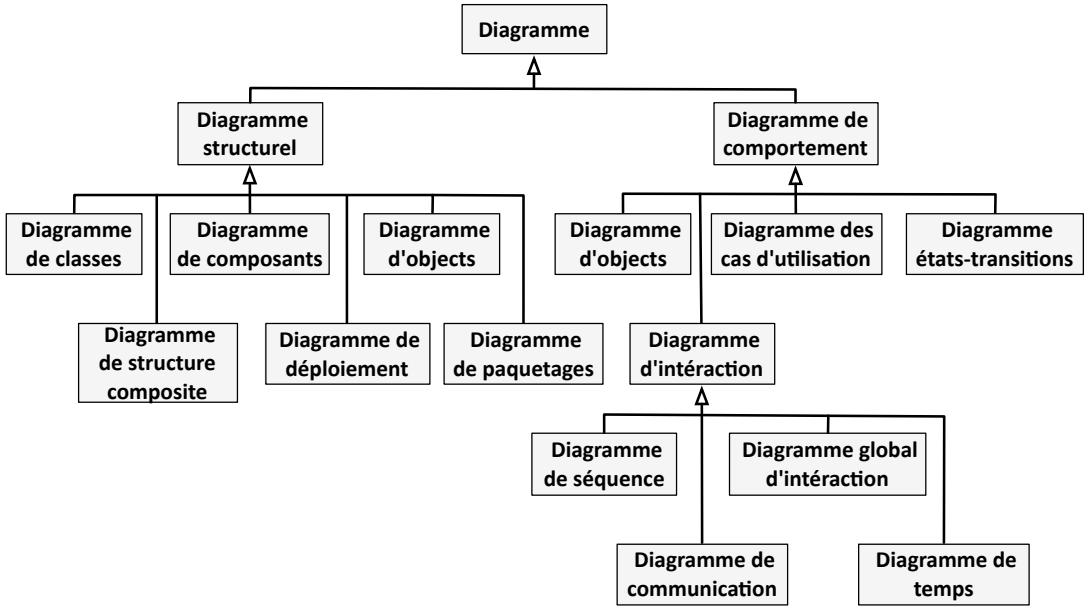


Figure 22 Diagrammes de l'UML (Pender, T. et al. 2003)

Des descriptions des diagrammes UML, et leurs étapes du cycle en V correspondantes, sont proposées ci-après :

Étape du cycle en V	Diagramme	Description
Spécification	Diagramme d'activité	Il permet de décrire, sous forme de flux ou d'enchaînements d'activités, le comportement du système ou de ses composants.
	Diagrammes de séquence	Représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.
	Diagramme de cas d'utilisation	Il permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est à dire toutes les fonctionnalités que doit fournir le système.
Conception détaillée	Diagramme de classes	Il représente les classes intervenant dans le système.
Autre	Diagramme états-transitions	Il permet de décrire sous forme de machine à états finis le comportement du système ou de ses composants.

Tableau 16 Descriptions des diagrammes UML et leurs étapes du cycle en V correspondants

Les avantages d'utiliser UML et OCL sont les suivantes (Cranefield, S. & Purvis, M. 1999):

- UML possède une communauté d'utilisateurs très vaste et en expansion. UML peut accepter la technologie des systèmes d'information distribués entre les

communautés d'utilisateurs finaux nouvelles parce que les utilisateurs d'infrastructures des systèmes d'information sont plus susceptibles d'être familiarisés avec UML qu'avec d'autres langages comme KIF ou d'autres logiques de description.

Si l'on compare UML à KIF, KIF peut fournir toute la puissance expressive de la logique sous-jacente du premier ordre, mais pour raisonner sur les ontologies avec KIF, cela requiert des capacités théorème-preuve générales. En revanche, la description logique fournit un langage beaucoup plus structuré et moins général, pour décrire des ontologies, et donc des inférences spécialisées peuvent être effectuées sur des ontologies décrites, en utilisant la description logique (Nebel, B. 1990).

- UML a une représentation graphique standard pour les modèles. Cette représentation graphique est importante pour permettre aux utilisateurs de systèmes d'information distribués de parcourir une ontologie et de découvrir des concepts qui puissent apparaître dans leurs requêtes. En revanche, la logique de description comporte une syntaxe linéaire mais aucune représentation graphique standard.
- L'*Object Constraint Language* (OCL) est puissant. Il permet l'expression de contraintes qui ne peuvent être décrites par la description logique.

Nous devons savoir quels types d'inférence sont souhaitables et possibles pour soutenir des ontologies représentées dans UML. Ceci dépend, en partie, du type de système d'ontologies choisis. Nous ne suggérons pas que UML puisse être considéré comme une alternative à la description logique dans toutes les situations. Par exemple, bien que (Haimowitz, I. J. 1988) aient trouvé les outils de raisonnement de la connaissance inadéquats pour la modélisation d'ontologies dans un système expert médical, UML ne peut pas fournir une alternative simple pour les modélisations d'ontologies. Il serait soit nécessaire d'exprimer la sémantique des diagrammes de classes UML dans la logique du système déductif ou bien, un système hybride devrait être construit de sorte que les déductions possibles, en raison de la sémantique (implicites) de l'ontologie, puissent être intégrés au raisonnement déductif explicite du système.

Pour les systèmes où le type de raisonnement requis pour les ontologies peut être limité à répondre à des questions spécialisées spécifiques, UML est un candidat plus

solide. Il y a plusieurs étapes au cours desquelles des inférences particulières sur les ontologies peuvent être nécessaires :

- La construction initiale de l'ontologie : c'est le domaine bien soutenu par les logiques de description qui prévoient des mécanismes d'inférence pour vérifier l'intégrité de la construction de l'ontologie.
- Aider les utilisateurs à former des requêtes au sein d'une ontologie. UML est utile comme système d'aide aux utilisateurs, pour découvrir les concepts qui peuvent apparaître dans les requêtes. Par exemple : la recherche et l'affichage de tous les plus courts chemins de navigation, allant d'une classe donnée, à des classes ou à des attributs, avec des noms correspondant à un motif fourni par l'utilisateur.
- Décomposer et traduire les requêtes exprimées dans une ou plusieurs ontologies de domaine de haut niveau, ou dans le cadre de requêtes impliquant des ontologies pour des sources de données spécifiques. Cela nécessite, à la fois, une représentation des relations entre ontologies et un mécanisme de raisonnement à leur sujet.

Étude de cas : processus clinique

Cette étude se focalise sur la description du processus clinique à travers l'utilisation d'un diagramme d'analyse associé à un diagramme de classe. Le diagramme d'analyse est basé sur une description simple du processus clinique (diagnostic – pronostic – traitement – suivi thérapeutique). Le modèle est composé d'un diagramme de classe décrivant les informations relatives au patient (Figure 23), et deux diagrammes d'activités (un diagramme principal qui décrit le flux global du processus clinique (Figure 24) et un diagramme axé sur la gestion des complications observées durant le suivi thérapeutique (Figure 25).

UML définit plusieurs types de diagrammes qui peuvent être utilisés pour modéliser le comportement statique et dynamique d'un système. Le diagramme de classes est utilisé pour décrire les classes et leurs relations dans le domaine. Dans cette section, nous décrivons la notation UML utilisée dans la figure 23.

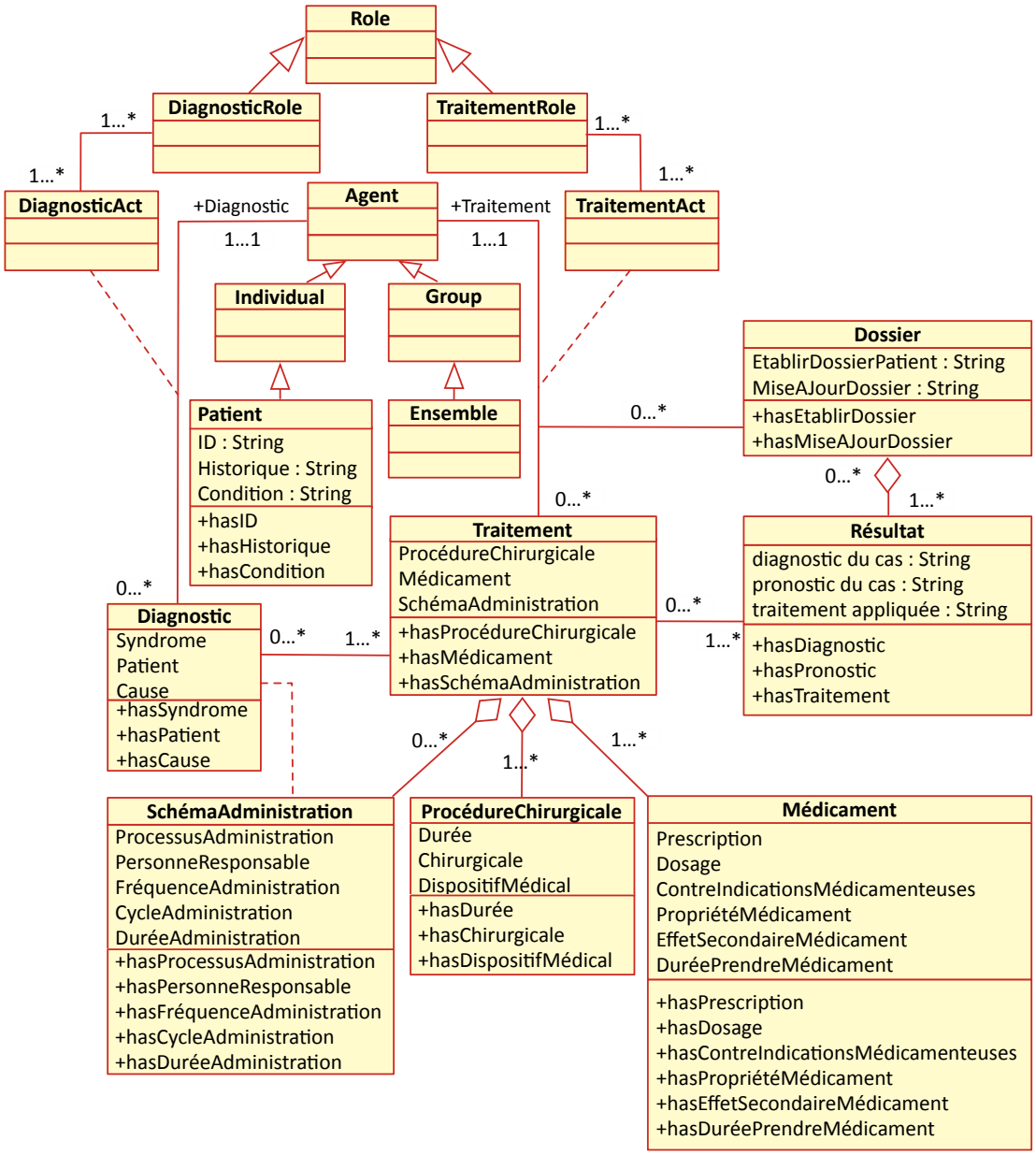


Figure 23 Diagramme de classes décrivant la structure des données relatives au patient

Dans un diagramme de classes, les classes sont représentées par trois parties : le nom de la classe, les attributs de la classe (spécifiés par leur nom, le type et la visibilité) et les opérations de la classe (spécifiées par nom, liste d'arguments, le type de retour et la visibilité). Aux fins de représenter les ontologies, tous les attributs peuvent être considérés comme ayant une visibilité publique. Dans ce cas, une ontologie est une vue publique partagée d'un domaine.

La figure 23 montre trois types importants de relation qui peuvent être utilisés entre les classes :

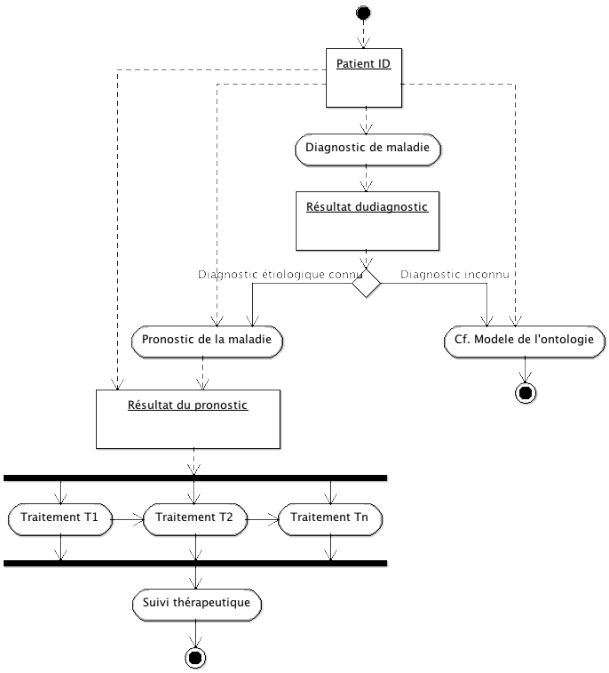
- *Généralisation*, représentée par des lignes avec de grandes têtes de flèches creuses pointant vers la superclasse (par exemple, la classe « Individu » et la classe « Patient ») ;
- *Association*, représentée par des lignes solides entre deux classes avec des extrémités nommées optionnellement, ou des rôles (par exemple la classe Diagnostic est associée à la classe Traitement au milieu de la figure) ;
- *Agrégation* : c'est une association (par exemple la classe Résultat sur la droite de la figure entretient une relation d'agrégation avec la classe Dossier). UML comporte un type d'agrégation encore plus fort (agrégation composite, notée par un diamant noir solide) qui implique la propriété des parties liées à l'agrégat. Nous ne faisons pas de distinction entre ces deux types d'agrégation dans nos ontologies à l'heure actuelle.

Les extrémités des relations d'agrégation peuvent être annotées avec des indicateurs de multiplicité donnant une série de numéros (e.g. '*' représente l'infini). Ces numéros indiquent qu'à cette extrémité de la relation, les instances de la classe peuvent être associées à chaque instance de la classe de l'autre extrémité.

Plusieurs autres constructions de UML sont utilisées dans la figure 23. La classe TraitementAct, dans l'angle en haut à droite, est une classe d'association : une classe attachée à une association. Celles-ci peuvent être utilisées pour les associations qui nécessitent des attributs.

Le processus principal (Figure 24) débute par un état initial requérant des informations relatives au patient (données relatives au diagramme de classe). L'objet patient possède des données sortantes vers chacune des activités modélisées tout au long du processus. La première activité consiste à requérir les informations relatives au diagnostic. La valeur résultante est stockée dans l'objet résultat du diagnostic. Dépendant du diagnostic, l'état du patient est soit positif soit négatif, un losange de décision dirige vers les activités associées. Dans le cas où le diagnostic est positif, un pronostic est établi et plusieurs traitements possibles en découlent. Du suivi thérapeutique découlent plusieurs cas en fonction de l'évolution de l'état du patient : le diagramme d'activité de gestion des complications décrit le flux d'activités correspondant (Figure 25). Au cours du suivi thérapeutique, plusieurs cas peuvent être observés : le traitement n'est pas approprié ; le diagnostic n'est pas conforme ; le bilan est normal. Des sorties correspondent à chacun des cas, respectivement S1, S3 et S2. Pour les sorties S1 et S3, des renvois au diagramme d'activité principal sont

nécessaire (respectivement, ils renvoient au pronostic de la maladie ou à son diagnostic).



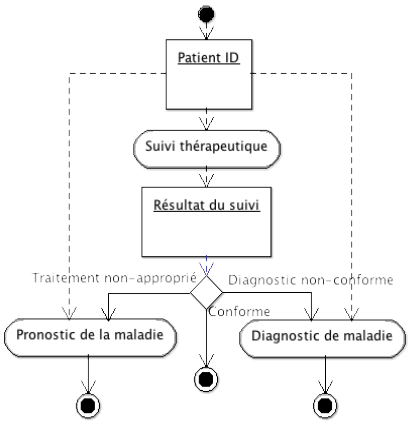


Figure 24 Diagramme d'activité principal du processus clinique

Figure 25 Diagramme d'activité de gestion des complications observées durant le suivi thérapeutique

4.3 XML (*Extended Markup Language*)

Le XML (*Extended Markup Language*) (Bray, T. et al. 1997) est proposé par le W3C (*World Wide Web Consortium*) pour décrire la structure arborescente de documents, à l'aide d'un système de balises pour marquer les éléments qui composent la structure et les relations entre eux. Ce langage a été conçu pour faciliter l'échange, le partage et la publication de données sur des réseaux basés sur Internet. Son format est de plus en plus utilisé dans les applications commerciales.

Les documents XML contiennent leur propre structure, son analyseur syntaxique peut accéder à cette structure et récupérer les données. La syntaxe des langues utilise les balises de départ et les balises de fin, pour marquer les éléments d'information (par exemple <office> et </office> dans la figure 26). Les éléments peuvent être enrichis en attachant des paires nom-valeur appelées attributs (par exemple, <address country="FR"> dans la figure 26).

La syntaxe simple est facile à traiter par la machine, compréhensible pour les humains. XML est extensible parce que l'on peut définir de nouvelles balises et les

noms des attributs, afin de paramétrer ou qualifier sémantiquement les données et documents. Les structures peuvent être imbriquées à n'importe quel niveau de complexité, de sorte que les schémas de bases de données ou des hiérarchies orientées objet peuvent être représentés.

```
<adresse professionnelle>
 <name>Ying SHEN</name>
 <address country="FR">
  <lab>MoDyCo<lab>
  <office>Bureau 401B<office>
  <university>ParisX<university>
  <postal>92001</postal>
 </address>
 <email>ying.shen@u-paris10.fr</email>
 <last_contact>2013-11-13</last_contact>
</adresse_professionnelle>
```

Figure 26 Exemple XML

Le XML Schéma (**XML-S**), une nouvelle recommandation du W3C, est un langage XML pour définir les grammaires d'arbres caractérisant des documents (notion de validité syntaxique). Le XML Schéma ne nous permet pas de décrire la sémantique, mais il permet de saisir des documents et des données. La figure 27 montre un exemple avec le numéro de téléphone et la date du dernier contact avec contrainte par un type.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="adresse professionnelle">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
     <xs:element name="address" minOccurs="0" maxOccurs="1">
      <xs:complexType>
       <xs:attribute name="country" type="xs:string" use="required"/>
       <xs:sequence>
        <xs:element name="lab" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="office" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="university" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="postal" type="xs:string" minOccurs="1" maxOccurs="1"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="email" type="xs:string" minOccurs="1" maxOccurs="1" />
     <xs:element name="last_contact" type="xs:date" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 27 Exemple de XML Schéma

XML est utilisé pour structurer un document définissant ses propres lignes directrices nécessaires et sans tenir compte de l'importance de cette structure et des systèmes d'information qu'elle exploitera. W3C propose des normes telles que **XPath**

et **XQuery**, pour naviguer et interroger l'arborescence du document XML, et la représentation des données de documents XML.

Le **XSL** (*Extensible Stylesheet Language*) (Adler, S. et al. 2001) est défini avec le formalisme d'XML et permet de générer les feuilles de style, qui ont des capacités de manipulation de documents au-delà du style. Un document peut être transformé en d'autres documents afin de l'adapter aux besoins. XSL est constitué de deux formalismes : XSL FO (*Formatting Objects*) et XSLT (*XSL Transformation*). XSLT (*XSL Transformation*) permet la transformation de l'arbre XML, et la génération de XML, HTML, et les fichiers texte.

```
<xsl:template match="/">
<HTML>
 <HEAD>
  <TITLE>email</TITLE>
 </HEAD>
 <BODY>
   <xsl:apply-templates />
 </BODY>
</HTML>
</xsl:template>

<xsl:template match="adresse professionnelle">
 <xsl:value-of select='name'>
 <xsl:text>: </xsl:text>
 <xsl:value-of select='email'><BR/>
</xsl:template>
```

Figure 28 Exemple de XSLT

```
<HTML>
 <HEAD>
  <TITLE>email</TITLE>
 </HEAD>
 <BODY>
  <p>Ying SHEN: ying.shen@u-paris10.fr</p>
 </BODY>
</HTML>
```

Figure 29 Résultat de HTML

4.4 Logiques de description

Au cours des dernières années, les recherches sur les systèmes basés sur la connaissance et les langages pour la représentation des connaissances ont émergé. Depuis, un grand nombre d'études ont été proposées sur la logique des prédicats, les réseaux sémantiques et les cadres langues. Ces langages se fondent sur les formalismes de représentation suivants : les frames, les graphes conceptuels et les logiques de description. Les logiques de description (DL) (Baader, F. 2003) (Grosof, B.N. et al. 2003) sont les plus appropriées pour la représentation des ontologies formalismes. Les logiques de description sont des formalismes liés à des réseaux sémantiques et à des systèmes de châssis réservés à la représentation de connaissances (Todirascu, A. et al. 2000). Ils fonctionnent sur de puissants mécanismes d'inférence, pour la déduction automatique, le raisonnement, etc.

Les logiques de description sont utilisées pour de nombreuses applications, telles que le web sémantique (représentation d'ontologies (Baader, F. et al. 2005), et la recherche d'informations basées sur la logique), le médecine/bioinformatique (représenter et gérer des ontologies biomédicales), le Traitement automatique des langues (TAL) (Fehrer, D. et al. 1994), l'ingénierie de processus (représenter des descriptions de service), l'ingénierie de la connaissance (représenter des ontologies) et l'ingénierie logicielle (représenter la sémantique des diagrammes de classe UML) (Berardi, D. et al. 2001).

Les logiques de description permettent de représenter les connaissances d'un domaine d'application d'une manière formelle et structurée, en utilisant les notions de concept (classes d'individus), de rôle (relations binaires entre classes) et d'individu. Un concept et un rôle possèdent une description structurée définie à partir d'un certain ensemble de constructeurs (Chaabani, M. et al. 2010).

4.4.1 Ontologies et Logiques de description

Les Logiques de Description sont développées pour décrire des connaissances ontologiques. Il existe trois primitives de base pour la syntaxe de ces logiques :

i. Les noms de concepts, e.g. médicaments, personnes.
ii. Les noms de rôles, comme *hasMaladieCoexistant*.
iii. Le concept d'individu (les éléments d'un concept).

Avec les concepts, les rôles et les individus des logiques de description, on peut expliquer l'approche ontologique qui décrit les individus d'un domaine, en requérant des définitions générales d'individus et les relations logiques que les individus ou catégories peuvent entretenir entre eux.

La relation de subsomption des logiques de description est utilisée pour organiser les concepts et les rôles des hiérarchies qui opèrent sur les processus de classification et l'instanciation : ces opérations sont à la base d'un raisonnement sur les descriptions ou le raisonnement terminologique. La classification s'applique aux concepts et aux rôles appropriés et détermine la position d'un concept et d'un rôle dans leurs hiérarchies respectives, tandis que l'instanciation permet de trouver les concepts dont un individu est susceptible d'être une instance.

Comme nous l'avons présenté dans le dernier sous-chapitre, le Web sémantique est organisé en une architecture en couches, XML produit la couche de transport syntaxique, tandis que OWL est considéré comme un standard de langage d'ontologie de W3C basé sur les logiques de description. Les axiomes et les constructeurs d'OWL sont restreints pour que le raisonnement soit décidable.

4.4.2 Syntaxe des logiques de description

Les logiques de description ont une base commune qui décrit les propriétés des valeurs. Cette base commune peut s'enrichir de différentes extensions :

Opérateur DL	**Expression logique**	**Interprétation DL**
C = (SOME Rel D)	$\exists \mathrm{Re}l.D$	Il existe au moins un objet appartenant à D, lié par une relation Rel avec les objets de C
C= (ALL Rel D)	$\forall \mathrm{Re}l.D$	Limite le co-domaine de la relation Rel
C = (AND D1 D2)	$D_1 \cap D_2$	Conjonction des descriptions conceptuelles
C= (OR D1 D2)	$D_1 \cup D_2$	Disjonction de descriptions conceptuelles
C = NOT D	$\neg D$	Complément d'un concept
C = ≥ *nRel*.D	$\exists y1 \ldots yn\left(1 \le i \le n, R\left(x, yi\right) \cap D\left(yi\right)\right)$	Il existe au moins n objets de D par rapport (Rel) à C

Tableau 17 Expression logique de DL

Exemple

(define-concept Woman (AND Human (ALL isFemale Female)(All hasAge Age)))

La définition du concept « Woman » est interprété comme : une femme est une femelle humaine dont l'âge est ≥ à 18 ans. Pour chaque instance du concept « Woman », toutes les instances liées par hasAge doivent être un individu du concept « Age ».

Les logiques de description structurent le domaine des connaissances sur deux niveaux :

- **Niveau Terminologique (T-Box)** contient les axiomes qui définissent les classes d'objets du domaine (nommés concepts), avec leurs propriétés et leurs relations (rôles) avec d'autres objets, le service de raisonnement principal disponible en T-Box est une subsomption entre deux concepts, la détermination dont le concept est le plus général.

- **Niveau Assertionnel (A-Box)** contient des objets et des classes abstraites (les individus). Le test d'instanciation détecte quelle description conceptuelle est instanciée par une instance donnée ; l'inférence d'extraction permet d'extraire les individus qui appartiennent à un concept donné. La ABox doit être associée à une TBox, car les assertions s'expriment en termes de concepts et de rôles de la TBox.

L'exemple du tableau 18 comprend les individus nommés suivants : Martin et Alice.

TBox	**ABox**
$Femme \equiv Humain \cap Femelle$ $Homme \equiv Humain \cap \neg Femelle$ $Femelle \subseteq \forall \cap \neg Male$ $Male \subseteq \forall \cap \neg Femelle$ $Parent \subseteq Humain \cap aEnfant$ $Mere \subseteq Parent \cap Femme$ $Pere \subseteq Parent \cap Homme$	Alice (Femme) Martin (Homme) epouse_de (Martin, Alice) aEnfant (Martin, Alice)

Tableau 18 Base de connaissances composées d'une TBox et d'une ABox

4.4.3 Simplification des requêtes

Des modules TAL interprètent les requêtes de l'utilisateur, pour la représentation sémantique. Les concepts identifiés dans la requête sont utilisés pour récupérer les instances.

La raison d'utiliser DL comme formalisme de représentation des connaissances de domaine et les méthodes TAL peu profondes, c'est sa capacité à gérer des données incomplètes ou erronées, basées sur les caractéristiques suivantes :

I. Identification des concepts pertinents sans identification rigoureuse des structures syntaxiques :

Supposons qu'une erreur de syntaxe s'est produite dans la requête :

*« *Les patient avoir un diarrhée virale »*

Les morceaux sémantiques sont : «les patients» et « un diarrhée virale ». Les concepts associés à ces morceaux sont **Patient** et **DiarrhéeVirale**. L'erreur n'influence pas la compréhension de la requête.

II. Raisonnement sur des données de domaine pour compléter l'information. Les incohérences sémantiques sont détectées par la vérification des Logiques de Description :

« Les patients ont la diarrhée bactérienne bénigne avec la septicémie. »

Cette requête vise à chercher les cas de patients qui ont une diarrhée bactérienne et la septicémie en même temps. En fait, la septicémie et la diarrhée bactérienne sont possibles ensemble. La description conceptuelle attribuée à cette expression est :

(AND Patient (SOME hasDisease DiarrhéeBactérienneBénigne) (ALL hasMaladieCoexistant Septicémie)

La représentation de la notion Diarrhée Bactérienne Bénigne est :

(define-concept DiarrhéeBactérienneBénigne (AND Disease (ALL hasSyndrome Immunodépression) (ALL hasSyndrome InsuffisanceRénale) (ALL hasSyndrome **Valvulopathie**))

Mais la représentation de la notion Septicémie est :

(define-concept Septicémie (AND Disease (ALL hasSyndrome (NOT Immunodépression)) (ALL hasSyndrome (NOT InsuffisanceRénale)) (ALL hasSyndrome **(NOT Valvulopathie)**)))

Selon ces deux représentations de la notion, on peut trouver que la requête est non conforme. L'information en retour par le système est vide.

S'il y a, dans le dictionnaire, un texte ontologique correspondant à la requête d'entrée, par exemple, l'utilisateur demande :

"Le patient a une diarrhée à germe invasif mais n'a pas de bactéries ou de parasite sévère."

(AND Patient (SOME hasDisease DiarrhéeGermeInvasif) (ALL hasSyndrome (NOT BactérieSévère)) (ALL hasSyndrome (NOT ParasiteSévère)))

Dans ce qui suit, le système revient à l'ensemble des documents contenant des instances de ce concept.

4.5 Prototype

Prototype est utilisé pour tester les différents concepts et exigences et montrer aux clients les fonctions que l'on veut mettre en œuvre. Lorsque le client a donné son accord, le développement suit souvent un cycle de vie linéaire.

Notre travail s'inscrit plus particulièrement dans une approche informatique manipulant des prototypes. Cette approche de représentation ne préprogramme ni de manière figée, ni toutes les opérations pour un traitement automatique de la construction du sens, ni les processus de représentation des unités lexicales manipulées.

4.5.1 Cadre de représentation

Système prototype pour la construction d'ontologie automatique

Le prototype du système intègre plusieurs modules de traitement du langage naturel, ainsi que certains modules de l'inférence logique. Pour tester le prototype, nous allons utiliser une partie expérimentale du corpus français, des articles de journaux et articles TAL. La plupart des exemples présentés sont extraits du corpus de diarrhée aiguë et cancer de l'estomac.

Les exigences générales et spécifiques doivent être remplies et identifiés par le système de prototype, ainsi que leur description détaillée. Les **exigences générales** suivantes ont été définies pour le cadre de la construction de l'ontologie :

- Construire les systèmes multi-agents.
- Extraire des termes d'un corpus de textes.
- Extraire des associations d'un corpus de textes.
- Construire une ontologie avec les termes et les relations sélectionnés selon la fréquence des mots.

À part ces exigences générales, d'autres **exigences spécifiques** ont été identifiées :

- Ajouter les synonymes générés et les propres synonymes de l'utilisateur à des concepts dans les systèmes multi-agents,
- Générer une liste de concepts et des associations dans le système multi-agents,
- Mettre à jour le système multi-agents.

Dans cette thèse, le système prototype permet de réaliser toutes les étapes du cadre de la construction d'ontologie automatiquement, il permet donc la construction de l'ontologie en utilisant un corpus de textes et des systèmes multi-agents.

4.5.2 Méthodologie

En ce qui concerne la description détaillée du cadre de description de l'ontologie, une idée claire des fonctionnalités requises pour le prototype commence à apparaitre. Par conséquent, un processus de développement de logiciel a été choisi.

La description détaillée des exigences établies au cours des étapes précédentes suppose que chaque exigence a été analysée et divisée en trois modules (ou entités de conception) : **Modèle de l'ontologie**, **Modèle de Traitement** et Modèle d'interconnexion. Afin de faciliter la conception et la description de l'ensemble du système, plusieurs composants sont mis en œuvre en parallèle. Toutes les fonctionnalités du modèle ont été testées individuellement après avoir été codées. Enfin, les fonctionnalités ont été réunies et liées à l'interface utilisateur graphique.

Le Prototype est utilisé pour tester les différents concepts et exigences et montrer aux clients les fonctions que l'on veut mettre en œuvre. Lorsque le client a donné son accord, le développement suit souvent un cycle de vie linéaire. (Colloc J. 2011)

Il y a deux approches opérables pour construire les ontologies : soit en mode « Protégé », soit avec un langage plus natif en utilisant OWL ou XML, en coopération avec un environnement Java.

L'avantage de « Protégé », c'est qu'il existe déjà tout un environnement et que l'on peut voir les modes de communication et interactions entre les bases de connaissances et les ontologies. Par exemple, certaines fonctionnalités Protégé-OWL qui peuvent être utilisés pour la construction de l'ontologie, sont énumérées ci-dessous, d'autres fonctionnalités peuvent être trouvées dans (Blomqvist, E. et al. 2006):

- Partitions de valeur : permettre la création d'une liste de concepts, et l'application de conditions sur les concepts de la partition.
- Matrice de restriction : permettre la création de restrictions existentielles sur un concept ou un groupe de concepts.
- Créer une classe : permettre d'ajouter un concept à l'ontologie.
- Classes disjointes : permettre la spécification d'un concept disjoint d'un autre.

- Propriétés : permettre la création d'associations entre deux concepts d'un modèle ou d'une ontologie.

Mais il faut nécessairement organiser une coopération entre le SMA et Protégé, ce qui est parfois difficile.

Dans cette thèse, nous utilisons OWL, UML et XML, en coopération avec un environnement Java. Notre système de prototypes est développé en Java, afin de faciliter la compatibilité avec des outils réutilisables. Le langage Java est l'un des langages programmation les plus commun pour l'extraction de termes, et il peut être aisé d'adapter de nouveaux composants à ses fonctionnalités. Avec la capacité de la représentation commune de la connaissance du domaine et son sens (en description logiques), Java fournit des mécanismes d'inférence puissants, capables de traiter des données erronées, incomplètes. Il intègre des techniques de traitement du langage naturel peu profonds pour les documents texte.

4.6 Le Raisonnement à partir de Cas (RàPC)

La mise en place d'acquisitions et la restitution de connaissance dans les systèmes d'aide à la décision cliniques peuvent être réalisées à travers la création automatique de pôles de comportements linguistiques. Parallèlement, le RàPC mémorise et restitue l'expérience et la connaissance de résolutions de problèmes similaires.

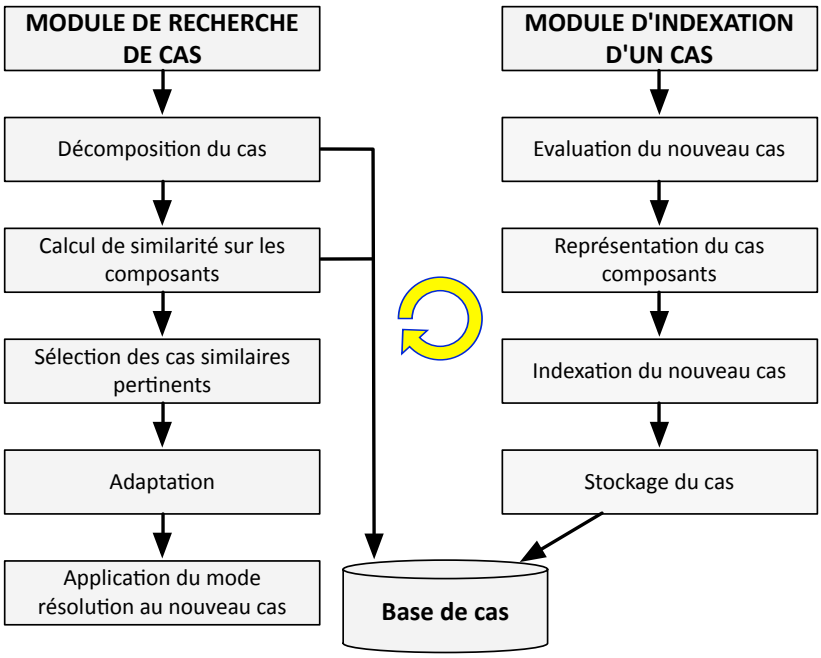


Figure 30 Raisonnement à partir de cas

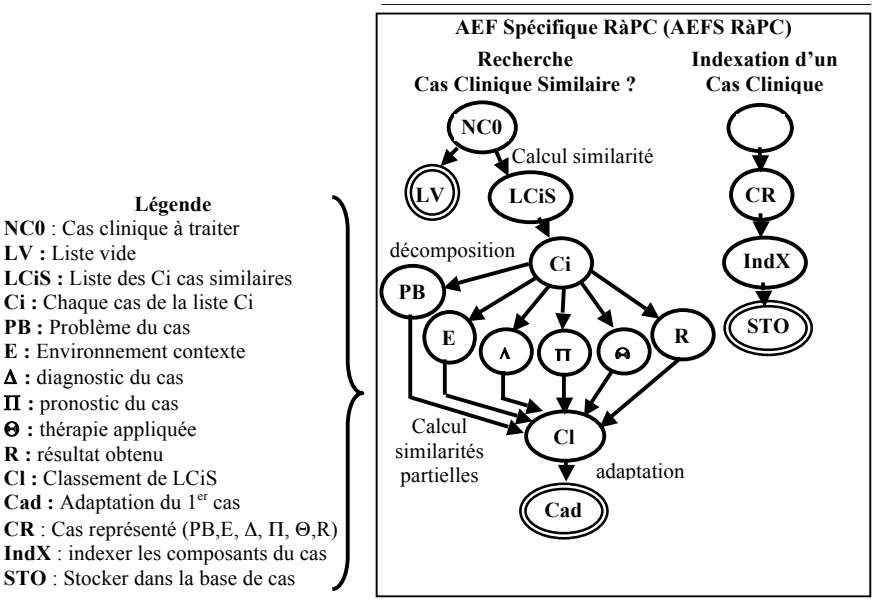


Figure 31 Raisonnement à partir de cas (RàPC) sur le domaine médical (Colloc J. & Sybord C. 2003)

Le raisonnement à partir de cas (RàPC) est un mode de résolution analogique des problèmes qui compare de nouveaux cas à partir de cas précédents indexés (Aamodt A. & Plaza E. 1994). Il y a certains inconvénients au RàPC. Par exemple, il demeure du domaine de la recherche et il est difficile actuellement à implanter sur un ordinateur. En outre, au départ, la base de cas est vide, il faut l'alimenter (modélisation et saisie) ou bien, il faut utiliser conjointement un autre système à base de connaissances. Mais le RàPC est encore utilisé dans notre travail parce que c'est un modèle mixte, qualitatif et quantitatif ; il prend en compte chaque nouveau problème comme un cas unique, pour résoudre de nouveaux cas par analogie avec d'anciens cas déjà traités (succès et échecs). Les cas sont comparés à l'aide d'une distance mathématique : par exemple, une fonction d'évaluation s'appuyant sur la logique floue, peut être utilisée. Les cas sont modélisés sous la forme d'objets et stockés dans une base de cas. Lorsqu'un nouveau cas se présente, les cas les plus similaires sont recherchés, sélectionnés et adaptés en espérant que ce qui fut approprié une fois le sera plusieurs fois (Gupta, U. G. 1994). Le détail des apports du RàPC dans le SMAAD est déjà présenté dans « [l']Architecture de système SMAAD et TACG », au chapitre 1.

Chapitre 5 ELABORATION DES ONTOLOGIES

Ce chapitre introduit un processus de représentation de connaissances linguistiques à partir du texte. Les éléments importants des textes sont filtrés pour extraire les substantifs (objets), les adjectifs (attributs ou propriétés des objets) et les verbes et adverbes (actions ou associations). Cette approche utilise des savoirs extraits sur des corpus et des associations remarquables pour construire automatiquement des structures qui décrivent ces savoirs puis les classent. Ce travail de représentation est conditionné par une phase d'extraction de savoirs sur corpus réalisé par l'analyse lexicale et l'analyse syntaxique. La démarche de représentation suivante utilisera la programmation en Java, OWL, les logiques de description et à prototypes pour la construction d'ontologies.

Ce chapitre décrit la mise en place d'acquisition de connaissance, à partir de prototypes et propose des outils pour la construction d'ontologies. Le prototypage aide à la construction automatique d'ontologies qui produit des résultats, l'ajustement de représentations et le classement des prototypes suivant leurs comportements linguistiques. Cette manière de faire est proche des méthodes MERISE et AGILE.

Ce chapitre enfin explique comment les comportements et les interactions entre modules sont générés et coordonnés dans le SMAAD. La génération de nouveaux savoirs nécessite d'affiner les processus auto-descriptifs des objets, de définir des règles pour la génération des nouveaux objets, et de créer un dictionnaire d'erreurs. Des outils de contrôle et de traçage évaluent les résultats construits : les représentations exploitées par un agent superviseur.

5.1 Introduction

Le dictionnaire ontologique est un modèle d'un type d'agent ontologique. La méta-connaissance sur les modèles décrit leurs concepts, leurs limites et circonscrit leur pouvoir de représentation. Dans les types abstraits de données, des axiomes et des tautologies décrivent le comportement du type et son emploi, lorsqu'il est instancié ou utilisé.

La principale idée pour la construction d'une ontologie est l'extraction de termes et de concepts, à partir d'un corpus de textes (un corpus de textes est un ensemble de fichiers texte), et en sélectionnant le module pour extraire des termes et des

associations nécessaires à la construction de l'ontologie. Les étapes suivantes sont destinées à construire une ontologie de manière semi-automatique :

i. Extraire les termes à partir d'un corpus de textes, et expliquer les termes utilisés pour décrire le déroulement clinique habituel,
ii. Extraire les associations à partir d'un corpus de textes,
iii. Construction d'ontologies avec hiérarchie conceptuelle,
iv. Instancier l'ontologie,
v. Procéder à l'extension de la hiérarchie par l'extraction de concepts supervisés, la génération de nouveaux savoirs (requêtes), pour l'ontologie, et la création des prototypes de mots.
vi. Indexer l'ontologie et établir les liens d'interactions entre le Dictionnaire Ontologique (DO) et le Module de Communication.

En cela, le développement de l'ontologie peut se décomposer selon les étapes suivantes : définition, implémentation, intégration et documentation.

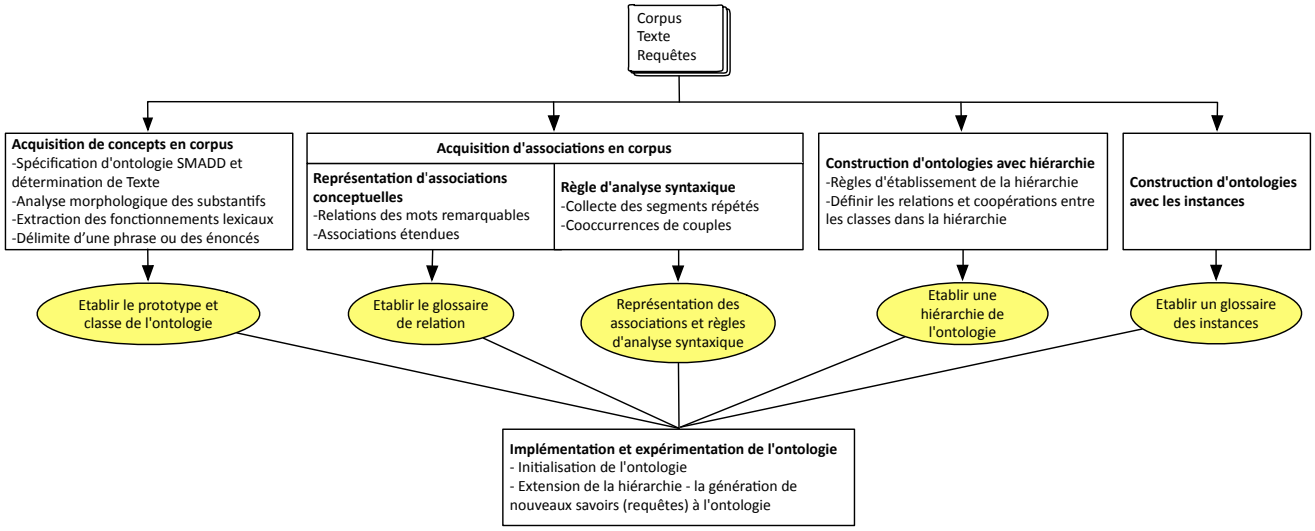


Figure 32 Processus de conception d'une ontologie à partir de textes médicaux

Après avoir traité le « processus de conception d'une ontologie à partir de textes médicaux » (figure 32), sous la supervision des ingénieurs des connaissances, trois principales procédures travaillent ensemble pour la conception d'une ontologie et l'élaboration de la réponse à une requête. Ces procédures comprennent :

- « Concevoir un SMA pour la construction d'ontologie » ;
- « Établir une ontologie semi-automatique » ;
- « Vérification et application des connaissances ».

La figure.33 présente l'agent superviseur de ces activités.

L’étape « Concevoir un SMA pour la construction d’ontologie » est supervisée par les ingénieurs des connaissances, elle vise à fournir les contraintes et les spécificités du SMA et des étapes de fonctionnement d’un agent superviseur. Ce processus a été présenté au chapitre.1.

Lorsque la procédure « établir une ontologie semi-automatique » a bien reçu les résultats de l’analyse linguistique, les contraintes et spécificités de coopération des agents et modules du domaine, elle réalise la construction ou bien renouvelle l’ontologie et transfère les connaissances élaborées à l’étape de « vérification et application des connaissances ».

Dans la procédure « vérification et application des connaissances », la modélisation est déroulée afin de présenter les processus de décision fondés sur la démarche clinique, et d’organiser les connaissances du cas d'étude. Cette étape permet de tirer des conclusions à partir de l’ontologie du domaine et répond la requête.

Les informations, en retour, fournies par les utilisateurs finaux, à travers des méthodes et interfaces, sont transférées à chaque module de l’agent afin d’améliorer leur coopération.

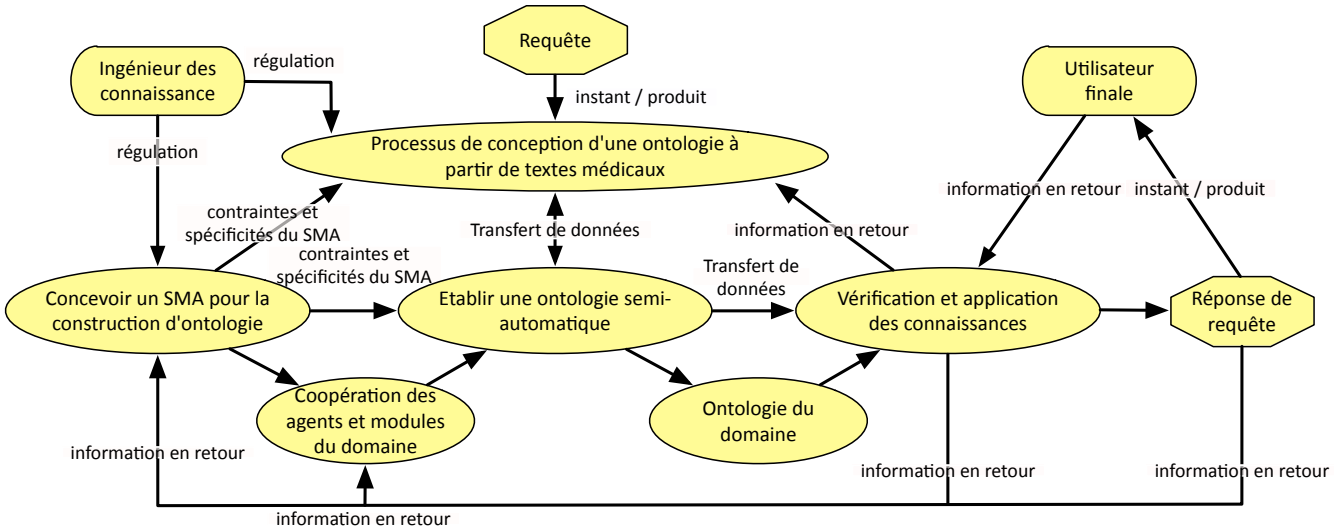


Figure 33 Agent superviseur de conception d’une ontologie et de réponse à une requête

Les questions d'acquisition de données, d'annotation de corpus, de représentation de connaissances et de constitution de bases de connaissances terminologiques et associations sont traitées dans la suite de la thèse.

5.2 Acquisition de concepts en corpus

L'extraction de termes constitue l'une des étapes de l'extraction d'information. L'extraction d'information nécessite d'obtenir l'information la plus pertinente, soit à partir de documents structurés, comme des pages HTML, soit à partir de documents non structurés comme des documents en langage naturel. L'extraction d'information intervient dans deux étapes de la construction de l'ontologie :

Il convient d'abord d'extraire des termes à partir d'un corpus de textes, puis d'extraire des associations entre ces termes. Les termes produits dans système multi-agents SMAAD sont des syntagmes nominaux, utilisés pour exprimer l'information obtenue de manière à faciliter la mise en évidence des relations existant entre eux.

La phase d'acquisition des données linguistiques consiste à collecter les données nécessaires à la représentation. Trois types d'objets linguistiques sont distingués : les dénominations, les instances et les relations. Du point de vue de leurs formes, les dénominations sont des substantifs simples ou des syntagmes nominaux figés ; les instances sont des dénominations au sein d'un syntagme nominal et les relations sont *a priori* contenues dans les autres signes, plus spécifiquement les verbes et certaines conjonctions, prépositions, ou adverbes.

Diverses approches d'analyse interviennent dans l'identification des éléments d'information présents dans cette thèse. Il s'agit d'identifier les éléments eux-mêmes, c'est-à-dire les mots significatifs, au travers d'une analyse morphologique, lexicale et d'un « *parsing* » ; ensuite, les relations entre ces mots, grâce à l'analyse syntaxique ; enfin, l'analyse sémantique permet de connaître la signification des mots dans leur hiérarchie.

Les résultats finaux obtenus sont évalués par comparaison entre eux, avec le texte d'origine, souvent en termes de taux de rappel et de précision, ou encore par consultation d'experts du domaine ou d'ontologies existantes. Les résultats finaux servent à constituer des ontologies médicales dans cette thèse.

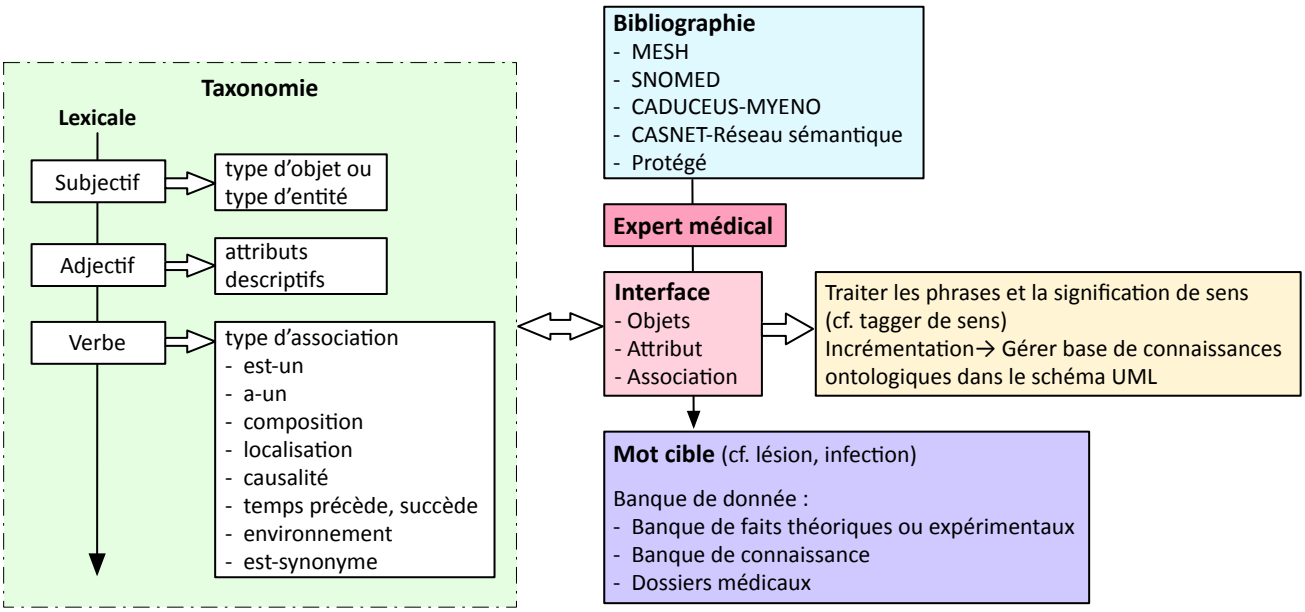


Figure 34 Analyse semi-automatique des mots cibles

En résumé, notre première étape met l'accent sur la détermination semi-automatique des mots cibles :

i. Importation d'éléments bibliographiques (cf. MESH, SNOMED, CASNET-Réseau sémantique, CADUCEUS-MYENO etc.)
ii. Extraire des synonymes du mot cible (**cf. 5.2.3**)
iii. Utiliser le *tagger* de sens, afin d'analyser les phrases et la signification de sens (**cf. 5.2.4**)
iv. Dérouler la méthode pour analyser l'objet, l'attribut et l'association.
    - Analyse lexicale – substantif : type d'objet ou type d'entité,
    - Analyse lexicale – adjectif : attributs descriptifs
    - Analyse lexicale – verbe : type d'association (est-un, a-un, composition, localisation, causalité, temps, précède et suit, environnement, est-synonyme etc.) (**cf. 5.3**)
v. Gérer la base de connaissances ontologiques dans le schéma UML

5.2.1 Spécification d'ontologie SMAAD et détermination de Texte

Qu'est-ce que "***texte***" ?

Des documents uniques à corpus étendus.

Qu'est-ce que "***l'information structurée***" ?

Des taxonomies de sujets destinées aux ontologies entières.

Le tableau, ci-dessous, vise à définir notre Ontologie SMAAD (Ontologie de Système multi-agents pour l'aide à la décision clinique), afin de déterminer le texte source pour l'extraction de données terminologiques :

Ontologie	Ontologie SMAAD (Ontologie de Système multi-agents pour l'aide à la décision clinique)
Objectifs	Aider à la décision clinique dans le système multi-agents avec les dictionnaires et les corpus existants.
Source	Les noms de maladies, l'acte et l'anatomie sont fondés sur le domaine riche en terminologies tel que CIM-10, le MeSH et SNOMED. À propos des noms de médicaments, nous utilisons les données du site médical Doctissimo. Pour mieux caractériser les symptômes et les maladies, on utilise le MeSH (Medical Subject Headings), thésaurus biomédical de référence. C'est un outil d'indexation, de catalogage et d'interrogation des bases de données de la NLM (*National Library of Medicine*), notamment MEDLINE/PubMed. Un corpus d'unités thématiques sur la Diarrhée aiguë est aussi établi sur la base de documents français issus des corpus PubMed et Science Direct, car les corpus existants sur internet sont en anglais et peu ciblés. Le corpus établi est assez étendu et on le mémorise aux formats xml et txt pour l'analyse et la représentation des termes et associations.
Unité thématique	Diagnostic, pronostic, traitement et suivi-thérapeutique des maladies
Utilisateurs	Utilisateurs humains : professionnels de santé médicale, professionnels de santé paramédicaux, ingénieur des connaissances, auteurs ou annotateur de documents, linguistes, etc.
Scénario d'utilisation	- Répondre aux requêtes des utilisateurs finaux en interrogeant la base de connaissances. - Ajouter de nouveaux termes, synonymes, associations et textes, en étudiant automatiquement les données d'entrée - L'ingénieur des connaissances peut vérifier les schémas et les règles OWL - L'auteur ou l'annotateur de documents peuvent modifier les concepts, les associations et les synonymes ajoutés, heuristiques sélectionnées sous la supervision de l'ingénieur des connaissances. - Le système peut donc construire l'ontologie automatiquement, à partir des termes et associations extraits, synonymes entrés et heuristiques sélectionnées.

Tableau 19 Spécification d'ontologie SMAAD et détermination le Texte

Après avoir spécifié l'ontologie du SMAAD et déterminé le texte, il faut présenter les mots invariables pour délimiter les frontières. Les limites des syntagmes nominaux sont simples comme, par exemple, verbe, pronom, préposition, conjonctions de coordination, conjonctions de subordination, etc.

Prépositions
Ces mots de liaison introduisent un groupe nominal (ou un pronom), un verbe à l'infinitif ou un adverbe, en le faisant dépendre d'un autre mot de la phrase **Se situer dans l'espace** sous, entre, devant, derrière, dans, chez **Situer dans le temps** avant, après, vers, depuis, pendant, pour **Le mouvement** vers, à, jusqu'à/au, de, par
Conjonctions de coordination
Servent à réunir deux mots ou deux groupes de mots mais, ou, et, donc, or, ni, car
Conjonctions de subordination
Servent à réunir une proposition subordonnée à une principale **si, comme, quand, que, au moment, où, et les composés de que :** Lorsque, bien que, alors que, pour que, afin que, parce que, depuis que, aussitôt que, après que, avant que, en attendant que, jusqu'à ce que, pendant que, tandis que, puisque, sous prétexte que, d'autant que, au point que, de façon que, tant que, tellement que, de crainte que, de peur que, à moins que, suivant que, pourvu que, selon que, à supposer que

Tableau 20 Le Glossaire de mots invariables (Mots invariables 2013)

5.2.2 Analyse morphologique des substantifs

L'analyse morphologique s'intéresse à la formation des mots au travers des processus de flexion (marques de genre, nombre, de conjugaison...), dérivation (formation de vendeur à partir de vend-, vendre) et composition. (Daille, B. et al. 2002) Une analyse morphologique complète est réalisée, soit en utilisant des bases lexicales existantes, soit à l'aide de véritables systèmes d'analyse, plus à même de traiter les formes non répertoriées.

L’analyse morphologique reconnaît les formes, variantes d'un vocable donné. Il est utilisé pour générer des systèmes de racinisation. Il s'agit du premier processus de traitement du langage naturel avant le marquage des mots (*tagging*).

Les algorithmes de racinisation les plus connus sont ceux de (Lindberg, D.A. et al. 1993) et (Porter, M. F. 1980). Ces divers algorithmes procèdent en deux étapes : un pas de dé-suffixation qui consiste à ôter aux mots des terminaisons prédéfinies les plus longues possibles, et un pas de recodage qui ajoute aux racines obtenues des terminaisons prédéfinies. Il est important de noter que les racines fournies par l’algorithme de Porter ne sont pas forcément de véritables morphèmes. En extraction de connaissance, l’analyse morphologique la plus employée est la lemmatisation (cf. fiche lemmatisation) qui permet d’associer à une forme fléchie une forme conventionnelle ainsi que de calculer les traits flexionnels. (Daille, B. et al. 2002)

Pour simplifier les mots dans le texte, par exemple, les formes variantes du mot cible {contre-indiqué}, {contre-indiqués}, {contre-indiquées}, {contre-indiquée}, {contre-indiquent}, et {contre-indique} peuvent être présentés sous la forme de {contre-indiqu*}.

Afin de réaliser l’analyse morphologique en langage Java, il faut d’abord définir la classe `Morphologie` et sa sous-classe `Segmenteur` pour obtenir `Syntagme` avec la méthode `getMots()`. Après avoir collecté les combinaisons d’étiquettes avec la méthode `getCombinaisons()`, on doit définir les `Exceptions`.

La « Programme.1 Analyseur morphologique-Lemmatisation » est présenté en Annexe.4.

5.2.3 Extraction des fonctionnements lexicaux

L’analyse lexicale reconnaît la structure et les significations au niveau du mot. Elle sert à construire des listes de mots invariables et le thésaurus, et à détecter et marquer les types de mots (ou catégories) : nom propre, verbe, substantif, etc.

Certaines méthodes sont réutilisables et l’on peut les associer à d’autres, pour réduire l'incertitude. On peut spécifier deux types de méthodes : celle du dictionnaire et l'analyse lexicale.

Dictionnaires existants

Selon les engagements ontologiques, il existe des accords pour utiliser le vocabulaire d'une manière cohérente et uniforme, et il existe une connexion entre le

vocabulaire de l'ontologie et la signification des termes d'un tel vocabulaire. (Gruber, T. & Olsen, G. 1994) Un agent est conforme à une ontologie s'il « agit » en conformité avec les définitions. (Guarino, N. et al. 1994)

S'il existe un dictionnaire du domaine, nous pouvons utiliser les ressources terminologiques existantes afin de localiser les occurrences des termes dans un texte. Autrement dit, un ensemble de concepts est enregistré dans le dictionnaire et ensuite ce dictionnaire est réutilisé avec la méthode d'apprentissage d'informations pour extraire des termes.

Par exemple, « le Petit Larousse » est un dictionnaire encyclopédique de langue française des éditions Larousse. Dans certains domaines, il peut fournir une bonne partie des dénominations. D'autres dictionnaires peuvent être utilisés. L'utilisation du Littré ou du Robert (Nouveau Littré), dictionnaires qui existent en ligne, serait à retenir car Émile Littré, médecin lui-même, lexicologue et lexicographe remarquable, est à l'origine de la fixation de nombreux termes médicaux.

Notre « Programme.2 Synonyme » est détaillé dans l'annexe.4. Ce programme vise à extraire les synonymes de mot cible avec les classes tel que `StringTokenizer`, `synonymSet`, `HashSet<String>` etc. Prendre le mot « infection » par exemple, nous pouvons présenter les termes relatifs suivants.

fétidité, pestilence, puanteur, infestation, contagion, contamination

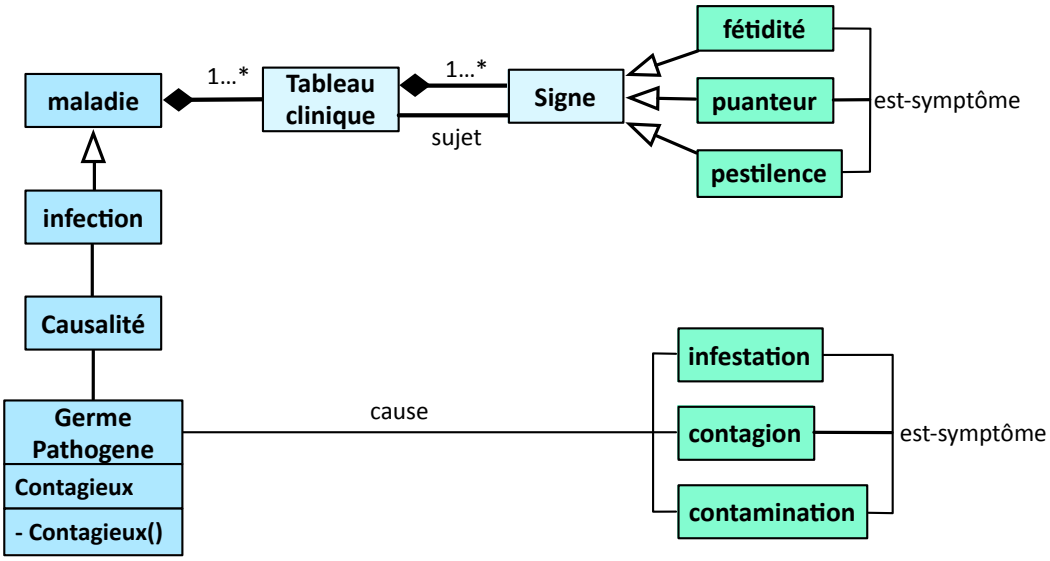


Figure 35 Relations entre le mot «infection» et ses synonymes

Bien que cette liste soit incomplète, il offre toujours de l'aide à notre analyse lexicale.

Analyse lexicale

Pour identifier des dénominations, il faut d'abord les lister, avant d'étudier le lexique des dénotations.

Il faut choisir des termes pertinents du domaine, procéder à la normalisation sémantique, en précisant les relations de similarités et de dis-similarité que chacun des concepts entretient avec d'autres, de même niveau d'analyse. Nous collectons des informations en classant les quatre phases cliniques essentielles : diagnostic, pronostic, traitement, suivi-thérapeutique.

En outre, dans le corpus, on collecte les chaînes de caractères entre crochets comprenant la dénomination à représenter, on repère les adjectifs, les dénominations-arguments, les substantifs qui ne sont pas des dénominations (par exemple *forme*), et les termes annotés, regroupés de manière adéquate. « [L]'Approche statistique » combine le programme pour représenter le caractère lexical d'un mot.

Ci-dessous l'exemple de « infection » extrait de 217 phrases contenant le mot «infection » dans notre corpus sur la diarrhée aiguë. Toutes les formes ont été mises au singulier.

INFECTION

- ***Arguments***

  - *Adjectifs*

    ~ naturelle, sévère, asthénie, aiguë, débutante, virale, fréquente, focale, principales, typique, répété, expérimentale, directement, opportuniste, possible

    ~ entérique, entérale, entéro-invasive

    ~ digestive

    ~ urinaire

    ~ parasitaire (bronchite, angine, sinusite, grippe, gastro-entérite, mycose, abcès, etc.)

    ~ nosocomiale

~ vénérienne

~ pédiatrique

~ respiratoire

~ bactérienne

~ choléri forme

~ plurimicrobienne

~ manuportée

~ chez le volontaire sain

- *Verbe*

~ faire [vomir l'enfant], causer [salmonella paratyphi], entraîner [l'activation du système nerveux entérique], témoigner [de la gravité de l'épidémie], indiquer, évoluer [sous forme d'épidémies hivernales]

~ agir sur [le fonctionnement], due à [virus], lier à [la présence d'une virémie], suite à.

~ être responsable de [altérations des fonctions de digestion]

- *Substantifs Nom*

~ de [entérocytes], de [entérocyte], du [cathéter veineux central], de [plaies cutanées visibles chez l'immunodéprimé],

~ à [Shigella], à [Escherichia coli], à [Vibrio cholerae], à [Campylobacter sp.], à [Salmonelles], à [Yersinia sp.], à [rotavirus]

- *Substantifs non-Nom*

- éliminer~, associer~, contre~, accompagner~, concerner~, diminuer~, favoriser~, limiter~, provoquer~, entraîner~,

- propagation de~, incidence de~, source de~, sévérité de~, physiopathologie de~, manifestation de~, pathogénie de~, origine de~, la durée de~, ce type de~, diagnostic de~, survenue de~, risque de~, émergence de~, acquisition de~,

- lors de~, en cas de~, entre le~,

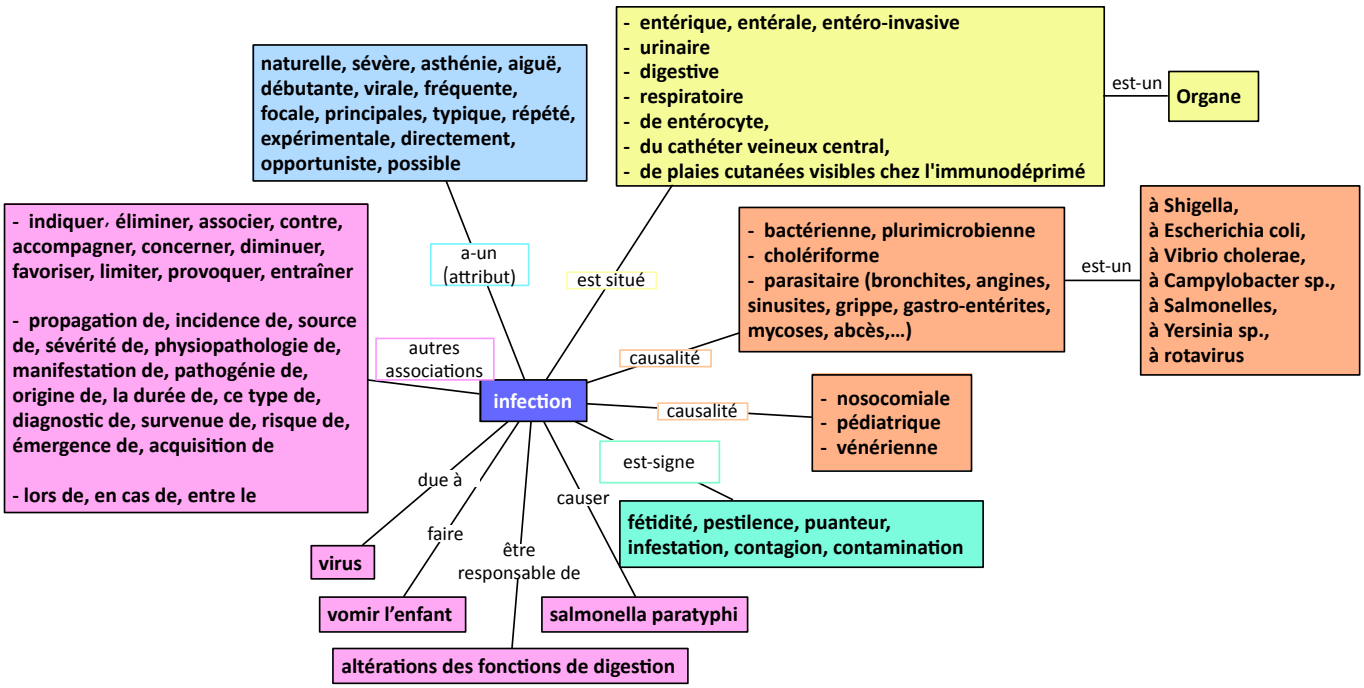


Figure 36 Le mot «infection» et ses caractéristiques lexicales et sémantiques

Construire un lexique est la première représentation intermédiaire de l'ontologie. Il faut obtenir les termes et leurs définitions puis on choisit une et une seule occurrence de définition (c'est-à-dire, une seule acception). Les mots ciblés de l'analyse lexicale sont les substantifs significatifs ayant un sens important, les adjectifs décrivent les attributs ou propriétés des objets ; les verbes et adverbes présentent les actions et les associations. La désambiguïsation des termes est rendue nécessaire par la polysémie, par exemple :

Extraits

« …Le soir, la **fièvre** atteint 38.5°C … »

« …La palpation montre un **abdomen** sensible mais souple, pas de **déshydratation**… »

« …L'**ionogramme** sanguin est normal… »

Définition

ABDOMEN n.m. (lat. *abdomen*) La partie caudale du corps qui suit le thorax, circonscrite en haut par le diaphragme, en bas par le bassin, en arrière par les vertèbres lombaires et en avant par des aponévroses et des muscles.

Définition d'interprétation DL

(define-concept abdomen (AND Organ (ALL isBody Caudal)(All hasLocation followThorax)(ALL hasLocation aboveDiaphragme)(ALL hasLocation belowBassin)(ALL hasLocation backByVertèbresLombaires)(ALL hasLocation forwardAponévroses)(ALL hasLocation forwardMuscles))).

Définition

FIÈVRE n.f. (lat. *febris*) La fièvre est une température corporelle anormalement élevée, qui dépasse 38 °C. Généralement, elle résulte d'une infection bactérienne ou virale, comme par exemple un rhume ou une grippe, mais elle peut aussi être le signe d'un problème plus grave. Une forte fière peut entraîner des lésions cérébrales.

Définition d'interprétation DL

(define-concept fièvre (AND Maladie (SOME hasInfection BactérienneInfection)(SOME hasInfection ViraleInfection)(ALL hasTempérature dépasse38°C)(SOME hasResult causerLésionsCérébrales)))

Toutes les définitions peuvent être une réécriture en Description logiques (DL), qui fournit des mécanismes d'inférence puissants, liés à des réseaux sémantiques ; les modèles frame (Minsky, M. 1975) sont réservés à la représentation de connaissances.

Ensuite, on enrichit la structure du lexique en classant les concepts, propriétés et attributs dans différentes tables ; on améliore le lexique en intégrant des aspects sémantiques pertinents (par exemple : subsumption). Les squelettes taxonomiques (*top-down* ou *bottom-up*) sont utiles dans cette étape.

Classe	Domain	Range	Traits	SuperClasse
Abdomen	Corps	Entomologie	Abstrait	Corps
			Concret +comptable ✓ – comptable	
Fièvre	Maladie	Écologie Écologie physiologique	Abstrait ✓	Maladie
			Concret + comptable – comptable	

Classe	subClass (Méronymes)	Synonymes	Antonymes	Apparentés étymologiques
Abdomen	cavité abdominale paroi abdominale viscère abdominal	bide panse ventre bas-ventre opisthosome		abdominal
Fièvre	Syndrome Durée Sévérité Emplacement Historique	ardeur hyperthermie pyrexie	hypothermie	antifébrile fébricitant fébricule fébrifuge fébrile

Tableau 21 Lexique de «Abdomen» et «Fièvre»

5.2.4 Délimite une phrase ou des énoncés

Deux approches (POS tagger et *tagger* de sens) sont utilisés dans cette étape.

La tâche de marquage (*tagging* en anglais), ce qui ajoute des informations lexicales, syntaxiques ou sémantiques au texte brut, rend les textes plus précis. La précision de marquage dépend de plusieurs aspects (Manning, C. D., & Schütze, H. 1999). Par exemple, la quantité de données apprises, la granularité de l'ensemble de marquage (*tagging set*), les occurrences de mots inconnus, etc.

Le *tagger* de sens est le processus d'attribution du sens approprié de certains lexiques sémantiques, appliqué à tous les mots ambigus dans le texte naturel. (Boguraev, B. & Pustejovsky, J. 1996) Il est similaire à la technologie POS tagging, plus largement employé, mais les marquages assignées en tagger de sens, sont sémantiques, issus d'un dictionnaire, plutôt que des marquages grammaticaux (morphologiques), attribués par un POS tagger.

Le *tagger* de sens traite de la désambiguïsation du sens **WSD** (*Word Senses Desambigusation*), il ne doit pas être confondu avec le WSD. Le *tagger* de sens peut être considéré comme une véritable tâche de l'ingénierie linguistique, car il est chargé du traitement robuste de grands corpus, et il fournit des informations utiles pour le traitement ultérieur de l'ingénierie linguistique, tandis que les objectifs du WSD sont limités et principalement de nature expérimentale.

5.2.4.1 POS tagger

Notre programme de POS tagger est formé, pour le français, d'un ensemble de données fournies par la bibliothèque tokenize Java. Il identifie les mots de contenu (noms, adjectifs, verbes) et des mots fonctionnels (prépositions, conjonctions, etc.) en utilisant un ensemble de règles contextuelles et lexicales (fondées sur les préfixes et les suffixes d'identification), tirés de textes annotés.

Les méthodes définies par le «Programme.3 POS tagger» comprennent `LexicalEntry()`, `HasFeatureCache()`, `TokenWrapper()`, `PosTaggedTokenWrapper()`, etc. Ces méthodes fonctionnent ensemble pour « *tagger* » le mot cible, afin de déterminer son genre, le nombre, sa nature temporelle, etc. (cf. Annexe.4)

Texte d'entrée

La baie de myrtille, riche en tanins, possède une action antidiarrhéique et un effet bénéfique sur les douleurs et les spasmes intestinaux lors de gastro-entérites.

Texte marqué

La(det.) baie(noun) de(prep.) myrtille(noun), riche(adj.) en(prep.) tanins(noun), possède(verb) une(det.) action(noun) antidiarrhéique(adj.) et(coord.) un(det.) effet(noun) bénéfique(adj.) sur(prep.) les(det.) douleurs(noun) et(coord.) les(det.) spasmes(noun) intestinaux(adj.) lors de(prep.) gastro-entérites(noun).

5.2.4.2 Tagger de sens

Le processus de « *tagger* » de sens comporte quatre étapes :

Etape 1. Le texte est d'abord traité par «Programme.1 Analyseur morphologique - Lemmatisation», pour être scindé en phrases, en ne laissant que les racines morphologiques. Dans cette étape, les mots appartenant à une liste de mots dits vides sont retirés (*cf.* «Tableau 13 : Glossaire de mots invariables»).

Etape 2. L'étape « Analyse lexicale » classe les mots en certaines catégories, comme l'entité (virus, outil, maladie, médicament, etc.), la personne, l'organisation, l'emplacement et certains domaines exclusifs, tels que : diagnostic, pronostic, traitement, suivi-thérapeutique, etc.

Exemple

<Un homme de 40 ans est hospitalisé pour une insuffisance rénale à diurèse conservée>

Liste de segmentation	Corresponde fonction
1) un **homme** de 40 ans	Exposer l'âge, le sexe, l'identité (femme enceinte, cardiaque,...) et d'autres informations personnelles du patient.
2) est hospitalisé	Verbes passifs - pour exposer la structure hiérarchique de la phrase.
3) pour une insuffisance **rénale**	Le causalité, provoque, cause,…
4) à	Les associations de réseaux sémantiques (prépositions du mouvement)
5) **diurèse** conservée	Confirmer le symptôme (Δ)

Etape 3. Le texte est ensuite marqué par POS Tokenization.

Etape 4. Le *tagger* de sens accède à l'ensemble des éléments (mots, syntagmes et catégories lexicales) et leur sens est attribué par les ingénieurs des connaissances. Le *tagger* de sens contient un **algorithme d'alignement** (écrit en Java, présenté en Annexe.4) pour annoter les séquences de mots correspondant à leurs descriptions sémantiques.

Exemple de Tagger de sens

Il présente [**une fièvre a 39-40°C**]/FIEVRE, accompagnée de [**frissons, vomissements, diarrhée et céphalées diffuses**]/SYMPTOME. [**Le frottis et la goutte épaisse**]/OUTIL montrent l'existence de nombreux trophozoites de [**plasmodium falciparum**]/PARASITE. Le diagnostic retenu est celui d'[**accès palustre simple**]/DIAGNOSTIC à [**plasmodium falciparum**]/VIRUS vraisemblablement [**chloroquino-résistant**]/PROPRIETE.

5.2.5 Établir le prototype et la classe de l'ontologie

La création automatique des prototypes de mots détermine les comportements par l'affinage des processus auto-descriptifs des objets et la définition de règles pour la génération des nouveaux objets.

On établit le prototype et la classe de l'ontologie avec les résultats des analyses morphologique, lexicale, et le « *tagging* », présentés dans les paragraphes précédents. Prenons le mot « médicament » par exemple, le tableau ci-dessous en montre les différentes acceptions.

Nom	Synonymes	Description	Type
médicament	drogue, remède, potion	substance préparée et employée pour soigner	concept
traitement	thérapie, cure, soin	façon de soigner	concept
pronostic	prévision, prédiction, présage	conjecture sur l'avenir ou l'issue d'une affaire, fondée sur un raisonnement, des probabilités ou certains indices	concept

Tableau 22 Glossaire de terme

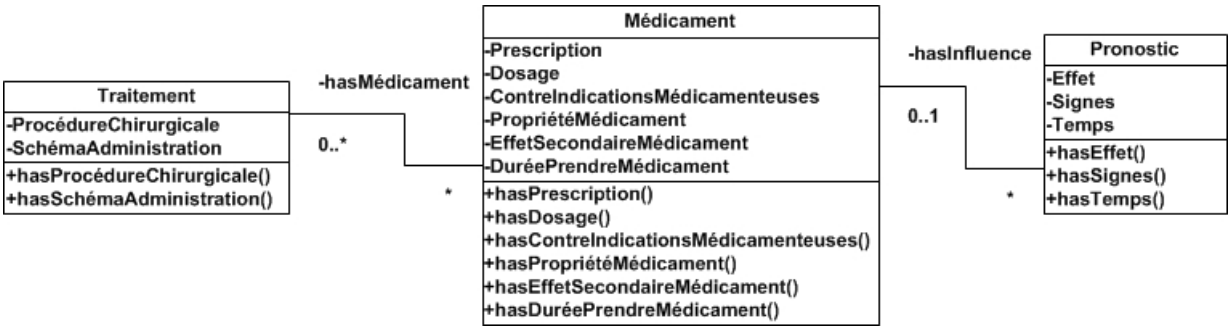


Figure 37 Classe « médicament »

5.3 Acquisition d'associations en corpus

Dans l'étude qui suit, on décrira un ensemble de procédures applicables à des textes, dont l'objet est de faciliter le repérage visuel des relations qui, ensuite, peuvent servir au cogniticien pour construire l'ontologie, ou permettre au système d'analyser le texte. L'objectif est de parvenir à ce but en réduisant au minimum les interventions manuelles.

Tout d'abord, les phrases avec les relations remarquables (instanciation /anti-instanciation, généralisation/spécialisation, composition/décomposition, etc.) et les associations étendues sont extraites du corpus par l'algorithme d'alignement, et elles sont ajoutés à la classe RelationExample, codée en Java. Ensuite, les programmes sont écrits pour présenter automatiquement la phrase et les relations entre les différents termes. Ensuite, ces associations servent au cogniticien à

construire l'ontologie. Enfin, on résume les règles d'analyse syntaxique avec la théorie de co-occurrence. Les règles d'analyse syntaxique sont résumées et mémorisées dans le système multi-agents, afin d'extraire les associations de textes automatiquement. L'idée clé de la théorie de co-occurrence est d'identifier les relations entre un ensemble de termes et un autre, en analysant la façon dont souvent les termes apparaissent ensemble dans plusieurs structures linguistiques similaires (Lovins, J.B. 1968). La théorie de la co-occurrence est utilisée ici, pour récupérer les relations entre les différents termes.

5.3.1 Représentation d'associations conceptuelles

Sur le domaine réseau sémantique de systèmes complexes, les relations sont les attributs descriptifs entre deux mots. Les différents types de relations montrés ci-après peuvent nous aider de simplifier et améliorer l'analyse et l'extraction de relations. Le tableau présenté ci-après montre les types de relations sur le domaine réseau sémantique. Ce tableau est inspiré du projet Idéliance (Hamon J.P. 2002).

Enoncés	Types de relations
Romain est le collègue de **Sébastian** **Sébastian** est le collègue de **Romain**	Symétrie / réciprocité
Romain est plus grand que **Sébastian** **Sébastian** est plus petit que **Romain**	Symétrie / différentiel
Romain peut distribuer le travail à **Sébastian** **Sébastian** ne peut pas distribuer le travail à **Romain**	Antisymétrie / non réciprocité
Romain pense être l'ami de **Sébastian** **Sébastian** déteste **Romain**	antisymétrie d'états
Romain travaille dans le même bureau que **Sébastian** **Sébastian** travaille dans un bureau différent de celui de **Romain**	antisymétrie de positions
Romain observe **Sébastian** **Sébastian** «observation indéterminée sur» **Romain** **Sébastian** est observé par **Romain**	 relation inachevée relation par défaut

Tableau 23 Types de relations sur le domaine réseau sémantique

5.3.1.1 Relations des mots remarquables

Dans ce qui suit, il faut mettre en exergue l'importance de la sémantique des relations remarquables (association d'instanciation (créer un objet à partir d'une

classe), sa réciprocité : anti-instanciation (définir une classe à partir d'un objet), généralisation/spécialisation (est-un), composition/décomposition (est-partie-de) et l'association (a-une ou possède une propriété) car elles s'associent à la logique sémantique des phrases. Ces quatre associations remarquables et les contraintes qu'elles génèrent sont indispensables à la définition des objets et à leur autonomie.

Par exemple, la classe « Médicament » possède des propriétés ou attributs « nom », « marque », « validité » qui sont des propriétés statiques, tandis que la méthode CalculePrix() est une propriété dynamique. Les propriétés sont unies à la classe « médicament » par l'association (a-un), c'est-à-dire : un médicament a un nom, une marque, une validité.

À propos de la relation généralisation/spécialisation, il faut détailler l'association "est-un" ou "est une sorte de". L'objet est une instance de classe, on peut donc considérer qu'il s'agit d'une classe qui ne comporte qu'un seul objet. Dans la théorie des ensembles, un singleton montre la notion d'objet unique, spécifique.

Comme nous l'avons déjà indiqué plus haut, la généralisation/spécialisation (est-un) est une association logique entre classe et sous-classe. Par exemple, Pénicilline G est un «Antibiotique», sous-classe de «Médicament», dont «Pénicilline G» hérite de toutes les propriétés, plus des propriétés spécifiques : «contreIndicationAntibiotique», «effetSecondaireAntibiotique», etc.

La composition/décomposition (est-partie-de) : « gélule » est un composant des médicaments présentés en gélule. L'objet « gélule » a des propriétés ou attributs comme « couleur » « composition ». Pour créer un objet « Médicament » comme Amoxicilline, il faut instancier un objet « capsule » de la classe « gélule » et donner une valeur à la propriété ou attribut « couleur » = 'blanc' et 'rouge'.

Avec les relations essentielles et autres relations importantes concernant les dimensions temporelle et spatiale, la causalité et ainsi de suite, on extrait les phrases correspondantes sur le corpus «Diarrhée aiguë», pour présenter les classes et sous-classes, et ses relations intrinsèques.

Après avoir défini les propriétés statiques, on définit les propriétés dynamiques à partir de différentes méthodes ou fonctions qui servent à exprimer le comportement des objets et les relations importantes de causalité et de temporalité.

Relations essentielles
– noms avec les adjectifs. (NAdj) – généralisation/spécialisation (est-un) (class→sub-class) – composition/décomposition (est-partie_de, est-composé_de) (relations d'agrégation) – instanciation/anti-instanciation (a_un) (l'objet est porteur d'une caractéristique) – Autres associations exemple : est-propriétaire (possède) et, est-synonyme lié_fonctionnellement_à lié_conceptuellement_à verbes importants : acheter, vendre, prendre etc.
CASNET
– temporelles (lié_temporellement_à: précède, suit,…)
Logique de Allen formes de prédicats temporels
– spatialité (lié_physiquement_à: est dessus, est dessous, est à proximité de, est-situé, est associé à, est voisin de etc.) – causalité - association entre l'objet qui cause un effet sur un autre objet : produire, provoquer, causer, évoquer
Logiques causales
– similarité ou comparaison – logique d'induction – logique d'abduction

Tableau 24 Relations entre mots remarquables

Exemple

is_a ([être]-[un])

est_un

- Le salicylate de bismuth est un dérivé de l'acide salicylique.

est_une

- La diarrhée du voyageur est une affection très fréquente qui survient le plus souvent au cours de la première semaine du séjour.

est_le

- Le SRO est le traitement de première intention des enfants atteints de GEA.

est_la

- La giardiase est la principale étiologie dont l'expression clinique est très polymorphe.

sont_les

- Les trois principales infections bactériennes observées chez l'homosexuel sont les salmonelloses, les shigelloses et les campylobactérioses.

sont_des

- Les adénovirus entériques sont des virus à ADN double brin, non enveloppés.

être_une_sorte_de

- Le Phénicoles est une sorte des antibiotiques.

has_a (a une propriété)

a_un

- Le rotavirus a un tropisme spécifique pour les entérocytes matures du sommet des villosités de l'intestin grêle.

a_une

- Le racécadotril (Tiorfan®) a une activité antisécrétoire sans effet sur la motricité intestinale et a démontré son efficacité dans la diminution du débit des selles avec une diminution de la durée de la diarrhée.

a_le/la

- Il a la peau grisâtre.

ont_les/des

- Ils ont les yeux creux.

est-propriétaire (possède)

- Les composés antidiarrhéiques possèdent divers mécanismes d'action : altération de la motilité intestinale, altération de la sécrétion, adsorption de toxines ou de liquides et altération de la flore intestinale.
- La baie de myrtille, riche en tanins, possède une action antidiarrhéique et un effet bénéfique sur les douleurs et les spasmes intestinaux lors de gastro-entérites.

est_partie_de

- Le nombre de selles peut commencer par augmenter. Ce processus fait partie de la guérison de l'intestin.

associated_with (est associé_à)

- Parmi tous les indicateurs nutritionnels, seul le faible périmètre brachial (< 125 mm) était associé à la présence de diarrhée.
- La diarrhée est associée à une augmentation du risque d'infection de cathéter veineux central par contamination fécale.

physically_related_to (lié_physiquement_à)

(est-situé, est-voisin_de, est dessus, est dessous, est à proximité de…)

- La tomodensitométrie thoracoabdominale a confirmé la présence d'une masse médiastinale postérieure située au niveau de la gouttière costo-vertébrale gauche en regard de T6 à T11.
- Les entérotoxines thermolabiles des ETEC sont des entérotoxines cytotoniques voisines de la toxine produite par Vibrio cholerae et de celles produites par d'autres bactéries comme Aeromonas hydrophila, Campylobacter jejuni, Salmonella typhi murium.

temporally_related_to (lié_temporellement_à)(précède, suit, avant…)

- Leur présentation est globalement assez proche, associant en général des nausées et des vomissements (parfois au premier plan) qui précèdent de quelques heures ou accompagnent les diarrhées, hydriques, qui durent en moyenne 1 à 4 jours.
- Une analyse de la documentation scientifique des cinq à dix dernières années sur les recommandations à l'égard de la réhydratation orale et du traitement de la diarrhée a permis d'établir ce qui suit.
- Pour diminuer le risque de contamination, les adultes et les enfants doivent bien se laver les mains après chaque changement de couche, après être allés aux toilettes et avant de manger ou de préparer un repas.

causalité (produit, provoque, cause, évoque)

- Clostridium difficile produit une entérotoxine A mais aussi une cytotoxine B : après pénétration intracellulaire, elles provoquent une

désorganisation du cytosquelette et modifient le fonctionnement de la jonction intercellulaire.

- La mortalité n'est pas négligeable puisqu'une cinquantaine de décès par an en France sont causés par des GEA.
- Les éléments suivants doivent faire évoquer devant une DA d'autres causes qu'une gastroentérite aiguë : douleur abdominale avec abdomen sensible à la palpation, avec ou sans défense, pâleur, ictère, oligoanurie, diarrhée sanglante, altération de l'état général hors de proportion avec l'importance de la déshydratation.

Autres associations : achète, vend, prendre…

- Les solutions de réhydratation orale sont vendues en pharmacie sous forme de préparations prêtes à servir, de sucettes glacées et de sachets de poudre.
- En cas de diarrhée infectieuse, il peut être conseillé de prendre une goutte d'HE d'origan compact (Origanum compactum) sur un comprimé neutre ou dans une cuillère à café d'huile d'olive à laisser fondre en bouche avant ou après les trois repas.

Tous les résultats présentés ci-dessus seront analysés à partir des associations étendues.

5.3.1.2 Associations étendues

Il reste maintenant à créer les *frames* des relations. On repère et rassemble ici des séquences entre deux dénominations pour présenter les relations. Nous donnons l'exemple présenter (X, Y) avec les contextes du corpus sur la diarrhée aiguë. Le premier exemple montre la relation entre «antibiothérapie» et «infection» avec 4 contextes. D'autres exemples présentent les relations entre «traitement» et «antidiarrhéique», «médicament» et «antidiarrhéique», «diarrhée» et «antidiarrhéique» respectivement.

PRÉSENTER (X, Y)

X1 : **antibiothérapie**

Y1 : **infection**

formes associées :

- orale est limitée aux

- et d'un traitement antidiarrhéique a prouvé son efficacité, notamment l'association ciprofloxacine et lopéramide testée chez des voyageurs adultes lors d'

- émergence d'

- peut être hautement efficace contre les

1) L'antibiothérapie orale est limitée aux infections à Shigella et aux autres infections bactériennes en cas de sepsis grave ou de terrain fragile.

2) L'association d'une courte antibiothérapie et d'un traitement antidiarrhéique a prouvé son efficacité, notamment l'association ciprofloxacine et lopéramide testée chez des voyageurs adultes lors d'infections à Shigella sp. ou Escherichia coli entéro-invasif.

3) Il peut s'agir de risques individuels pour le patient (allergie, effets secondaires digestifs de l'antibiothérapie, émergence d'infections à Clostridium difficile).

4) L'antibiothérapie peut être hautement efficace contre les infections à Shigella, à Escherichia coli et à Vibrio cholerae, et le métronidazole est indiqué en cas de colite à Clostridium difficile.

X2 : **traitement**

Y2 : **antidiarrhéique**

formes associées :

- médical souvent avec prise d'

- antispasmodique, les médicaments

- à l'aide de composés

1) La diarrhée est dite persistante quand elle évolue depuis plus de 14 jours, malgré un premier traitement médical souvent avec prise d'antidiarrhéiques seuls.

2) Outre le traitement antispasmodique, les médicaments antidiarrhéiques trouvent là une bonne indication.

3) Cependant, les nouvelles données sont présentées sur la composition des SOR et sur les traitements à l'aide de composés antidiarrhéiques.

X3 : **médicament**

Y3 : **antidiarrhéique**

formes associées:

- qui ait démontré une activité

- comme

- (lié)

1) Le racécadotril est le seul médicament qui ait démontré une activité antidiarrhéique, avec une diminution du débit des selles de l'ordre de 50 %.

2) On considère un médicament comme antidiarrhéique s'il entraîne une diminution d'au moins 30 % du débit des selles par rapport à un placebo.

3) Outre le traitement antispasmodique, les médicaments antidiarrhéiques trouvent là une bonne indication.

X4 : **diarrhée**

Y4 : **antidiarrhéique**

formes associées :

- on associe au lait des aliments classiquement

- est dite persistante quand elle évolue depuis plus de 14 jours, malgré un premier traitement médical souvent avec prise d'

- peut être traitée par l'utilisation d'

1) Lorsque l'alimentation était déjà diversifiée avant l'apparition de la diarrhée, on associe au lait des aliments classiquement « antidiarrhéiques » (carottes, pommes-coings, riz, bananes).

2) La diarrhée est dite persistante quand elle évolue depuis plus de 14 jours, malgré un premier traitement médical souvent avec prise d'antidiarrhéiques seuls.

3) La diarrhée peut être traitée par l'utilisation d'antidiarrhéiques comme les inhibiteurs de l'enképhalinase et le bismuth, par la modification de la flore intestinale grâce à des lactobacilles ou par l'administration de suppléments de zinc.

La construction du réseau sémantique (concepts et relations établis) permettent de résumer les relations entre les différents sujets : le sujet est un nœud central du réseau. Certaines conditions initiales existantes ont valeur d'apprentissage. Par exemple, la

règle définie par (Hamon J.P. 2002) est « QUI a écrit QUOI, sur quel SUJET, à quels DESTINATAIRES et à quel MOMENT », etc.

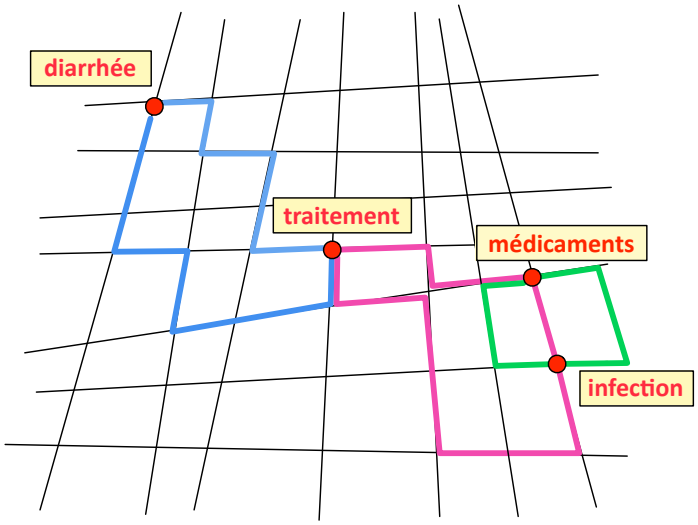


Figure 38 Condensation des réseaux sémantiques entre les mots «diarrhée» et «infection»

La «Programme.9 Extraire des relations des mots cibles» (cf. Annexe.4) vise à extraire des relations des mots cibles. Tout d'abord, on importe le RelationExample (cf. Programme.8). Ensuite, après avoir extrait tous les mots qui forment la phrase, on définit le wordNode pour chaque mot et l'on extrait le prédicat d'une phrase. Dans ce cas, l'extraction des relations d'un mot cible est opérable. Afin de réutiliser ces relations extraites, on va changer la relation du type «String» au type «RelationExample», et passer la forme binaire de certaines relations à la forme unaire.

Avec les codes Java, on peut extraire des réseaux sémantiques entre les mots «diarrhée», «traitement», «médicament» et «infection» dans des corpus, puis indiquer des sens 1.1, 1.2, 2.1 2.2 ... entre les mots cibles dans la figure ci-après.

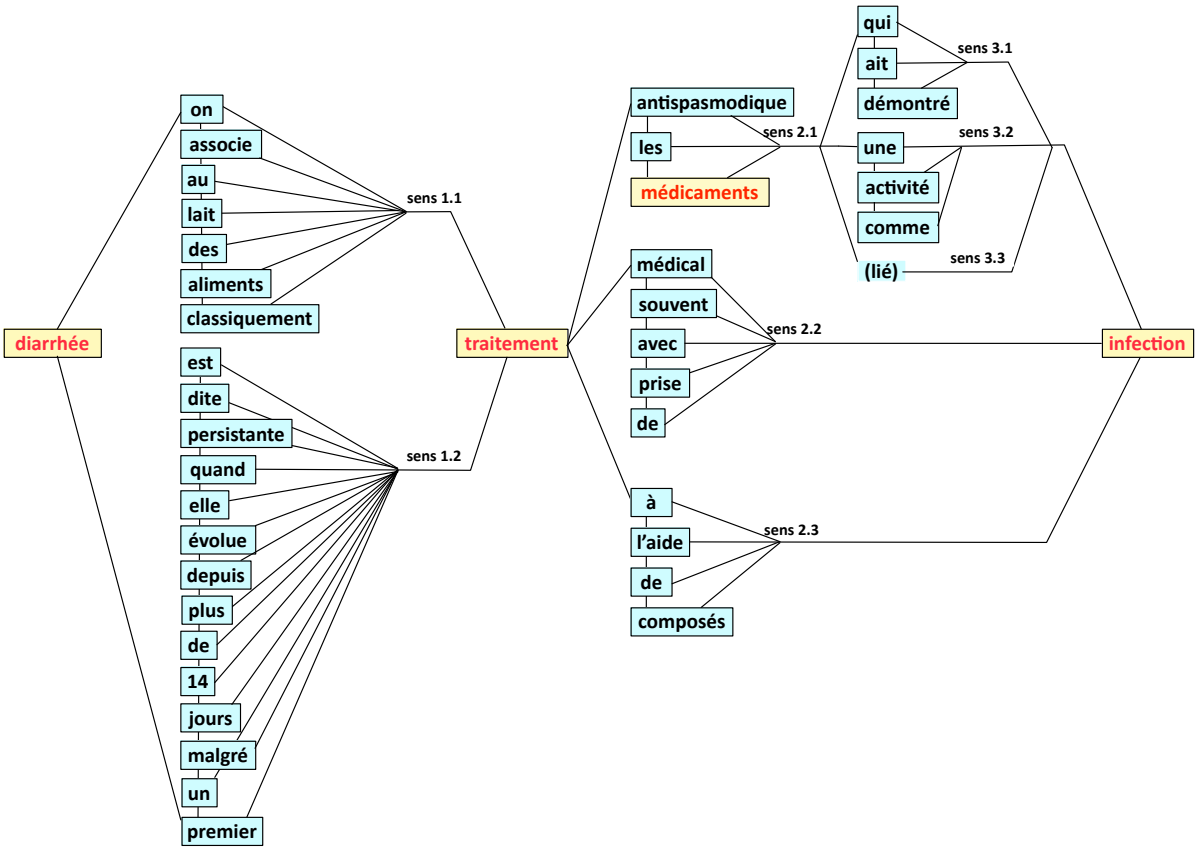


Figure 39 Extrait de réseaux sémantiques entre les mots «diarrhée», «traitement», «médicament» et «infection»

5.3.1.3 Glossaire de relation

Le tableau suivant présente certaines des relations existantes dans notre «Dictionnaire ontologique», en montrant les concepts sources et cibles et les phrases correspondantes de chacune de ces relations.

Relation	Concept source	Concept cible	Exemple de corps « Diarrhée aiguë »
hasSyndrome	Diagnostic	Syndrome	Leur présentation est globalement assez proche associant en général des nausées et des vomissements (parfois au premier plan) qui précèdent de quelques heures ou accompagnent les diarrhées, hydriques, qui durent en moyenne 1 à 4 jours.
hasCause	Diagnostic	Cause	Devant une DA, les éléments suivants doivent faire évoquer d'autres causes qu'une gastroentérite aiguë : douleur abdominale avec abdomen sensible à la palpation, avec ou sans défense, pâleur, ictère, oligoanurie, diarrhée sanglante, altération de l'état général hors de proportion avec l'importance de la déshydratation.
hasDonnéeStatistique	Cause	Donnée Statistique	Parmi tous les indicateurs nutritionnels, seul le faible périmètre brachial (< 125 mm) était associé à la présence de diarrhée.
hasEffet	Pronostic	Effet	Le nombre de selles peut commencer par augmenter. Ce processus fait partie de la guérison de l'intestin.
hasProcédureChirurgicale	Traitement	Procédure Chirurgicale	Le SRO est le traitement de première intention des enfants atteints de GEA
hasMédicament	Traitement	Médicament	La diarrhée peut être traitée par l'utilisation d'antidiarrhéiques comme les inhibiteurs de l'enképhalinase et le bismuth, par la modification de la flore intestinale grâce à des lactobacilles ou par l'administration de suppléments de zinc. Le racécadotril (Tiorfan®) a une activité antisécrétoire sans effet sur la motricité intestinale et a démontré son efficacité dans la diminution du débit des selles avec une diminution de la durée de la diarrhée.
hasDosage	Médicament	Dosage	En cas de diarrhée infectieuse, il peut être conseillé de prendre une goutte d'HE d'origan compact (Origanum compactum) sur un comprimé neutre ou dans une cuillère à café d'huile d'olive à laisser fondre en bouche avant ou après les trois repas.
hasFonctionMédicament	Médicament	Fonction Médicament	Les composés antidiarrhéiques possèdent divers mécanismes d'action : altération de la motilité intestinale, altération de la sécrétion, adsorption de toxines ou de liquides et altération de la flore intestinale. La baie de myrtille, riche en tanins, possède une action antidiarrhéique et un effet bénéfique sur les douleurs et les spasmes intestinaux lors de gastro-entérites.
hasPrévention	Suivi-thérapeutique	Prévention	Pour diminuer le risque de contamination, les adultes et les enfants doivent bien se laver les mains après chaque changement de couche, après être allés aux toilettes et avant de manger ou de préparer un repas.

Tableau 25 Glossaire de relations

5.3.2 Règle d'analyse syntaxique

Dans l'étude qui suit, on décrira un ensemble de procédures applicables à des textes dont l'objet est faciliter le repérage de relations. Nous concevons un module applicable : Description Logiques des mécanismes d'inférence, ainsi que des règles syntaxiques, afin de combiner les descriptions conceptuelles associées à chaque fragment sémantique.

Les règles syntaxiques sont d'abord établies manuellement par les ingénieurs des connaissances sur la base du corpus de test POS marqué et annoté manuellement, avec des descriptions conceptuelles dans les dernières sections.

Les étapes d'application d'une règle sont les suivantes :

a) Identifier un mot déclencheur;

b) Collecter des segments répétés de probabilité la plus élevée,

c) Rechercher les règles de contexte entre cooccurrences de couples,

d) Représenter des associations et des règles d'analyse syntaxique.

L'analyse syntaxique se concentre sur la structure des phrases ; elle peut être utilisée pour cartographier des formes passives et des formes actives, en participant à la normalisation de la forme de la phrase avant l'interprétation sémantique.

Cette approche de recherche est inspirée par (Frath, P. et al. 2000).

5.3.2.1 Collecte des segments répétés

Pour la représentation des relations (servir à la réalisation de propriétés d'ontologie), le «Programme.10 Compteur de mots» (cf. annexe.4) est utilisé pour traiter et calculer la fréquence de répétition de segments de longueur déterminée dans le texte.

En cela, le segment de longueur « n » est réglé pour être inférieur ou égal à 6 (≤ 6), pour éviter la redondance de résultat.

contre-indiqu*

médicament(s) [être] **contre-indiqu*** (11)
médicament(s) [être] contre-indiqu* chez (7)
médicament(s) [être] contre-indiqu* chez […] enfant(s) (5)
médicament(s) [être] contre-indiqu* chez […] enfant(s) de (5)
médicament(s) [être] contre-indiqu* chez […] enfant(s) de moins (4)
médicament(s) [être] contre-indiqu* chez […] enfant(s) de moins de cinq ans (1)

contre-indiqu* en (8)
contre-indiqu* en cas (8)
contre-indiqu* en cas de (8)
contre-indiqu* en cas de syndrome (3)
contre-indiqu* en cas de syndrome dysentérique (2)
contre-indiqu* en cas de syndrome douloureux (1)

lopéramide contre-indiqu* (7)
lopéramide contre-indiqu* avant (6)
lopéramide contre-indiqu* avant deux (1)
lopéramide contre-indiqu* avant deux ans (1)
lopéramide contre-indiqu* avant l'âge de deux (1)

l'indication mais contre-indiqu* (2)
l'indication mais contre-indiqu* chez (1)
l'indication mais contre-indiqu* chez l'enfant (1)
médicaments ayant l'indication mais contre-indiqu* chez l'enfant (1)

Il est intéressant de regrouper les segments répétés, et aussi nécessaire de regrouper des segments répétés qui se différencient seulement par leur forme morphologique. Cette représentation collective sert à l'analyse de « Cooccurrences de couples ».

5.3.2.2 Cooccurrences de couples

« Programme.10 Compteur de mots », avec sa fonction « filtre », est conçu pour produire un nombre réduit de couples représentant un nombre accru de contextes, désormais faciles à manipuler.

On émet l'hypothèse que ces couples représentent des entités conceptuelles présentes dans le texte. Le fait de disposer d'un petit nombre de couples représentant un grand nombre de contextes augmente les chances de découvrir des cooccurrences de contextes susceptibles de révéler l'existence de relations (Frath, P. et al. 2000).

Pour la fonction « filtre », certaines fonctions sont attendues :

i Pour chaque mot de contenu (nom, adjectif, verbe et certains adverbes), il va stocker les contextes à gauche et à droite.

ii Les concepts primitifs ou mots couples sont instanciés par des mots de contenu. Les concepts spécifiques sont identifiés à partir des contextes (les phrases correspondantes) des instances de concepts primitifs.

Les exemples ci-dessous affichent les co-occurrents de chaque couple, en plaçant en tête de liste les couples les plus co-occurrents :

Exemple n° 1 : Couples : {contre-indiqu*, médicament*} et {en raison de, effet*}

(1) Ces **médicaments** sont **contre-indiqués** chez les enfants **en raison de** leurs **effets** secondaires au potentiel grave.

(2) Ce **médicament** est **contre-indiqué** chez l'enfant de moins de deux ans **en raison du** risque d'**effets** neurologiques centraux de type somnolence et surtout d'iléus paralytique ayant entraîné plusieurs décès.

(3) Les autres spécialités (**médicament**) **contre-indiquées** concernaient des micro-organismes, **en raison des** formes galéniques inadaptées chez le nourrisson.

Exemple n° 2 : Couples : {infection*, à} et {observ*, à}

(4) Les **infections à** Yersinia sp. sont habituellement **observées à** l'occasion de déplacements dans les pays développés ;

(5) Les **infections à** rotavirus sont **observées au** printemps, principalement dans les petites collectivités (crèches, services de pédiatrie).

Exemple n° 3 : Couples {infection*, à} et {[être], fréquent*}

(6) Les **infections à** salmonelles **sont** très **fréquentes**, allant du portage asymptomatique aux formes sévères à type de septicémie ou de colectasie.

(7) Les **infections à** Campylobacter **sont** assez **fréquentes**, mais sans particularité chez l'homosexuel.

Exemple n° 4 : Couples {antidiarrhéique, diarrhée} et {en cas [de], [prendre]}

(8) **En cas de** diarrhée modérée bien tolérée sans fièvre, un **antidiarrhéique** (lopéramide ou racécadotril) peut être **pris** selon l'intensité de la **diarrhée**.

(9) Ils ne sont pas indiqués **en cas de diarrhée** modérée chez l'adulte, chez lequel le risque de déshydratation est faible notamment **en cas de prise** associée de médicaments **antidiarrhéiques**.

Exemple n° 5 : Couples {au cours [de], diarrhée*} et {cause*, infectieuse*}

(10) **Au cours des diarrhées** aiguës, les **causes infectieuses** sont les plus fréquentes.

(11) **Au cours des diarrhées aiguës**, les **causes infectieuses** sont principalement d'origine virale chez l'enfant.

(12) **Au cours des diarrhées** aiguës, les **causes infectieuses** nécessitent toujours une coproculture.

(13) **Au cours des diarrhées aiguës**, les **causes infectieuses** ne récidivent jamais avec le même germe.

(14) **Au cours des diarrhées aiguës**, les **causes infectieuses** peuvent être facilitées par un état d'immunodépression.

5.3.2.3 Représentation des associations et règles d'analyse syntaxique

Le but de l'analyse est de mettre en évidence d'éventuelles relations entre les couples de manière à établir des règles d'analyse syntaxique.

Maintenant, l'acquisition de ces schémas est semi-automatique : l'ingénieur des connaissances résume la représentation des associations et règles d'analyse syntaxique, puis le système vérifie les synonymes de phrases cibles (par «Programme.2 Synonyme»), pour étudier plus spécifiquement les relations syntaxiques et sémantiques de cooccurrences de couples. Par exemple, on peut généraliser « En cas de diarrhée modérée bien tolérée sans fièvre, un antidiarrhéique peut être pris selon l'intensité de la diarrhée. » en [**EN_CAS_DE** (condition(X))] {[(objet) action] SELON (condition(Y))}. En outre, cette règle peut être généralisé encore en [**SI** (condition(X))] {[(objet) action] SELON (condition(Y))} car [en cas de] et [si] est synonyme.

Tous les cadres sémantico-syntaxiques sont mémorisés dans le «dictionnaire ontologique» de notre système multi-agents pour être indexés et réutilisés.

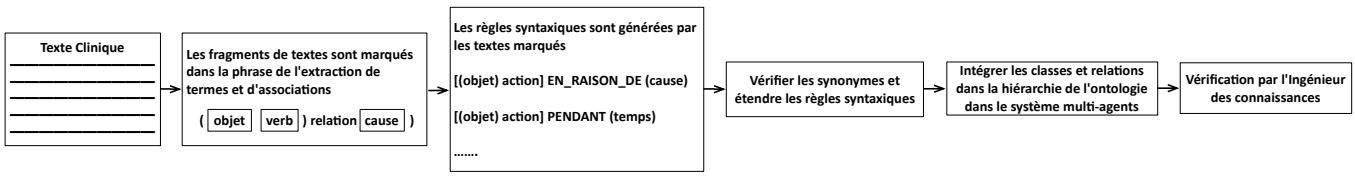


Figure 40 Extraction de connaissances à partir de textes : génération de règles syntaxiques

Exemple n° 1 : Couples : {contre-indiqu*, médicament*} et {en raison de, effet*}

(1) Ces **médicaments** sont **contre-indiqués** chez les enfants **en raison de** leurs **effets** secondaires au potentiel grave.

(2) Ce **médicament** est **contre-indiqué** chez l'enfant de moins de deux ans **en raison du** risque d'**effets** neurologiques centraux à type de somnolence et surtout d'iléus paralytique ayant entraîné plusieurs décès.

(3) Les autres spécialités (**médicament**) **contre-indiquées** concernaient des micro-organismes, **en raison des** formes galéniques inadaptées chez le nourrisson.

C'est le cas pour les **phrases 1, 2 et 3,** où on peut identifier un schéma du type :

[médicament*(contre-indiqu*)] EN_RAISON_DE (effet*)

ou

[médicament*(contre-indiqu*)] EN_RAISON_DE (risque*)

généralisé en :

[(objet) action] EN_RAISON_DE (cause)

Comme ce schéma apparaît plusieurs fois dans notre corpus, cela suggère qu'il est assez significatif dans ce texte. En ce qui concerne la relation EN_RAISON_DE, il s'agira de l'interpréter et de l'étiqueter pour comparer avec d'autres phrases de ce type recueillies ailleurs.

Exemple n° 2 : Couples : {infection*, à} et {observ*, à}

(4) Les **infections à** Yersinia sp. sont habituellement **observées à** l'occasion de déplacements dans les pays développés ;

(5) Les **infections à** rotavirus sont **observées au** printemps, principalement dans les petites collectivités (crèches, services de pédiatrie).

{[(infection*) À (virus)] (être_observ*)}PENDANT (temps)

généralisé en :

[(objet) action] PENDANT (temps)

Exemple n° 3 : Couples : {infection*, à} et {[être], fréquent*}

(6) Les **infections à** salmonelles **sont** très **fréquentes**, allant du portage asymptomatique aux formes sévères à type de septicémie ou de colestasie.

{[(infection*) À (virus)] ETRE (fréquent*)}(range)

généralisé en :

[(objet) ETRE (propriété)](range)

(7) Les **infections à** Campylobacter **sont** assez **fréquentes**, mais sans particularité chez l'homosexuel.

[(infection*) À (virus)] ETRE (fréquent*) MAIS (condition)

généralisé en :

[(objet) ETRE (propriété)] MAIS (condition)

Exemple n° 4 : Couples : {antidiarrhéique, diarrhée} et {en cas [de], [prendre]}

(8) **En cas de** diarrhée modérée bien tolérée sans fièvre, un **antidiarrhéique** (lopéramide ou racécadotril) peut être **pris** selon l'intensité de la **diarrhée**.

[EN_CAS_DE (condition)] {(antidiarrhéique) (être_prendre) SELON (propriété_diarrhée)}

généralisé en :

[EN_CAS_DE (condition(X))] {[(objet) action] SELON (condition(Y))}

généralisé encore en :

[SI (condition(X))] {[(objet) action] SELON (condition(Y))}

(9) Ils ne sont pas indiqués **en cas de diarrhée** modérée chez l'adulte, chez lequel le risque de déshydratation est faible notamment **en cas de prise** associée de médicaments **antidiarrhéiques**.

[EN_CAS_DE (diarrhée_propriété)] {(symptôme) EN_CAS_DE [(prendre) (médicaments_antidiarrhéiques)]}

généralisé en :

[EN_CAS_DE (maladie_propriété)] {(symptôme) EN_CAS_DE [action(médicament)]}

Exemple n° 5 : Couples : {au cours [de], diarrhée*} et {cause*, infectieuse*}

(10) **Au cours des diarrhées** aiguës, les **causes infectieuses** sont les plus fréquentes.

[AU_COURS_DE (diarrhée*_aiguë*)] (cause*_infectieuse*) ETRE (fréquent*)

généralisé en :

[AU_COURS_DE (maladie(X))] (cause(Y)) ETRE (propriété)

(11) **Au cours des diarrhées aiguës**, les **causes infectieuses** sont principalement d'origine virale chez l'enfant.

[AU_COURS_DE (diarrhée*_aiguë*)] (cause*_infectieuse*) D'ORGINE (chez l'enfant)

généralisé en :

[AU_COURS_DE (maladie(X))] (cause(Y)) D'ORGINE (objet(Z)))

(12) **Au cours des diarrhées** aiguës, les **causes infectieuses** nécessitent toujours une coproculture.

[AU_COURS_DE (diarrhée*_aiguë*)] (cause*_infectieuse*) NECESSITER (coproculture)

généralisé en :

[AU_COURS_DE (maladie(X))] (cause(Y)) NECESSITER (objet(Z)))

(13) **Au cours des diarrhées aiguës**, les **causes infectieuses** ne récidivent jamais avec le même germe.

[AU_COURS_DE (diarrhée*_aiguë*)] [(cause*_infectieuse*) (ne_récidivent_jamais) AVEC (le_même_germe)]

généralisé en :

[AU_COURS_DE (maladie(X))] {[(cause(Y)) action] AVEC (objet(Z))}

(14) **Au cours des diarrhées aiguës**, les **causes infectieuses** peuvent être facilitées par un état d'immunodépression.

[AU_COURS_DE (diarrhée*_aiguë*)] [(cause*_infectieuse*) (facilit*) PAR (un_état_d'immunodépression)]

généralisé en :

[AU_COURS_DE (maladie(X))] {[(cause(Y)) action] PAR (objet(Z))}

Exemple n° 6 : Phrases longues et complexes

Il présente une **fièvre** a 39-40°C, **accompagnée de** frissons, vomissements, **diarrhée** et céphalées diffuses. **Le frottis et la goutte épaisse** montrent l'existence de nombreux trophozoïtes de **plasmodium falciparum**.

(présent*(fièvre)) AVEC (symptôme)

généralisé en :

(action (maladie)) AVEC (symptôme)

(montr*(virus)) PAR (outil)

généralisé en :

(action (virus)) PAR (objet)

Exemple n° 7 : Phrases longues et complexes

Pour diminuer le risque de contamination, les adultes et les enfants doivent bien se laver les mains après chaque changement de couche, après être allés aux toilettes et avant de manger ou de préparer un repas

POUR (diminu* (risque*), (lav*, main*)

généralisé en :

POUR [action(X) (objet(X)), [action(Y) (objet(Y))]

Qu'en est-il de « diminuer le risque de contamination » et de « se laver les mains » ? On peut considérer les termes « risque de contamination », les verbes « diminuer », « doivent bien » et « laver les mains ». L'idée principale de cette phrase est « pour atteindre le objectif, il faut faire quelque chose ». Donc on peut généraliser la phrase en **[POUR [action(X) (objet(X)), [action(Y) (objet(Y))]]**, sur la base d'analyse linguistique de précédents chapitres. Dans cette approche, l'accent est mis d'emblée sur les termes considérés comme des variables.

5.4 Construction d'ontologies avec hiérarchie

Beaucoup d'attention a été accordée à la définition et à la conceptualisation de la hiérarchie (Bales, R.F. et al. 1951) (Benoit-Smullyan, E. 1944) (Bernstein, P.A. & Goodman, N. 1981) (Lenski, G.E. 1954) (Maedche, A. & Staab, S. 2002). En général, la hiérarchie est définie comme un ordre de classement des individus, selon une ou plusieurs dimensions socialement importantes (Gruenfeld, D.H. & Tiedens, L.Z. 2010).

Dans le domaine de l'ontologie, les concepts sont organisés en une hiérarchie de subsomption, calculée selon les définitions de concepts. Les classes sont structurées en une hiérarchie d'héritage, définie par le lien "a-kind-of" « est une sorte de ». Les relations peuvent également être définies et organisées en hiérarchies comme des classes.

Dans ce cas, les langages de modélisation graphique (UML, OMT) ont été proposés : ils ressemblent à des langages graphiques orientés objet.

5.4.1 Règles d'établissement de la hiérarchie

Transitivité des relations hiérarchiques

Les relations hiérarchiques comme la spécialisation de classe en sous-classes est transitive : c'est le cadre pour l'extension de la hiérarchie dans cette thèse.

Si B est une sous-classe de A, et C est une sous-classe de B,

Alors C est une sous-classe de A

Par exemple, nous pouvons définir une classe « médicament », puis définir une classe « antibiotique » comme une sous-classe de « médicament ». Ensuite, nous définissons une classe « antimycosiques» comme une sous-classe de « antibiotique ». La transitivité de la relation de spécialisation permet de déduire que la classe « antimycosiques » est aussi une sous-classe de « médicament ».

Parfois, on fait la distinction entre les sous-classes directes et sous-classes indirectes. Nous appelons « sous-classe directe » la sous-classe immédiate de la classe dans la hiérarchie (Noy, N. F. & McGuinness, D. L. 2001). Dans notre exemple, la classe « antimycosiques » est une sous-classe directe de la classe « antibiotique » mais pas une sous-classe directe de « médicament ».

La composition d'objet : is-part-of, qui unie un objet composé et des objets composants, est également une relation antisymétrique et transitive. Elle permet de

construire une hiérarchie de composition et un héritage multiple ascendant, allant des objets composants vers les objets composés (Colloc J. 1985).

Mode de raisonnement déductif

Le mode de raisonnement déductif s'appuie sur la logique des propositions et le *modus ponens* (Colloc J. 2011).

- Les propositions Vraies permettent de déduire d'autres propositions Vraies lorsque l'on sait que ces propositions sont liées par une règle d'inférence.
- La déduction, qui s'appuie sur un mécanisme d'inférence en chaînage avant, peut permettre d'affirmer des propositions inconnues (*Modus ponens*) ou en chaînage arrière permettre de réfuter des propositions (*Modus tollens*).

La déduction est utilisée dans les syllogismes :

Les hommes ont l'âge, x est un homme donc x a l'âge.

Modus ponens : A est vrai, A⇒B, B est vrai

Modus tollens : B est faux, A⇒B, A est faux

Le mode de raisonnement déductif permet de faire du diagnostic mais il est assez mal adapté au diagnostic de situations complexes.

Selon (Colloc, J. & Boulanger, D. 1987), la Médecine est une science basée sur la pratique et l'observation de l'organisme humain dont la complexité interdit une représentation exhaustive du fonctionnement et, *a fortiori*, de la genèse des maladies. Malgré les progrès de la science, le médecin doit raisonner dans un univers incertain. La séméiologie médicale établit une classification des signes cliniques observés chez le malade. Les théories médicales basées sur l'observation des « Signes cliniques » et sur l'expérimentation peuvent varier selon les écoles de médecine.

- Variation selon les individus : une même maladie peut s'exprimer d'autant de manières différentes qu'il y a de malades.
- Variation suivant la localisation : certaines maladies sont multifocales, elles présentent une forme différente selon l'organe ou l'appareil atteint (la Tuberculose osseuse, pulmonaire), par exemple.
- Variation selon la périodicité : Les maladies sont des processus dynamiques, un tableau clinique n'est qu'un cliché instantané d'une pathologie à un stade donné de son évolution. Cette dernière peut être de type linéaire (la syphilis possède trois phases successives) ou bien cyclique comme l'ulcère

gastroduodénal, l'Herpès. Le temps est donc un facteur prépondérant dans la description des pathologies.

- Variation selon le traitement instauré : un traitement inadéquat ou une autre maladie sous-jacente peuvent interférer et donner un tableau clinique inhabituel, voir faire apparaître une maladie iatrogène.
- La connaissance médicale est en perpétuel changement : la médecine est basée sur l'observation de certains faits corrélés entre eux.

Selon les contraintes liées à la nature du raisonnement, le mode de raisonnement déductif fonctionne avec la règle de transitivité des relations hiérarchiques, afin de traiter la complexité du domaine médical.

Le mode de raisonnement par analogie (ou analogique)

Le raisonnement analogique est utilisé pour étudier plus spécifiquement les hiérarchies de l'ontologie. Le raisonnement analogique exploite les relations de correspondances et de dépendance, afin de transposer des connaissances d'un Univers de problème U1 à un autre U2. (Colloc J. 2011) Les principes sont présentés ci-dessous :

- Les relations de dépendance entre des connaissances sur l'Univers U1 sont généralement antisymétriques.
  $A_1 \rightarrow B_1$ par une relation R dans U_1 s'exprime par le triplet (A_1, R, B_1)
- La relation de similarité notée S permet de mettre en correspondance A_1 et B_1 avec A_2, B_2, sur le fait qu'il existe une similarité entre A_1 et A_2 et B_1 et B_2 respectivement.
- Cette relation de similarité S peut prendre des formes sémantiques très différentes mais c'est généralement une relation symétrique.

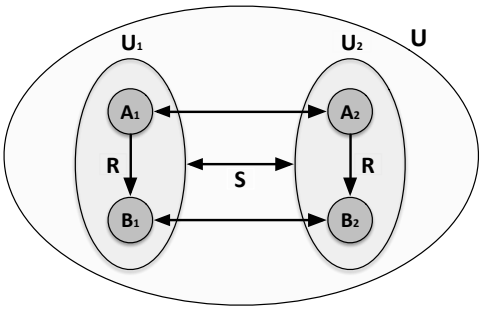


Figure 41 Exemple de Raisonnement par analogie (ou analogique)

Exemple d'application du modèle :

U_1: Mammifères A_1 et B_1 sont unis par la relation **R** de filiation : B_1 est fils de A_1.

U_2 : Oiseaux A_2 et B_2 sont unis par la même relation **R**.

La relation de similarité **S** provient du fait qu'il $\exists$ U / $U_1 \subset U$ et $U_2 \subset U$.

Le raisonnement analogique est le plus naturel et le plus utilisé par l'être humain, mais hélas on ne dispose pour l'instant que de peu de solutions sur le plan informatique pour pouvoir valablement le mettre en œuvre, car le raisonnement analogique a besoin de deux outils : un corpus qui doit contenir beaucoup d'exemples et de cas existants, et la méthode doit être établie entre ce corpus et le système d'aide à la décision.

Dans le cadre de cette thèse, nous avons choisi d'utiliser, d'une part, les concepts des modèles orientés objet et, en particulier, la généralisation/spécialisation de classes d'objet, la composition/décomposition d'objet et, d'autre part, la déduction dans un contexte de logique propositionnelle. Dans nos travaux actuels et futurs, nous envisageons de développer des outils de raisonnement analogiques qui permettront la comparaison d'objets à l'aide de distances sémantiques fondées sur la logique floue.

5.4.2 Hiérarchie de l'ontologie utilisée dans un Système Multi-Agents d'Aide à la Décision (SMAAD)

Nous construisons une hiérarchie initiale du domaine manuellement.

Les textes initiaux ont été sélectionnés manuellement, à partir des résultats de recherche par mots clés, retournée par un moteur de recherche. Les concepts initiaux sont un ensemble de mots-clés représentatifs, extraits de la liste des segments répétés d'un ensemble de textes initiaux. Les relations entre les concepts sont également définies par les ingénieurs des connaissances, en utilisant les concepts de taxonomie (cf. chapitre 2).

Il est difficile de décider quels seront les concepts inclus dans la hiérarchie de spécialisation, leur degré de spécificité et leur granularité. Les principaux critères d'inclusion utilisés sont la fréquence des segments répétés. Un autre critère est induit par la relation de subsomption, nous gardons les concepts les plus généraux dans la hiérarchie initiale (Malaisé, V. 2005).

Exemples

Certains exemples de conceptualisation de la hiérarchie :

(define-primitive-concept **Diagnostic**)

(define-concept **Historique** (AND Patient (SOME hasFréquence Fréquence) (SOME hasAntécédentsFamillaux AntécédentsFamillaux) (SOME hasTraitementPrécédent TraitementPrécédent)))

Selon la définition, « Diagnostic » est la superclasse de « PatientCondition », tandis que « [l']Historique » et « [l']Etat» sont ses sous-classes.

Nous présentons également la sous-classe de la classe « Historique » ici, qui comprend : « Fréquence », « AntécédentsFamillaux » et « TraitementPrécédent ». Toute les hiérarchies sont presentées à la suite.

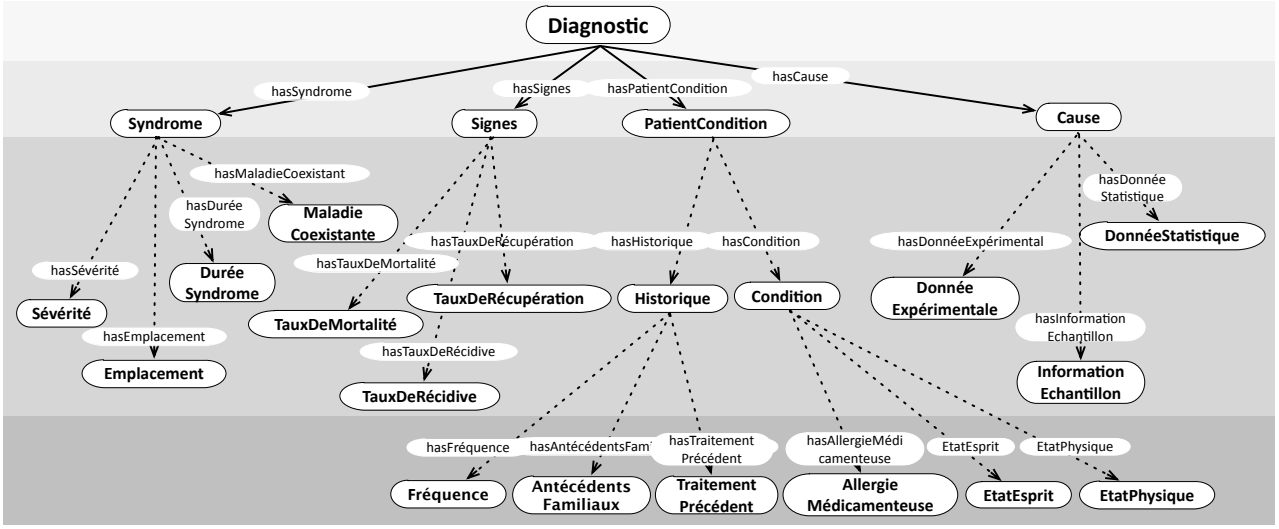


Figure 42 Hiérarchie de diagnostic

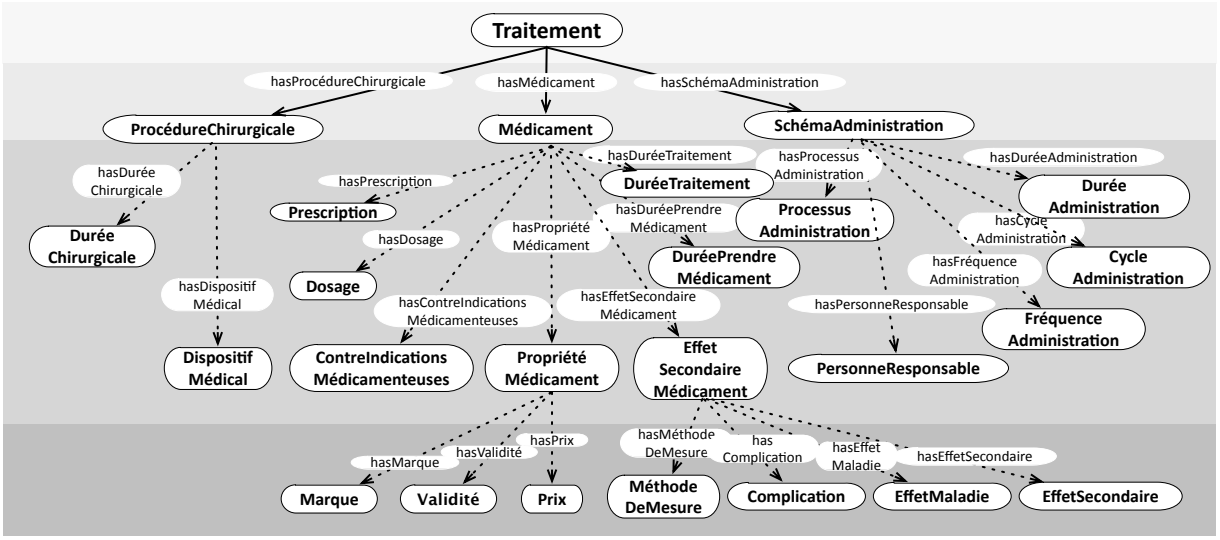


Figure 43 Hiérarchie de traitement

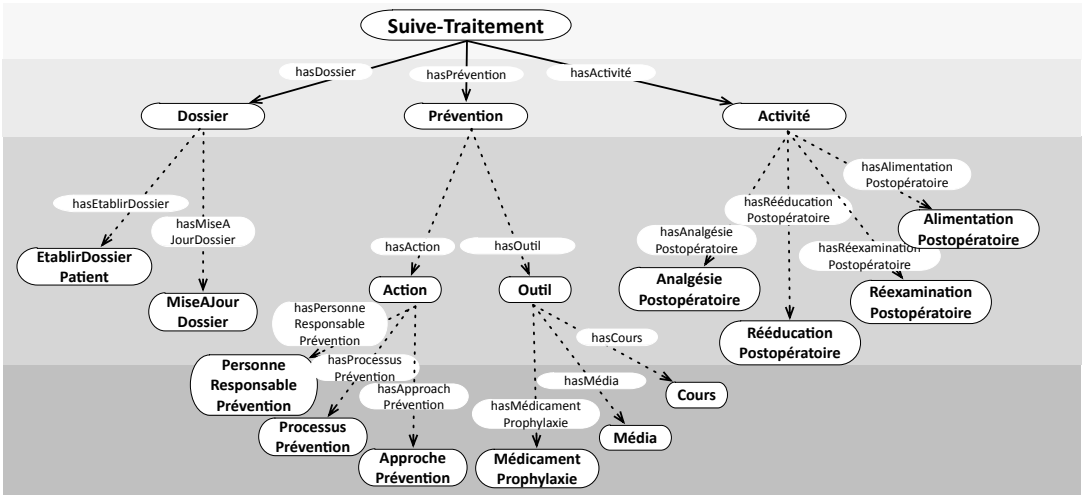


Figure 44 Hiérarchie de suivi-thérapeutique

Il faut également détailler la relation et les coopérations entre les classes diagnostic, pronostic, traitement, et suivi-thérapeutique.

Les hiérarchies de composition des objets cliniques : diagnostic, pronostic, traitement et suivi thérapeutiques sont décrites dans les figures précédentes (42 à 44) conformément aux travaux de (Colloc, J. & Sybord, C. 1997) (Colloc J. & Sybord C. 2003) (Shen, Y. et al. 2012) : l'AEF clinique générale et l'AEF spécifiques Δ, Π, Θ et RàPC présentés par (Colloc, J. & Sybord, C. 1997). Dans un SMA supervisé, l'agent superviseur gère la communauté des agents diagnostic Δ, pronostic Π, thérapie Θ et suivi thérapeutique SΘ (figure 45). Tous les agents encapsulent des modèles de connaissances et de décision différents. Le superviseur utilise un Automate d'Etat Fini Clinique Général (AEFCG) qui définit l'ensemble des étapes nécessaires à la démarche de décision clinique qui charge et déclenche les automates d'état finis contrôlant des tâches décisionnelles spécifiques AEFS Δ, Π, Θ, RàPC.

Des automates d'états finis (AEF), chargés de planifier et d'organiser les tâches de décision, assurent la cohérence des transactions.

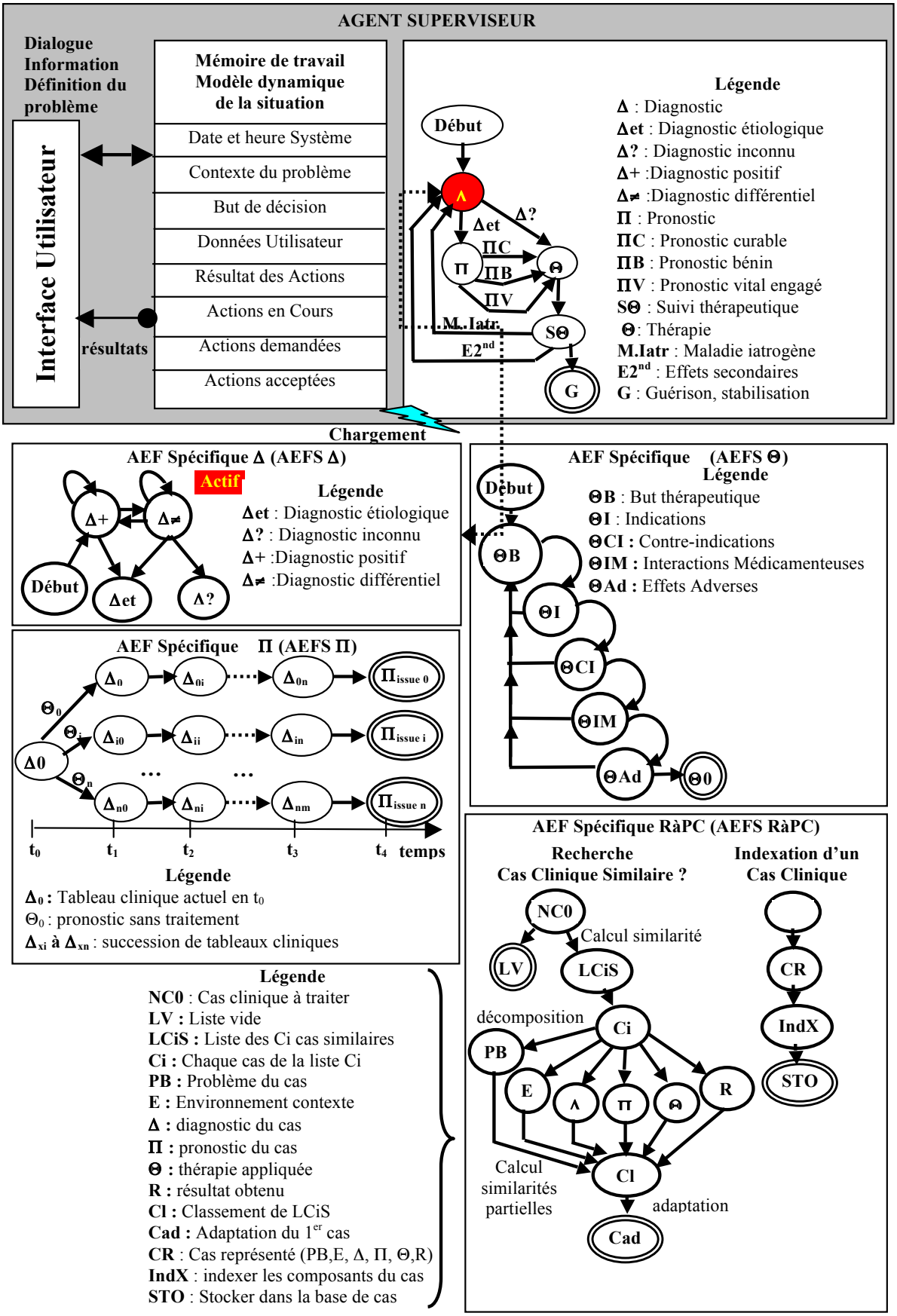


Figure 45 Relations et coopérations entre les classes diagnostic, pronostic, traitement, et suivi-thérapeutique (Colloc, J. & Sybord, C. 1997)

Selon la théorie de Colloc & al (2013), nous définirons les flux de données dans ou entre les classes diagnostic, pronostic, traitement, et suivi-thérapeutique. Les lignes et flèches bleue indiquent les flux de données au sein de chaque classe, Les flèches rouges présentent les transferts de données entre classes.

Il existe quatre types de diagnostics différents. Après avoir vérifié le type de diagnostic (e.g. le diagnostic étiologique), la classe pronostic transfère les résultats du pronostic à la classe de traitement. La classe SchémaAdministration surveille les classes « médicament » et « procédure chirurgicale ». Avec le diagnostic, le pronostic du cas, et le traitement appliqué, on peut établir le dossier du patient qui, en retour, peut fournir de nouvelles informations à différents classes.

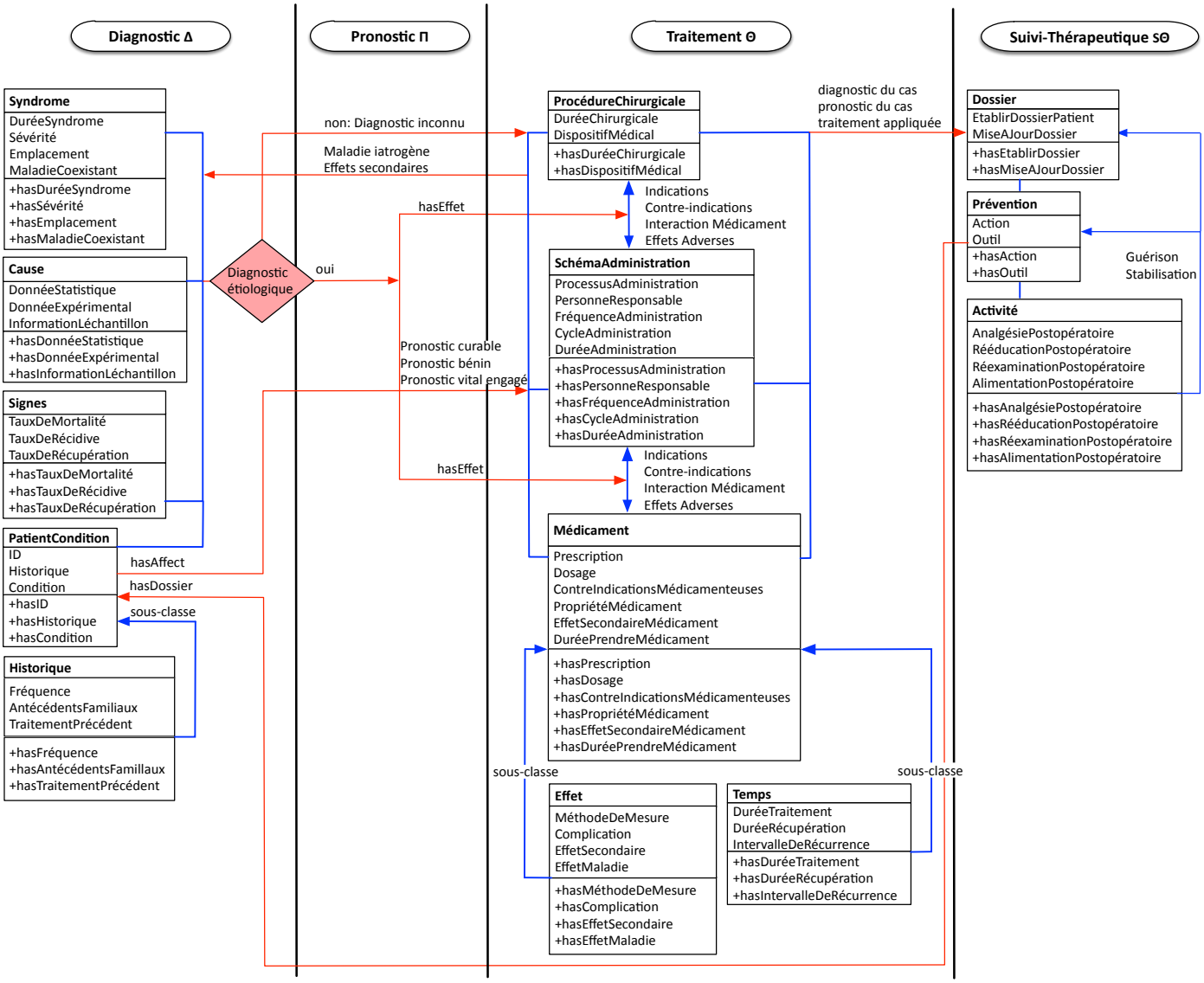


Figure 46 Relations et coopérations entre les classes diagnostic, pronostic, traitement, et suivi-thérapeutique

5.4.3 Alignement de nouveaux termes à hiérarchie existante

On réalise l'alignement de nouveaux termes à la hiérarchie existante à travers trois approches :

i. On établit et étend les hiérarchies selon les règles d'établissement de la hiérarchie.
ii. Le programme « Synonyme » vise à chercher les synonymes de mots cibles existant dans le dictionnaire (e.g. Larousse). Les synonymes ainsi trouvés sont placés dans le même niveau de hiérarchie que le mot cible.
iii. Enfin, les ingénieurs de connaissance établissent les hiérarchies (de spécialisation et de composition) manuellement.

5.5 Construction d'ontologies avec les instances

Une classe doit être instanciée pour être utilisée. On peut créer un objet appartenant à cette classe avec l'opérateur new. Les instances sont les objets instanciés finaux d'une classe. Les instances sont liées à leur classe par un lien "est-une instance de". Une instance possède un identifiant et des attributs uniques. Chaque attribut comporte une liste de facettes donnant la valeur et les caractéristiques de l'attribut.

L'exemple suivant présente le constructeur construit des objets à partir d'une classe d'objets. Il y a 3 instances de ID : Jamie, Alice, et Lorraine sont des objets de la classe ID, on peut en créer n (n est fini).

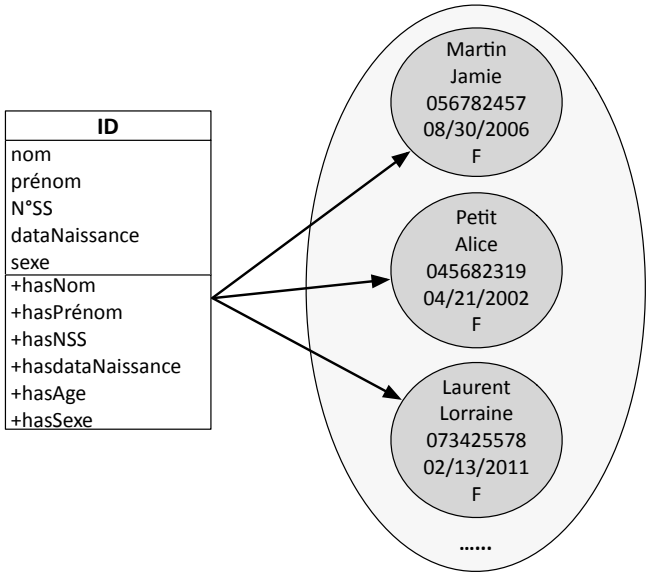


Figure 47 Exemple : Instanciation des objets à partir d'une classe d'objet

Nom des attributs des objets instances	Description	Type de valeur	Cardinalité	Unité de mesure	Précision	Valeur Intervalle
Antibiotiques nom	Le nom de l'antibiotique	*String*	(0,N)			
Prix de l'antibiotique	Le prix de médicament antibiotique	*Float*	(0,1)	euros	0,01	0-
Validité de l'antibiotique	La durée de validité de l'antibiotique	Date	(0,1)	jours		

Certaines instances extraites de corpus sur la diarrhée aiguë sont présentées ci-dessous, pour établir un glossaire des instances.

Class	Instance
Médicament	Le lopéramide (et son dérivé, l'oxyde de lopéramide), le racécadotril, le salicylate de bismuth, la leucine, les composés antidiarrhéiques, la réhydratation orale, Theriaque®, lactobacillus acidophilus en sachet-dose, lactobacillus acidophilus, Saccharomyces boulardii, le glucose, lactoprotéines méthyléniques…
Bactérie	Salmonella typhi, campylobacter, clostridium botulinum, shigella, enterobacter, pseudomonas aeruginosa, …
Syndrome	Nausées vomissements, douleur abdominale, pâleur, ictère, oligoanurie, diarrhée sanglante…
Patient	Jamie MARTIN, Alice PETIT, Laurent LORRAINE sont des patients
Examens Paraclinique	Tomodensitométrie thoracoabdominale, Scanner, Hématologie, Ionogramme et tous les examens qui peuvent être pratiqués pour connaître l'état du patient.

Tableau 26 Glossaire des instances

5.6 Implémentation et expérimentation de l'ontologie

Dans les paragraphes précédents, nous avons présenté le modèle objet permettant d'exprimer les données et les traitements des classes utiles durant la démarche clinique (diagnostic, pronostic, traitement, suivi thérapeutique), les associations, les hiérarchies de spécialisation et de composition et, enfin, nous avons montré comment instancier ces classes. Ce paragraphe décrit la phase d'exploitation des connaissances pour l'implémentation de l'ontologie.

La modélisation est utilisée lors de la structure d'un type d'agent ontologique. Le travail du concepteur est d'exprimer la réalité dans les termes du modèle, puis de la coder dans la machine.

La modélisation de la réalité comporte 3 étapes. Tout d'abord, il faut analyser les objets du monde réel et leurs interactions (flux de données ou flux de documents, matériels, etc.). Ensuite, on doit concevoir des schémas dans les termes du modèle pour appréhender cette réalité et ses interactions possibles. Finalement, on établit les méthodes correspondant aux besoins en connaissances des utilisateurs lors des différentes étapes cliniques.

La figure 48 montre comment un schéma UML décrivant le domaine de connaissance clinique est analysé sur le plan morphologique, lexical, syntaxique et sémantique pour établir une reconnaissance des éléments utiles dans le texte clinique et instancier des objets UML correspondants.

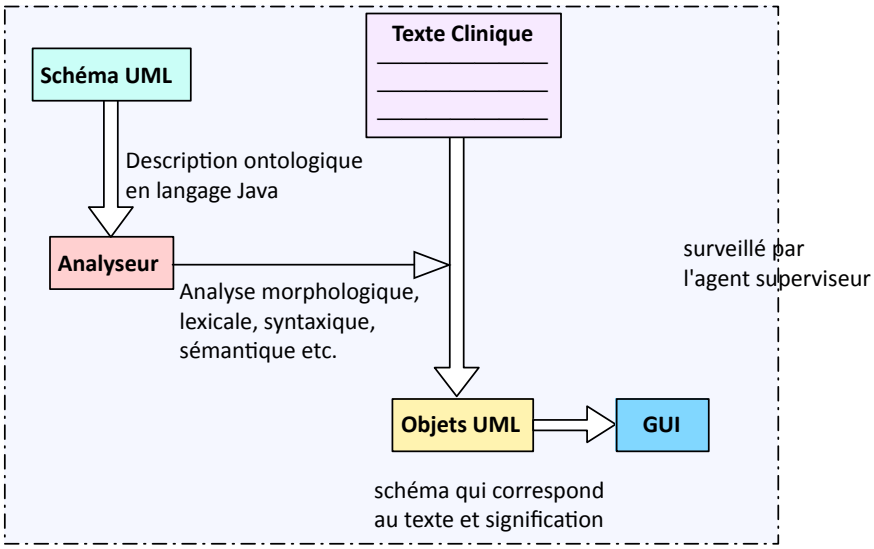


Figure 48 Organiser le contenu de texte à l'objet UML et réaliser la visualisation d'ontologie

Class en-tête	catégorie	groupe	nombre
médicament	nom	objet	singulier
	notions en rapport	**type**	**Super Classe**
	Posologie, Pharmacocinétique, Pharmacodynamie, Indication, Contre-indication, Association déconseillée, Précaution d'emploi, À prendre en compte, Synergie, Potentialisation, Antagonisme	Anesthésiants, Antalgiques, Sédatifs, Antidépresseurs, Anxiolytiques, Psychotropes, Antibiotiques...	traitement

Class en-tête	genre	synonymes	traits
médicament	masculin	drogue, remède, potion	Abstrait
			Concret + comptable✓ – comptable
	Sous-classe	**méthode**	**description annotation**
	Prescription	hasPrescription()	- Il est la sous-classe de « Traitement » - Il a de l'influence de « Pronostic »
	Dosage	hasDosage()	
	Contre Indications Médicamenteuses	hasContreIndications Médicamenteuses()	
	Propriété Médicament	hasPropriétéMédicament()	
	Effet Secondaire Médicament	hasEffetSecondaire Médicament()	
	Durée Prendre Médicament	hasDuréePrendre Médicament()	

Tableau 27 Le prototype de « médicament »

5.6.1 Initialisation de l'ontologie

Les modules TAL décrits ci-dessus sont chargés de traiter les documents et les requêtes des utilisateurs et de les inclure dans le corpus. Dans l'annexe.2, nous présentons toutes les opérations, méthodes et variables chargées d'établir des ontologies exploitables au sein d'un système multi-agents. Voir par exemple « OPERATION-12. Construction des ontologies », en Annexe 2.

Titre	Construction des ontologies
But de la fonction	Le système doit construire l'ontologie automatiquement à partir des termes et associations extraits, synonymes entrée et heuristiques sélectionnées.
Stabilité	Stable
Plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	La construction de l'ontologie est établie automatiquement

Tableau 28 Paramètres et définition de l'opération : « construire une ontologie »

Ingénieur de connaissances	Prototype de construction d'ontologie automatique
1. L'ingénieur de connaissances démarre le « **Modèle de l'Ontologie** » pour la construction de l'ontologie	
	2. Demander de choisir un emplacement pour enregistrer la nouvelle ontologie (« **Dictionnaire ontologique** »)
3. Sélectionner un emplacement pour enregistrer l'ontologie	
	4. Demander d'entrer un nom pour la nouvelle ontologie
	5. Construire l'ontologie avec : - Les termes des OPERATION-2 et 4, - Les synonymes des OPERATION-5 et 6, - Les associations des OPERATION-3 et 7, - Les heuristiques d'OPERATION-9, - Les schémas et les règles OWL d'OPERATION-10.
6. Amélioration et autorisation de **l'ingénieur de connaissances**	
	7. Enregistrer le nom de l'ontologie avec les termes, les associations et les synonymes générés

Tableau 29 Séquence de réponse et stimulus de construction d'une ontologie

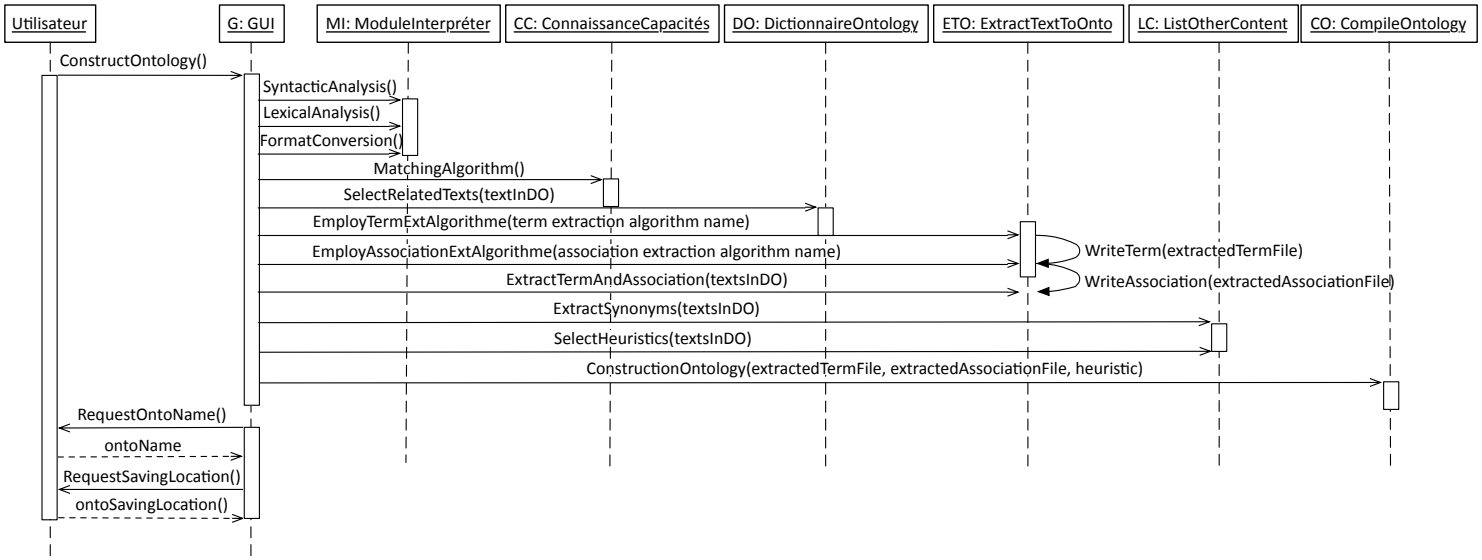


Figure 49 Diagramme de séquence : Construction du « Dictionnaire ontologique »

Voici un diagramme séquence UML à propos de la construction d'ontologies.

- Le module GUI visualise l'ontologie avec des codes JAVA.
- Le ModuleInterpréter vise à réaliser l'analyse lexicale et syntaxique, avec les méthodes SyntacticAnalysis() et LexicalAnalysis().
- La méthode FormatConversion() permet de convertir le format texte en format OWL, avec l'algorithme.
- Le module ConnaissanceCapacités vérifie si le texte ou la requête existe déjà dans le corpus, avec la méthode MatchingAlgorithm().
- Autres méthodes : ExtractTermAndAssociation(), l'algorithme d'alignement travaille avec ses variables pour l'extraction et la génération de termes et les associations (cf. notre présentation de « Chemin d'accès »).

Ce processus permet à l'utilisateur de choisir les méthodes et les heuristiques les mieux adaptées à la génération de l'ontologie choisie. Après avoir proposé d'entrer un nom et un emplacement, pour enregistrer une nouvelle ontologie, le système construit l'ontologie selon la méthode choisie et il enregistre sa description dans un fichier OWL.

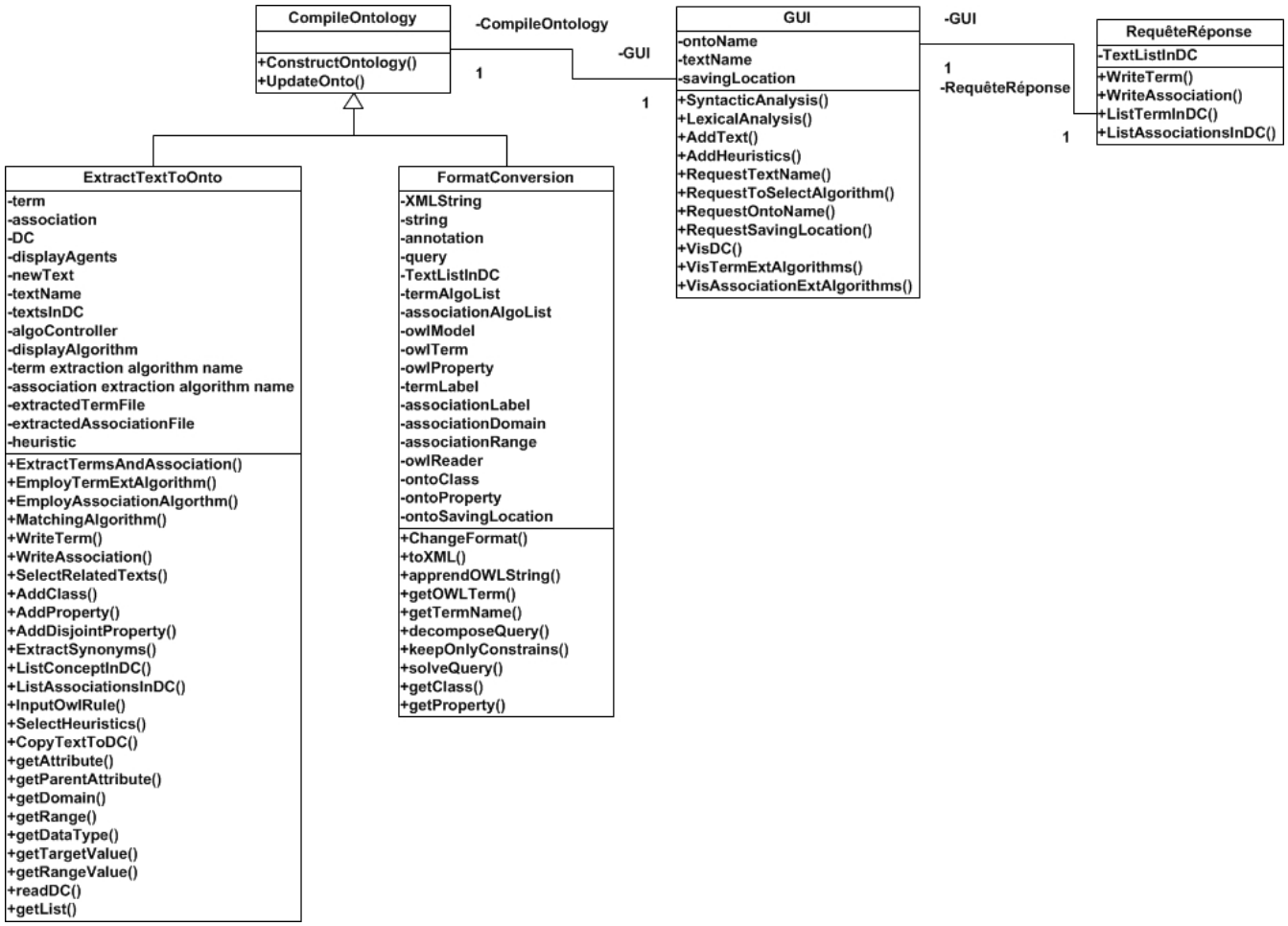


Figure 50 Diagramme de classe : Système prototype pour la construction d'ontologies et les requêtes réponses

Le diagramme de classes vise à présenter le système prototype pour la construction d'ontologies et les requêtes réponses.

Une ontologie s'établit en deux étapes : il faut tout d'abord extraire les termes, associations et les textes à SMA pour étendre l'ontologie. Il faut également réaliser le format de conversion de textes à OWL.

Pour obtenir la réponse, le programme « alignement » va rechercher les termes et les relations nécessaires dans le « Modèle de l'Ontologie ». Il s'agit de vérifier si la connaissance correspondant à la requête est disponible dans notre corpus. Ensuite, le SMA transfère la réponse à l'utilisateur.

Dans ce système multi-agents, on peut interroger l'ontologie pour récupérer les connaissances. Ces requêtes visent à connaître :

- les actions nécessaires dans le processus de construction du « Dictionnaire ontologique » : CompileOntology, ExtractTextToOnto, FormatConversion, AnalyseLinguistics
- les variables et les méthodes nécessaires dans ce processus de construction du « Dictionnaire ontologique »

ExtractTextToOnto et **ModuleInterpréter** :

La méthode *ExtractTermsAndAssociation()* vise à extraire les nouveaux objets et les associations entre objets, à partir d'un corpus de textes, et à expliquer les objets utilisés pour décrire le déroulement clinique habituel. Cette étape peut être réalisée à l'aide des méthodes *LexicalAnalysis()* et *SyntacticAnalysis()* qui déclenchent « [l']analyseur morphologique » (lemmatisation de la langue française) et « [l']analyseur synonyme », pour l'analyse lexicale, et le « POS Disambiguïsation » pour l'analyse syntaxique.

Rappelons que dans le cadre de l'approche objet et dans les langages C++ ou Java, on appelle méthode tout traitement (ou fonction) associé à une classe d'objets et exécutable par chaque objet instance de cette classe.

Les méthodes *LexicalAnalysis()* et *SyntacticAnalysis()* correspondent à « Méthode d'analyse lexicale » et « Méthode d'analyse syntaxique » d'un métalangage pour le type d'objet application.

La méthode *getAttribute()* est utilisée pour obtenir des attributs de la classe ciblée dans la hiérarchie.

Les méthodes *EmployTermExtAlgorithm()* et *EmployAssociationAlgorithm()* fonctionnent avec la variable « algoController », pour le contrôle de l'algorithme, qui contient tous nos algorithmes et programmes pour l'analyse et l'extraction des nouveaux objets et associations entre objets. Il fournit des programmes et les algorithmes adaptés au système, à la demande de l'utilisateur et/ou conformes au réglage par défaut défini par l'ingénieur de connaissances.

FormatConversion :

La méthode *FormatConversion()* effectue la conversion de format du texte en OWL. Il faut d'abord décomposer la requête entrée par la méthode decomposeQuery(), puis déterminer les titres, les notes explicatives et tous les

autres aspects stockés dans la chaîne de caractère en OWL, à l'aide de la méthode *apprendOWLString()*.

CompileOntology :

La méthode *ConstructOntology()* démarre le processus de construction et établit une ontologies, avec la liste des termes et des associations correspondantes, les heuristiques sélectionnées et les schémas et règles OWL entrées. L'ontologie résultante est enregistrée dans un fichier OWL.

La méthode *UpdateOnto()* correspond à l'étape de réutilisation qui détecte tous les paramètres qu'il faut mettre à jour dans l'ontologie existante.

- Elle définit les actions à effectuer pour visualiser le résultat à l'aide de l'interface graphique GUI.

Pour les aspects liés au processus de décision clinique en médecine :

- les structures qui participent au processus clinique sont : « Diagnostic », « Pronostic », « Traitement » et « Suivi-thérapeutique »
- les personnes évaluées dans le processus clinique sont les patients : « Patient »
- Comment trouver la cause potentielle de la maladie : DonnéeStatistique, DonnéeExpérimental, InformationLéchantillon, …

etc.

Pour réaliser automatiquement les étapes mentionnées ci-dessus, on établit une ontologie pathologie en code JAVA avec les classes : « attribut, objet, action, patient, diagnostic, pronostic, traitement, suiveThérapeutique, pathologie, signe, médicament, traitementChirurgicale, autre » etc. (*cf.* annexe.4 Programme.11)

Prenons la classe « attribut » comme exemple, nous ajoutons les objets «transmissible, infectieux, maligne, sévère, résistant, positif, progressif, viral, mortel, chronique, aigu, soudaine, rapidement, occasionne, microscopique, gangreneux, bactérienne, septicémique, caséeux, osseux, immunodépressif, nécrotique, aplasique, tumoral, nerveux, rénal, respiratoire, hématopoïèse, cardiovasculaire, abdominal, interne, ventre, digestif, buccal, occulo_nasal, plusieurs_moyens, multi_factoriel » comme les sous-classes à la classe « attribut ».

Cette étape combine les classes des différents niveaux des hiérarchies présentées en UML (*cf.* 5.4.2), à code JAVA.

Avec les relations extraites et les hiérarchies définies, nous établissons une ontologie pathologie et nous enregistrons cette nouvelle ontologie dans des fichiers

OWL, puis nous effectuons sa visualisation, produite à l'aide du GUI développé en JAVA. (*cf.* annexe.4 Programme.13)

5.6.2 Extension de la hiérarchie - la génération de nouveaux savoirs (requêtes) à l'ontologie

Ce paragraphe décrit la méthode utilisée pour cartographier les concepts de candidats dans le domaine de l'ontologie (Figure 51). Sa conceptualisation repose sur la cartographie des concepts candidats à la hiérarchie de cette dernière.

Le contenu de la hiérarchie est étendu de manière dynamique, de nouveaux documents prélevés sur le Web ou préparés par un expert humain, ainsi que des requêtes formulées par des utilisateurs, contenant des instances de concepts inconnus sont ajoutés à la base de connaissances. L'objectif du système est d'acquérir ces nouveaux concepts et de les intégrer aux hiérarchies de domaine existant.

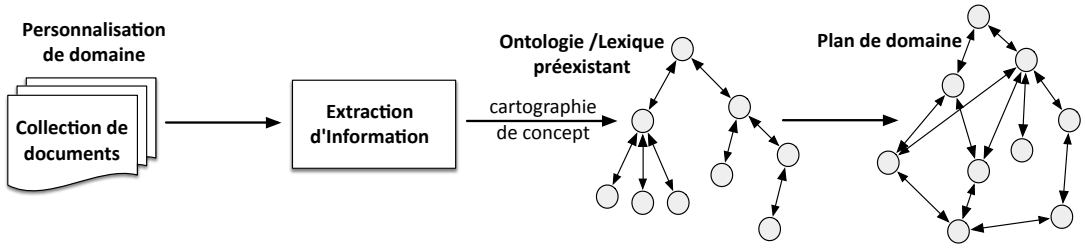


Figure 51 Étapes de l'alignement entre le contenu et le domaine spécifique

Les documents et les requêtes des utilisateurs sont traités comme un ensemble de mots clés. Le système extrait un certain nombre de documents contenant ces mots-clés. Ces documents sont ensuite traités par des modules TAL, pour affiner les résultats de recherche et identifier de nouveaux concepts. Quand un nouveau document est indexé, le système en identifie les concepts et complète le modèle de domaine :

I. Le document d'entrée ou requête d'utilisateur est d'abord traité par le «Compteur de mots », pour l'extraction des plus fréquents avec leurs contextes (*content words*). Les contextes des mots sont ensuite traitées par les modules TAL pour établir indexer de nouveaux concepts.
II. Les informations lexicales sont affectées à chaque mot.
III. Le POS *tagger* et le *tagger* de sens (fonctionne avec la description logique) étiquettent les mots avec les descriptions conceptuelles. Leur existence dans la hiérarchie du domaine est vérifiée. S'ils n'existent pas, ils sont classés dans la hiérarchie existante.

IV. Les règles syntaxiques et les règles heuristiques sont utilisées pour encoder les connaissances syntaxiques.

V. Les nouveaux concepts, avec les résultats de l'analyse linguistique (morphologie, lexicale, syntaxe, *tagging*, synonyme, etc.) sont validés par les ingénieurs des connaissances, puis ajoutés à la hiérarchie du domaine.

Lors du traitement du texte fourni par l'utilisateur, les instances des concepts sont extraites de la hiérarchie du domaine. Si de nouveaux documents ont été traités, alors la hiérarchie de domaine peut être enrichie de nouveaux concepts.

Toutes les étapes présentées ci-dessus nécessitent l'apprentissage de nouveaux termes, synonymes, associations, etc. Elles sont détaillées dans les Annexes 2 et 3.

Afin de réaliser ces étapes automatiques dans notre système multi-agents, on écrit les codes java pour ajouter, supprimer, modifier ou énumérer des concepts, des relations, des hiérarchies et des instances d'ontologie. (cf. l'annexe.4 Programme.12)

5.7 Synthèse

Le chapitre 5 a décri un dispositif pour la représentation et le classement automatiques des mots et de leurs comportements, et il a montré comment exprimer les connaissances à partir de savoirs extraits d'un corpus. Nous envisageons de construire des représentations informatiques des mots et d'attacher à ces objets leurs comportements syntaxiques et sémantiques en ne prédéterminant pas complètement les savoirs à représenter. Pour cette représentation, nous utilisons le corpus UMLS/MEDLINE, à partir duquel sont extraits des savoirs pour construire des structures de représentations lexicales sous la forme de prototypes.

Les détails de l'acquisition de concepts et associations en corpus pour la construction d'ontologies sont présentés ci-après :

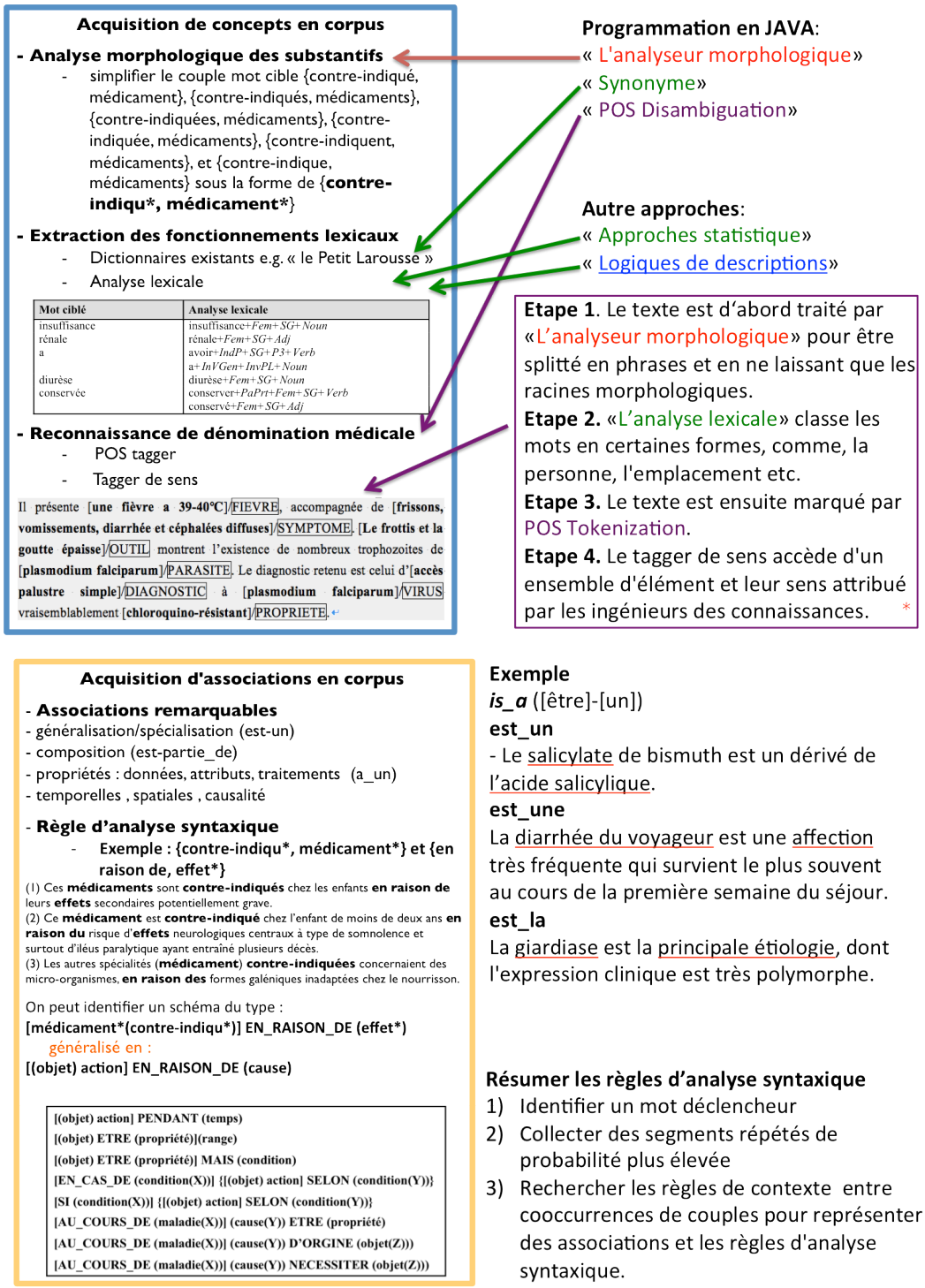


Figure 52 Bilan d'acquisition de concepts et associations en corpus

Les résultats obtenus ont mis en évidence la nécessité d'une méta-connaissance pour la description des objets. Les problèmes posés mettent en avant un point fondamental pour la résolution d'un traitement automatique des faits de langue (également pour la construction du sens de la classification des connaissances linguistiques). Le TAL exploite de nombreuses connaissances, et méta-connaissances

tout en reconstruisant dynamiquement de nouvelles connaissances. Ce dernier point est illustré dans notre cadre linguistique. Une meilleure compréhension des mécanismes humains d'élaboration des connaissances pourrait améliorer ce procesus. Cependant, il faut poursuivre la recherche sur l'utilisation de méta-connaissances pour le traitement automatique du langage naturel, quitte à limiter le champ d'étude sur des phénomènes restreints permettant d'affiner encore les tâches à effectuer.

Chapitre 6 APPLICATION DES ONTOLOGIES POUR AIDER À LA DECISION CLINIQUE

Avant les années 80s, 70 % des applications informatiques s'appliquaient surtout à la gestion, les méthodes de conception étaient souvent orientées sur la gestion. Cependant, les nouveaux types d'application sont apparus sur le domaine CAO, FAO, Simulation (de circuits, de programmes), Cartographie, Architecture etc. Une forte interactivité avec l'utilisateur, une représentation et une manipulation d'objets complexes, des temps de réponse courts, et des interfaces homme-machines (IHM) conviviales caractérisent ces applications.

Les propriétés des approches orientées objet ont surtout été utilisées en génie logiciel : La réutilisation intensive des classes (types) d'objets existantes, pour construire de nouvelles classes ou (types) d'objets à l'aide de la spécialisation, a conduit à de nouvelles conceptions et réutilisations.

Par ailleurs, la composition d'objets a permis de mettre en œuvre des interfaces de haut niveau et d'établir des standards d'environnement (CORBA) (Siegel, J. 2000), de tenir compte de critères ergonomiques, de fournir des bibliothèques de composants logiciels, d'objets interfaces (ascenseurs, fenêtres, boutons) et d'objets métiers.

Par ailleurs les approches orientées objet ont permis la modélisation de connaissances sous la forme d'objets connaissances en interaction (Martin, J., & Odell, J. J. (1994).

La difficulté de la prise de décision, en particulier en médecine et en santé publique, provient de l'incertitude, au sens large: l'incertitude sur les connaissances, l'incertitude sur les faits et l'incertitude du langage employé., en particulier dans les sciences humaines et sociales, où la notion d'incertitude existe de fait, puisqu'il s'agit de l'humain. depuis le diagnostic, jusqu'au suivi du patient, en passant par le pronostic et le traitement, il s'agit de prélever un maximum d'indices sémantiques médicaux pertinents.

Un type d'agent ontologique utilise des connaissances du domaine, médical ici, et repose sur l'analyse sémantique et l'organisation des connaissances utiles dudit domaine, en vue de la conception d'un système d'interrogation robuste. L'agent ontologique est utilisé pour étendre la hiérarchie de domaine, ainsi que l'interprétation des requêtes des utilisateurs.

Dans ce chapitre, un exemple illustre le fonctionnement de l'agent. L'objectif est de présenter les applications d'ontologies dans les systèmes d'aide à la décision clinique, en montrant comment elles tirent parti et profit d'une ontologie.

6.1 Caractéristique des systèmes d'aide à la décision clinique

La médecine est une discipline scientifique mais aussi une discipline d'action, du domaine de l'humain, qui requiert souvent une prise de décision. Ce processus résulte de la confrontation d'un problème réel à l'expérience acquise et à un corpus de connaissances théoriques. (Informatique médicale 2014)

Il faut détailler les caractéristiques de notre système d'aide à la décision afin d'analyser et spécifier le type de problème posé, le mode d'intervention, le domaine d'études, la nature de l'interaction, et les méthodes employées.

Dans le système multi-agents proposé par notre équipe, des bases de données et des dossiers informatisés interagissent ensemble pour faciliter la prise de décision, en améliorant l'accès aux données pertinentes à travers des interfaces définies. Précisions ici, d'un point de vue linguistique, que nous ne travaillons pas sur un corpus unifié mais sur plusieurs corpus d'informations, inter-corrélés, dans le cadre du système multi-agents. Dans ce contexte, nous mettons l'accent sur les recherches de deux types de problèmes avec l'aide du raisonnement à partir de cas (RàPC) :

- problème de classement ou de diagnostic : après avoir recherché l'incertitude sur la situation réelle de l'objet d'étude (patient, organe, population), il faut décider les mesures efficaces ;
- Il s'agit donc de retrouver les cas connus similaires dans une base d'expérience clinique (appelée encore base de cas) en comparant les circonstances d'apparition, le pronostic, l'évolution, les stratégies thérapeutiques employées et leurs résultats (Colloc, J. & Léry, L 2008).
- À propos de l'adaptation, le but est de choisir la démarche la plus efficace (par exemple, une stratégie thérapeutique) et de l'adapter au cas présent selon l'objectif thérapeutique et les contraintes afférentes : l'état d'esprit du patient, les contre-indications médicamenteuses, les effets secondaires potentiels des médicaments, etc.). Ces opérations nécessitent une analyse précise des termes et des connaissances disponibles.

Plusieurs modes de fonctionnement de ces programmes informatiques sont possibles, tels que le mode **semi-actif** qui correspond à un système dont le déclenchement automatique répond à une intervention humaine, et le mode autonome, à déclenchement automatique, sans intervention ou surveillance du décideur. Dans nos recherches, nous utilisons le mode interactif qui suppose l'intervention explicite de l'utilisateur pour décrire le problème (par exemple, les symptômes de la maladie, l'état du patient etc.) et interroger le système. En retour, le système consultant fournit une conclusion ou un conseil (par exemple : un diagnostic, un pronostic, des médicaments de cas similaire etc.). Notre système multi-agents permet de commenter ou de critiquer la conclusion, en indiquant les failles du raisonnement.

Utilisateurs et populations cibles

L'objectif d'aide à la décision clinique consiste a fournir de meilleurs soins, à élever le niveau de la santé individuelle et collective. Nous pouvons identifier les différentes catégories d'utilisateurs :

- Médecin généraliste : accès aux bases de données, formation.
- Médecin hospitalier : accès aux bases de données, responsable de fournir les informations en retour aux systèmes d'aide à la décision, afin d'élever la qualité de surveillance automatique du traitement et d'aide au traitement.
- Décideur : accès aux bases de données, responsable d'améliorer les systèmes d'aide à la décision en santé publique.
- Population générale : éducation (pré-consultation, prévention).
- Étudiants : enseignement, formation : apprentissage, simulation de cas cliniques.

6.2 Modélisation du processus de décision médicale

6.2.1 Modèle de la démarche clinique

Le système multi-agents (SMA) que nous proposons s'appuie sur des travaux précédemment effectués par l'équipe (Colloc, J. & Sybord, C. 1997) (Colloc J. & Sybord C. 2003) et ceux d'autres auteurs (Montani S. et al. 2001) (Stanley Y.W. et al. 1995). Dans les SMA non-supervisés, la communication entre agents a lieu à l'aide de langages : ACL ou KQML (Labrou, Y. et al. 1999). Dans un SMA supervisé, l'agent superviseur gère la communauté des agents (Colloc J. 1985) (figure 45, p. 156).

Des automates d'états finis **(AEF)**, chargés de planifier et d'organiser les tâches de décision assurent la cohérence des transactions. Le superviseur utilise un Automate d'Etat Fini Clinique Général (**AEFCG**) qui définit l'ensemble des étapes nécessaires à la démarche de décision clinique ; il charge et déclenche les automates d'état finis contrôlant des tâches décisionnelles spécifiques : AEFS Δ,Π,Θ, RàPC (figure 53). L'AEFS RàPC offre un mode particulier de consultation et de restitution des cas cliniques traités. Il constitue la mémoire et l'expérience du SMAAD. Ces AEFS sont chargés à partir d'une base (figure 2 et 53). La mémoire de travail décrit la situation courante et recueille les résultats intermédiaires. L'AEFCG établit l'ordonnancement de l'ensemble du processus de décision. Chaque AEFS gère le déroulement d'une étape clinique spécifique : diagnostic Δ, pronostic Π, thérapie Θ, suivi thérapeutique SΘ (non représenté par concision). Ces agents encapsulent des modèles de connaissances et de décision différents. Le superviseur assure le dialogue avec l'utilisateur. Il contrôle la prise en charge et l'exécution des tâches cliniques par les agents disponibles compétents. Un agent doit abandonner une tâche si le contexte ou l'objectif actuel fixé par le superviseur est modifié : une solution trouvée par un ou plusieurs autres agents a été validée par l'utilisateur, le délai imparti a expiré. Chaque agent utilise ses connaissances réflexives (figure 3 et 4) pour déterminer s'il peut participer à la tâche demandée par le superviseur. Son degré de spécialisation, la nature de la tâche, les connaissances disponibles dans la base, le mode de raisonnement interviennent dans ce choix. Si l'agent accepte, le superviseur l'inscrit dans les actions en cours. L'agent devient actif. Il consulte les données de l'utilisateur dans la mémoire du superviseur, il sollicite des informations *via* l'interface utilisateur. Les réponses sont classées et disponibles dans la mémoire de travail du superviseur pour tous les agents activés par la tâche courante. Lorsque l'objectif est atteint, le superviseur charge l'AEFS de l'étape suivante conformément aux transitions de l'AEFCG (figure 53).

Notre approche définit de manière simple, robuste et modulaire, d'abord, l'ensemble du protocole décisionnel clinique par un AEFCG, puis chaque tâche spécifique par un AEFS Δ, Π, Θ, SΘ (Colloc, J. & Sybord, C. 1997). Le développement de nouveaux AEFCG et AE facilite l'adaptation du SMAAD à de nombreux domaines où un protocole clinique est pertinent. Ainsi, le dialogue avec

l'utilisateur peut avoir lieu *via* une interface (figure 3) et la gestion de la cohérence dans un SMA supervisé est plus facile. Les inconvénients proviennent de la surcharge de l'interface du superviseur et d'une relative diminution de l'autonomie des agents. Les figures 3 et 4, décrivent le Type d'Agent Clinique Général (TACG), dont les modules sont fonctionnellement liés les uns aux autres, et assure la communication, la négociation, la décision, l'évaluation du contexte (exprimé par le superviseur) et la gestion des tâches.

6.2.2 Modèle de décision fondée sur la démarche clinique

L'analyse de décision est une méthode probabiliste et quantitative destinée à la modélisation de problèmes, dans des situations d'incertitude. Pour prendre une décision, tout d'abord, nous recueillons des informations nécessaires pour structurer le problème, puis évaluer les conséquences des alternatives, et enfin décider d'une action. Le but de l'analyse de décision est de clarifier les dynamiques et les compromis impliqués dans la sélection des indices à l'origine d'une stratégie dans un ensemble d'alternatives.

Le modèle de décision est choisi pour représenter le problème clinique. Le modèle de décision requiert d'émettre une série d'hypothèses pour la modélisation ; hypothèses qui doivent être assez simples pour être comprises, mais suffisamment complexes pour circonscrire l'essentiel du problème. L'analyse de décision médicale est une méthode formelle de combinaison d'indices pour définir clairement les alternatives, les événements et les résultats. Il s'agit d'un processus d'acquisition des données utiles, afin d'aider les soignants à prendre des décisions médicales correctes dans un contexte d'incertitude (chaque cas possède une évolution unique) et de responsabilité (bénéfices, risques, coût) des actes effectués.

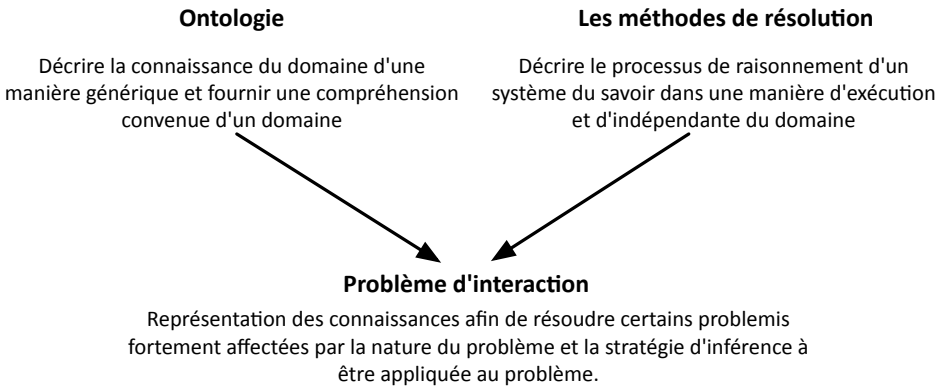


Figure 53 Composants de connaissances réutilisables (Bylander Chandrasekaran, B. 1988)

Différents modèles de connaissances ont été proposés pour l'élaboration de décisions médicales : les règles de production modélisent des heuristiques, les objets connaissances décrivent des situations typiques, les données statistiques (population, ex : épidémiologie), les bases de données, et les bases de cas (RàPC). Il s'agit de réfléchir à comment utiliser ces modèles conjointement.

La figure ci-après explique le principe d'un SMA d'aide à la décision clinique (Sybord, Colloc, 1997). L'agent cas clinique représente l'état du patient. L'agent diagnostic, pronostic, traitement et suivi-thérapeutique montre l'effet de la thérapie sur le pronostic, et l'agent RàPC stocke et indexe des cas. L'agent superviseur surveille toutes les opérations, tandis que l'agent ontologique fournit l'axiomatisation du domaine clinique.

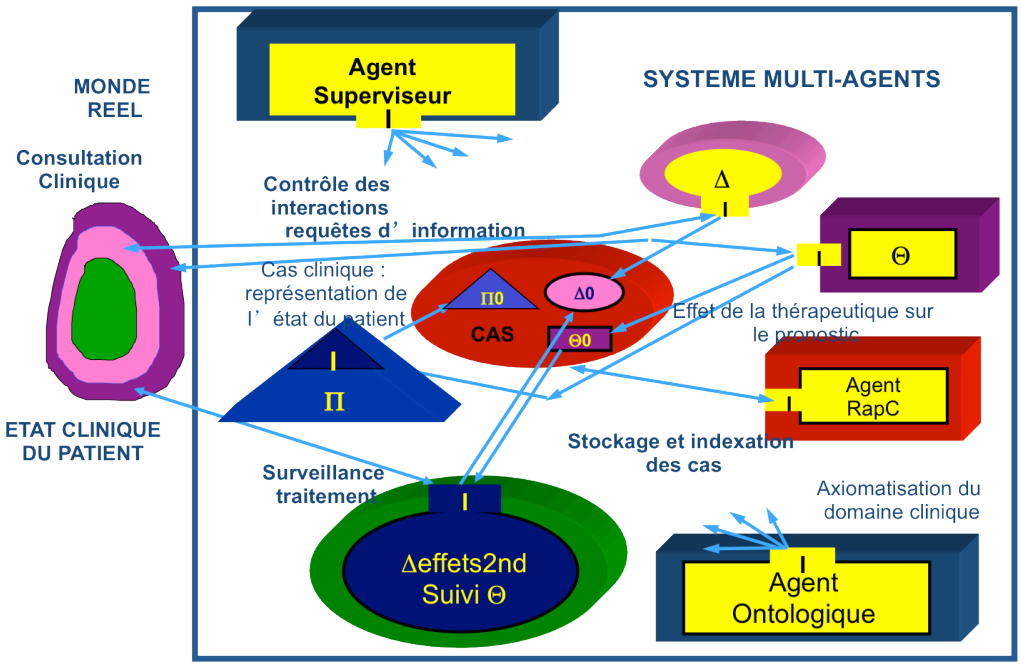


Figure 54 SMA d'aide à la décision médicale (Colloc J. & Sybord C. 2003)

Étapes du processus de décision

Pour expliquer les étapes du processus de décision, le cancer de l'estomac est pris comme exemple, ici. À propos du diagnostic, pronostic et traitement de ce cancer, il faut combiner plusieurs spécialités médicales pour planifier et utiliser de manière rationnelle les traitements tels que la chirurgie, la chimiothérapie, la radiothérapie et la thérapie ciblée. Selon les cas, les objectifs sont : guérir (éradiquer totalement les cellules cancéreuses) ou contrôler la tumeur, prolonger la vie des patients et améliorer leur qualité de vie physique, psychique, autrement dit, personnelle et sociale.

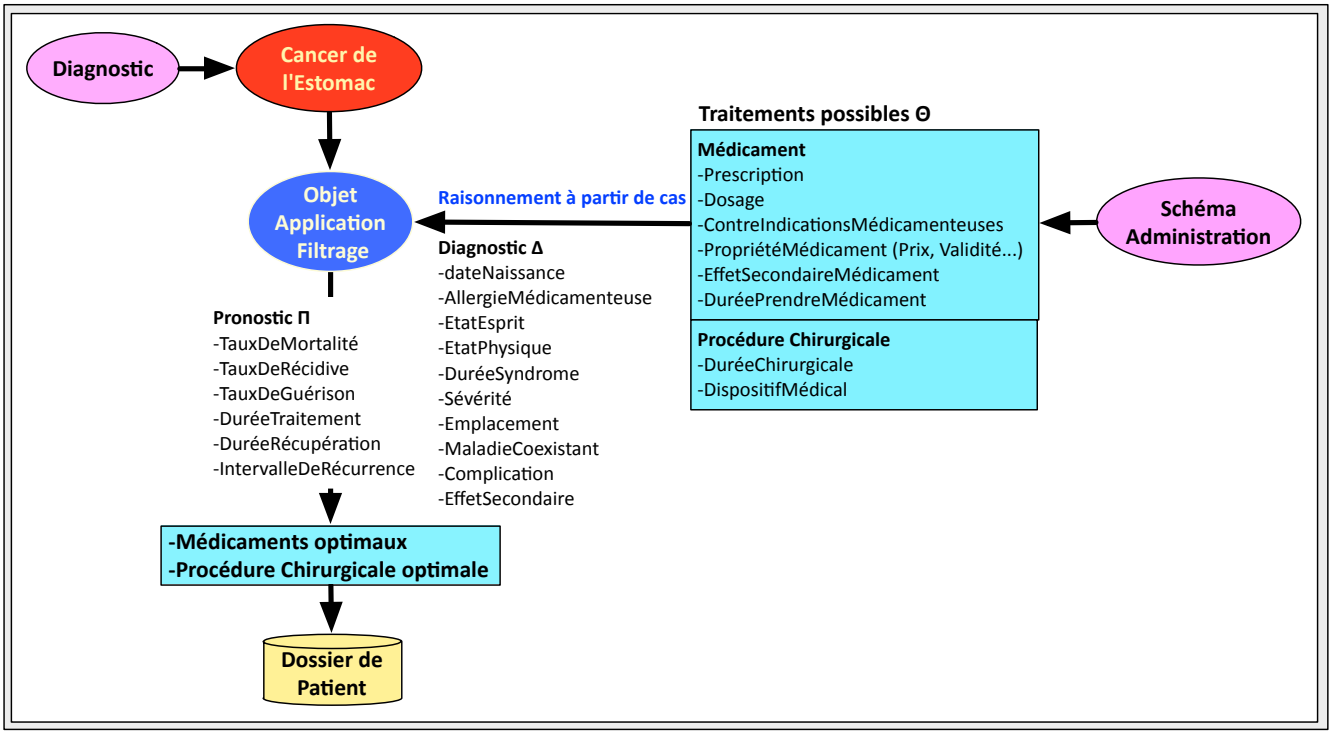


Figure 55 Étapes du processus de décision : trouver les médicaments et les procédures chirurgicales optimaux

Les informations de dictionnaire ontologique : celles qui concernent le patient (ID, historique, condition), le diagnostic (syndrome, cause), le pronostic (effet, signes, temps), le traitement (ProcédureChirurgicale, Médicament, SchémaAdministration) et le Suivi-thérapeutique (Dossier) sont accessibles à travers des interfaces définis (tels que hasID, hasHistorique, hasSchémaAdministration etc.).

La mise en place de l'acquisition et de la restitution de connaissances dans les systèmes d'aide à la décision cliniques peut être réalisée à travers la création automatique de systèmes de traitement linguistique. Parallèlement, le raisonnement à partir de cas (RàPC) mémorise et restitue l'expérience et la connaissance de résolution de problèmes similaires.

Le RàPC fait intervenir des distances sémantiques pouvant être élaborées selon différentes approches : des algorithmes de similarité structurale (Colloc J. & Boulanger D. 1993); l'apprentissage statistique comme celui proposé par le système SIPINA (Sebban, M. et al. 1997) (Zighed D.A. et al. 1992) ; des approches numériques issues des réseaux de neurones ou de la logique floue. Les travaux concernant les distances sémantiques tendent à réconcilier les aspects symboliques et numériques (Bouchon-meunier, B. et al. 1990) (Dubitzky W. et al. 1997) (Schuster, A.

et al. 1997), c'est un apport nécessaire pour travailler dans le domaine de l'humain, c'est-à-dire, dans le domaine de l'incertitude. Le RàPC constitue une voie de recherche prometteuse, sur le plan technique, dans le domaine de la santé, comme en témoigne le flux important de publications afférentes (Lenski, G.E. 1954).

Voici un exemple d'extraction d'information, à partir d'un corpus : ici, un dossier médical. Ce patient est atteint d'un cancer de l'estomac stade 3. On peut extraire les détails de ses syndromes : historique et cause, avec classe, sous-classe et interface correspondants.

Classe	**Sub-classe**	**Interface**	**Résultat**
ID	nom	hasNom	Martin
	prénom	hasprénom	Patrick
	N°SS	hasNSS	274079921633533
	dateNaissance	hasDateNaissance	28 Juillet 1974
Historique	AntécédentsFamiliaux	hasAntécédentsFamiliaux	Il avait des antécédents d'hypertension. Il est en bonne santé et son histoire familiale est négative.
Syndrome	DuréeSyndrome	hasDuréeSyndrome	En janvier, survient une gêne abdominale supérieure, accompagnée de reflux acides, d'éructations, de douleurs postprandiales occasionnelles. Absence de vomissements et d'autres symptômes. Le 15 mai : vomissements le matin, maux de gorge. Un diamètre de 4 cm masse palpable en quadrant supérieur droit sensible à la pression.
	Sévérité	hasSévérité	
	Emplacement	hasEmplacement	
	MaladieCoexistant	hasMaladieCoexistant	Obstruction du pylore
Cause	DonnéeStatistique	hasDonnéeStatistique	Le Juin 12, **rapports de pathologie** : (antre) haute - adénocarcinome différencié. **Rapports de CT** (abdomen, échographie pelvienne + renforcée + reconstruction gastrique) : épaississement de la paroi ventrale et épaississement anormal de la muqueuse ; en relation avec la maladie de la vésicule biliaire ; côté du pylore et la petite courbure, présence de plusieurs ganglions lymphatiques ; épaississement de la paroi de la vessie, hypertrophie de la prostate augmentation ; segment initial du coeliaque ondulé sous pression.
	DonnéeExpérimental	hasDonnéeExpérimental	
	InformationEchantillon	hasInformationEchantillon	
	QuantitéTemporelle	hasQuantitéTemporelle	

Tableau 30 Les informations extraites du patient à travers des interfaces définis

Après avoir pris en compte l'âge du patient, son état physique, sa maladie, etc., le RàPC aide « [l']Objet Application Filtrage » à chercher les médicaments et les procédures chirurgicales de cas similaires dans le corpus. Le résultat dépend du contenu de la base de connaissances.

Dans les étapes de recherche des médicaments optimaux, il faut tenir compte de la condition de patient (AllergieMedicamenteuse), des caractéristiques des médicaments : ContreIndicationsMédicamenteuses, EffetSecondaireMédicament, les interactions médicamenteuses, etc. Ensuite, il faut déterminer ceux qui sont compatibles, et les ordonner selon leurs risques potentiels d'effets secondaires. La procédure chirurgicale optimale est surveillée selon le protocole opératoire.

On établit les ontologies initiales d'un cancer de l'estomac manuellement. Elles constituent le cadre de l'extension des ontologies : c'est un corpus linguistique initial dont la forme est donc relativement spécifique. On a collecté les connaissances cliniques élémentaires dans un dictionnaire médical qui, lui, est un autre corpus, très spécifique. Par exemple, certains médicaments de chimiothérapie sont listés ici.

Médicaments de chimiothérapie		
Indications	**Médicament**	**Détails**
Médicament pour tirer le cancer gastrique du danger	Taxol (Paclitaxel)	Le paclitaxel est un inhibiteur mitotique utilisé dans la chimiothérapie du cancer. Le paclitaxel est utilisé pour traiter les patients atteints de cancer du poumon, le cancer de l'ovaire, cancer du sein, le cancer de la tête et du cou, et des formes avancées de la maladie de Kaposi. Le paclitaxel est également utilisé pour la prévention de la resténose. Les effets indésirables fréquents incluent la nausée et les vomissements, la perte d'appétit, le changement de goût etc.
Promédicament de 5-FU	Xeloda (Capecitabine)	Xeloda (capécitabine) est un carbamate fluoropyrimidine avec une activité antinéoplasique. Il s'agit d'un précurseur de médicament systémique de la 5'-désoxy-5-fluorouridine (5'-DFUR) qui est converti en 5-fluoro-uracile. Les signes d'une réaction allergique : urticaire, respiration difficile, gonflement du visage, des lèvres, de la langue ou de la gorge (œdème de Quincke). Effets secondaires : diarrhée sévère, nausées, perte d'appétit.
Médicament de choix pour le traitement du cancer gastrique avancé	5FU (5-Fluorouracil)	Il s'agit d'un médicament qui est un analogue de pyrimidine, qui est utilisé dans le traitement du cancer. Il s'agit d'un inhibiteur du cycle cellulaire, cytotoxique car il produit l'apoptose.. Il fonctionne par inhibition irréversible de la thymidylate synthase. Il appartient à la famille de médicaments appelés antimétabolites.
Cancers gastriques avancé localement et récurrent/métastatique	HDFL (High dose 5-FU and Leucovorin)	Une chimiothérapie combinée à base de HDFL fonctionne bien dans le traitement du cancer gastrique avancé. Perfusion hebdomadaire de 24 heures, avec haute dose de 5-fluorouracile et de leucovorine (HDFL) est efficace dans le traitement du cancer gastrique.

Les patients atteints d'un adénocarcinome gastrique métastatique qui est inefficace du traitement HDFL	Oxaliplatin (Eloxatin)	L'oxaliplatine est un agent antinéoplasique à base de platine, utilisé en chimiothérapie du cancer. Typiquement, l'oxaliplatine est administrée avec le fluorouracile et la leucovorine, en une combinaison connue comme FOLFOX pour le traitement du cancer colorectal.
Cancer gastrique métastatique	Tegafur	Le tégafur (DCI) est un promédicament du fluorouracile chimiothérapeutique utilisé dans le traitement des cancers. Il s'agit d'un composant de la combinaison médicament tégafur-uracile. Lorsqu'il est métabolisé, il devient 5-fluoro-uracile qui est cytotoxique.
	Carboplatin (Paraplatin)	Carboplatine ou cis-diamine (1,1-cyclobutanedicarboxylato) platine (II) est un médicament de chimiothérapie utilisé contre certaines formes de cancer. Il a été introduit dans les années 1980 et a il a gagné en efficacité dans le traitement clinique parce que, comparé à cisplatine, un composé parent, il réduit considérablement les effets secondaires. Le cisplatine et le carboplatine appartiennent au groupe des agents antinéoplasiques à base de platine qui interagissent avec l'ADN et interfèrent dans la réparation de l'ADN.
	Taxotere(Docetaxel)	
Médicament de thérapie ciblée		
	Herceptin (Trastuzumab, Cetuximab) : anticorps anti-HER-2	
	Bevacizumab (Avastin®)	
	Cetuximab (Erbitux®) : anticorps anti-EGFR	

Tableau 31 Echantillon de médicaments de chimiothérapie et thérapie ciblée

Notre modèle ontologique se fonde sur la richesse sémantique des modèles orientés objet. Sur les figures : Ontologie des agents Procédure Traitement, Ontologies des pathologies, Ontologies des médicaments, et Ontologies du Dossier Médical, on montre qu'un diagramme de classes UML offre des capacités sémantiques et graphiques suffisantes pour coder une ontologie. Cela rend compte aussi de la complexité linguistique des données manipulées et de la variété des corpus corrélés.

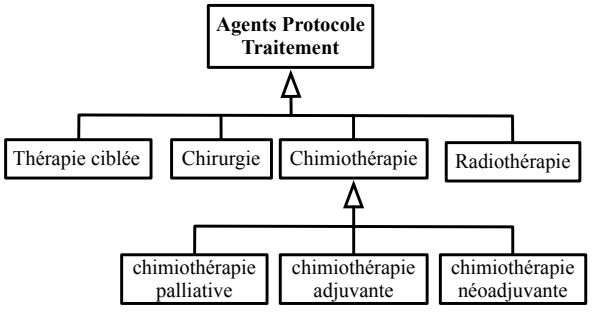


Figure 56.1 Ontologie des agents Protocole Traitement

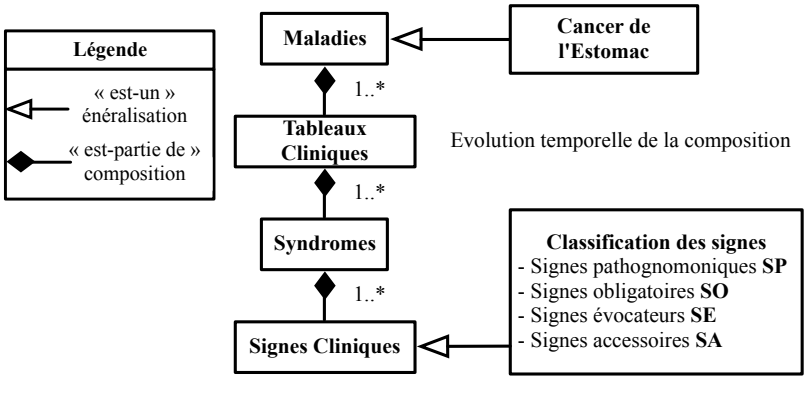


Figure 56.2 Ontologies des pathologies

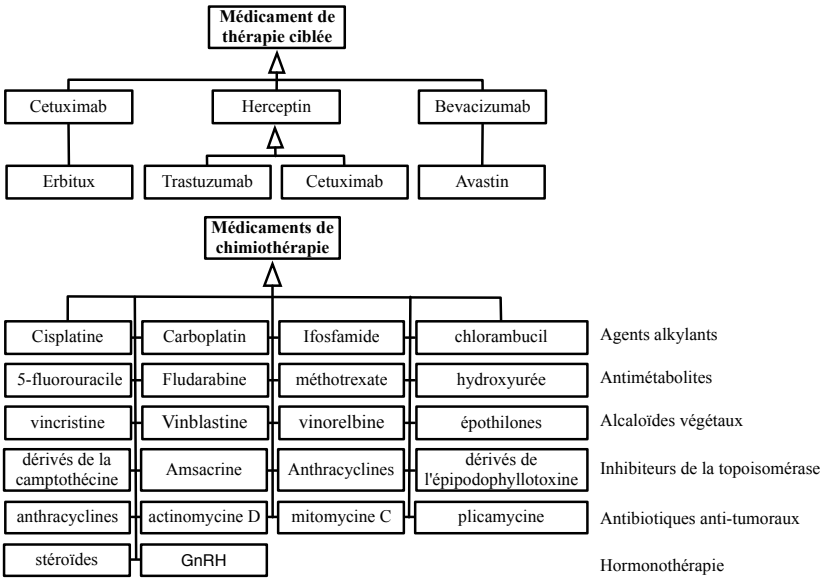


Figure 56.3 Ontologies des médicaments

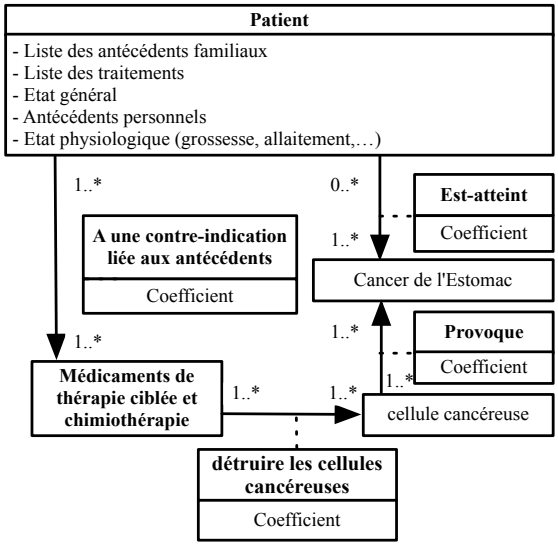


Figure 56.4 Ontologies du Dossier Médical

Figure 56 Modèle d'ontologie à propos d'un cancer de l'estomac

Après avoir vérifié les contre-indications entre le « Protocole Chirurgical » et les médicaments, un traitement personnalisé optimal est présenté avec certains détails tels que la spécialité prescrite (Contre-indication liée à la voie d'administration), le coût journalier du traitement, l'existence d'un générique, et les participants au protocole chirurgical.

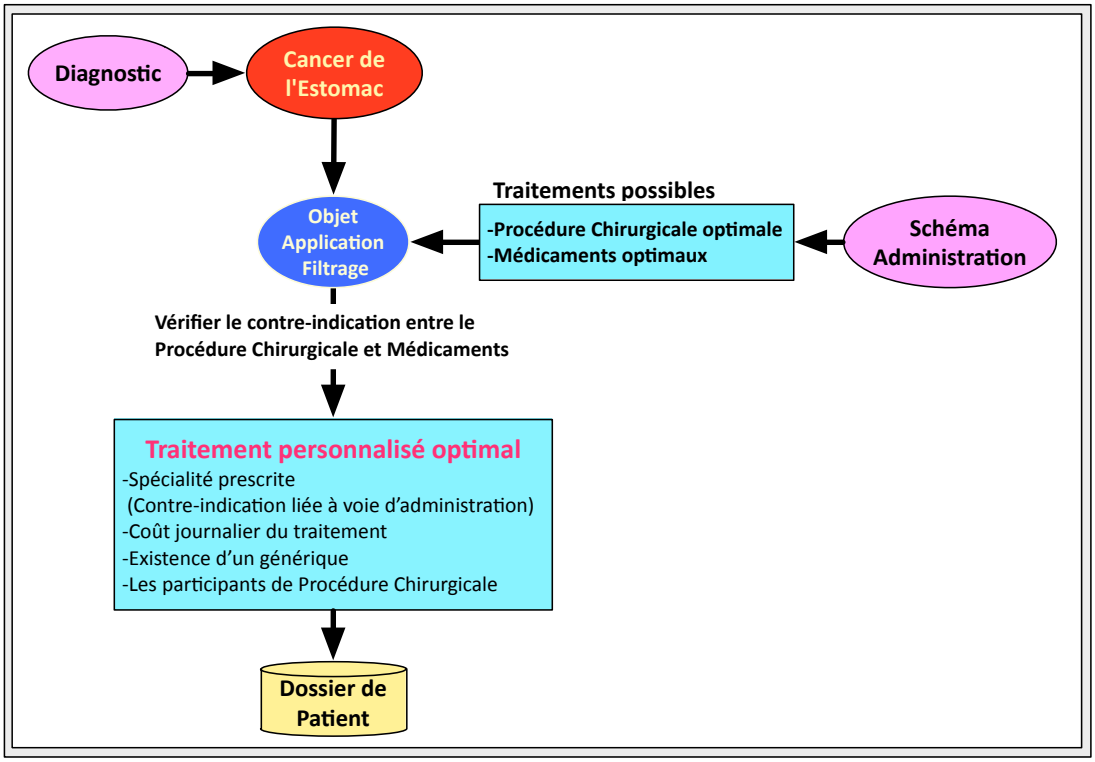


Figure 57 Étapes du processus de décision : présenter le traitement personnalisé optimal

6.2.2 Organiser les connaissances du cas d'étude

Afin d'analyser la requête d'entrée, il faut d'abord dérouler les *scenarii* pour identifier intuitivement les applications et les solutions possibles, et recomposer la requête. Ensuite, il faut comparer les requêtes des phrases avec les terminologies formelles en associant les mots des phrases aux classes, aux relations, aux attributs, axiomes et instances correspondants. Prenons, par exemple, la requête « Trouver les cas de maladies qui comprennent le cancer de l'estomac de stade 3 ». Ces classes incluent les substantifs « cas », « maladie », « cancer » et « estomac », la relation est «compren*», et l'attribut est « stade » avec comme valeur 3 : « stade 3 ». L'objectif de cette requête est de trouver toutes les instances « cas 1, 2,…n. ».

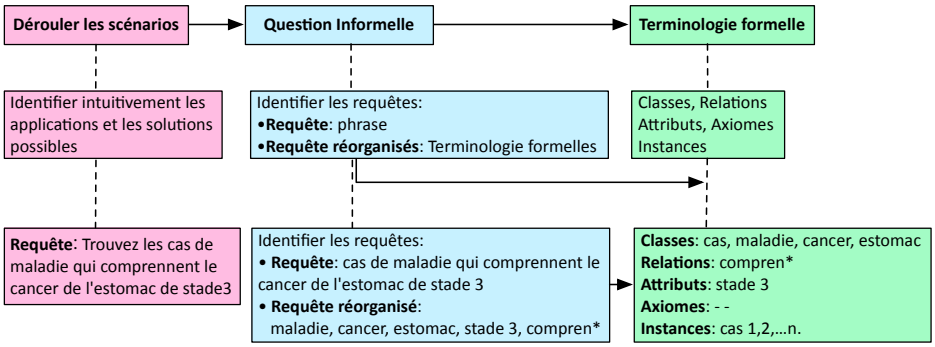


Figure 58 Obtenir les terminologies des questions

L'analyse morphologique et lexicale montre les racinisations et les synonymes des mots cibles. Les classes, relations et attributs extraits sont utilisés pour les questions réponses.

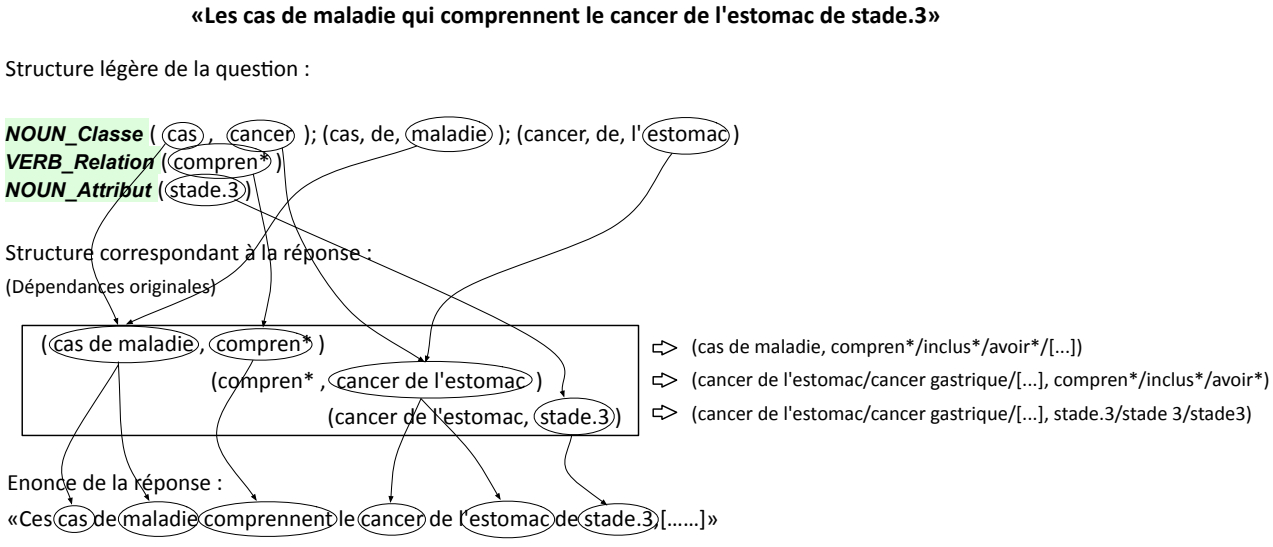


Figure 59 Exemple d'interrogation de la structure informationnelle avec son expansion

Ce cas typique est présenté pour montrer l'architecture associée à une maladie incluant le diagnostic, le pronostic et le traitement d'un cancer gastrique.

Le tableau ci-dessous illustre la démarche clinique implantée par le SMAAD (Colloc, Sybord, 2003). L'Automate d'Etats Finis Clinique Général du superviseur (AEFCG) (figure 45) charge l'Automate d'Etat Fini Spécialisé (AEFSΔ) destiné au diagnostic (Δ) d'un cancer de l'estomac. La présence initiale d'un cancer de ce type (SO1) peut conduire aux diagnostics Δ1 à Δ6, selon les signes cliniques SE1 à SE6 (tableau 32.2). Les détails de procédure chirurgicale PC1 à PC6 sont présentés pour l'étape de pronostic et de traitement (tableau 32.3). Ensuite, l'AEFSΠ définit les pronostics associés : Π1 à Π6, puis l'AEFSΘ définit les stratégies thérapeutiques correspondantes : Θ1 à Θ6 (tableau 32.4). Les suivis thérapeutiques (SΘ) ne sont pas présentés par concision.

Le tableau 32.5 présente le cas similaire clinique. Nous cherchons à identifier leurs significations avec des termes importants tels que des adjectifs et des verbes qui sont porteurs d'associations efficaces (e.g. maladie coexistant, stade d'un cancer de l'estomac, etc.). L'AEF RàPC (Colloc J. & Sybord C. 2003) stocke et indexe successivement les cas rencontrés et les liens de composition de leurs composants dans la base de cas avec, les éléments (Δ, Π, Θ, SΘ) (Colloc J. et al. 2007) et un groupe de mots clés définissant le problème du cas (PB), l'environnement (fiche du patient) (E), le résultat obtenu (R).

Tableau 32.1 Légende Signes cliniques obligatoires (SO), évocateurs (SE)

SO_1	La tumeur s'accroît dans la lamina propria, muqueuse musculaire, ou dans la sous-muqueuse (couches internes de la paroi de l'estomac).
SE_1	aucun signe de métastases ganglionnaires.
SE_3	La tumeur s'accroît dans la musculeuse (couche musculaire de l'estomac).
SE_4	La tumeur se développe à travers toutes les couches du muscle dans le tissu conjonctif à l'extérieur de l'estomac, mais pas de propagation dans la paroi peritonéale ou séreuse.
SE_5	La tumeur s'accroît à travers toutes les couches du muscle dans le tissu conjonctif en dehors de l'estomac et se développe dans le péritoine ou séreuse ou les organes environnants l'estomac.
SE_6	Une récidive de cancer post traitement. Probablement une récidive localisée (revient à l'endroit où il a commencé), ou peut être une métastase à distance (revient dans une autre partie du corps).

Figure 32.2 diagnostic et pronostic d'un cancer gastrique

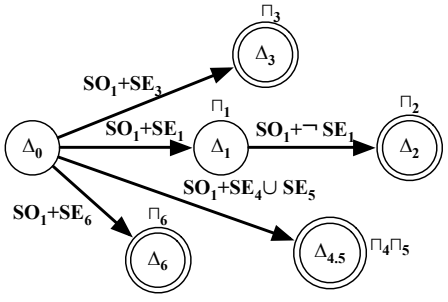


Tableau 32.3 Protocoles de Traitement (PT)	
PT_1	**Chirurgie**: La laparotomie peut vérifier si la résection palliative est opérable. Sinon, la jéjunostomie permet le soutien nutritionnel. Pendant Stade 4 de cancer gastrique, la chirurgie ne peut pas être effectuée.
PT_2	**Chimiothérapie palliative**: S'applique chez le patient après avoir eu une résection récurrente ou palliatif, ou s'applique chez le patient sur qui la résection palliative n'est pas opérable mais l'ensemble du corps est en bon état, et la fonction des organes majeurs est normale.
PT_3	**Chimiothérapie adjuvante**: Une chimiothérapie adjuvante est destinée à augmenter ou stimuler les effets positifs d'une autre forme de traitement, comme la chirurgie ou la radiothérapie. Objet d'un traitement comprent: les patients au Stage 1b de postopératoire pathologique , et avec métastases ganglionnaires; les patients au Stade 2 et plus de postopératoire pathologique.
PT_4	**Chimiothérapie néoadjuvante**: Un traitement néo-adjuvant est un prétraitement administré afin de réduire la taille d'une tumeur cancéreuse (maligne) avant une intervention chirurgicale (et donc limiter l'exérèse) ou une radiothérapie (accroître l'efficacité). Elle est adaptée pour les patients de Stade 3 et Stade 4.
PT_5	**Radiothérapie**: La radiothérapie est une méthode de traitement locorégional des cancers, utilisant des radiations pour détruire les cellules cancéreuses en bloquant leur capacité à se multiplier. Elle peut être utilisée seule ou associée à la chirurgie et à la chimiothérapie. Ses indications sont liées au type de tumeur, à sa localisation, à son stade et à l'état général du patient.
PT_6	**Thérapie ciblée**: Les cibles moléculaires sont essentiellement sur le récepteur du facteur de croissance épithélial (EGFR), facteur de croissance vasculaire endothélial (VEGFR), récepteur du facteur de croissance épidermique humain-2 (HER-2) etc.

Tableau 32.4 Diagnostic, pronostic et traitement d'un cancer gastrique					
Diagnostics		**Pronostics**		**Thérapie**	
Δ_0	Cancer gastrique				
Δ_1	Cancer gastrique au début et aucun signe de métastases ganglionnaires	$\sqcap_1$	Survie à 5 ans après la résection du cancer gastrique précoce jusqu'à environ 90%.	$\ominus_1$	PT_1 Thérapie endoscopique ou chirurgie, sans radiothérapie adjuvante ni chimiothérapie postopératoire.
Δ_2	Cancer gastrique avec métastases ganglionnaires	$\sqcap_2$		$\ominus_2$	PT_1 + PT_3 Excision chirurgicale curative selon la profondeur d'invasion tumorale. Selon les circonstances, la chirurgie radicale peut être opérée directement ou après chimiothérapie néoadjuvante.
Δ_3	Cancer gastrique localement avancé	$\sqcap_3$	Soulager les symptômes/Vital	$\ominus_3$	PT_3 + PT_4 Pris à la thérapie combinée basée sur la chirurgie. Après la mise en œuvre réussie de la chirurgie radicale, il faut décider du programme de thérapie assistée selon le stade de pathologique (chimiothérapie adjuvante, radiothérapie adjuvante...)
$\Delta_4\Delta_5$	Cancer gastrique avancé et inopérable	$\sqcap_4$ $\sqcap_5$	La réponse complète (CR) de taux est inférieur à 30%, le temps de survie moyen allant d'environ 6 à 12 mois	$\ominus_4$ $\ominus_5$	PT_2 Avec la chimiothérapie palliative à base de 5-FU, l'objectif principal est de soulager des symptômes causés par le cancer, tels vomissements, ascite, distension et douleurs abdominales.
Δ_6	Cancer gastrique récurrent ou métastatique	$\sqcap_6$	Soulager les symptômes/Vital	$\ominus_6$	PT_2 +PT_5 +PT_6 Prise de thérapie combinée basée sur des traitements médicamenteux, et adopter les traitements locaux tels que la chirurgie palliative, radiothérapie, thérapie interventionnelle, ablation par radiofréquence etc.

Tableau 32.5 Cas clinique similaire (RàPC)

Cas clinique similaire Un homme de 62 ans avec cancer gastrique de stade post-opératoire **IIIa** et **obstruction du pylore**. Il avait des antécédents d'hypertension. Il est en bonne santé et son histoire familiale est négative. [Hallissey, M. T. et al. 1994]

Mars 2001, subi une gastrectomie radicale. Rapports de pathologie: adénocarcinome stade II-III, impliquant l'ensemble de la couche, 2/20 métastases ganglionnaires. Opérer 8 cycles de chimiothérapie adjuvante post-opératoire. [Taxol(85mg/m2)+CF(400mg/m2)+5-Fu(0.5g)+5-Fu(3.0g/m2) (répétée chaque 2 semaines)] Le patient avec une toxicité nerveuse périphérique significative, et d'autres effets indésirables plus léger.

Décembre 2002 (21 mois après opération) douleurs abdominales (CA19-9>5000ng/L). PET-CT : Tissus mous de la sous capsulaire du lobe droit épaississent, tissus mous nodulaire sus-ombilical apparaissent l'ombre, et présentent une grande métabolique.

Janvier 2003, Traitement = Laparotomie + résection partielle rétropéritonéale + chirurgie de réduction tumorale abdominale. Une vaste masse granulaire est trouvée dans le diaphragme et péritoine, la plus grande est de 4*5cm. Postopératoire: chimiothérapie intrapéritonéale avec DDP 2 fois ; La chimiothérapie systémique: Docétaxel (60 mg) + Xeloda (1500 mg) (4 cycles, répété chaque 3 semaines). Évaluation: Préopératoire: CA19-9 5029u/ml; Après 4 cycles de chimiothérapie: 233.9u/ml. Les raisons de l'interruption de la chimiothérapie: pendant le cinquième cycle de chimiothérapie, il y a obstruction de l'intestin grêle incomplète adhésif, donc arrêter de la chimiothérapie.

Février 2004 (12 mois après la chimiothérapie intrapéritonéale) CA19-9 fortement augmente. CT: métastases au foie, les poumons et la cavité abdominale.

Mars-Septembre 2004, Chimiothérapie : CPT-11(120mg)+5-Fu(300mg) (4 cycles, 4 semaines consécutives, repos de deux semaines, répéter chaque 6 semaines). CA19-9 a diminué à la gamme normale. Les métastases du foie et des pulmonaires rétrécissent.

Décembre 2005

Réadmission car obstruction intestinale incomplète, et métastases du foie, abdominopelvien et des poumons sont étendues. La laparotomie montre que il y a différentes tailles de nodules blancs dans la tumeur abdominopelvien, et des parties de l'intestin grêle adhérent entre-elles. Traitement = Excision partielle nodulaire + côlon ascendant-iléale côté-à-côté. Postopératoire=Chimiothérapie DDP +5-FU 2 fois.

Février 2006

Obstruction de l'intestin grêle, ascite massive, selles sanglantes, difficulté à respirer, abandonner la rescousse, décès.

Cas clinique qui recherche la suggestion

Homme de 40 ans avec cancer gastrique de stade **IIIa** et **obstruction du pylore** (Maladie Coexistant). En janvier, survient une gêne abdominale supérieure, accompagnée de reflux acide, d'éructations, de douleurs postprandiale occasionnelles. Absence de vomissements et d'autres symptômes. Le 15 mai vomissements du matin, maux de gorge. Un diamètre de 4cm masse palpable en quadrant supérieur droit sensible à la pression.

Le Juin 12, rapports de pathologie: **(antre) haute - adénocarcinome différencié**. Rapports de CT (abdomen, échographie pelvienne + renforcée + reconstruction gastrique): épaississement de la paroi antrale et épaississement anormal de la muqueuse ; En relation avec la maladie de la vésicule biliaire ; Côté du pylore et la petite courbure, présence de plusieurs ganglions lymphatiques; épaississement de la paroi de la vessie, hypertrophie de la prostate augmentation; segment initial du coeliaque ondulé sous pression. Légère sténose luminale, peut être causé par la compression du ligament de la voûte plantaire.

Tableau 32-1 à 5 Diagnostic, pronostic et traitement d'un cancer gastrique

L'ontologie utilisée dans cette thèse est présentée ci-après : le patient 1 participe à la phase de diagnostic, pronostic, traitement et suivi-thérapeutique. Les signes cliniques obligatoires et évocateurs conduisent à des pronostics différents (П1 à П6). Notre dictionnaire ontologique (corpus) fournit la liste des traitements possibles au système SMA et au médecin, à travers des recherches de cas similaires. Après avoir renouvelé le dossier du patient, l'agent SchémaAdministration surveille toutes les phases de traitements, et peut changer les programmes thérapeutiques, suivant l'état du patient. Si la tumeur récidive, un nouveau cycle de procédure médicale sera lancé.

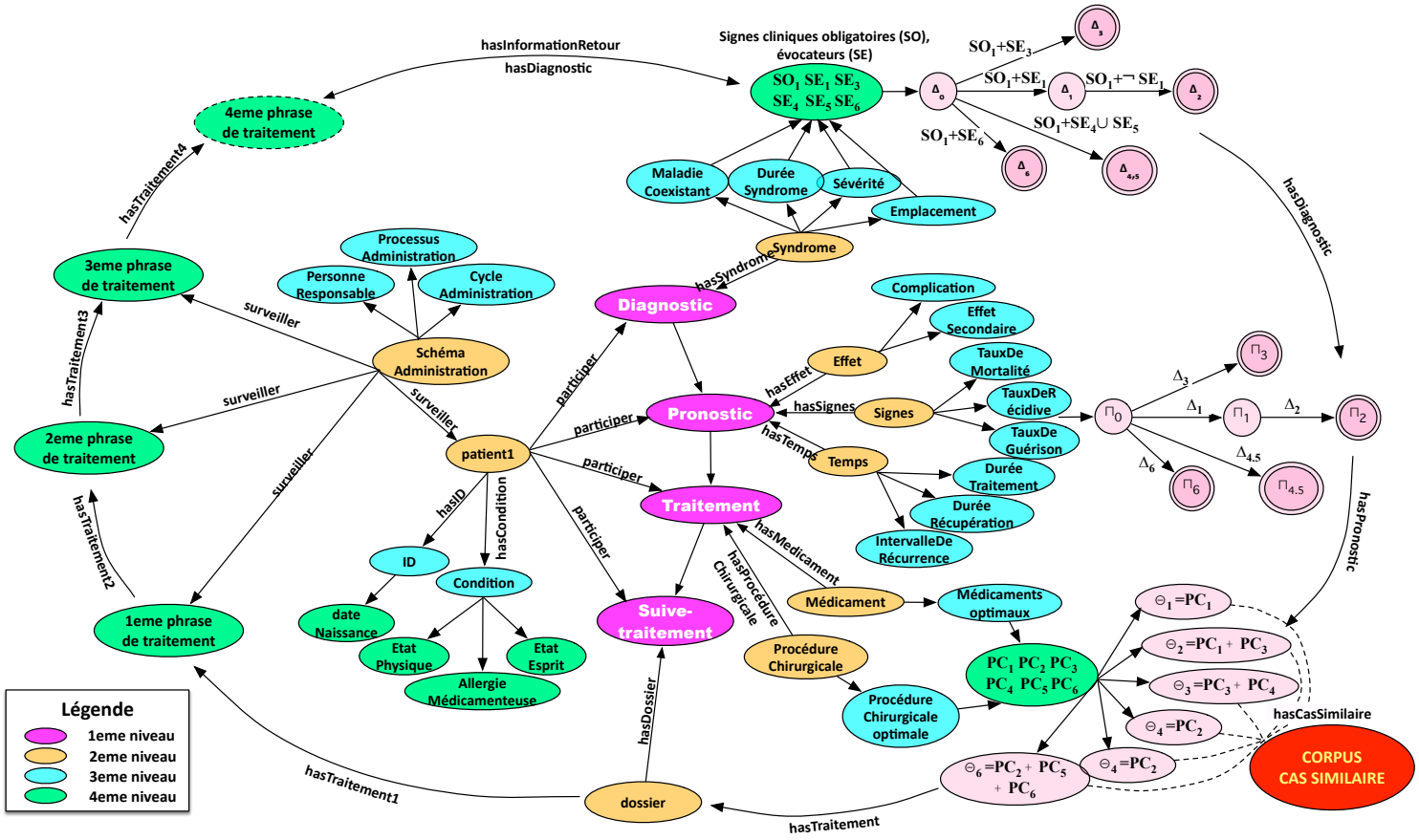


Figure 60 Exemple d'Ontologie de diagnostic, pronostic et traitement d'un cancer gastrique

6.3 Apport de l'ontologie sur les antibiotiques et sur les maladies infectieuses dans le programme SIAMED

Aujourd'hui, les européens utilisent l'informatique pour aider à la décision clinique et aux recommandations thérapeutiques. Ce sous-chapitre montre un exemple d'utilisation des systèmes à bases de connaissances ; il décrit les étapes de décision et les bonnes pratiques en antibiothérapie.

Le travail présenté ici corrèle la linguistique et l'informatique de façon plus précise, après les indications évoquées ci-dessus, sachant que ces divers corpus ont des structures respectives sont hétérogènes.

Nous recherchons le mode de représentation des connaissances mieux adapté. Nous montrerons que les langages orientés objets apportent des éléments de solution. Les langages orientés objet ont en commun avec les frames : arborescence, héritage de propriétés; Un objet est le représentant d'une classe. Les classes sont organisées hiérarchiquement. Chaque sous-classe hérite des « méthodes » (-actions) et des « champs » (- attributs) de sa surclasse. Ils possèdent des facettes (valeurs possibles, valeur par défaut, restriction...) rendant le système expert évolutif (Colloc, J. & Boulanger, D. 1987).

Dans ce sous-chapitre, tout d'abord, nous rappelons des spécifications de SIAMED en 1985. Puis nous détaillons les méthodes de recherches, afin de montrer comment nous obtenons les ontologies qui doivent impérativement représenter le résultat d'une analyse de textes. L'analyse linguistique et le programme en code JAVA, décrits au chapitre 5, sont utilisés pour construire un système d'aide à la décision. Les classes d'objets, les associations et les attributs sont encapsulés dans le système multi-agents, pour présenter la connaissance sous la forme d'objets complexes hiérarchisés. Les connaissances sur les antibiotiques sont conçues en UML, puis converties en OWL, afin de réaliser les ontologies nécessaires à la prescription d'un antibiotique.

L'apport des ontologies est de déterminer comment est organisé le domaine de connaissances, concernant les antibiotiques, et comment ces connaissances peuvent être employées pour choisir un traitement antibiotique approprié à un patient atteint d'une pathologie infectieuse.

6.3.1 Connaissance à propos des antibiotiques et des infections

L'infection est une maladie qui résulte de la pénétration dans l'organisme d'une bactérie, d'un virus, d'un parasite, d'un champignon ou de leurs produits (toxines etc.) Les spécialistes des maladies infectieuses et les épidémiologistes de la santé publique s'intéressent aux causes et observent les symptômes pertinents constitués de signes : fièvre, éruptions cutanées, nausées, vomissements, altération de l'état général, inflammation, douleurs, courbatures, frissons, etc. L'infection peut être traitée par la

prescription d'antibiotiques, en particulier lorsqu'une bactérie est suspectée d'en être responsable.

Waksman en donne en 1941 (Waksman, S.A. & Woodruff, H.B. 1941) la définition de l'antibiotique suivante :

> On a longtemps appelé antibiotique toute substance chimique isolée d'un micro-organisme (bactérie ou champignon) et capable d'inhiber et même de détruire d'autres micro-organismes.

Actuellement, l'antibiotique est une substance chimique capable d'inhiber une, au moins, des étapes essentielles du métabolisme des bactéries et ceci quel que soit son mode de production (Colloc, J. 1985).

Les recherches sur les antibiotiques reposent sur des compétences en antibiothérapie, réparties dans différentes spécialités médicales fortement concernées par la prescription d'anti-infectieux et par la lutte contre l'acquisition de résistances bactériennes aux antibiotiques. Il s'agit, notamment, des services de bactériologie, de réanimation, de médecine d'urgence, d'onco-hématologie, d'anesthésie, de chirurgie, de médecine interne, de pneumologie, de gériatrie, de pédiatrie, etc. (Raoult, P.D. et al. 1985). La prescription des antibiotiques est déterminée par le germe, la localisation, les propriétés des médicaments etc. (Colloc, J. 1985). Ils constituent les facteurs importants pour les recherches du projet SIAMED :

- Germe

  Le plus souvent, la clinique permet de faire le diagnostic qui peut incriminer tel ou tel germe. Parfois, il est plus difficile à identifier et il peut faire suspecter un ensemble de germes, avec des probabilités variées d'être responsables de la maladie. Enfin, parfois, seul un prélèvement (culture et antibiogramme) permet d'identifier le germe responsable et de connaître sa sensibilité aux antibiotiques. Ainsi, soit le germe est connu (bactériogramme), soit, on se trouve en présence d'un tableau statistique fondé sur des études épidémiologiques :

Étiologies des pneumonies communautaires en France	
Age < 18 mois, n =75	**Age > 18 mois, n = 104**
Virus 35 (dont VRS 24, surinfectés 10)	Virus 30 (dont VRS 10)
Pneumocoque 7	Pneumocoque 12
Haemophilus influenzae 2	Pneumocoque + Mycoplasme 2

Coqueluche (*bordetella pertussis*)	1	*Mycoplasma pneumoniae*	41
Tuberculose *Mycobacterium tuberculosis*	1	*Chlamydia pneumoniae*	1
Staphilococcus aureus	1	*Staphilococcus aureus*	1
Inconnu	27	Inconnu	17

Tableau 33 Statistiques des germes provoquant des pneumonies communautaires en France (Gendrel, D. 2002)

- Propriétés des médicaments

  Interaction médicamenteuse

  Il s'agit de l'effet produit par deux médicaments administrés conjointement. L'effet peut être, soit une potentialisation du premier sur le deuxième, du second sur le premier ou même des deux à la fois, soit la création d'un nouveau composé qui peut être toxique. Ceci est généralisable à un nombre supérieur de molécules.

  Contre-indication médicamenteuse

  Une contre-indication est un état pathologique ou un état physiologique particulier augmentant de manière importante la probabilité de survenue d'un incident ou un accident quand on prescrit le médicament considéré. Deux points sont importants : d'une part, la gravité de la pathologie engendrée par la prescription et, d'autre part, sa très forte probabilité de survenue lorsque le médicament est prescrit chez ce type de patient.

  Effets secondaires toxiques

  Un effet secondaire est un incident ou un accident dont la probabilité de survenue est faible et qui atteint un sujet contre toute attente lorsqu'on lui administre le médicament considéré.

  Action synergique de plusieurs antibiotiques

  À chaque fois que c'est possible, il faut prescrire un seul antibiotique, et il faut réserver les associations soit à des affections présentant une certaine gravité soit au traitement d'infections à germes résistants (la monothérapie est la règle).

- Localisation

  Il faut pouvoir atteindre le germe au niveau du site de l'infection et diffuser l'antibiotique dans le tissu infecté (pénétration). Par exemple, certains germes

sont intracellulaires de sorte que seuls des antibiotiques à bonne pénétration cellulaire seront actifs (ex. les tétracyclines, les phénicolés,...).

Les connaissances mentionnées ci-dessus sont extraites du texte, comme précisé au chapitre 5, puis elles sont représentées en UML et codées en OWL et, enfin, elles sont encapsulées dans l'agent ontologique, comme nous le montrons à la suite.

Ces connaissances sont très importantes car, certaines maladies infectieuses sont rarement rencontrées par les médecins, dans leur pratique : exemples de certaines maladies rares en Europe (anthrax cutané, charbon pulmonaire, botulisme, rage, pestes bubonique et pneumonique, variole, tularémie, typhus exanthématique, borrélioses,…). Pourtant, le diagnostic précoce de ces maladies est indispensable, non seulement pour sauver des vies mais aussi pour protéger les populations des épidémies.

6.3.2 Rappel des spécifications de SIAMED en 1985 (Colloc, J. 1985)

En 1985, Joël Colloc a conçu et proposé le système SIAMED (Système Informatisé d'Antibiothérapie MEDicale) (Colloc, J. 1985). Une première version de ce système a été développée à l'aide d'une approche procédurale. Dans une version plus élaborée, l'objectif a été de modéliser et d'implanter des connaissances concernant les antibiotiques et les maladies infectieuses, à l'aide d'une approche orientée objet. Elle prend en compte le processus de prescription utilisé habituellement par le médecin et certaines heuristiques concernant les indications, les contre-indications, les interactions médicamenteuses, les effets secondaires, les voies d'administration, les spécialités médicamenteuses disponibles.

Ce système comporte des connaissances profondes qui décrivent la nature biochimique des antibiotiques, la sensibilité des bactéries, les caractéristiques biologiques des maladies infectieuses, les accidents (parfois mortels) ou les incidents susceptibles de se produire lors de la prescription de ces médicaments. Ce système propose des recommandations en antibiothérapie, fondées sur des paramètres cliniques, caractérisant chaque patient.

SIAMED comprend un ensemble de dictionnaires qui recensent les termes que le système utilise et un ensemble de matrices chargées de définir les interactions entre les entités modélisées.

Les connaissances du système sont exprimées sous la forme de matrices de pondération concernant, d'une part, les germes habituellement responsables des maladies infectieuses et, d'autre part, le spectre, la diffusion tissulaire, les contre-indications, les interactions médicamenteuses, les effets secondaires des antibiotiques. Des mécanismes de filtrage, représentant les étapes successives de raisonnement du médecin lorsqu'il choisit un antibiotique, permettent d'exploiter les matrices du système. (Raoult, P.D. et al. 1985)

Le processus de décision décrit dans SIAMED 2 (Colloc, J. 1985) fait intervenir tour à tour différentes catégories de connaissances décrites par des ontologies que nous avons précisées dans les paragraphes précédents. Le processus de décision du système à base de connaissances SIAMED exploite l'algorithme de filtrage par affinements successifs de l'ensemble des antibiotiques initiaux, selon deux phases :

- **La phase de filtrage** (figure 61) ou de sélection détermine l'ensemble des antibiotiques susceptibles d'être actifs sur le germe responsable de la maladie et de pénétrer le site de l'infection.

  Les connaissances du système sont représentées sous la forme de matrices de pondération concernant les germes habituellement responsables des maladies infectieuses, le spectre, la diffusion tissulaire, les contre-indications, les interactions médicamenteuses, les effets secondaires des antibiotiques. Comme nous l'avons dit, les matrices du système sont exploitées par des mécanismes de filtrage représentant les étapes successives de raisonnement du médecin lorsqu'il choisit un antibiotique.
- **La phase d'élimination** (figure 61) est chargée d'écarter les antibiotiques susceptibles d'être contre-indiqués, de provoquer des interactions médicamenteuses néfastes ou bien présentant un risque d'effets secondaires dont la gravité et/ou la fréquence est disproportionnée eu regard à l'état pathologique du patient. Certains antibiotiques peuvent se révéler particulièrement dangereux et provoquer de graves lésions voire le décès du patient.

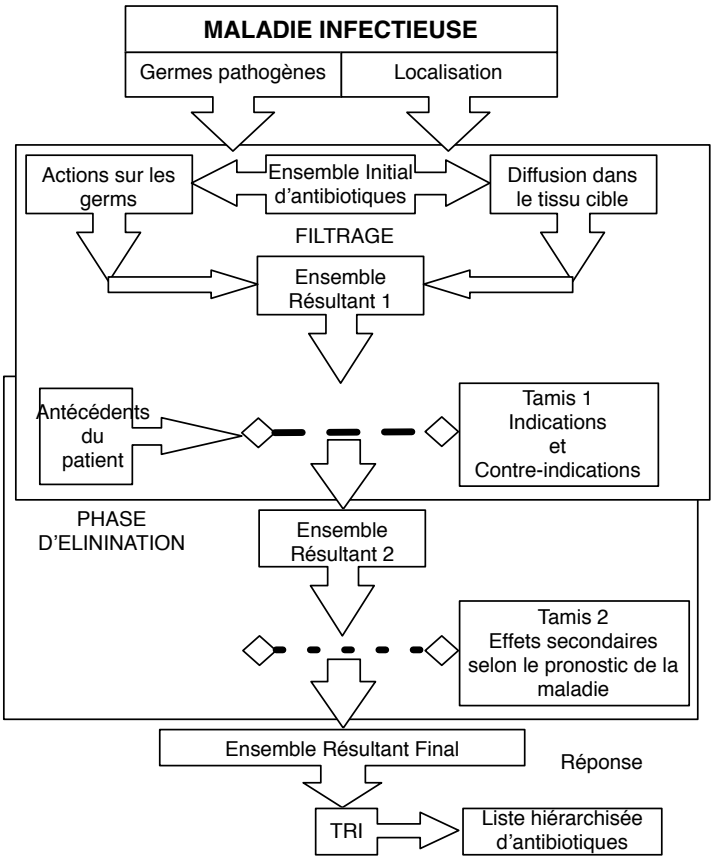


Figure 61 Processus de filtrage de SIAMED (Colloc, J. 1985)

La première étape de raisonnement (recherche des antibiotiques indiqués) est montrée à la figure 62. Le diagramme des flux de données permet de modéliser les interactions entre les différentes classes d'objets, les connaissances impliquées dans chaque phase de raisonnement.

Chaque phase correspond à un objet application, chargé d'obtenir les données et d'exécuter les tâches nécessaires.

Dans le cas de l'antibiothérapie, la classe initiale des antibiotiques (lexique des antibiotiques) va être progressivement filtrée, pour accéder à la sous-classe des antibiotiques pouvant être prescrits (extraction d'un sous-ensemble).

Les phases de raisonnement successives concernent la détermination des antibiotiques indiqués, la détection des contre-indications, la recherche d'éventuelles interactions médicamenteuses, la comparaison des risques d'effets secondaires et enfin le choix parmi les antibiotiques pouvant être prescrits.

L'objet Maladie infectieuse (on suppose le diagnostic réalisé) fournit une liste de germes potentiellement responsables et une ou plusieurs localisations possibles pour la maladie, par exemple. L'utilisateur doit préciser le site de l'infection.

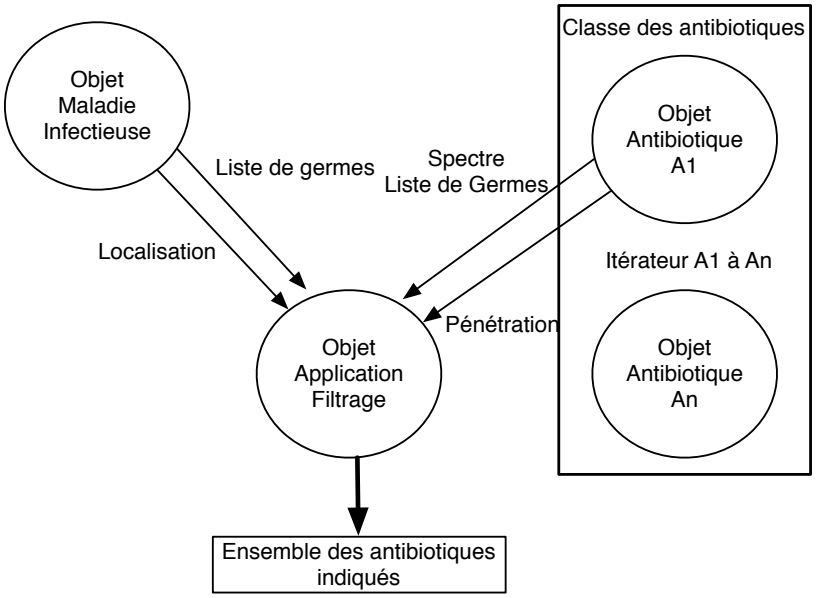


Figure 62 La détermination des antibiotiques indiqués (Colloc, J. 2000)

Pour chaque objet, la classe « Antibiotiques » fournit antibiotique :

- Son spectre : la liste des germes sur lesquels il est actif,
- Le coefficient de pénétration dans le compartiment de l'organisme atteint.

Une fonction d'évaluation (une méthode de l'objet application) détermine l'ensemble des antibiotiques indiqués.

Les autres étapes de raisonnement sont modélisées de la même manière.

6.3.3 Processus d'élaboration d'ontologies pour l'antibiothérapie dans le SMA

À partir de ces textes, il faut expliquer comment l'on définit les ontologies nécessaires décrites dans les chapitres dernières pour construire un système d'aide à la décision similaire au SIAMED. Tous les étapes sont encapsulés et réalisés dans le système multi-agent. Parce que le *diagnostic* étant déjà réalisé, on met l'accent sur l'analyse d'étape *thérapeutique* Θ de la démarche clinique, et d'étape du *pronostic* qui sert à définir le degré de gravité de la maladie.

6.3.3.1 Méta-modèle modifié de l'ODMG

Le méta-modèle (Guarino, N. et al. 1994) (IBM 2003) exprime les concepts de schémas UML (en particulier le diagramme de classe) et les associations remarquables associées à la sémantique de phrases. L'UML s'appuie sur une représentation graphique utilisée pour la conception de systèmes d'information (Berardi, D. et al. 2005) (Cranefield, S. J. S. & Purvis, M. K. 1999). Nous souhaitons exploiter la sémantique des concepts d'UML pour modéliser des ontologies médicales

et leurs requêtes. Dans un diagramme de classes, ces dernières sont définies par leur nom, leurs attributs (chacun spécifié par son nom, son type et sa visibilité) et leurs méthodes (chacune caractérisée par son nom, sa liste d'arguments, son type de retour et sa visibilité). Les propriétés (attributs et méthodes) sont unies à leur classe par l'association (a-un).

L'ODMG (Object Database Management Group) (Bartels, D. et al. 1997) propose un méta-modèle qui décrit la structure des diagrammes de classes UML. Notre méta-modèle propose une extension de celui de l'ODMG. Il précise la spécialisation des méthodes d'exécution, et détermine quels sont les sous-types de méthodes spécialisées et leurs significations (hiérarchie en vert sur la figure 64).

Comme indiqué sur la figure 63, le type d'objet implémente la « Classe » qui instancie l'« Objet ». Une « Association » concerne un ou plusieurs types d'objet. Une propriété est soit un «Attribut» soit une «MéthodeExécution» ces deux spécialisations s'excluent mutuellement (XOR). La «MéthodeExécution» détermine le déroulement de l'exécution d'une méthode et en particulier sa signature, les paramètres, le type de retour et les exceptions. Ainsi, la collection, la cardinalité d'éléments morphologiques, lexicaux, syntaxiques et le tagger, déclenchent l'analyse linguistique et en établissant les interfaces *créer_itérateur*(), *initialiser*(), *générer*() etc. Toute la syntaxe des phrases est codée Java.

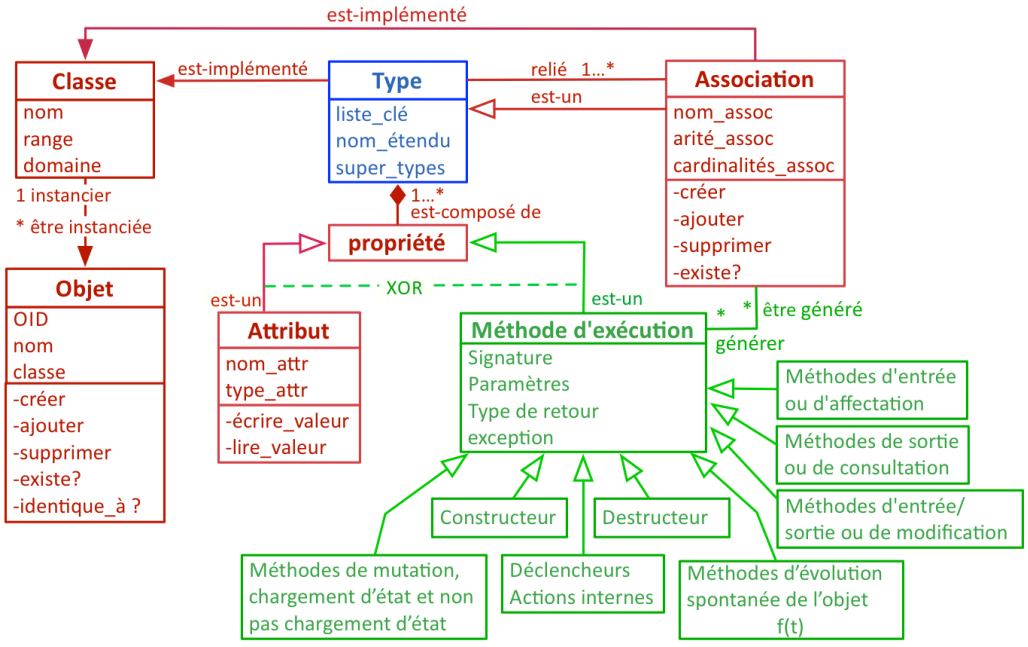


Figure 63 Proposition d'une extension du méta-modèle de l'ODMG

Les variables et les interfaces définies aident la « MéthodeExécution » à détecter et extraire les associations et à réaliser les activités comme : ajouter ou supprimer une association, vérifier son existence ou non, etc.

Le type d'objet gère le déclenchement et le résultat des méthodes décrites par « MéthodeExécution », cela constitue ses propriétés dynamiques : les actions dont il est capable. Une instance du sous-type hérite de l'ensemble des attributs (données) et des méthodes de traitement du super-type. Son état et son comportement particuliers proviennent des attributs et méthodes définies dans la classe spécialisée.

L'héritage des propriétés des super-types par les sous-types constitue une extension antisymétrique. Il faut donc tenir compte de la sémantique des relations remarquables pour construire un nouveau méta-modèle, mieux adapté, et le connecter à l'acquisition de nouveaux mots du texte dont on souhaite analyser la substance sémantique.

6.3.3.2 Fonctionnalités de TACG du projet SIAMED

Dans le chapitre 5, notre travail était de déterminer comment **l'analyse de texte** concernant l'antibiothérapie a permis de construire les ontologies nécessaires à la mise en œuvre de ce protocole, décrit sur la figure 62. Les étapes correspondantes seront principalement réalisées dans le « Dictionnaire ontologique » du système multi-agents.

Pour extraire les connaissances, à partir de textes *via* l'analyse linguistique, le premier objectif consiste à extraire des objets représentatifs et des classes d'objets exprimant les connaissances du domaine (i.e. des termes désignant des instances, des concepts de l'ontologie). Le deuxième objectif consiste à extraire des relations et des attributs entre ces termes (i.e. des instances des relations de l'ontologie). On réalise toutes les étapes à l'aide de l'analyse linguistique dans le « Modèle de l'Ontologie » et les programmes en Java du « Modèle de Traitement ».

Dans cette section, les notions sur le TAL, détaillées dans les précédents chapitres, permettent l'extraction de termes et de relations, à partir des textes, sont rappelées pour traiter les connaissances.

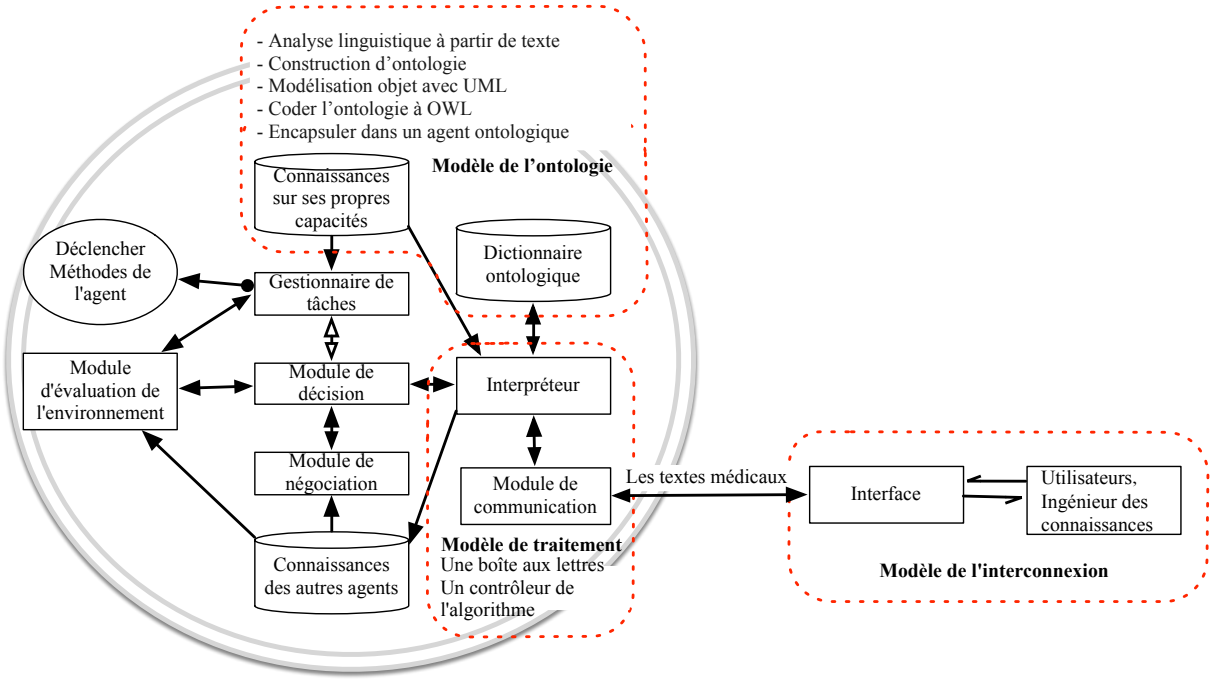


Figure 64 Architecture du Type d'Agent Clinique Général (TACG) analysé de texte médical pour élaborer l'ontologie

Après avoir reçu les textes médicaux de l'utilisateur final ou de l'ingénieur des connaissances, le «Module d'Interface» transfère les textes au «Modèle de Traitement» qui fonctionne comme une boîte aux lettres et un contrôleur de l'algorithme. Avec les programmes et les algorithmes du «Modèle de Traitement», le «Modèle de l'Ontologie» analyse et extrait les termes et les associations pour construire une ontologie.

Pour le traitement automatique de la langue naturelle, nous considérons que le texte est une source essentielle et riche pour l'acquisition des connaissances. Donc dans le «Modèle de l'Ontologie», tout d'abord, nous devons analyser les données brutes, pour repérer les patterns de texte, ce qui nous permettront de classer les documents selon leur type et leur contenu. Les fréquences des mots (occurrences mots) et les structures syntaxiques (phrases) sont utiles pour aider les moteurs de recherche à trouver les références relatives à la pathologie spécifiques d'un patient. Pour les définitions de dictionnaire, nous faisons référence aux définitions de deux dictionnaires : *Dictionnaire de la langue française*, d'Émile Littré, en ligne aussi (Littré, É. 1956-1957 & 1957-1958), à l'origine de la fixation de nombreux termes médicaux, et le Dictionnaire Garnier, Delamarre et al., la dernière édition.

Le module «Connaissances sur ses propres capacités» vérifie si la connaissance qui correspond au texte est disponible dans le corpus.

6.3.3.3 Analyse du texte correspondant aux processus de décision pour l'antibiothérapie

Nous partons d'un texte fourni par l'Agence du médicament, l'extrait étudié ici, correspond aux pages 5 et 6 et il a pour titre : Principes d'Antibiothérapie1. A partir de ce texte, on détermine les bonnes conduites de l'antibiothérapie et le processus de décision afférent. L'analyse du texte est effectuée avec l'application Tagger. Elle permet d'extraire des substantifs, adjectifs, adverbes et les verbes qui définissent la méta-connaissance nécessaire au choix d'un antibiotique, comme implanté dans le système SIAMED. On y retrouve les différentes étapes qui ont déjà écrites plus haut, à partir d'exemples issus des précédents travaux de J. Colloc (1985).

À partir du texte, on retrouve les différentes étapes : comment choisir un antibiotique, sur des critères bactériologiques, cliniques, pharmacocinétiques, risques et médico-économiques. On y retrouve également les informations utiles concernant les antibiotiques qui ont déjà été analysés dans d'autres textes : Espèces habituellement sensibles, Espèces résistantes, Espèces modérément sensibles ou de sensibilité intermédiaire et Espèces inconstamment sensibles.

Exemple :

« Critères cliniques : le tableau clinique, le site de l'infection, les données épidémiologiques générales permettent de présumer de la bactérie responsable donc de sa sensibilité usuelle (exemple : Pénicilline pour une angine présumée streptococcique). »

Il y a deux catégories d'antibiotiques : antibiotiques bactéricidiques et bactériostatiques (attribut de la classe antibiotique). Un antibiotique est prescrit quand l'infection n'est pas lié à un virus, à un champignon, à un parasite. Si une bactérie est la cause donc on prescrit l'antibiotique.

---

[1] Ministère des Affaires Sociales de la Santé et de la Ville, « Le Bon Usage du Medicament », Agence du Médicament ed. Comité Français d'Education Pour la Santé (CFES), 1993, 159 p.
Les catégories précisées correspondent aux scores suivants dans SIAMED :
Espèces résistantes : 0
Espèces inconstamment sensibles : 1
Espèces modérément sensibles ou de sensibilité intermédiaire : 2
Espèces habituellement sensibles : 3 .

À travers de l'analyse, le tableau clinique, le site de l'infection, et les données épidémiologiques générales sont les points clés pour déterminer la source d'infection. La méta-connaissance des critères cliniques incluant la liste des bactéries susceptibles de provoquer la maladie (streptocoque, staphylocoque, haemophilus..).

```
>>>
Please input the statment:
Critères cliniques : le tableau clinique, le site de l'infection, les données ép
idémiologiques générales permettent de présumer de la bactérie responsable donc
de sa sensibilité usuelle (exemple : Pénicilline pour une angine présumée strept
ococcique
Terminologie :  critère
Adjectif :  clinique
ArticleDefini : le
Terminologie :  tableau
Adjectif :  clinique
ArticleDefini : le
Terminologie :  site
Preposition : de
Terminologie :  infection
ArticleDefini : les
Terminologie :  donnée
Verbe :  épidémiologiques
Verbe :  générales
Verbe :  permettent
Preposition : de
Preposition : de
ArticleDefini : la
Terminologie :  bactérie
Adjectif :  responsable
Conjonction : donc
Preposition : de
Verbe :  usuelle
Terminologie :  pénicilline
Preposition : pour
ArticleDefini : une
Terminologie :  angine
>>>
```

Figure 65 Exemple1. Les résultats du programme de l'extraction des termes

Pour le site de l'infection comme exemple, il conditionne le tissu cible de l'antibiotique. La liste de 16 tissus cible (le site de l'infection) indique la pénétration des antibiotiques dans ces tissus.

Le processus pour l'antibiothérapie y compris le détermination de l'interaction médicamenteuse, le contre-indication médicamenteuse, les effets secondaires, etc. Les temps d'utilisation d'antibiotique doivent coordonner le processus d'antibiothérapie.

Il faut également réfléchir les paramètres liés à l'âge (nouveau-né, personne-âgée et conditions physiologiques : grossesse allaitement).

6.3.3.4 Analyse de texte médical pour élaborer l'ontologie

Construction du corpus

Nous établissons six corpus, sur le domaine des antibiotiques et de l'antibiothérapie : corpus des substantifs, des adjectifs, des prépositions, des adverbes, des conjonctions, des verbes, pour réaliser l'extraction des relations et des termes composant la terminologie.

Le corpus des verbes comporte deux catégories : les radicaux et les suffixes de verbes.

- Les radicaux de verbes contiennent tous les radicaux des verbes du français, sans les suffixes de terminaison : -er, -ir, -re et -oir.
- Les suffixes des verbes présentent les marqueurs de la conjugaison en français : le premier groupe {-er, -e, -es, -e, -ons, -ez, -ent, -ais, -ait, -ai, -aient, etc.}, le deuxième groupe { -ir, -is, -it, -issons, -issez, -issent, -issais, etc.} et le troisième groupes (liste exhaustive en annexe).

  Les cas particuliers de verbes sont détaillés dans le corpus : les verbes du premier groupe, en -er (eg. Lancer), -e-<consonne>-er (eg. jeter, ramener), -é-<consonne(s)>-er (eg. sécher), et -yer (eg. payer, appuyer, grasseyer) ; les verbes « haïr » et « fleurir », du deuxième groupe (-ir) ; ainsi que tous les verbes du troisième groupe, avec des conjugaisons irrégulières.

Notre approche

L'approche de la modélisation d'un domaine à partir de textes consiste à (i) identifier les termes caractérisant le domaine, pour ensuite (ii) extraire les relations sémantiques qui les unissent.

Cette approche peut être utilisée dans un contexte de génération à partir des textes du domaine. Nous préconisons ici, que le repérage d'une relation fréquente (de nombreuses fois reproduite) dans une ontologie, indique que cette phrase peut être porteuse d'informations. Donc, disposant de l'ensemble des relations et de l'ensemble des termes qui peuvent être liés par ces relations, l'approche que nous proposons pour la génération d'une ontologie se décompose en deux étapes (présentées dans la Figure 67): (i) identification des termes constituant les arguments de cette relation, et (ii) détection d'une instance de relations dans le texte.

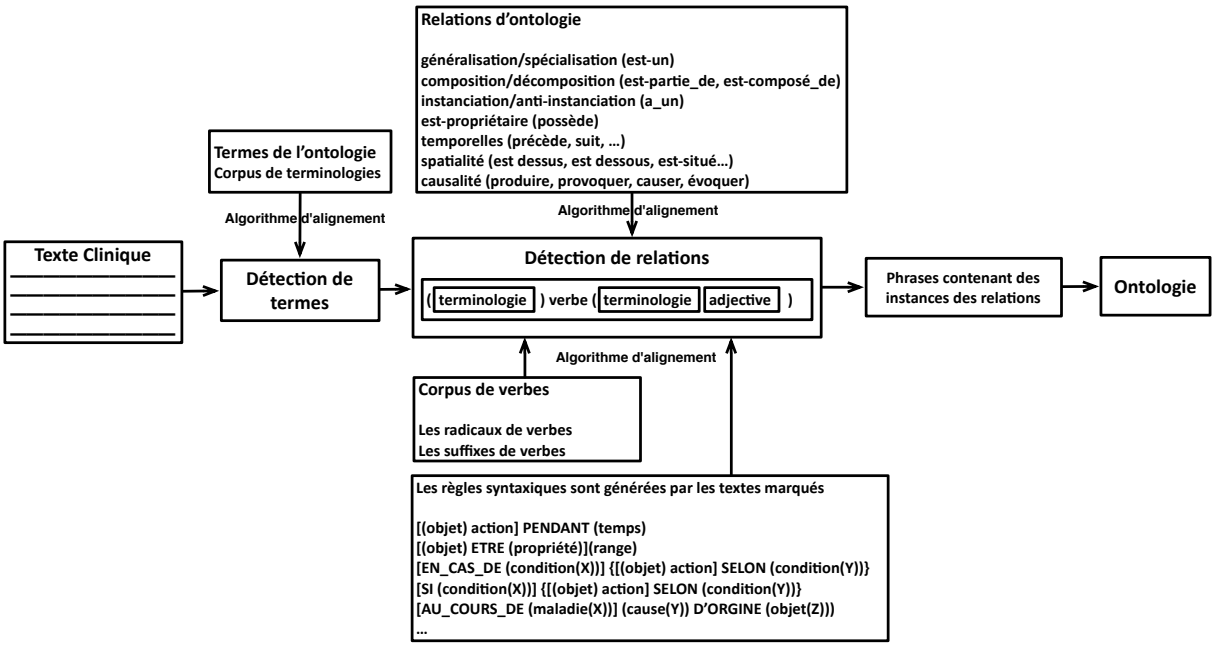


Figure 66 Les étapes de la génération des verbes et des terminologies basées à partir du texte

En nous fondant sur cette approche, nous avons proposé une méthodologie pour la génération des connaissances à partir de textes : elle se décompose en quatre phases :

- La première phase consiste à effectuer une série d'analyses linguistiques sur le texte : marquage des adjectifs, adverbes et prépositions, afin de préparer les phases d'extraction ;
- La deuxième phase consiste à analyser les phrases pour en extraire les termes associés aux concepts de l'ontologie ;
- La troisième vise à repérer les instances des relations de l'ontologie. Tout d'abord, avec le programme, le corpus des radicaux des verbes se déroulera le processus ergodique[2] avec les suffixes de verbes, afin de les repérer dans des phrases. En outre, nous considérons qu'un ensemble de verbes et de syntagmes verbaux caractérise chaque relation. L'apparition d'un de ces syntagmes dans le texte peut être considérée comme une instanciation de cette relation ;
- Enfin, la quatrième et dernière phase consiste à collecter toutes les informations issues des phases précédentes (adjectifs, prépositions, adverbes et conjonctions), pour formaliser un diagramme de classes UML, nécessaire à la génération d'une ontologie, dans nouvelles étapes.

[2] Ergodique : ensemble des possibilités combinatoires du paradigme des radicaux verbaux et des suffixes de terminaison, filtrés par leur existence dans la langue.

***Mapping* des relations, des termes et des structures de phrases**

Dans ce contexte, on effectue le *mapping*, à partir d'expressions textuelles, représentatives des termes et des relations extraits, et des relations remarquables (est-un, est-partie de, a-un), complémentaires (temporelles : précède, suit ... ; causales : cause, provoque ...) ; le tout est associé au vocabulaire choisi (analyse lexicale) et aux schémas de phrases (structures morphosyntaxiques).

L'implémentation et l'expérimentation de la représentation des objets, des classes d'objets, des associations et des attributs convergent sur la transformation des connaissances. Après avoir modélisé le texte et analysé les objets et leurs interactions, on établit les interfaces correspondantes aux besoins en connaissances des utilisateurs.

Afin de relier les connaissances extraites et le diagramme de méta-modèle de l'ODMG, nous utilisons le langage de modélisation graphique UML qui permet une traduction aisée en Java, associée à des langages sémantiques, comme XML et ses dérivés OWL. Nous proposons un diagramme de séquences UML, pour détailler le processus de la représentation des objets et autres éléments importants du projet SIAMED.

- Le *ModuleInterpréter* vise à réaliser l'analyse lexicale, le POS tagger et le tagger de sens, et l'extraction de relations avec les méthodes *LexicalAnalysis*(), *Tagger*() et *RelationExtraction*().
- Le module *ConnaissanceCapacités* vérifie si les connaissances existantes déjà dans le corpus, avec la méthode *MatchingAlgorithm*(), *VerifyTermExistence*(), et *VerifyAssociationExistence*().
- Le méthode *ExtractTermAndAssociation*() fonctionne pour extraire et générer de termes et les associations.
- Le module *UML* permet de visualiser les connaissances aux utilisateurs.

Ce processus permet la génération des termes et des associations, et le *mapping* des objets, des classes d'objets, des associations et des attributs entre méta-modèle et expressions textuelles. En fin de compte, il fournit un diagramme visuel aux utilisateurs.

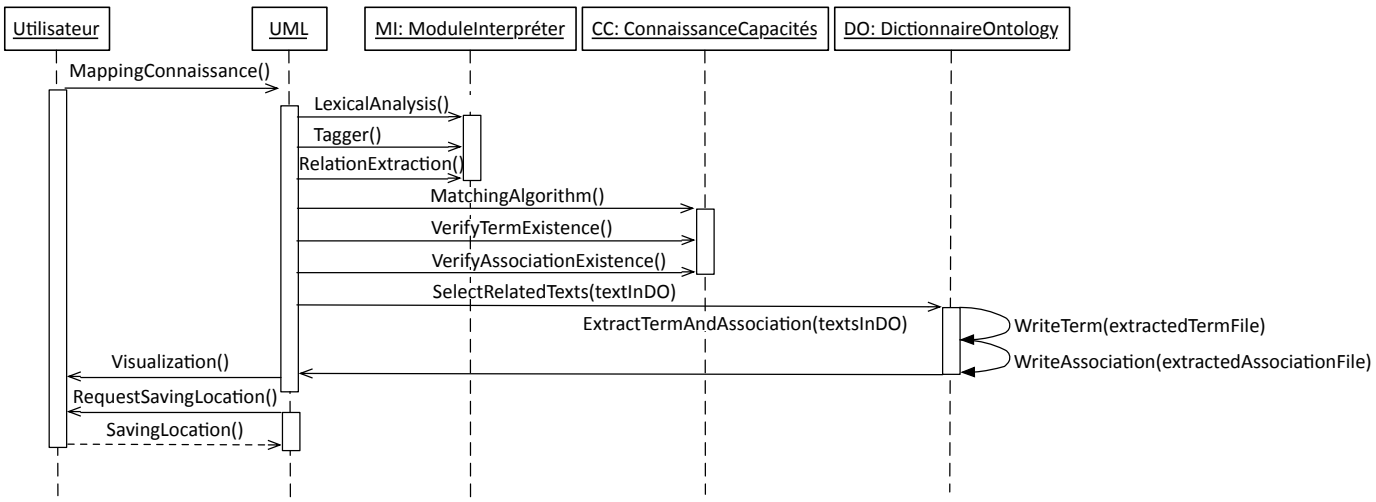


Figure 67 Diagramme de séquence : Représentation des objets, des classes d'objets, des associations et des attributs du projet SIAMED

Le contenu de la hiérarchie est étendu de manière dynamique, de nouveaux documents sont ajoutés à la base de connaissances. L'objectif du système est d'acquérir de nouveaux termes et associations, *via* le méta-modèle, et de les placer dans les hiérarchies de domaines existants.

Comment choisir un antibiotique ?

Le choix est raisonné et doit prendre en compte différents critères tels que les critères cliniques, bactériologiques, pharmacocinétiques.

Le tableau clinique, le site de l'infection, les caractéristiques médicamenteuses, les critères cliniques permettent de présumer de la bactérie responsable, donc de sa sensibilité usuelle. Ils correspondent à la liste des bactéries susceptibles de provoquer la maladie (eg. streptocoque, staphylocoque, haemophilus…).

I. Tableau clinique

Si nous nous référons à l'exemple de l'enfant, un certain nombre de syndromes sévères peuvent être observés, qui contre-indiquent de manière absolue l'utilisation d'un certain nombre de produits :

- Les phénicolés engendrent chez le nouveau-né et le prématuré une cyanose grisâtre qui conduit à la mort dans un tableau de collapsus cardio-vasculaire.
- Les tétracyclines engendrent une coloration définitive des dents chez l'enfant.
- Les sulfamides engendrent une érythrodermie grave ou syndrome de Lyell, chez le jeune enfant.

Il faut donc éviter hormis les cas où la gravité de la maladie l'exige (Typhoïde ...) de prescrire ces médicaments chez l'enfant.

La manipulation automatique de textes écrits en langage naturel, quelle que soit la langue utilisée, requiert souvent une première analyse des entités formant ces textes. L'objectif de cette étape est de prélever toute information permettant de caractériser le sens et les emplois d'un mot dans ses contextes d'énonciation.

À propos de l'antibiothérapie, il faut montrer comment, dans le texte, à partir des substantifs, on repère les objets, les classes nécessaires, les attributs des objets (les valeurs représentées) et les verbes (associations et actions) utiles pour effectuer le choix de l'antibiotique le mieux adapté.

Le résultat de l'extraction des relations et des termes de phrase « Les sulfamides donnent chez le jeune enfant une érythrodermie grave ou syndrome de Lyell. » est présenté ci-après.

```
>>> ================================ RESTART ================================
>>>
Please input the statment:
Les phénicolés engendrent chez le nouveau-né et le prématuré une cyanose grisâtr
e qui conduit à la mort dans un tableau de collapsus cardio-vasculaire.
ArticleDefini : les
Terminologie :  phénicolé
Verbe :  engendrent
Preposition : chez
ArticleDefini : le
Terminologie :  nouveau-né
Conjonction : et
ArticleDefini : le
ArticleDefini : une
Terminologie :  cyanose
Adjectif :  grisâtre
Pronom : qui
Verbe :  conduit
Preposition : à
ArticleDefini : la
Verbe :  mort
Preposition : dans
ArticleDefini : un
Terminologie :  tableau
Preposition : de
Terminologie :  collapsu
Adjectif :  cardio-vasculaire
>>>
```

Figure 68 Exemple2. Les résultats du programme de l'extraction des termes

Ces résultats du programme peuvent déterminer le substantif ou nom (type d'objet ou type d'entité), l'adjectif (attributs descriptifs) et le verbe (relation).

Les méthodes basées sur les règles sont également utilisées dans nos recherches. Elles reposent sur la création manuelle de règles d'extraction fondées sur les particularités spécifiques à une classe de termes. Par exemple, les mots se terminant par -ase et -in peuvent être considérés comme des enzymes ou des protéines (Fukuda, K. et al. 1998). Ainsi, les termes vérifiant l'expression régulière [az]+[0-9] peuvent

être considérés comme des gènes (une séquence de lettres suivie d'une séquence de chiffres) (Hobbs, J.R. et al. 1997).

Pour approfondir et vérifier les résultats de l'extraction de termes et des relations, l'analyse lexicale est utilisée car elle permet de rechercher l'existence des mots et des expressions dans un dictionnaire ontologique, et de confirmer ou d'infirmer l'existence des morphèmes identifiés par l'analyse morphologique.

Avec les résultats de l'analyse, les taggers sont utilisés pour représenter les structures syntaxiques d'un texte sous forme symbolique (*cf*. 5.2.4). Il s'agit de la mise en évidence des structures d'agencement des catégories grammaticales (nom, verbe, adjectif, etc.), pour en exprimer les relations formelles ou fonctionnelles (par exemple, sujet, verbe et complément). Ci-après, nous présentons un exemple de POS Tagger (Part Of Speech Tagger : système d'étiquetage) de cette phase.

> Les(art.déf.) sulfamides(n.) donnent(verb) chez(prep.) le(art.déf.) jeune(adj.) enfant(n.) une(art.déf.) érythrodermie(n.) grave(adj.) ou(conj.) syndrome(n.) de lyell(n.).

À chaque *token* (occurrence), l'étiquetage grammatical du texte associe une catégorie grammaticale (Nom, Verbe, Adjectif, etc.), en tenant compte des résultats de l'extraction et des définitions des hiérarchies d'ontologies (*cf*.5.4.2). Voici un exemple de Tagger de sens.

> Les **[sulfamides]**/MEDICAMENT **[donnent]**/RELATION **[chez le jeune enfant]** /PERSONNE une **[érythrodermie]**/MALADIE grave ou **[syndrome de Lyell]**/SYMPTOME.

Notre travail consiste à analyser un texte sur les antibiotiques et à permettre de générer les relations existantes entre les objets et les types d'objets. Ces approches se fondent sur les traces linguistiques qui marquent les relations sémantiques dans le texte pour construire des taggers qui permettent la détection de ces relations. Un tagger peut être considéré comme une formule linguistique dont les mots indiquent une relation à vérifier dans le texte.

Cette identification consiste à déterminer les différentes formes syntaxiques d'apparition des relations dans le texte. Ces formes peuvent être considérées comme des instances possibles de relations formalisées dans l'ontologie. Nous pouvons résumer cela par la phrase :

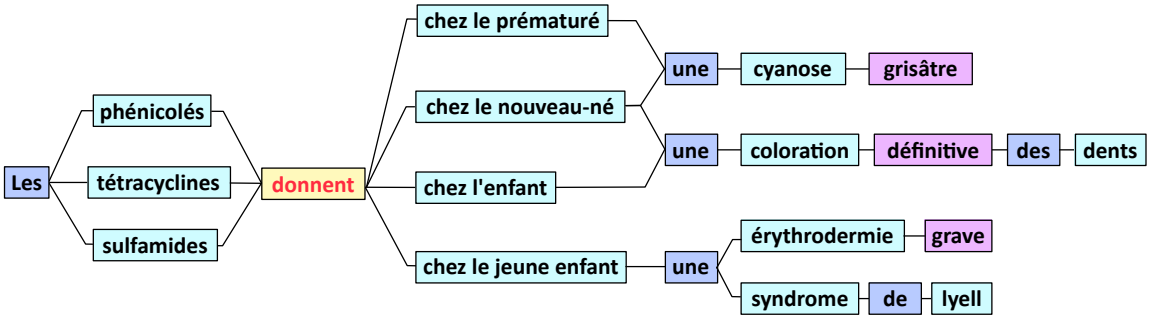


Figure 69 Représentation des termes et des relations de phase d'extraction

Le découpage de la phrase permet de repérer les frontières de chaque segment, afin de les distinguer, en vue d'un traitement sémantique, nécessaire à l'analyse lexico-grammaticale. Grâce aux résultats de l'analyse lexicale des taggers et de l'extraction de relations, les contenus de phrases sont segmentés puis représentés par les fonctions de correspondance (*Matching functions*), telles que montrées ci-dessous :

Liste de segmentation	**Fonctions de correspondance** (*Matching functions*)
1) Les **sulfamides**	Présenter le nom de médicament
2) **donnent**	**Verbes** – l'association de réseaux sémantiques (propositions d'action).
3) **chez le jeune enfant**	Indiquer le patient qui prend le médicament
4) une **érythrodermie** grave	Détailler la maladie qui est provoquée par de médicament.
5) ou	Conjonction de coordination, qui sert à exprimer une alternative ou une équivalence
6) **syndrome de Lyell**	Montrer le symptôme (Δ) Exposer le degré de gravité de la maladie du patient.

Les relations et les termes extraits, de même que les fonctions de correspondance, permettent d'établir le diagramme de classe en UML, afin de formaliser les connaissances.

II. Site de l'infection

Le site de l'infection est le deuxième facteur important pour déterminer les bactéries provoquant la maladie. Il faut pouvoir atteindre le germe dans le site de

l'infection et, pour cela, la diffusion de l'antibiotique dans le tissu infecté doit être suffisante pour obtenir des taux dépassant la CMI.[3]

Certains antibiotiques diffusent très bien dans certaines zones de l'organisme, alors qu'ils diffusent peu dans d'autres. En conséquence, l'extraction de connaissance est nécessaire pour déterminer quel antibiotique est adapté à l'infection. Prendre ces trois phrases comme exemple :

- Les Ampicillines sont considérablement concentrées au niveau de la bile ce qui peut être utile en cas d'infection des voies biliaires.
- Certains antibiotiques, qui ne sont pas absorbés par voie orale, peuvent traiter électivement des infections du tube digestif tout en limitant la toxicité (eg. Néomycine).
- Certains germes sont intracellulaires de sorte que seuls des antibiotiques, ayant une bonne pénétration cellulaire, seront actifs (eg. les tétracyclines, les phénicolés,...).

L'extraction des relations et des concepts de la phrase « Les Ampicillines et notamment la Métampicilline sont considérablement concentrées au niveau de la bile ce qui peut être très utile en cas d'infection des voies biliaires. » est montrée ci-après.

```
>>> ================================ RESTART ================================
>>>
Please input the statment:
Les Ampicillines sont considérablement concentrées au niveau de la bile ce qui p
eut être utile en cas de infection des voies biliaires.
ArticleDefini : les
Terminologie :  ampicilline
Verbe :  sont
Adverbe :  considérablement
Adjectif :  concentrée
ArticleDefini : au
Terminologie :  niveau
Preposition : de
ArticleDefini : la
Terminologie :  bile
Pronom : ce
Pronom : qui
Verbe :  peut
Verbe :  être
Adjectif :  utile
Preposition : en cas de
Terminologie :  infection
Preposition : des
Verbe :  voies
Terminologie :  biliaire
>>>
```

Figure 70 Exemple3. Les résultats du programme de l'extraction des termes

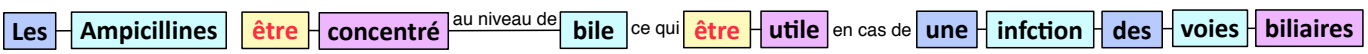


Figure 71 Représentation des termes et des relations pour déterminer le site de l'infection

[3] CMI : concentration minimum inhibitrice, à laquelle un antibiotique est actif sur un germe.

Avec le même principe, les relations de ces trois phrases peuvent être représentées par le Tagger de sens.

Les **[Ampicillines]**/ANTIBIOTIQUE **[sont]**/RELATION considérablement concentrées au niveau de la **[bile]**/CORPS LIQUIDES ce qui **[peut être]**/RELATION très utile en cas d'**[infection des voies biliaires]**/MALADIE.

Certains **[antibiotiques]**/ANTIBIOTIQUE **[ne sont pas absorbés]**/RELATION par **[voie orale]**/MOYEN **[peut traiter]**/RELATION électivement des **[infections du tube digestif]**/MALADIE tout en limitant la toxicité (eg. **Néomycine**/EXEMPLE).

Certains **[germes]**/GERME **[sont]**/RELATION intracellulaires de sorte que seuls des **[antibiotiques]**/ANTIBIOTIQUE **[ont]**/RELATION bonne **[pénétration]**/PENETRATION cellulaire **[seront]**/RELATION actifs (eg. **[les tétracyclines, les phénicolés,...]**/EXEMPLE).

Nous pouvons établir les diagrammes UML avec les classes et les relations générées ci-dessus.

III. Interactions médicamenteuses

L'interaction médicamenteuse est un domaine compliqué et indispensable pour éviter les accidents thérapeutiques. Les caractéristiques des pénicillines et des céphalosporines sont présentées ci-après.

Les pénicillines

- Le Probénécide inhibe l'excrétion tubulaire ce qui entraine un retard d'élimination et un renforcement de l'action des pénicillines.
- L'association de l'Allopurinol aux amoxicillines augmente le risque de voir apparaître des accidents cutanés.

Les céphalosporines

- L'acide éthacrynique entraine un retard d'élimination des céphalosporines avec une recrudescence possible de leurs effets secondaires.

- Les anticoagulants sont potentialisés par les céphalosporines ce qui peut entrainer des hémorragies.

```
>>> ================================ RESTART ================================
>>>
Please input the statment:
Le Probénécide inhibe l'excrétion tubulaire ce qui entraine un retard d'éliminat
ion et un renforcement de l'action des pénicillines.
ArticleDefini : le
Terminologie :  probénécide
Verbe :  inhibe
Terminologie :  excrétion
Adjectif :  tubulaire
Pronom : ce
Pronom : qui
Verbe :  entraine
ArticleDefini : un
Terminologie :  retard
Terminologie :  élimination
Conjonction : et
ArticleDefini : un
Adverbe :  renforcement
Preposition : de
Terminologie :  action
Preposition : des
Terminologie :  pénicilline
>>>
```

Figure 72 Exemple4. Les résultats du programme de l'extraction des termes

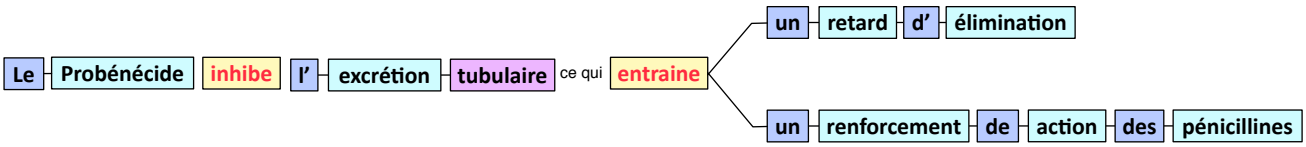


Figure 73 la représentation des terminologies et des relations pour montrer les interactions médicamenteuses

Selon les résultats de l'extraction, le facteur des pénicillines peut être visualisé sous la forme de tagger.

Le **[Probénécide]**/INHIBITEUR **[inhibe]**/RELATION l'**[excrétion tubulaire]**/EXCRÉTION ce qui **[entraine]**/RELATION un **[retard d'élimination]**/SYNDROME et un **[renforcement de l'action]**/EFFECT des **[pénicillines]**/ANTIBIOTIQUE.

Le principe étant le même, et par concision, l'approche résumant d'autres règles sur les médicaments n'est pas représentée.

IV. Contre-indication médicamenteuse

Les contre-indications liées à un terrain physiologique particulier tels que l'âge, la grossesse et l'allaitement.

Les contre-indications sont liées à un état pathologique préexistant tels que la mononucléose infectieuse, la myasthénie, la migraine, l'asthme et les allergies, le déficit en glucose 6 phospho-déshydrogènase, l'insuffisance hépatique, l'insuffisance rénale etc.

Pour la grossesse et l'allaitement, on préfère des antibiotiques de la famille des pénicillines et des macrolides.

Pour les migraineux, il faut éviter la prescription d'antibiotiques de la famille des macrolides et plus particulièrement l'Oléandromycine.

```
>>> ================================ RESTART ================================
>>>
Please input the statment:
Pour la grossesse et l'allaitement, on préfère des antibiotiques de la famille d
es pénicillines et des macrolides.
Preposition : pour
ArticleDefini : la
Terminologie :  grossesse
Conjonction : et
Adverbe :  allaitement
Pronom : on
Verbe :  préfère
Preposition : des
Terminologie :  antibiotique
Preposition : de
ArticleDefini : la
Terminologie :  famille
Preposition : des
Terminologie :  pénicilline
Conjonction : et
Preposition : des
Terminologie :  macrolide
>>>

>>>
Please input the statment:
Pour les migraineux, il faut éviter la prescription d'antibiotiques de la famill
e des macrolides et plus particulièrement l'Oléandromycine.
Preposition : pour
ArticleDefini : les
Pronom : il
Verbe :  faut
Verbe :  éviter
ArticleDefini : la
Terminologie :  prescription
Terminologie :  antibiotique
Preposition : de
ArticleDefini : la
Terminologie :  famille
Preposition : des
Terminologie :  macrolide
Conjonction : et
Preposition : plus
Adverbe :  particulièrement
Terminologie :  oléandromycine
>>>
```

Figure 74 Exemple5. Résultats du programme de l'extraction des termes

Nous générons deux règles de la contre-indication médicamenteuse en cas de grossesse et en cas de migraine :

Pour **[la grossesse]**/PERSONNE et **[l'allaitement]**/PERSONNE, on **[préfère]** / RELATION des antibiotiques de la famille des **[pénicillines]**/ANTIBIOTIQUE et des **[macrolides]**/ANTIBIOTIQUE.

La **[migraine]**/MALADIE doit **[faire]** **[éviter]**/RELATION la **[prescription]**/ PRESCRIPTION d'antibiotiques de la famille des **[macrolides]**/ANTIBIOTIQUE et plus particulièrement **[l'Oléandromycine]**/ANTIBIOTIQUE.

V. Effets secondaires toxiques

La classification, incluant les réactions allergiques au niveau du système nerveux, du sang, des reins, du tube digestif, des dents et des os, permet de détailler les effets secondaires.

Au niveau du **système nerveux périphérique**, il peut s'agir d'une atteinte diffuse touchant les nerfs, on peut observer :

- Un blocage neuromusculaire, avec les antibiotiques polypeptidiques,... ce peut être mortel en cas d'atteinte des muscles respiratoires.
- Une polynévrite peut être observée avec les nitrofuranes, l'isoniazide et l'éthambutol.
- Des paresthésies peuvent être provoquées avec les antibiotiques polypeptidiques et les antituberculeux.

```
>>> ================================ RESTART ================================
>>>
Please input the statment:
Un blocage neuromusculaire, avec les antibiotiques polypeptidiques, ce peut être
 mortel en cas d'atteinte des muscles respiratoires.
ArticleDefini : un
Terminologie :  blocage
Terminologie :  neuromusculaire
Preposition : avec
ArticleDefini : les
Terminologie :  antibiotique
Terminologie :  polypeptidique
Pronom : ce
Verbe :  peut
Verbe :  être
Adjectif :  mortel
Preposition : en
Preposition : des
Terminologie :  muscle
Adjectif :  respiratoire
>>>
```

Figure 75 Exemple6. Résultats du programme de l'extraction des termes

Certains effets secondaires toxiques du système nerveux périphérique sont représentés, avec des classes et des relations extraites, dans la figure ci-dessous.

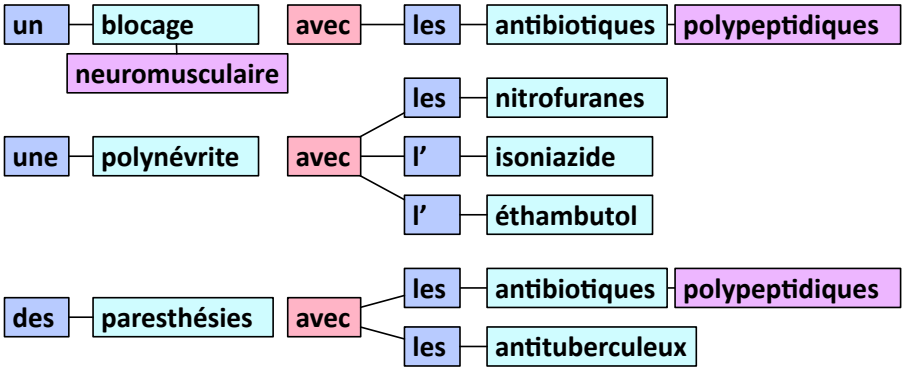


Figure 76 Représentation des termes et des relations syntagmatiques représentant les effets secondaires toxiques du système nerveux périphérique

Soit d'une atteinte élective d'un nerf sensitif :

- L'atteinte des nerfs optiques, avec une diminution de l'acuité visuelle peut être provoquée par les antibiotiques polypeptidiques, les antituberculeux, le chloramphénicol, les quinolones...
- L'atteinte des nerfs cochléaires et vestibulaires se manifeste par une diminution de l'acuité auditive et/ou l'apparition de vertiges. Ce sont essentiellement les aminosides qui sont en cause et, tout particulièrement, la streptomycine et la kanamycine.
- On observe une atteinte des nerfs vestibulaires avec les antibiotiques polypeptidiques...

6.3.3.5 Améliorer la prescription des antibiotiques à l'aide d'ontologies et de schéma UML

l'UML (*Unified Modeling Language*) est un langage orienté objet, capable de représenter différents points de vue nécessaires au développement, puis de déployer les systèmes. Cette section présente notre investigation consacrée à l'UML qui fonctionne comme un outil de modélisation ontologique pour faciliter le *mapping* entre la connaissance et les modèles informatiques.

Pour modéliser et formaliser les connaissances, les ontologies servent à représenter formellement les connaissances afin de faciliter leur partage.

Comme déjà mentionné, l'ontologie est la spécification explicite d'une conceptualisation qui fonctionne comme une base complète de connaissances. Les ontologies peuvent être utilisées comme des outils d'acquisition de connaissances, collectées dans le domaine, ou bien de générer des bases de données ou des systèmes à base de connaissances. Un modèle d'ontologie peut faciliter les processus d'analyse et de représentation de connaissances.

Il existe deux principales composantes d'un système d'aide à la décision : les connaissances de domaine et les connaissances de résolution de problème. Les ontologies influencent les connaissances de résolution de problème, avec les fonctions d'analyse, la modélisation et la représentation des connaissances du domaine (Wang, X. & Chan, C.W. 2001). L'ontologie de domaine du système à base de connaissances est une spécification explicite des objets, des concepts, et des autres entités qui sont censés exister dans certains domaines d'intérêt et entretenir des relations entre eux (Gruber, T.R. 1995). Ce dernier définit un ensemble de termes et les relations d'un domaine, indépendamment de toute méthode pour les résolutions de problème.

Les connaissances sur les antibiotiques et l'antibiothérapie sont extraites, puis représentées sous forme UML. Les médecins expriment souvent la connaissance médicale par des recommandations qui font intervenir des objets issus de niveaux de granularité différents, selon deux acceptions : niveau de spécialisation (une classe) ou un niveau de composition (un composant).

Notre modèle ontologique se fonde sur la richesse sémantique des modèles orientés objet. Nous utilisons donc la notation UML (*Unified Modeling Language*), pour exprimer les relations sémantiques définissant la terminologie d'un domaine de connaissance (Li, S.T. et al. 2010). Sur les figures 78 à 82, nous montrons qu'un diagramme de classes UML offre des capacités graphiques suffisantes, pour coder correctement et de façon rigoureuse une ontologie. Ces ontologies sont exploitées par SIAMED 2 (Système Informatique d'Antibiothérapie MEDicale) (Colloc, J. 1985)

Les terminologies et les relations importantes, extraites à partir de textes, sont marquées par le cadre (☐) et le soulignement (____), respectivement. Ces connaissances sont importées au logiciel OmniGraffle pour générer les diagrammes UML semi-automatique.

Ontologies de monothérapie antibiotique et associations d'antibiotiques

Une monothérapie antibiotique est suffisante dans la plupart des cas. Dans la pathologie infectieuse courante tels que infection ORL, respiratoire basse, urinaire etc., la mono-antibiothérapie est de règle.

Le recours à une association d'antibiotiques est adapté pour les infections sévères (septicémie, méningites, endocardites etc.), les germes présentés pluri-résistants, et les infections à Pseudomonas aeruginosa, etc. Les infections caractérisées tels que brucellose, tuberculose, staphylococcémie... justifient une association.

Les médecins emploient également une association lorsque l'on veut atténuer certains effets secondaires ou empêcher la sélection de mutants résistants.

Couple bactéries-antibiotiques à risque d'émergence de résistances :

- Entérobactéries du groupe 3 (*cf.* enterobacter, serratia, citrobacter freundii, providencia, morganella) et céphalosporines de 3^{e} génération,
- Staphylococcus aureus et fluoroquinolones, rifampicine, acide fusidique ou fosfomycine,
- Entérobactéries résistantes à l'acide nalidixique et fluoroquinolones, etc.

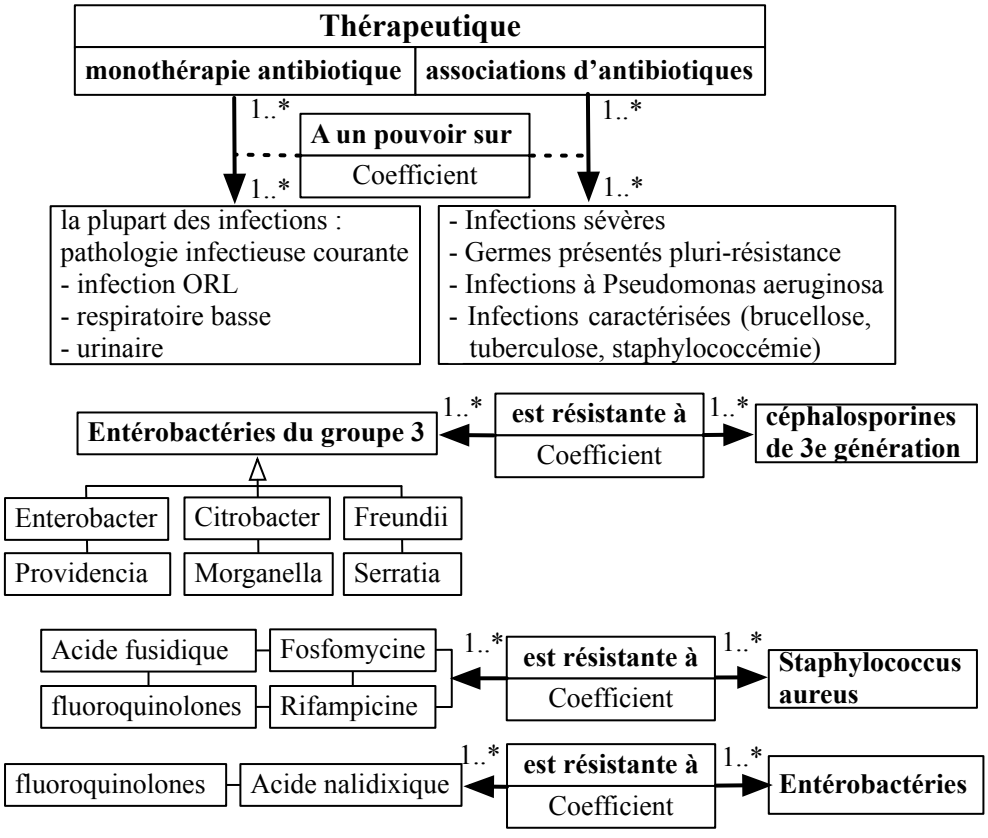


Figure 77 Ontologie des thérapeutiques sous la forme de UML

Ontologies de la durée d'utilisation d'antibiotique

Le traitement dépend de la gravité de l'infection et de sa nature.

L'antibioprophylaxie chirurgicale (Fatras, J.Y. & Goudet, B. 1993), suppose certaines consignes associées au protocole.

- Pendant l'induction anesthésique, effectuer l'injection intraveineuse 1 heure au maximum avant l'incision cutanée ;
- La durée est le plus souvent limitée à celle de l'acte opératoire et ne dépasse pas 24 heures.

Antibiothérapie curative

- L'antibiothérapie curative ne dépasse généralement pas une semaine.
- 15 jours pour une infections urinaires ou infections sévères.

Réévaluation de l'antibiothérapie

- La réévaluation de l'antibiothérapie à propos du maintien d'une éventuelle association doit être discutée entre la 24e heure et la 72e heure ;
- Normalement, le maintien d'une association d'antibiotiques ne doit pas être poursuivi plus de 3 jours, sauf dans de rares situations.

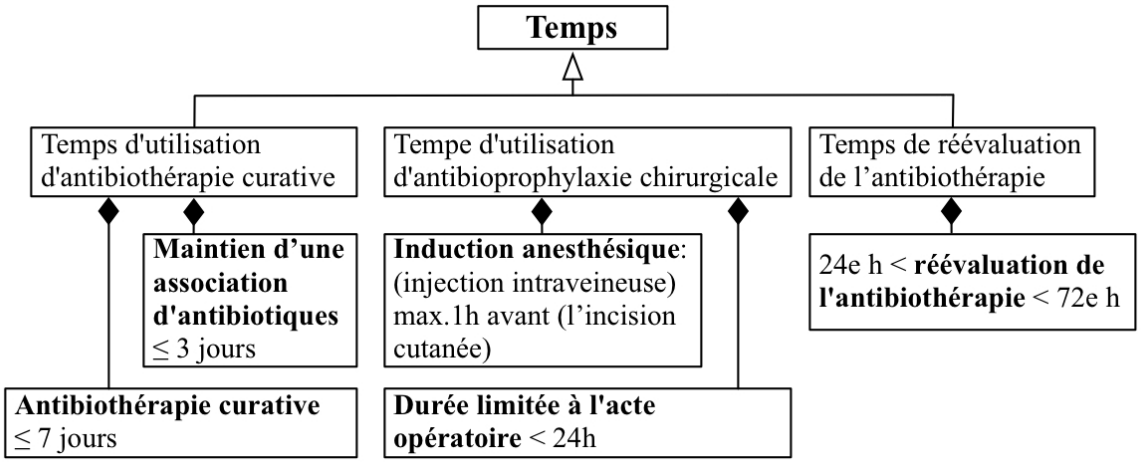


Figure 78 Ontologie des temps d'utilisation d'antibiotique en langage UML

Les ontologies présentées ci-après sont formalisées par les mêmes approches :

Ontologies des médicaments

L'ontologie de la figure 80 décrit la classification des antibiotiques. Les types d'antibiotiques se déclinent en sous-types qui représentent leurs différentes familles d'antibiotiques Beta-Lactamines, Macrolides, Cyclines, Aminosides...

Les antibiotiques constituent une classe complexe de médicaments (17 familles), très utilisées et particulièrement difficiles à prescrire.

Les résistances aux antibiotiques sont des connaissances dynamiques et évolutives. Certaines bactéries deviennent résistantes à un seul antibiotique et, parfois, à une classe entière d'antibiotiques (plasmides). Il est donc essentiel de mettre à jour cette classe de connaissances.

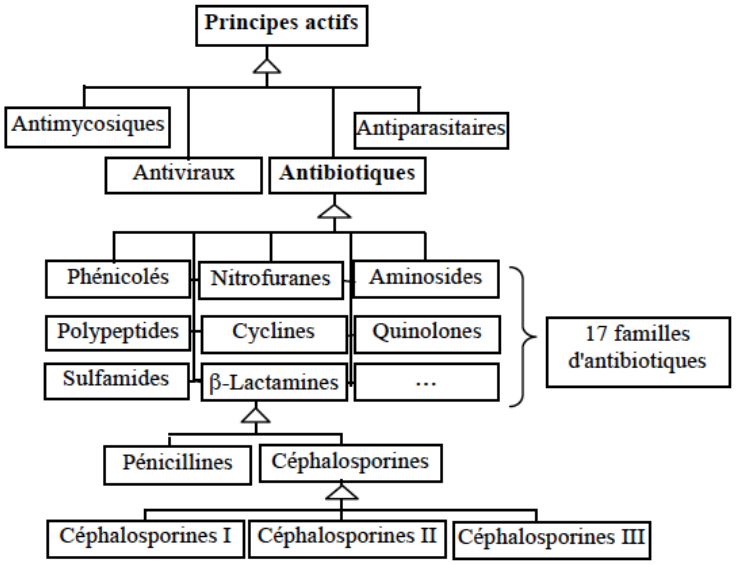


Figure 79 Ontologie des antibiotiques sous la forme de UML (Shen Y. et al. 2012)

Ontologies des posologies et des modalités d'administration

À propos des posologies et des modalités d'administration adaptées aux antibiotiques et à la pathologie du patient, il s'agit de réfléchir au mode d'administration (voie orale, intramusculaire IM, Intraveineuse IV), à la dose de charge, au rythme (mono-dose ou multi-doses journalières) ou en perfusion continue, etc.

La figure 80 montre le fait qu'un surdosage peut être à l'origine de pathologies iatrogènes. Dans ce cas, le recours au dosage sérique des antibiotiques est utile pour certaines molécules : les glycopeptides, aminosides, etc., par exemple.

Cette figure présente certaines règles : le surdosage d'antibiothérapie peut provoquer des pathologies iatrogènes, tandis que le dosage sérique des antibiotiques peut prévenir la survenue de ces pathologies. En outre, le dosage sérique des antibiotiques détecte l'éventuel surdosage de certaines molécules (*cf.* glycopeptides, aminosides, etc.). La condition physique du patient a une incidence, elle peut affecter la prescription d'antibiotiques.

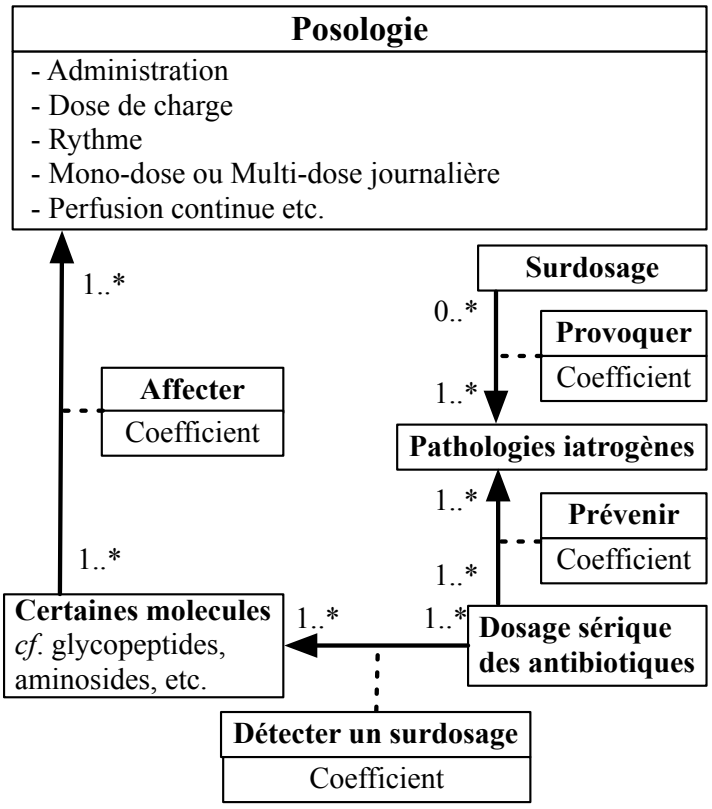


Figure 79 Ontologie des posologies en langage UML

D'autres règles sont à prendre en compte également :

- Les posologies seront diminuées pour l'enfant, le vieillard et lorsqu'il existe une contre-indication relative.
- Les doses sont exprimées en Million d'unités internationales (MUI) ou en mg/Kg/jour pour les pénicillines.
- Les doses sont généralement de l'ordre de 10 à 50 mg/Kg/jour mais il existe des variations importantes selon les produits.

Ontologies du Dossier Médical

La figure 81 exprime le fait qu'une ou plusieurs bactéries peuvent provoquer une maladie infectieuse bactérienne.

Nous souhaitons donc pouvoir disposer d'un véritable héritage conceptuel, concernant cette association « action bactéricide » et la propriété *coefficient* correspondante. Il s'agit de l'héritage de la valeur du coefficient exprimant la capacité de l'antibiotique (ou de la famille d'antibiotiques) à agir sur un germe (ou une famille de germes). La valeur héritée doit pouvoir être masquée, lors de la spécialisation de sous-classes, pour pouvoir exprimer les spécificités d'action de certains antibiotiques ou les résistances de certaines souches bactériennes (Colloc, J. 2000).

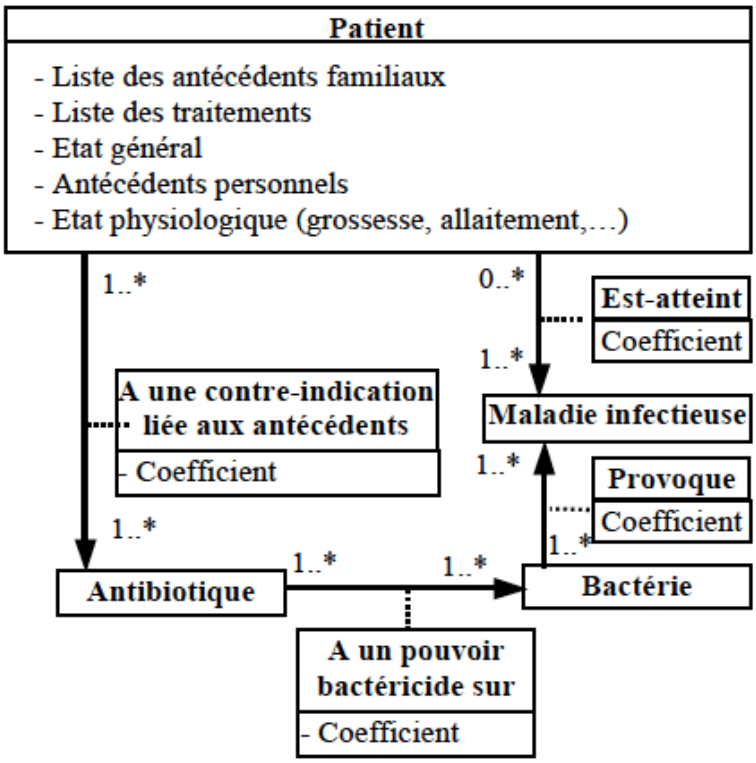


Figure 80 Ontologie de l'antibiothérapie en UML (Shen Y. et al. 2012)

Capitalisation des connaissances *via* l'UML pour l'aide à la décision : choix d'un antibiotique

Pour illustrer l'approche de modélisation intégrée en UML, nous l'appliquons au développement de l'ontologie du projet SIAMED 3 qui repose sur une base de connaissances, concernant les antibiotiques (connaissances) et les règles de l'antibiothérapie (processus de décision), stockées dans le « Modèle d'ontologie ».

Les tableaux 34 et 35 évoquent trois niveaux d'action, intitulés : « Action de l'antibiotique sur le germe », et trois niveaux d'importance, désignés sous l'expression « Poids du germe dans la maladie ». Concernant les informations sur un antibiotique,

le spectre d'activité antibactérienne d'un antibiotique est noté : SPECTRE = Liste des bactéries sensibles à l'antibiotique. La possibilité de décrire l'activité en quatre classes pour clarifier l'activité des différents antibiotiques pour le prescripteur potentiel a été substituée aux trois catégories précédemment connues (sensible, intermédiaire, résistant).

Les catégories précisées correspondent aux scores suivants dans SIAMED :

Code activité = 0 : Espèces résistantes

Code activité = 1 : Espèces sensibles de façon inconstante.

Code activité = 2 : Espèces modérément sensibles ou de sensibilité intermédiaire

Code activité = 3 : Espèces habituellement sensibles

On cherche à déterminer des règles de conduite, pour la prescription des antibiotiques, à partir de la nature du germe, afin d'aider le médecin à trouver le ou les antibiotique(s) les plus actifs, pour éradiquer les germes les plus fréquemment responsables de la maladie.

Ces deux tableaux montrent des données d'interaction entre l'action de l'antibiotique sur le germe et le poids du germe dans la maladie.

		Action de l'antibiotique sur le germe		
	X/Y	**1**	**2**	**3**
Poids du germe dans la maladie	**1**	0.0625	0.125	0.25
	2	0.125	0.25	0.5
	3	0.25	0.5	1

Tableau 34 Score 1 - Tests de formules d'aide à la décision au choix d'antibiotiques

	Action de l'antibiotique sur le germe		
X/Y	**1**	**2**	**3**
1	4	8	16
2	8	16	32
3	16	32	64

Tableau 35 Score 2 - Tests de formules d'aide à la décision au choix d'antibiotiques

Le tableau 36 montre comment l'on calcule l'action d'un antibiotique sur une maladie angine érythémato-pultacée, à l'aide d'une méthode empirique qui produit une fonction caractéristique floue définie dans les tableaux 34 et 35.

				Rang 3			Rang 1				Rang 2
Angine erythémato pultacée	poidsGerme	ampicilline	Score1	Score2	erythrocine	score1	Score2	Score max	PeniG	score1	Score2
Streptocoque	3	2	0.5	32	3	1	64	64	3	1	64
Staphilocoque	2	1	0.125	8	2	0.25	16	32	1	0.125	8
Ménincocoque	1	2	0.125	8	3	0.25	16	16	3	0.25	16
Hémophylus	2	2	0.25	16	2	0.25	16	32	1	0.125	8
Gonocoque	2	2	0.25	16	2	0.25	16	32	2	0.25	16
				80			128	176			122
			SC2/Scmax	0.454545455			0.727272727				0.636363636

Tableau 36 Classement des antibiotiques actifs sur l'ensemble des germes responsables de l'angine érythémato pultacée

Il faut extraire les classes et les relations pour la modélisation d'entités/associations « Entité-Association ». À partir de ces trois tableaux, certaines classes utiles peuvent être extraites :

Action de l'antibiotique sur le germe/[ACTION]
Poids du germe dans la maladie/[GERME]
Angine érythémato-pultacée/[MALADIE]
Streptocoque, Staphylocoque, Ménincocoque, Hémophylus, Gonocoque/[BACTÉRIES]
Ampicilline, Erythrocine, PeniG/[ANTIBIOTIQUE]

Les relations entre les différentes classes sont :

[Streptocoque]-[Ampicilline] : **2**
[Hémophylus]-[PeniG] : **1**
[Pénicilline]-[Ampicilline] : **is-a**
[Antibiotique]-[AngineÉrythématoPultacée] : **agit sur**

Dans cette section, les diagrammes d'Entité-Association sont utilisés pour fournir un cadre pratique pour la conception, le développement et la documentation de base de connaissances. Ils représentent les relations entre les entités impliquées dans un système d'information, et ils montrent les relations entre les classes de l'ontologie.

Certaines classes placées dans la hiérarchie ontologique sont montrées sur la figure UML ci-après. Les antibiotiques mis en évidence en bleu sont ceux utilisés dans le tableau 36.

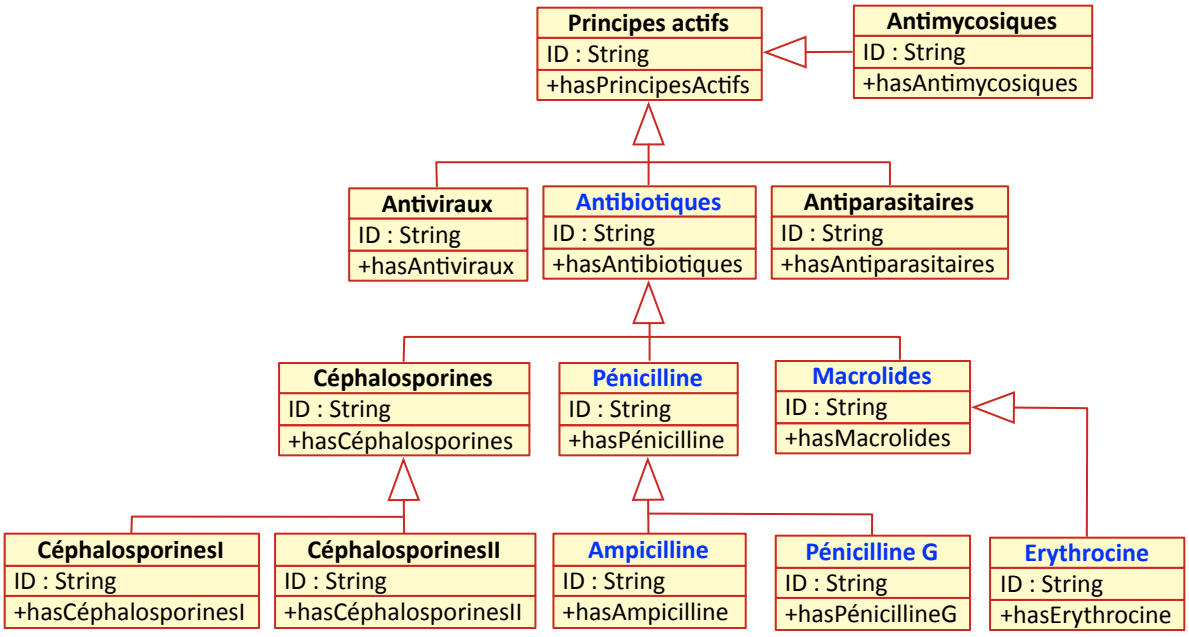


Figure 81 Diagramme de classe : Principes actifs

Avec les relations correspondantes du domaine, la figure 84 expose en détail l'interaction entre l'antibiotique et la maladie, et elle montre ses résultats (score1 et score2). La représentation de l'ontologie à travers des diagrammes UML formalise la connaissance.

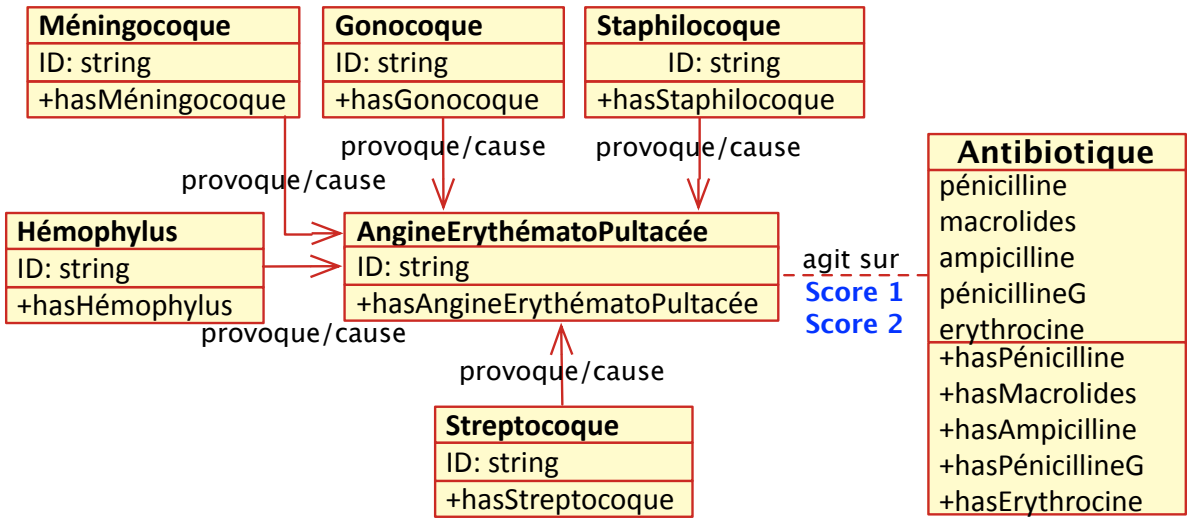


Figure 82 Diagramme de classes : interaction entre la maladie angine érythémato-pultacée et antibiotique

Notons, ici, que sur le diagramme en UML, les méthodes préfixées par « has » sont des méthodes statiques qui permettent de tester le fonctionnement des classes. Exemple : +hasGonocoque, dans la classe **Gonocoque**.

Le protocole, les objets, les classes d'objets, les associations et les attributs du processus de filtrage de SIAMED (figure 61) sont représentés ci-après.

La classe d'objets «*Maladie Infectieuse*» comporte l'attribut «*Germes pathogènes*» et l'attribut «*Tissu cible*». « [L']*Ensemble initial d'antibiotique*» agit sur le «*Germes pathogènes*» et diffuse sur le «*Tissu cible*», afin d'obtenir les données expérimentales et statistiques et produire «[l']*Ensemble Résultant 1*». Cette étape consiste à déterminer l'ensemble des antibiotiques qui peuvent être actifs sur l'organisme responsable de la maladie infectieuse et pour pénétrer jusqu'au tissu infecté par le germe pathogène.

Dans la phase d'élimination, les classes «*Antécédents du patient*» et «*Indication et Contre-Indication*» affectent les résultats de «[l']*Ensemble Résultant 1*», puis produisent « [l']*Ensemble Résultant 2*». Après avoir vérifié « [l']*Ensemble Résultant 2*» par la classe «*Effets secondaires selon le pronostic de la maladie*», nous pouvons obtenir « [l']*Ensemble Résultant Final*» qui vise à répondre aux requêtes de médecins, afin d'établir le traitement du patient. Cette étape est nécessaire pour éliminer les antibiotiques contre-indiqués, ceux qui provoquent des interactions médicamenteuses néfastes, ou qui font encourir un risque élevé d'effets secondaires selon l'état de patient.

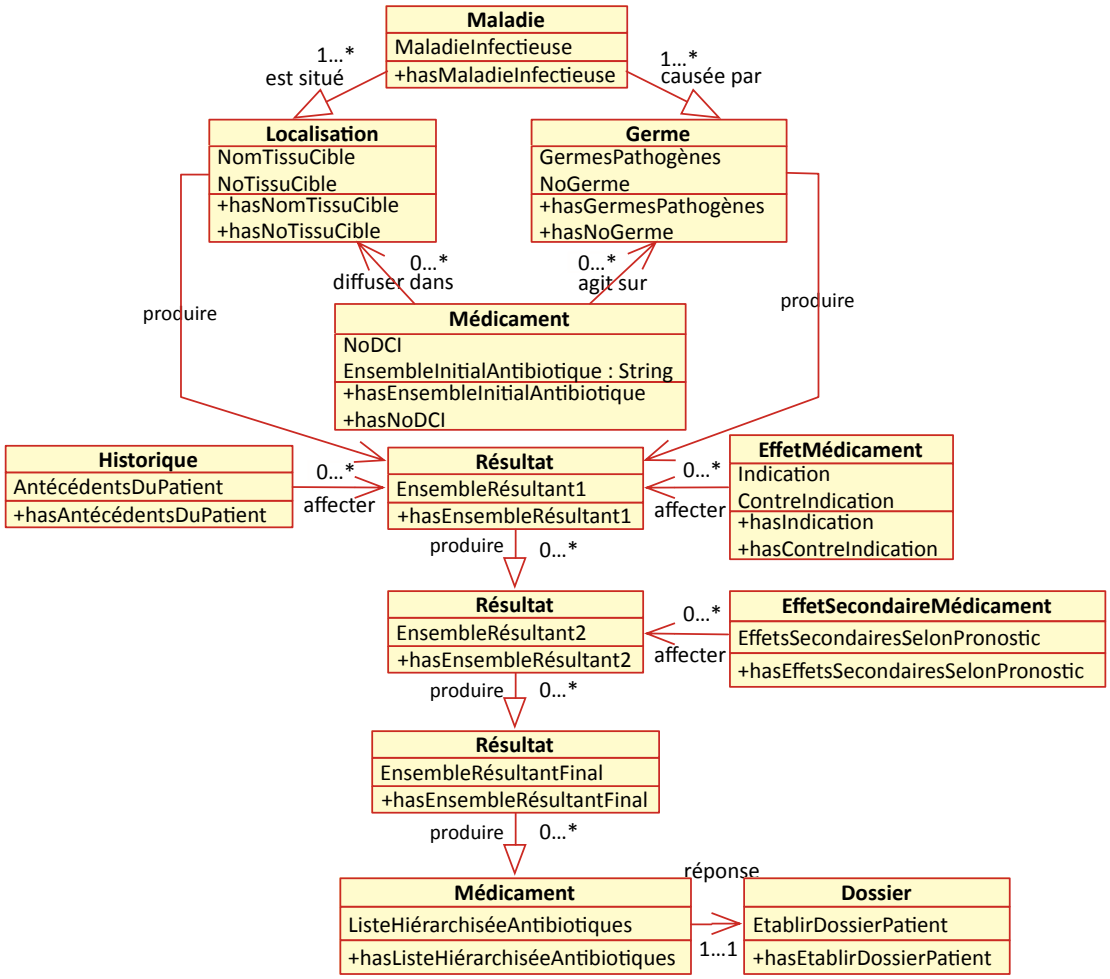


Figure 83 Diagramme de classe : Représentation des objets, des classes d'objets, des associations et des attributs du processus de filtrage de SIAMED

Avec les classes et les relations obtenues en UML, lors de l'analyse de textes dans le système multi-agents, on peut re-modéliser et simuler le processus et le protocole de détermination des antibiotiques indiqués avec les banques de données médicales à la figure 62. L'objectif est de simplifier l'approche et d'améliorer les connaissances recueillies.

Les classes «Maladie», «Localisation», «Antibiotique», «Germe», «Action» et «DossierPatient», définies dans les hiérarchies d'ontologie (*cf.* 5.4.2) alignent respectivement les objets différents du processus de filtrage SIAMED.

L'objet application filtrage fonctionne avec le Tamis.1 («Indications et Contre-indications»), que définit la relation *inverseOf*, et le Tamis.2 («Effets secondaires selon le pronostic de la maladie») qui est réalisé avec les relations « *provoquer* » et « *hasConstraint* ». Toutes les relations utilisées sont détaillées à l'étape de caractérisation des hiérarchies et du codage informatique de l'ontologie.

Ce processus de détermination des antibiotiques peut éviter l'inefficacité de l'antibiothérapie provoquée par :

- Le germe : si le germe en cause n'est pas connu, ou s'il est habituellement résistant à l'antibiotique prescrit.
- La localisation : l'insuffisance de diffusion de l'antibiotique au niveau du site de l'infection.
- Les contre-indications : la méconnaissance de contre-indications peut provoquer des accidents plus ou moins sévères.
- L'effet secondaire : la survenue inopinée d'un effet secondaire toxique doit faire cesser le traitement.
-

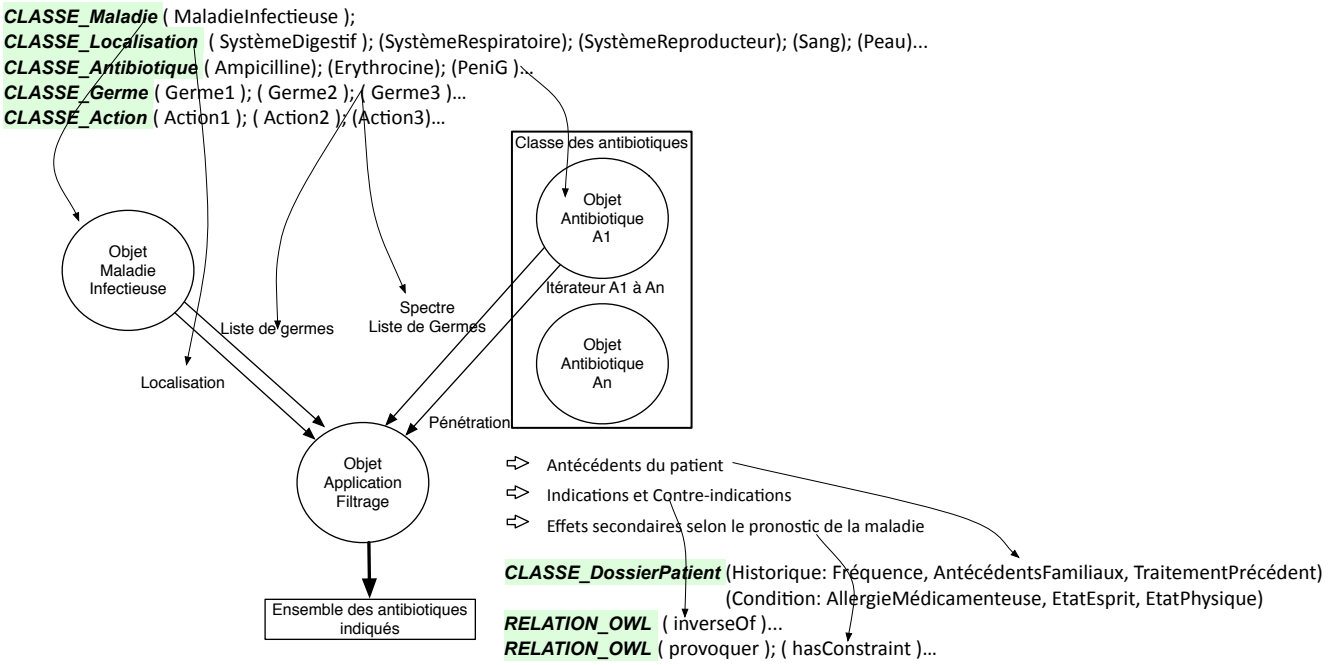


Figure 84 La représentation du processus de détermination des antibiotiques indiqués

Nous avons résumé et représenté les principales propriétés des antibiotiques nécessaires à l'antibiothérapie. Afin de formaliser les règles qui peuvent gouverner le choix dans ce domaine, nous allons réaliser le codage informatique dans la section suivante, pour construire le logiciel SIAMED objet.

6.3.3.6 Codage informatique des règles de bonne pratique

Dans le «Modèle de l'Ontologie», il faut formaliser et représenter les objets, les classes d'objets, les associations et les attributs de l'ontologie en OWL, puis stocker dans le «Dictionnaire Ontologique». L'OWL (*Web Ontology Language*) est un langage de balisage sémantique qui permet de représenter les connaissances d'un domaine spécifique d'une manière bien structurée.

Les antibiotiques sont classés par familles regroupant des molécules dont les propriétés thérapeutiques et biochimiques sont analogues. De prime abord, la modélisation objet d'une taxonomie des antibiotiques (regroupés en familles et sous-familles), sous la forme de classes (types d'objets), et leur spécialisation en sous-classes (ou sous-types d'objets), semblent assez triviale. De même, il existe une classification des germes pathogènes.

Le type d'agent ontologique crée une seule instance d'agent ontologique par étape clinique d'un domaine de connaissance ; il communique la terminologie commune à l'ensemble des agents spécialisés concernés par cette étape, par exemple : le diagnostic des maladies infectieuses, la prescription d'antibiotiques. Les ontologies sont des représentations symboliques, conceptuelles, collectives, sur lesquelles vont s'articuler les interactions entre agents spécialisés, afin qu'ils puissent collaborer à la réalisation de l'objectif commun de chaque étape clinique. Par exemple :

Ontologie des agents infectieux

En cas de maladie infectieuse, il faut se demander quel est la nature de l'agent infectieux en cause : est-ce une bactérie, un parasite, un champignon, un virus ?

- Un virus : les antibiotiques ne sont pas actifs sur les virus de sorte que leur prescription n'est justifiée dans les maladies virales que si l'on suspecte la présence de surinfections bactériennes. Lors d'une infection par le virus de la grippe, la dépression immunitaire favorise la survenue de surinfection bactérienne qu'il faudra traiter.
- Un parasite : il faut prescrire un médicament antiparasitaire adapté, auquel on pourra associer un traitement antibiotique, s'il existe des surinfections bactériennes (un sujet atteint de la gale [*sarcoptes scabiei*] souffre souvent d'une surinfection bactérienne, superficielle, un impétigo dû à un streptocoque). Le

traitement de la gale sera une lotion de benzoate de benzyle et, pour l'impétigo, le médecin prescrira un antibiotique antistaphylococcique et antistreptococcique : une bétalactamine.

- Une bactérie : dans la plupart des cas, il faudra choisir un antibiotique actif sur cette bactérie, afin de pouvoir la détruire. C'est le cas que nous traitons plus particulièrement dans SIAMED.

La figure 86 montre la classification des agents infectieux, en UML. Le type « agent infectieux » est spécialisé en sous-types : Bactéries, Virus, Parasites qui représentent les germes potentiellement capables de provoquer une maladie infectieuse. Le «Code OWL 1» montré ci-dessous vise à représenter l'ontologie des agents infectieux en OWL. La fonction **`subClassOf`** de OWL est utilisée pour indiquer les relations entre superclasses et sous-classes.

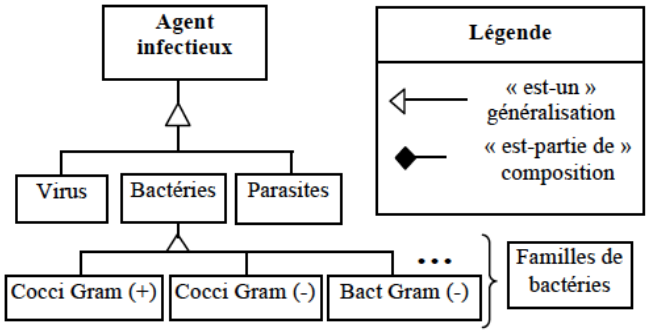


Figure 85 Ontologie des agents infectieux : diagramme de classes en UML

```
   <owl:Class rdf:ID="Virus">
       <rdfs:subClassOf rdf:resource="#AgentInfectieux" />
   </owl:Class> //indiquer que la classe «Virus» est la sous-classe de
_________________//«AgentInfectieux»

   <owl:Class rdf:ID="Parasites">
       <rdfs:subClassOf rdf:resource="#AgentInfectieux" />
   </owl:Class> //Idem. indiquer que la classe «Parasites» est la
_________________// sous-classe de «AgentInfectieux»

   <owl:Class rdf:ID="Bactéries">
       <rdfs:subClassOf rdf:resource="#AgentInfectieux" />
   </owl:Class>

   <owl:Class rdf:ID="CocciGram+">
       <rdfs:subClassOf rdf:resource="#Bactéries" />
   </owl:Class>

   <owl:Class rdf:ID="CocciGram-">
       <rdfs:subClassOf rdf:resource="#Bactéries" />
   </owl:Class>

   <owl:Class rdf:ID="BactGram-">
       <rdfs:subClassOf rdf:resource="#Bactéries" />
   </owl:Class>

   <owl:ObjectProperty rdf:about="#estUn">
```

Code OWL 1 Présentation de l'ontologie des agents infectieux en langage OWL

Ontologies des pathologies

Sur la figure 87, l'ontologie décrit les maladies comme une succession de tableaux cliniques, comportant des signes cliniques pathognomoniques, obligatoires, évocateurs ou accessoires, liés à une pathologie. Le type « *maladies infectieuses* » représente des objets maladies infectieuses provoquées par un agent infectieux (agent pathogène).

Le «Code OWL 2» travaille avec la fonction **`ObjectProperty`**, pour représenter la relation **`estPartieDe`**, pour réaliser la composition des syndromes, à partir de signes cliniques, des tableaux cliniques, à partir de syndromes, et des maladies, comme une succession de tableaux cliniques. L'antonyme de **`estPartieDe`** est la relation estComposéDe. La relation **`estCollection`** permet d'obtenir l'ensemble des composants de l'objet composé, et elle se définit à l'aide de la fonction **`inverseOf`**.

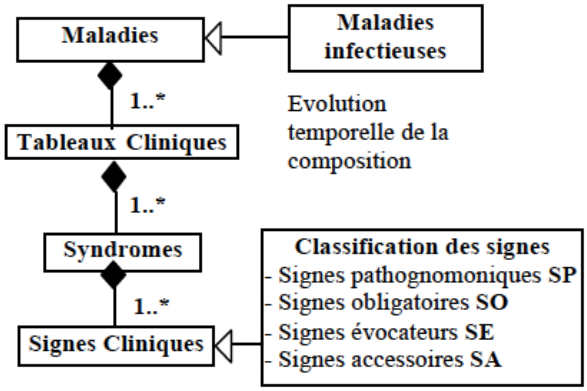


Figure 86 Ontologies des pathologies

```
<owl:Class rdf:ID="Maladies">
    <rdfs:subClassOf rdf:resource="#MaladiesInfectieuses" />
</owl:Class>

<owl:Class rdf:ID="SignesPathognomoniques">
    <rdfs:subClassOf rdf:resource="#SignesCliniques" />
</owl:Class>

<owl:Class rdf:ID="SignesEvocateurs">
    <rdfs:subClassOf rdf:resource="#SignesCliniques" />
</owl:Class>

<owl:Class rdf:ID="SignesPathognomoniques">
    <rdfs:subClassOf rdf:resource="#SignesCliniques" />
</owl:Class>

<owl:Class rdf:ID="SignesAccessoires">
    <rdfs:subClassOf rdf:resource="#SignesCliniques" />
</owl:Class>

<owl:Class rdf:about="#SignesCliniques">
    <rdfs:subClassOf>
        <owl:Class rdf:about="#Syndromes"/>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Syndromes">
```

```
        <rdfs:subClassOf>
            <owl:Class rdf:about="#TableauxClinique"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#TableauxClinique">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Maladies"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:ObjectProperty rdf:about="#estPartieDe">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <owl:inverseOf>
            <owl:ObjectProperty rdf:about="#estCollection"/>
        </owl:inverseOf>
        <rdfs:domain>
            <owl:Class rdf:about="#Maladie"/>
        </rdfs:domain>
    </owl:ObjectProperty>
    // représenter le relation estPartieDe et son antonym estCollection
    // à travers de la fonction ObjectPropert(et inverseOf).
```

Code OWL 2 Présentation de l'ontologie des pathologies en langage OWL

Les figures des ontologies présentées dans cette dernière section, tels que « [les] Ontologies des médicaments », « [les] Ontologies du Dossier Médical », « [les] Ontologies des posologies et des modalités d'administration », « [les] Ontologies de monothérapie antibiotique et associations d'antibiotiques » et « [les] Ontologies concernant la durée d'utilisation d'antibiotique » sont codées sous forme de OWL en l'Annexe 4 « PACKAGE-9 : Code OWL-3, 4, 5, 6, 7 » respectivement.

Organiser les connaissances du cas d'étude

Comme dans la plupart des domaines, les chercheurs en médecine visent à représenter, partager et réutiliser leurs connaissances. La communauté médicale s'est intéressée aux ontologies qui permettent d'exprimer ces connaissances, indépendamment de leur cadre d'utilisation et les rendent disponibles, pour la construction de systèmes d'aide à la décision en médecine.

Pour bien organiser les connaissances extraites, toutes les ontologies nécessaires du projet SIAMED sont représentées dans la figure ci-après : le patient 2 participe à la phase de diagnostic, pronostic, traitement et suivi-thérapeutique de la maladie infectieuse. Il faut tenir compte de la classe « Agent Infectieux » et des sous-classes, Bactéries, Virus et Parasites, capables de causer une maladie infectieuse. Les signes cliniques obligatoires et évocateurs qui composent la classe « Syndromes » déterminent des pronostics différents.

Pendant la phase de pronostic, avec les ontologies qui concernent la durée d'utilisation d'antibiotique, la classe « Temps » est formalisée avec ses sous-classes : «DuréeTraitement », « DuréeRécupération » et « IntervalleDeRécurrence ».

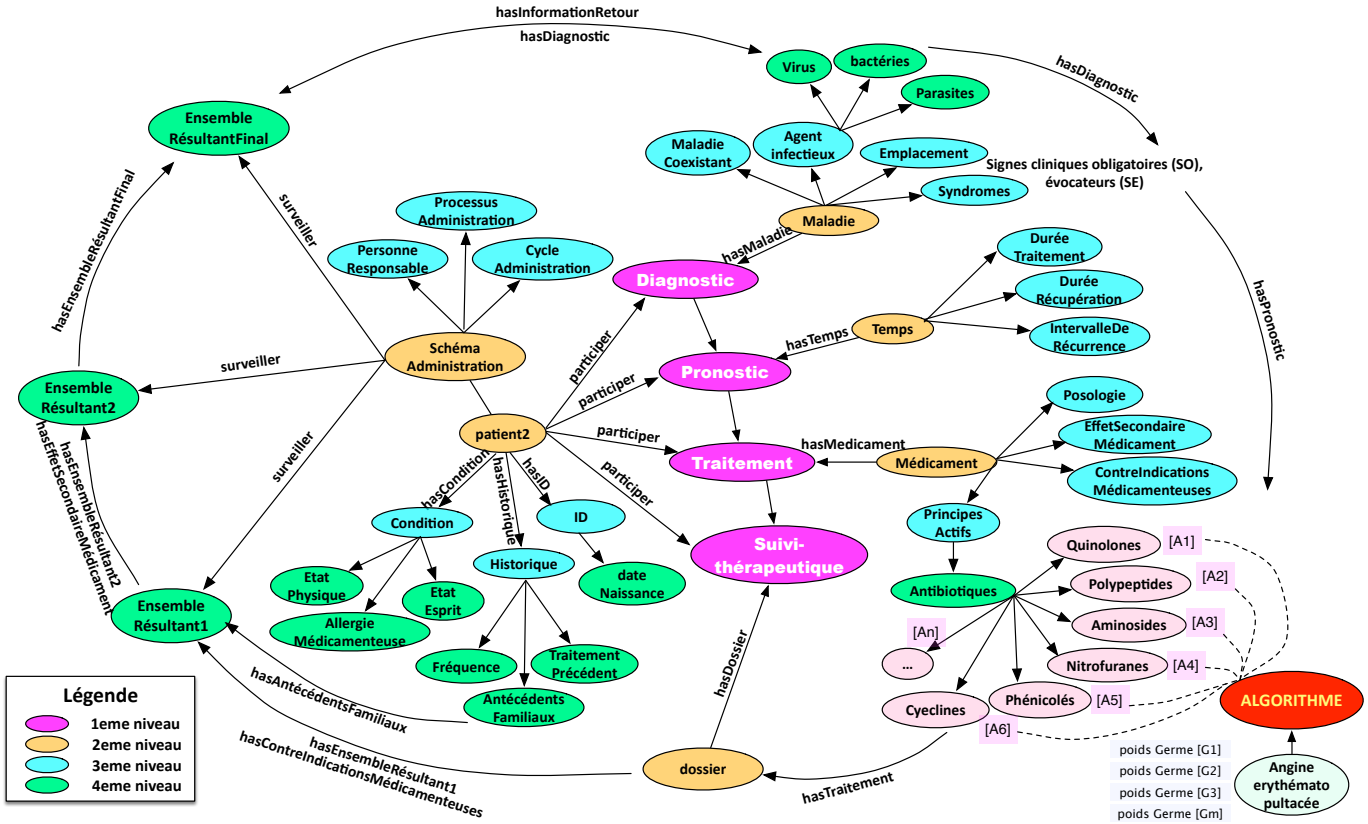


Figure 87 Exemple d'Ontologie de diagnostic, pronostic et traitement du projet SIAMED et Cycle de raisonnement pour calculer les score1 et score2 d'antibiotique

La classe «Traitement» et sa sous-classe «Médicament» sont des éléments nécessaires à l'élaboration de la prescription. La classe «Médicament» comporte quatre sous-classes : «Posologie», «EffetSecondaireMédicament», «ContreIndicationsMédicamenteuses» et «PrincipesActifs».

- La classe «Posologie» contrôle les posologies et les modalités d'administration, grâce aux ontologies définies plus haut.
- Les classes «EffetSecondaireMédicament» et «ContreIndications Médicamenteuses» aident au processus de filtrage de SIAMED et fonctionnent comme des tamis 1 et 2.
- La classe «PrincipesActifs» encapsule tous les types d'antibiotiques correspondants {[A1],...,[Ai],...,[Am]} au traitement de la maladie infectieuse. Un ensemble de poids (prépondérance), associé à chaque germe, susceptible de provoquer une angine erythémato-pultacée {[G1],...,[Gi],...,[Gn]} est

envoyé à des fonctions de calcul des score1 et score2. La fonction «Algorithme» travaille avec le «*Programme-14. Calculer l'action d'un antibiotique et comparer les scores de différents antibiotiques*», qui est détaillé dans l'annexe.4. Ce processus du cycle de raisonnement, qui calcule l'action d'un antibiotique, est implanté dans le «Module Interpréteur».

- Enfin, le mécanisme de raisonnement utilise des fonctions d'évaluation chargées de comparer les scores de différents antibiotiques, triés par ordre décroissant. Après avoir comparé les scores (Score2/Score max), le résultat du calcul est transféré à la base de connaissances, pour mettre à jour le corpus automatiquement.

Notre dictionnaire ontologique (corpus) fournit la liste de médicaments possibles au système multi-agents d'aide à la décision, pour le médecin ou l'étudiant en médecine. Les processus de filtrage de médicaments sont affectés par les méthodes écrites en langage Java :
hasContreIndicationsMédicamenteuses(), *hasEffetSecondaireMédicament()* et *hasAntécédentsFamiliaux()*.

Selon les caractéristiques et/ou les antécédents du patient, l'agent « SchémaAdministration » supervise les résultats du filtrage : «EnsembleRésultant1», «EnsembleRésultant2» et «EnsembleRésultantFinal».

6.3.4 Encapsulation d'ontologie et Synthèse

Dans ce chapitre, nous avons précisé une architecture multi-agents (SMAAD) permettant la prise en compte de l'ensemble d'un cycle de décisions cliniques, adaptables à de nombreux domaines médicaux, tels que le cancer de l'estomac et les maladies infectieuses, pour les exemples développés dans cette thèse. Le raisonnement à partir de cas (RàPC) mémorise et restitue l'expérience de résolution de problèmes similaires.

Nous présentons ci-après un « diagramme de séquence » (UML), pour présenter l'apport de l'ontologie sur les antibiotiques et les maladies infectieuses du projet SIAMED. Ce schéma décrit les étapes successives.

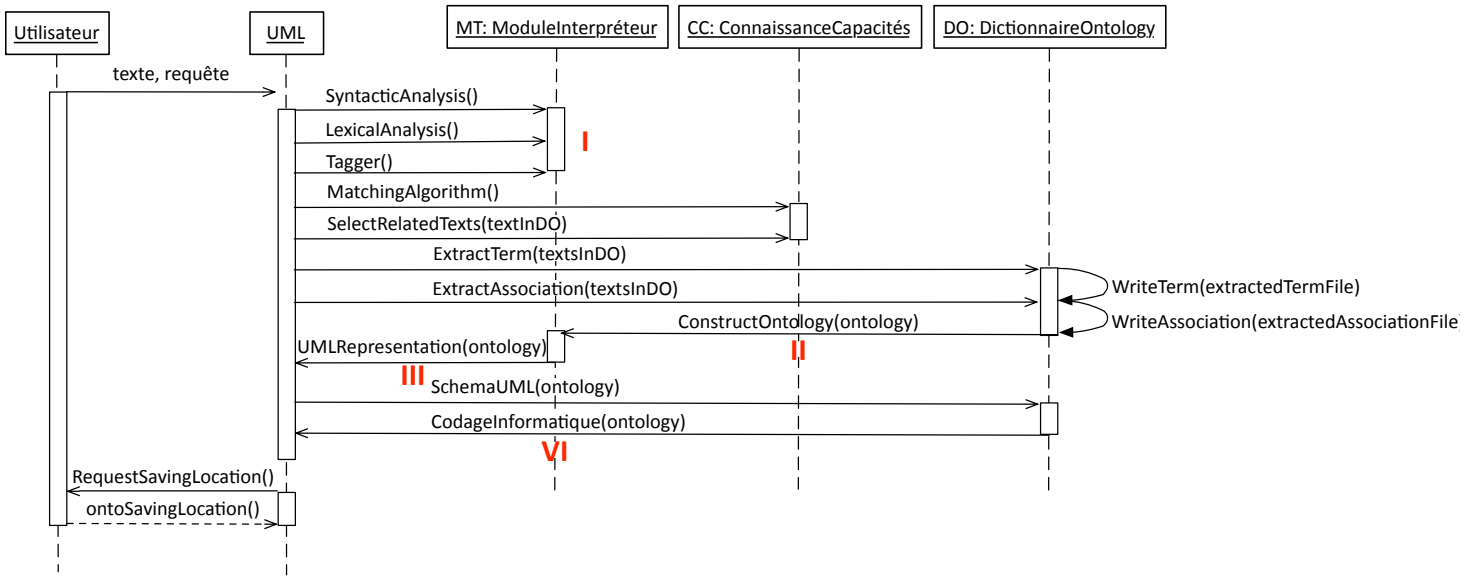


Figure 89 Diagramme de séquence : Apport de l'ontologie sur les antibiotiques et les maladies infectieuses du programme SIAMED

I. Analyse linguistique à partir de textes

Dans la phase d'extraction des connaissances, nous proposons des moyens et des outils pour faciliter l'extraction de connaissances pertinentes du domaine, à partir des différentes sources d'informations.

La mise en place de l'acquisition et la restitution de connaissances dans les systèmes d'aide à la décision clinique, peut être réalisée à travers la création automatique de ressources linguistiques. L'analyse linguistique à partir de textes est réalisée dans le «Dictionnaire Ontologique», à l'aide de programmes et d'algorithmes encapsulés dans « [l']Interpréteur ».

Durant la définition ou la mise à jour du « Dictionnaire Ontologique », les coopérations des différents modules, algorithmes et éléments permettent de tester les formules d'aide à la décision, pour le choix d'antibiotiques. L'extraction des données de la base de connaissances comprend plusieurs étapes :

- Il faut tout d'abord analyser les textes bruts, afin d'identifier la signification des termes importants : noms, adjectifs, verbes qui sont porteurs d'associations efficaces (chapitre 5)

  Pour représenter les propriétés dynamiques et statiques des processus logiques sous-jacents, il faut tenir compte de la sémantique des relations remarquables, afin de construire un nouveau méta-modèle, mieux adapté,

et le connecter à l'acquisition des mots du texte dont on souhaite analyser le sens. Ensuite, il faut effectuer le *mapping* des relations remarquables (est-un, est-partie de, a-un) et complémentaires (temporelles : précède, suit ... ; causales : cause, provoque ...) avec les termes employés (analyse lexicale) et les schémas de phrases (morphosyntaxe).

- Aligner les classes nécessaires (noms d'antibiotiques, actions, traitements, médicaments, etc.) et leurs relations et attributs correspondants (indication et contre-indication, effet secondaire…) à l'objet Maladie, l'objet Application Filtrage et l'objet Antibiotique respectivement.
- Vérifier et étendre les hiérarchies avec les synonymes et les sous-classes d'antibiotiques, ciblées à travers des programmes et des algorithmes stockés dans le «Modèle de Traitement».
- En fin de compte, il faut renvoyer les réponses aux utilisateurs. Dans ce cas, la visualisation de l'ontologie doit être réalisée. Elle est résumée dans la quatrième étape.

II. Construction d'ontologie

L'implémentation et l'expérimentation de l'ontologie, élaborées dans le «Dictionnaire Ontologique», convergent sur la transformation des connaissances extraites à partir de textes, pour la construction d'une ontologie. Pour la modéliser, nous avons analysé les objets (classes) et leurs interactions (relations). Finalement, nous établissons les interfaces correspondant aux besoins en connaissances des utilisateurs.

À propos des ontologies encapsulées, il faut partager les terminologies communes, enrichies de liens sémantiques grâce à des « agents ontologies ».

Grâce aux ontologies, nous pouvons représenter formellement les connaissances extraites et, ainsi, en faciliter le partage.

III. Réaliser la modélisation d'objet avec UML et formaliser le schéma UML

Durant la phase d'exploitation, nous fournirons des moyens pour une utilisation enrichissante et une formalisation des connaissances. Nous réalisons la visualisation d'ontologie en facilitant la tâche d'interprétation des informations et en proposant des visualisations pertinentes et conviviales, à la fois (UML graphes).

Les diagrammes de classes UML sont créés, à l'aide d'un ensemble de méta-modèles modifié de l'ODMG, afin de représenter l'ontologie.

Les diagrammes de classes «Représentation des objets, des classes d'objets, des associations et des attributs du processus de filtrage de SIAMED», «Principes actifs» et «L'interaction entre la maladie angine érythémato-pultacée et antibiotique» expriment les concepts de schémas UML et les relations remarquables qu'il faut associer à la structure sémantique et syntaxique des phrases.

Le diagramme de séquence «Représentation des objets, des classes d'objets, des associations et des attributs du projet SIAMED» déploie la séquence et les opérations pour la représentation de méta-connaissance. La mise à disposition et le partage de savoirs nécessitent d'affiner la méta-connaissance des concepts mis en œuvre, lors de la description des objets.

IV. Coder l'ontologie en OWL

Avec les ontologies construites, il faut les coder en OWL, afin d'organiser les connaissances et établir les interactions entre l'humain et l'ordinateur. Le codage des ontologies a généralement lieu à l'aide d'un langage de programmation logique.

À partir de la base de connaissances de SIAMED, nous avons codé des ontologies, pour formaliser et représenter les objets, les classes d'objets, les associations et les attributs obtenus d'un agent ontologique.

Il existe certaines règles pour les recherches d'antibiotiques, par exemple, un surdosage peut être à l'origine de pathologies iatrogènes ; les résistances aux antibiotiques ne sont pas des connaissances statiques, elles évoluent, etc. Selon ces règles, nous avons réécrit et formalisé les ontologies existantes en OWL dans le «Dictionnaire Ontologique».

En résumé, notre approche repose sur la spécialisation d'agents adaptés aux modèles de connaissances utilisés durant les étapes de la démarche clinique (diagnostic, pronostic, traitement, suivi thérapeutique). La construction d'ontologies repose sur l'organisation de connaissances, à partir de textes et elle se fonde sur des analyses linguistiques et des programmes de représentation et de visualisation de connaissances, *via* le déploiement de classes, d'associations, d'objets, exprimés en UML, à l'aide du « diagramme de classe » et du « diagramme de séquence ». Ensuite, le codage informatique des ontologies est réalisé en OWL, pour formaliser et implanter un agent ontologique, encapsulé dans le système multi-agents.

Chapitre 7 CONCLUSION

Cette thèse fait état de nos recherches et contributions en ingénierie des connaissances. Elles décrivent la conception, l'implémentation et l'utilisation potentielle d'un système de réalisation de génération semi-automatique d'ontologies, à partir de textes médicaux. Ces ontologies constituent le support du dialogue des utilisateurs avec un système à base de connaissances d'aide à la décision clinique.

En effet, les banques de données médicales existantes sont insuffisantes et ne permettent pas d'acquérir des connaissances facilement utilisables dans une démarche clinique. En outre, l'abondance de citations inappropriées constituent du bruit et nécessite un tri fastidieux incompatible avec l'exercice de la médecine.

Pour répondre à cette question, nous avons d'abord proposé une méthode et des outils de modélisation conceptuelle généraux. A partir de l'étude et de l'analyse de textes médicaux, notre méthode permet d'élaborer des agents ontologiques, exploitables dans des systèmes multi-agents d'aide à la décision.

Si l'on se réfère à l'état de l'art, sur cette question, L'utilisation des systèmes multi-agents pour la coopération de bases de connaissances n'est pas récente et elle a donné lieu à de nombreux travaux (Boulanger, D. & Colloc, J. 1992), (Boulanger, D. & Colloc, J. 1993) (Jennings, N. R. 1993), (Müller, J. P. 1996), (Tesauro, G.J. & Kephart, J.O. 2000). La notion d'ontologie a été proposée dans (Gruber, 1993). On trouve l'usage d'ontologies dans les systèmes multi-agents par exemple dans *la notion de dictionnaire ontologique de* (Falasconi, S. et al. 1996).

Le SMAAD (Système Multi-Agent d'Aide à la Décision) que nous avons proposé dans cette thèse est destiné à assurer la coopération de bases de connaissances médicales hétérogènes. Il s'appuie sur un méta-modèle de démarche clinique et un ensemble de travaux menés par le Pr. Colloc et ses collaborateurs (Sybord C. & Colloc J. 1997 ; Colloc, J. & Sybord, C. 2003 ; Dussart, C. et al. 2008). Il offre une approche interactive avec l'utilisateur *via* un agent superviseur. Le SMAAD assure l'interopérabilité des modèles de données et de connaissances encapsulés dans les agents. L'interopérabilité sémantique des connaissances est assurée par les « agents ontologies », couplés à des « agents connaissances ». En comparaison avec notre approche, SAPHIRE (Laleci, G.B. et al. 2008) est un système plus macroscopique, fondé sur des directives cliniques et des agents ontologiques. Il permet l'interopérabilité d'appareils, de bases de données et de connaissances. Le système

n'est pas guidé par un modèle de démarche clinique. Le SMAAD ne gère pas l'accès par plusieurs utilisateurs dans le cadre d'un workflow clinique comme dans les modèles que proposent (Bouzguenda, L. & Turki, M. 2012) ; (Marling, C. et al. 2009).

Certaines méthodes sont performantes, pour l'acquisition de connaissances : KOD (Vogel, C. 1989), KADS (Wielinga, B. J. et al. 1992), MACAO (Aussenac-Gilles N. et al. 1994), MKSM (Ermine J.L. et al. 1996), et MASK (Ermine J.L. et al. 1996). L'étude de ces méthodes s'est révélée utile pour fonder les bases théoriques de cette thèse. L'approche KADS nous a paru être la méthode la plus complète, pour la définition des bases de connaissances. Elle comporte de nombreux domaines d'application : étapes du processus de commercialisation, le biomédical etc. Le champ de ces recherches semble plus général, alors que nos travaux mettent l'accent sur les différentes étapes cliniques, spécifiques du diagnostic, pronostic, traitement et suivi thérapeutique, en médicine. KADS est une démarche descendante (*Top down*), tandis que MACAO est une démarche ascendante (*Botton up*). Dans nos recherches, les squelettes taxonomiques (*top-down* ou *bottom-up*) sont utiles, ensembles, pour enrichir la structure du lexique en classant les concepts, les propriétés et les attributs de mots cibles ; ils permettent aussi d'améliorer le lexique en intégrant des aspects sémantiques pertinents.

À partir des hypothèses émises, les apports de cette thèse tendent à répondre aux questions posées et traitées dans la première partie.

Chaque agent spécialisé exploite une base de connaissances qui définit les conduites à tenir, correspondant à l'état de l'art associé à une base ontologique qui exprime les relations sémantiques entre les termes du domaine considéré. Notre travail consistait à élaborer une méthode d'analyse de textes médicaux qui permette de construire des ontologies, encapsulées dans des agents ontologiques, exploités par des agents spécialisés dans les différentes étapes du cycle de prise de décision clinique. Les textes médicaux sont ciblés pour la présentation d'une maladie, incluant les paramètres du diagnostic, du pronostic, du traitement et du suivi-thérapeutique.

Si l'on se réfère au plan pratique, dans le système multi-agents étudié et développé, l'analyse linguistique et les programmes informatiques sont développés pour l'extraction de termes et de relations, à partir de textes, afin de réaliser l'organisation des connaissances et la construction d'ontologies. La coopération des différents

modèles permet de formaliser et d'implanter un agent ontologique, *via* des codages informatiques en OWL. Les algorithmes de notre système convertissent des classes, des associations et des attributs d'UML en OWL, pour fournir des réponses visuelles aux utilisateurs.

Dans la construction d'un agent ontologique, l'analyse linguistique est indispensable à la construction du sens, des moyens de représentation des connaissances et à l'approche générale de l'acquisition de connaissances textuelles. Pour être plus efficace dans la construction des modèles, les textes doivent être des sources de connaissances riches et utiles, surtout à la modélisation de certains domaines. Le traitement automatique des langues, fondé sur des principes linguistiques et statistiques, offre des outils performants, moyennant la définition d'interfaces de navigation et d'exploitation des résultats. L'approche à partir de textes est particulièrement adaptée à la modélisation de la terminologie et des ontologies elles-mêmes. Elle permet de leur associer une composante terminologique utile pour ensuite revenir vers les textes et les indexer.

Le module linguistique est conçu pour faciliter l'accès à la base de données médicales préexistantes : MESH, MEDLINE, SNOMED, CASNET-Réseau sémantique, etc., où des documents médicaux sont indexés (banques de données bibliographiques).

Dans la plate-forme d'exploitation, la base de connaissances constitue le réseau lexical, syntaxe et sémantique du domaine concerné. Un analyseur morphologique reconnaît les variantes d'un mot donné. Il simplifie les mots français pour générer des systèmes de lemmatisation : il s'agit du prétraitement automatique (TAL) nécessaire à l'extraction de termes et d'associations entre la requête et le dictionnaire existant. Les phénomènes de synonymie, la généralité et la spécificité sont pris en compte.

La construction d'ontologies pour le SMAAD et leur visualisation en UML sont fondées sur des bases théoriques linguistiques qui nous aident à résoudre les problèmes essentiellement sémantiques et de relations syntaxiques, face à des faits de langue complexes en langue de spécialité, médicale, ici.

Avec les termes, les relations et les attributs extraits *via* l'analyse linguistique, la construction d'ontologies devient opératoire.

Dans la phase de mise en place des ontologies, nous nous centrons sur l'introduction d'un processus de représentation de données linguistiques, à partir de savoirs extraits de textes. Les éléments importants des textes sont filtrés pour en

extraire les substantifs (objets), les adjectifs (attributs ou propriétés des objets) et les verbes et adverbes (actions ou associations).

Cette approche prend en compte des savoirs extraits de corpus et d'associations remarquables, pour construire automatiquement des structures décrivant et classant ces savoirs. Pour cela, nous avons utilisé des langages de programmation (e.g. le langage Java, les programmes en langage OWL et UML, des Logiques de Description, le Programme à Prototypes etc.), afin d'extraire des termes et des associations à partir d'un corpus de textes.

UML et sa syntaxe OWL sont utilisés pour la matérialisation de la mémoire, tandis que le système multi-agents et les techniques d'apprentissages automatiques, mentionnés au chapitre 1, sont adoptés pour l'exploitation de cette mémoire.

Dans l'étape d'application des ontologies pour aider à la décision clinique, le langage UML et OWL sont utilisés pour modéliser la représentation et réaliser la visualisation d'ontologie.

Les cas typiques, présentés dans la thèse, tels que le cancer de l'estomac et la maladie infectieuse sont expliqués pour illustrer les problèmes d'acquisition, de représentation et d'application des connaissances des experts médicaux. L'ontologie permet d'exploiter la base de connaissances en médecine.

Compte tenu des limites du système multi-agents et à l'aide de méta-modèles et de schémas UML, nous pouvons organiser des connaissances médicales à partir de textes, à l'aide d'analyses linguistiques et de programmes. Nous pouvons également adapter des agents aux modèles de connaissances utilisés lors des étapes de la démarche clinique. Ces deux étapes nous permettent d'obtenir des connaissances utilisables et de réduire effectivement les bruits de citations inappropriées.

Notre principale contribution est d'avoir formalisé un méta-modèle étendu des concepts orientés objet à partir de celui l'ODMG. Il permet d'exprimer la sémantique des textes médicaux par des ontologies représentées en UML sous la forme de diagrammes de classe et de diagrammes de séquence puis de les implanter au sein d'un agent ontologique. Le codage informatique des ontologies exprimées en UML en OWL détermine l'implantation d'un agent ontologique spécifique d'un domaine de connaissances. Ces connaissances sont disponibles sous la forme de diagrammes UML et OWL.

L'apport principal de cette thèse est de proposer une méthode générale, réutilisable pour construire des agents ontologiques à partir de textes qui est applicable à de

nombreux autres domaines que la médecine et en particulier à tous les domaines qui utilisent une démarche clinique fondée sur des connaissances empiriques et une approche systémique (la gestion, l'agriculture, la botanique, l'économie, la psychologie, la sociologie, la géographie...) et globalement toutes les sciences humaines.

Les résultats obtenus dans cette thèse mettent en évidence la nécessité d'une méta-connaissance pour la description des connaissances nécessaires à un système d'aide à la décision. Les problèmes posés mettent en avant un point fondamental, pour la résolution du traitement automatique de faits de langue (également pour la construction du sens et la classification des connaissances linguistiques). Le traitement des langues naturelles (TAL) nécessite de prendre en compte de nombreux items ou termes, pour construire de façon dynamique de nouvelles connaissances en évolution.

Aujourd'hui, la construction d'ontologies à partir de textes et l'utilisation du traitement automatique des langues et de l'apprentissage apportent la possibilité de réduire le coût et d'anticiper les problèmes de maintenance. Mais les représentations formelles très sophistiquées des bribes de connaissances présentent des limites, quand elles sont utilisées dans des contextes évolutifs, sans cesse renouvelés et qui affectent l'exactitude de l'extraction de l'information. La modélisation des connaissances de contexte demeure un verrou important de l'ingénierie des connaissances.

Les problèmes liés à la récursivité et à la complexité des connaissances conduisent à accepter une incomplétude des modèles proposés et notamment l'impossibilité encore d'accéder à une réalisation complétement automatisée d'ontologies sauf dans des domaines très simples, s'il en est.

Ces problèmes peuvent être atténués à travers de :

1. Augmenter la diversité (annotation, textes...) et la quantité (concepts, relations, attribut...) de la base de données existant, pour réduire l'impact des données dynamiques pendant l'extraction de l'information.
2. Définir toutes les variables et interfaces nécessaires, afin de simplifier les structures de phrases et de classer les concepts et les relations constituant la connaissance en différentes familles telles que les entités (virus, outil, maladie, médicament, etc.), les personnes, les organisations, les emplacements et les étapes de la démarche clinique tels que : diagnostic, pronostic, traitement, suivi-thérapeutique etc.

3. Superviser le recueil de connaissances et le dialogue avec le système à l'aide d'un agent superviseur. Dans des cas complexes où des éléments de connaissance peuvent être récursifs, les ingénieurs des connaissances doivent ajuster la granularité du système afin d'établir des conditions d'arrêt des traitements permettant d'obtenir une décision dans un temps acceptable compte-tenu des conditions d'utilisation.

Grâce à la méthode proposée, il nous semble intéressant d'élargir la recherche, d'abord dans le domaine de la santé et du médicosocial, si l'on se réfère aux parcours complexes du polyhandicap et des maladies chroniques, par exemple, qui supposent un suivi, hors des secteurs courants du soin : à domicile, en centres spécialisés.

Nos travaux font partie du substrat de construction des grosses bases de données (*Big Data*), médicales avec les modélisations en télémédecine, données qui peuvent être liées grâce à des systèmes multi-agents. Ce type de modélisation est applicable aux systèmes d'information de dossiers de santé (*Electronic health records* - EHRs) ou dossier médical informatisé (DMI), pour implémenter des évaluations fiables et de qualité (Electronic quality measures - eQM).

En ce qui concerne le dossier médical informatisé (DMI), nous allons nous centrer sur la création et l'application des outils et des ressources informatiques, pour exploiter des cas en biologie médicale, les maladies génétiques, afin de mettre en place des dossiers de santé numériques, à l'aide de programmes permettant leur analyse linguistique fine.

Dans les systèmes multi-agents, pour la génération semi-automatique d'ontologies médicales, à partir de textes, nous avons utilisé la notion de dossier médical informatisé souvent implantés à l'aide de bases de données (les connaissances dans le dictionnaire ontologique tels que les symptômes du patient, les effets secondaires médicamenteux etc. sont accessibles *via* des interfaces définies ou des procédures de requêtes chargées dans des objets). On peut construire les ontologies de dossiers de santé avec le *Web Ontology Language* (OWL).

Dans l'avenir, nous souhaitons explorer les *Big Data* dans le domaine de la biologie et médicale, pour la génération automatique des dossiers de santé électroniques (les images, les dossiers de la pharmacie, les notes cliniques, l'information du patient auto-déclarée), en vue de combiner les informations : données de santé, données financières et données de recherche, afin de fournir un aperçu de l'extraction de données à partir de plusieurs sources.

Le *Big Data* est utilisé dans nombreux domaines, tels que les médias et les réseaux sociaux (générer des données : Facebook, Twitter, YouTube, Flickr), les instruments scientifiques (collecteurs de toutes sortes de données), les appareils mobiles (qui suivent les objets en continu), les technologies et les réseaux de capteurs (qui mesurent une grande variété de données), le système multi-agents d'aide a la décision clinique (*cf.* quel type de traitement est adapté pour le cancer de l'estomac au stade 2b), etc. Les fréquences des mots et expressions sont utiles pour les moteurs de recherche dans le domaine des sciences médicales, afin de rechercher les plus récents travaux sur les maladies orphelines, chez certains patients.

Les dangers éthiques de l'utilisation du Big Data dans le domaine de la santé ne doivent pas être méconnus. En effet les données médicales des patients sont confidentielles et protégées par le secret professionnel en France. L'utilisation du Big Data est susceptible de remettre en cause les repères juridiques auxquels les professionnels de santé en France sont très attachés.

Dans ce cas, nous devons travailler dans le but de combiner des sources de données hétérogènes et d'en extraire des connaissances ; nous avons besoin de connecter différents types de données, tels que le texte, les images numériques, etc. Nous savons nous centrer sur l'analyse des relations entre de nombreuses sources de données hétérogènes, y compris les données structurées des systèmes existants, combinées avec les données non structurées ou des métadonnées. En outre, il est nécessaire d'explorer les associations entre les groupes (individus, organisations, sites web, etc.) afin de bénéficier de la recherche médicale en épidémiologie et en pharmacologie, parmi autres utilisations.

Par ailleurs, les mesures de qualité (eQM) sont des outils qui nous aident à mesurer ou à quantifier les processus de soins en santé, les résultats, les perceptions des patients et la structure organisationnelle et/ou des systèmes qui sont associés à la capacité de fournir des soins de qualité et/ou qui se rapportent à un ou plusieurs objectifs dans ce même domaine. Ces objectifs sont les suivants : efficacité, sauf, centré sur le patient, équité et soins en temps opportun.

Pendant le fonctionnement du système multi-agents, nous devons évaluer l'échange d'informations pour améliorer la qualité et la sécurité clinique. Actuellement, c'est une étape qui reste manuelle. Cela signifie que les médecins vérifient le diagnostic, le pronostic et le traitement par leurs connaissances cliniques. Une meilleure prise en compte du patient et du soignant, de la relation (personnel de santé – patient)

constitue une approche que nous souhaitons approfondir, dans le futur.

Enfin, ce projet doctoral est intégré au projet PICMAC, issu du consortium d'ASMA (Approche systémique de la modélisation des addictions). PICMAC comporte 29 participants de 11 institutions partenaires. Il a pour objectif de créer une plate-forme coopérative de lutte contre les addictions et cette thèse y a été intégrée.

PICMAC, projet interdisciplinaire mobilise des chercheurs en informatique, en neuropsychologie, en médecine et il requiert le développement d'ontologies, pour l'organisation et la coopération de connaissances au sein de la plate-forme.

Dans le cadre d'ASMA, l'objectif de recherche d'ontologies médicales (axe ONTO) est de réaliser la génération semi-automatique d'ontologies à partir de textes médicaux, afin d'aider à la décision clinique (axe CDMAS).

Comme nous l'avons indiqué plus haut, il est intéressant d'explorer les ontologies à partir d'autres domaines de spécialité : la gestion, l'agriculture, l'écologie et tous les domaines où une démarche clinique est utilisée et où il est nécessaire de disposer d'une ontologie décrivant le domaine de connaissances.

Dans le champ de l'agriculture, certaines recherches réalisées sont utiles telles que (Martin-Clouaire, R. & Rellier, J. P. 2009), (Chen, G. et al. 2003) et (Fileto, R. et al. 2003). L'article (Martin-Clouaire, R. & Rellier, J. P. 2009) présente un cadre de simulation fondé sur les ontologies des systèmes de production. Les ontologies présentées concernent les activités de production, les plans flexibles et les ressources matérielles. Les algorithmes d'interprétation opèrent sur des instances de l'ontologie dans un modèle de système de production construit en conformité avec l'ontologie.

Dans Chen, G. et al. (2003), une méthodologie est présentée pour construire des ontologies virtuelles à travers de la réutilisation des connaissances provenant d'autres domaines. Cette méthode élabore les ontologies, en combinant l'alignement sémantique, avec la pertinence du réseau sémantique des modèles de domaine existants et les ontologies préexistantes.

Les directions de recherche à approfondir pour l'ingénierie des connaissances se construisent, au fil des avancées technologiques. Elles permettent d'élaborer des applications qui facilitent les adaptations à de nouveaux contextes, de nouveaux domaines de connaissances scientifiques.

BIBLIOGRAPHIE

Références générales

[Garnier, M. Delamarre, V. et al 2012]

Garnier, M. Delamarre, V. et al (2012). *Dictionnaire illustré des termes de médecine*. (31e éd. rev. et aug.) Paris : Maloine.

[Littré, É. 1956-1957 & 1957-1958]

Littré, É. (1956-1957 & 1957-1958). *Dictionnaire de la langue française.* [Paris] : J.J. Paubert éd. (Vol. 1-4, 1956-1957)/[Paris] : Gallimard Hachette (Vol. 5-7, 1957-1958), 1541-1854-2095-2121-2059-2078-1976 p.

Articles et ouvrages

[Aamodt A. & Plaza E. 1994]

Aamodt A. & Plaza E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, *IOS Press*. *7*(1): 39-59.

[Adam, J.M. 2011]

Adam, J.M. (2011). *Les textes: types et prototypes: récit, description, argumentation, explication et dialogue*. 3e éd., rev. et aug. Paris : A. Colin.

[Adler, S. & al. 2001]

Adler, S., Milowski, A., Richman, J. & Zilles, S. (2001). *Extensible Stylesheet Language* (XSL)-Version 1.0.

http://www.w3.org/TR/2001/REC-xsl-20011015/.

[Agent UML 2001]

Agent UML (2001) [online]. [Access: 23 juillet 2014]. Available at: <http://www.auml.org/>.

[Antibioguide du CHU 2015]

Antibioguide 2014, CHU de Clermont-Ferrand [Accès : 1er février 2015]. Aviaible at:

http://cclin-sudest.chu-lyon.fr/Doc_Reco/guides/Antibioguide_2011.pdf/

[Arlow, J., & Neustadt, I. 2005]

Arlow, J. & Neustadt, I. (2005). *UML 2 and the unified process: practical object-oriented analysis and design*. Boston: Addison Wesley.

[Aronson, A.R. 2001]

Aronson, A.R. (2001). Effective mapping of biomedical text to the UMLS Metathesaurus. The MetaMap Program. In *Proceedings of the annual AMIA Symposium*, 2001, Nov. 3-7 (pp. 17-21), Washington DC: Hanley & Belfus, Americal Medical Information Association.

[Aussenac-Gilles, N. & Matta, N. 1994]

Aussenac-Gilles, N. & Matta, N. (1994). Making a method of problem solving explicit with MACAO. *International journal of human-computer studies*, *40*(2): 193-219.

[Baader, F. 2003]

Baader, F. (2003). *The description logic handbook: theory, implementation, and applications*. Cambridge: CUP.

[Baader, F. & al. 2005]

Baader, F., Horrocks, I. & Sattler, U. (2005). Description logics as ontology languages for the semantic web, in *Mechanizing Mathematical Reasoning* (pp. 228-248). Berlin, Heidelberg: Springer.

[Bachimont, B. 2000]

Bachimont, B. (2000). Engagement sémantique et engagement ontologique: conception et réalisation d'ontologies en ingénierie des connaissances. *Ingénierie des connaissances: évolutions récentes et nouveaux défis* (pp. 305-323). Paris : Eyrolles.

[Bales, R.F. & al. 1951]

Bales, R.F., Strodtbeck, F.L., Mills, T.M. & Roseborough, M.E. (1951). Channels of communication in Small groups. *American Sociological Review*, *16*(4): 461-468.

[Bartels, D. & al. 1997]

Bartels, D., Berler, M., Eastman, J. & al. (1997). *The object database standard: ODMG* / Dirk Bartels, Mark Berler, Jeff Eastman... [et al.], in R.G.G. Cattell, Douglas K. Barry, Ed. San Mateo: Morgan Kaufmann Publishers.

[Benjamins, V.R. & Fensel D.P.A. 1998]

Benjamins, V.R. & Fensel, D.P.A. (1998). Knowledge management through ontologies. Basel, Switzerland, *in* Ulrich Reimer (Ed), *Proceedings of the 2nd Conf. On Practical Aspects of Knowledge Management*. 1998, October 29-30, Zurich (Switzerland): Swiss Life, *13*(5): 1-12.

[Benoit-Smullyan, E. 1944]

Benoit-Smullyan, E. (1944). Status, status types, and status interrelations, *American Sociological Review*, *9*(2): 151-161.

[Berardi, D. & al. 2001]

Berardi, D., Calvanese, D. & De Giacomo, G. (2001). Reasoning on UML class diagrams using description logic based systems, In *Proc. of the KI'2001 Workshop on Applications of Description Logics*, 2001, Sept. 18, Vienna (Austria): *44*(1).

[Berardi, D. & al. 2005]

Berardi, D., Calvanese, D. & De Giacomo, G. (2005). Reasoning on UML class diagrams. *Artificial Intelligence*, *168*(1): 70-118.

[Bernaras, A. & al. 1996]

Bernaras, A., Laresgoiti, I. & Corera, J. (1996). Building and Reusing Ontologies for Electrical Network Applications', in *ECAI 1996*. 12th European Conference on Artificial Intelligence, August 11-16, 1996, Budapest, Hungary: *Proceedings*, in W. Wahlster (Ed.), organized by the European Coordinating Committee for Artificial Intelligence (ECCAI) in cooperation with AAAI, IJCAI, and PRICAI (pp. 298-302). Chichester (R.U.): Wiley.

[Berners-Lee, T. 1996]

Berners-Lee, T. (1996). WWW: Past, Present and Future. *Computer*, *29*(10): 69-77.

[Bernstein, P.A. & Goodman, N. 1981]

Bernstein, P.A. & Goodman, N. (1981). Concurrency control in distributed database systems. *ACM Computing Surveys (CSUR)*, *13*(2): 185-221.

[Biebow, B. & al. 1999]

Biebow, B., Szulman, S. & Clément, A.J. (1999). TERMINAE: A linguistics-based tool for the building of a domain ontology. In *Knowledge Acquisition, Modeling and Management*. Berlin, Heidelberg: Springer, pp. 49-66.

[Blomqvist, E. & al. 2006]

Blomqvist, E., Öhgren, A. & Sandkuhl, K. (2006) Ontology Construction in an Enterprise Context: Comparing and Evaluating two Approaches. *In Proceedings of 8th International Conference on Enterprise Information Systems*, Mai 2006 (pp. 86-93). Paphos (Cyprus).

[Bodenreider, O. 2004]

Bodenreider, O. (2004). The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research*, *32*(1): 267-270.

[Boguraev, B. & Pustejovsky, J. 1996]

Boguraev, B. & Pustejovsky, J. (1996). *Corpus processing for lexical acquisition*. Cambridge (Mass): MIT Press.

[Booch, G. & al. 1998]

Booch, G., Rumbaugh, J. & Jacobson, I. (1998). *The Unified Modeling Language*. Retrieved from http://www.uml.org

[Bouchon-Meunier, B. et al. 1990]

Bouchon-meunier, B., Despres, S., Dubois, D., Gascuel, O., Guenoche, A. & Prade, H. (1990). Aspects de l'interface entre symbolique et numérique. *In actes des 3èmes journées nationales PRC-GDR Intelligence Artificielle : autour de la monotonie*, 21-22 octobre 1990 (pp. 90-112). Paris : Hermès.

[Boulanger, D. & Colloc, J. 1992]

Boulanger, D. & Colloc, J. (1992). Detecting heterogeneity in a multidatabase Environment through an O-O Model, in *Proceedings of IFIP DS-5: Conference Semantics of Interoperable Database Systems*, Victoria (Australia), 16-20 Nov. 1992.

[Boulanger, D. & Colloc, J. 1993]

Boulanger, D. & Colloc, J. (1993). Cooperation between heterogeneous databases using an O.O.Model. *In proceedings Object Technology 93*, 1993, 30 March-1 April. Cambridge (England).

[Bourigault D. et Lame G. 2002]

Bourigault D. et Lame G. (2002). Analyse distributionnelle et structuration de terminologie. Application à la construction d'une ontologie documentaire du droit. *Traitement automatique des langues*, *43*(1) : 129-150.

[Bourigault, D. et al. 2004]

Bourigault, D., Aussenac-Gilles, N., & Charlet, J. (2004). Construction de ressources terminologiques ou ontologiques à partir de textes : Un cadre unificateur pour trois études de cas. *Revue d'Intelligence Artificielle*, *18*(1) : 87-110.

[Bouzguenda, L. & Turki, M. 2012]

Bouzguenda, L. & Turki, M. (2012). Coupling Clinical Decision Support System with Computerized Prescriber Order Entry and their Dynamic Plugging in the Medical Workflow System, *Int. Conf. on Information Technology and e-services*, ICITeS'12, 2012 March 24-26, Sousse (Tunisia). New York (USA): IEEE, pp. 1-6.

[Bray, T. & al. 1997]

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. & Yergeau, F. (1997). Extensible Markup Language (XML). *World Wide Web Journal*, *2*(4): 27-66. http://www.w3.org/XML/
https://owl.english.purdue.edu/owl/resource/560/10/

[Briot, J.P. 1989]

Briot, J.P. (1989). Actalk: A Testbed for Classifying and Designing Actor Languages in the Smalltalk-80 Environment. *In ECOOP89: Proceedings of the 1989 European Conference Object-Oriented languages and systems.* August 3-6, 1969 (pp. 109-130). Cambridge (UK): Cambridge University Press.

[Bylander Chandrasekaran, B. 1988]

Bylander Chandrasekaran, B. (1988). Generic Tasks in knowledge-based reasoning.: the right level of abstraction for knowledge acquisition. Ed. B.R. Gaines & J.H. Boose, *Knowledge Acquisition for Knowledge Based systems* (pp. 65-77). London: Academic Press.

[Capretz L.F. & Lee P.A. 1992]

Capretz L.F. & Lee P.A. (1992). Reusability and life cycle issues within an object oriented methodology, in *Proceedings of* TOOLS USA'92. *The 8th International Conference on Technology of Object oriented languages and systems*, August, 3-6, 1992. Santa Barbara, California (pp. 139-150). New Jersey (USA): Prentice Hall, Inc.

[Carroll, L. (Pseud) & Bartley, W.W. Ed. 1896/1977].

Carroll, L. (Pseud) & Bartley, W.W. Ed. (1896/1977). *Lewis Caroll's Symbolic Logic, I. together with letters from Lewis Carroll to eminent nine-teenth-century logicians*

and to his "logical sister", and eight versions of the Barber-shop paradox. Ed. with annotations and an introd. by William Warren Bartley, III : Part I Elementary (5th ed., 1896). London: Macmillan and Co.

[Chaabani, M. et al. 2010]

Chaabani, M., Mezghiche, M., Strecker, M. & Boumerdès, A. (2010). Vérification d'une méthode de preuve pour la logique de description ALC. *Proc. 10èmes Journées Approches Formelles dans l'Assistance au Développement de Logiciels*. Y. Ait-Ameur (Ed.), 2010, June 9-11, Poitiers, France (pp. 149-163). […] Poitiers] : LIAS.

[Chandrasekaran, B. & al. 1999]

Chandrasekaran, B., Josephson, J.R. & Benjamins, V.R. (1999). What are ontologies, and why do we need them? *Intelligent Systems and Their Applications*, *IEEE, 14*(1): 20-26.

[Charlet, J. 2003]

Charlet, J., Bachimont, B. & Troncy, R. (2003). Ontologies pour le Web sémantique. *Action spécifique*, *32 :* 43-63.
http://www.eurecom.fr/~troncy/Publications/Troncy-revue_i304.pdf

[Chen, G. & al. 2003]

Chen, G. Lu, R. & Jin, Z. (2003). Constructing virtual domain ontologies based on domain knowledge reuse [J]. *Journal of Software*, *3*: 71-77.

[Chen, P.P.S. 1976]

Chen, P.P.S. (1976). The entity-relationship model-toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, *1*(1): 9-36.

[Chen, P.P.S. 1981]

Chen, P.P.S. (1981). A Preliminary framework for entity-relationship Models, in *Proceedings of ER 81: the 2nd International Conference on the Entity-Relationship Approach to Information Modeling and Analysis.* Washington, DC, USA, October 12-14, 1981 (pp. 19-28). Netherland: North-Holland Publishing Co.

[Colloc, J. & Boulanger, D. 1987]

Colloc, J. & Boulanger, D. (1987). *Conception d'un système expert orienté objet destiné à l'aide au diagnostic médical*. Toulouse : SITEF.

[Colloc, J. & Boulanger, D. 1987]

Colloc, J. & Boulanger, D. (1987). *Problème du portrait-robot et de l'excès de déduction*. Toulouse : SITEF.

[Colloc J. & Boulanger D. 1993]

Colloc J. & Boulanger D. (1993). Automatic knowledge acquisition for object oriented expert systems. *In proceedings AVIGNON '9, 13th International Conference Artificial Intelligence, Expert Systems, Natural Language*, *May 24-28 1993.* Avignon, France (pp. 99-108). Ed. by EC2.

[Colloc, J. & Sybord, C. 1997]

Colloc, J. & Sybord, C. (1997). Représentation et apprentissage des processus de décision en médecine: une approche à base de cas. *Proceedings of LEARNING : From natural principles to artificial methods (AIDRI'97)*. 1997, June 23-26, Genève (pp. 43-48). Genève (Swiss): (AIDRI) International Association for the Development of Interdisciplinary Research.

[Colloc, J. 2000]

Colloc, J. (2000). Un système multi-agents neuronal: vers des systèmes d'information épigénétiques. *Revue Systèmes d'Information et Management*. *5*(4) : 55-71.

[Colloc J. & Sybord, C. 2003]

Colloc, J. & Sybord, C. (2003). A Multi-Agent Approach to Involve Multiple Knowledge Models and the Case Base Reasoning Approach in Decision Support Systems. *In proceedings of SSST'03: 35th IEEE Southeastern Symposium on System Theory*. 2003, March 17-18, Morgantown, USA (pp. 247-251). New York (USA): IEEE.

[Colloc J. & al. 2007]

Colloc, J., Cuvelier, M.H. & Summons, P. (2007). A Clinical Metaknowledge Model, *in Proccessing of ISC'2007: the 10th IASTED International Conference on Intelligent Systems and Control*, November 19-21, 2007. Cambridge (Mass) : ACTA Press.

[Colloc, J. et Jacquet-Andrieu, A. 2013]

Colloc, J. et Jacquet-Andrieu, A. (2013). « Système d'aide à la décision clinique, appliqué à la prise en charge neuropsychologique des aphasiques par AVC », in *Les systèmes informatisés complexes en santé. Banque de données, télémédecine : normes et enjeux éthiques*. [12e séminaire de l'Institut international de recherche en éthique biomédicale, IIREB, 19 et 20 mars 2013], sous la dir. de Ch. Hervé, M. Stanton-Jean, Éric Martinent, avec les contrib. de P. Avillach, N. Beaudry, Dr B. Begue... [et al.] Paris : Dalloz, pp. 183-207.

[Colloc, J. & Léry, L 2008]

Colloc, J. & Léry, L. (2008). Prise de décision dans l'éthique au quotidien. Comment décider le soin ? *Santé Décision Management*, *11*(1-2) : 243-254.

[Corcho, O. & al. 2003]

Corcho, O., Fernández-López, M. & Gómez-Pérez, A. (2003). Methodologies, tools and Languages for building ontologies. Where is their meeting point? *Data & knowledge engineering*, *46*(1): 41-64.

[Côté R.A. 1979]

Côté, R.A. & Robboy, S. (1980). Progress in Medical Information Management: Systematized Nomenclature of Medicine (SNOMED). *JAMA*, *243*(8): 756-762.

[Côté, R.A. 1993]

Côté, R.A. (1993). *The systematized nomenclature of human and veterinary medicine: SNOMED international*. Skokie. Illinois: College of American Pathologists. www.nlm.nih.gov/snomed/

[Cranefield, S.J.S. & Purvis, M.K. 1999]

Cranefield, S.J.S. & Purvis, M.K. (1999). UML as an ontology modeling language, in D. Fensel (Ed.), *Proceedings of the Workshop on Intelligent Information Integration.16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, *23*: 46-53. CEUR-WS.org. Retrieved from:

http://CEUR-WS.org?Vol-23/cranefiled-ijcai99-iii.pdf

[Daille, B. & al. 2002]

Daille, B., Fabre, C. & Sébillot, P. (2002). Applications of Computational Morphology. *Many morphologies* (pp. 210-234). Somerville (MA): Cascadilla Press.

[Dean, M. & al. 2004]

Dean, M., Schreiber, G., Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., ... & Stein, L.A. (2004). OWL web ontology language reference. Retrieved from http://www.w3.org/TR/owl-ref/

[DeLoach, S.A. & Hartrum, T.C. 1999]

DeLoach, S.A. & Hartrum, T.C. (1999). A Theory-Based Representation for Object-Oriented Domain Models. *IEEE Transactions on Software Engineering*, *26*(6): 500-517. New York (USA): IEEE.

[DeLoach, S.A. & Madhukar K. 2005]

DeLoach, S.A. & Madhukar K. (2005). Multi-Agent Systems Engineering: An Overview and Case Study, *Agent-Oriented Methodologies* (pp. 317-340). Pennsylvania: IGI Global.

[DSM-III 1980]

DSM-III. (1980). *Diagnostic and Statistical Manual of Mental Disorders*. Washington: American Psychiatric Association.

[Dubitzky W. & al. 1997]

Dubitzky W., Schuster A., Hughes J.G. & Bell D.A. (1997). An Advanced Case-Knowledge Architecture Based on Fuzzy Objects. *In Applied Intelligence : The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, Kluwer Academic Publishers, USA, *7*: 1-18.

[Dussart, C. & al. 2008]

Dussart, C., Pommier, P. & Siranyan, V. (2008). Optimizing clinical practice with case-based reasoning approach [J]. *Journal of evaluation in clinical practice*, *14*(5): 718-720.

[Ermine, J.L. et al. 1996]

Ermine, J.L., Chaillot, M., Bigeon, P., Charreton, B. & Malavieille, D. (1996). MKSM, méthode pour la gestion des connaissances. *Ingénierie des systèmes d'information*, *4*(4) : 541-575. AFCET. Paris : Hermès.

[Ermine, J.L. 2001]

Ermine, J.L. (2001). Capitaliser et partager les connaissances avec la méthode MASK. *Ingénierie et capitalisation des connaissances* (pp. 66-105). Paris : Hermès.

[Falasconi, S. & al. 1996]

Falasconi, S., Lanzola, G. & Stefanelli, M. (1996). Using Ontologies in Multi-Agents Systems. *Proceedings of KAW'96: the 10th Knowledge Acquisition for Knowledge Based Systems Workshop*. 1996, November 9-14, Banff (Canada), *28*: 1-20.

[Fatras, J.Y. & Goudet, B. 1993]

Fatras, J.Y. & Goudet, B. (1993). *RMI et santé* [J]. Paris : CFES.

[Ferber, J. & Drogoul, A. 1992]

Ferber, J. & Drogoul, A. (1992). Using reactive multi-agent systems in simulation and problem solving. *Distributed artificial intelligence: Theory and praxis*, *5*: 53-80.

[Ferber, J. & Perrot, J.F. 1995]

Ferber, J. & Perrot, J.F. (1995). *Les systèmes multi-agents: vers une intelligence collective*. Paris : InterEditions.

[Ferber, J. & Gutknecht, A. 1998]

Ferber, J. & Gutknecht, A. (1998). Un méta-modèle organisationnel pour l'analyse, la conception et l'exécution de systèmes multi-agents, in *Proceedings of Third International Conference on Multi-Agent Systems ICMAS*, 98, 1998, July 3-7, Paris, France (pp. 128-135). New York (USA): IEEE Computer Society.

[Ferber, J. 1999]

Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*. Reading: Addison-Wesley.

[Fehrer, D. et al. 1994]

Fehrer, D., Hustadt, U., Jaeger, M., Nonnengart, A., Ohlbach, H.J., Schmidt, R.A.,... & Weydert, E. (1994). Description logics for natural language processing, *in Proceedings of the International Workshop on Description Logics,* May 28-29, 1994. Bonn, Germany (pp. 80-84). Bonn: CEUR Workshop

[Fernández-López, M. & al. 1997]

Fernández-López, M., Gómez-Pérez, A. & Juristo, N. (1997). Methontology: from ontological art towards ontological engineering, in *Proceedings of the AAA197 Spring Symposium Series on ontological Engineering*, March 1997, Stanford, USA (pp. 33-40). California (USA): AAAI Press.

[Fileto, R. & al. 2003]

Fileto, R., Liu, L., Pu, C., Assad, E.D. & Medeiros, C.B. (2003). POESIA: An ontological workflow approach for composing Web services in agriculture. *The VLDB Journal. The International Journal on Very Large Data Bases*, *12*(4): 352-367.

[Foskett, D.J. 1997]

Foskett, D.J. (1997). Thesaurus, in *Readings in information retrieval*. San Francisco, California (pp. 111-134). Morgan Kaufmann Publishers Inc.

[Frath, P. et al. 2000]

Frath, P., Oueslati, R. & Rousselot, F. (2000). Identification de relations sémantiques par repérage et analyse de cooccurrences de signes linguistiques. *Ingénierie des connaissances. Évolutions récentes et nouveaux défis* (pp. 291-304). Paris : Eyrolles.

[Frath, P. 2005]

Frath, P. (2004). Acquisition automatique de connaissances : une méthode semi-automatique d'annotation de corpus. *Yazik i Literatura*. Retrieved from : www.res-per-nomen.org/respernomen/pubs/tal/TAL11-Tioumen.doc

[Frécon, L. & Kazar, O. 2009]

Frécon, L. & Kazar, O. (2009). *Manuel d'intelligence artificielle*. Lausanne (C.H) : Presses polytechniques et universitaires romandes.

[Gendrel, D. 2002]

Gendrel, D. (2002). Pneumonies communautaires de l'enfant: étiologie et traitement[J]. *Archives de pédiatrie*, *9*(3) : 278-288.

[Gómez-Pérez, A. 2001]

Gómez-Pérez, A. (2001). Evaluation of ontologies. *International Journal of intelligent systems*, *16*(3): 391-409.

[Gómez-Perez, A. & al. 2003]

Gómez-Perez, A., Corcho-Garcia, O. & Fernandez-Lopez, M. (2003). *Ontological engineering*. New York: Springer-Verlag, Inc.

[Greengrass, E. 2001]

Greengrass, E. (2001). *Information retrieval: A survey*. DOD Technical Report, TR-R52-008-001.
http://www.csee.umbc.edu/csee/research/cadip/readings/IR.report.120600.book.pdf

[Grosof, B.N. & al. 2003]

Grosof, B.N., Horrocks, I., Volz, R. & Decker, S. (2003). Description logic programs: Combining logic programs with description logic. *In Proceedings of www2003: the 12th international conference on World Wide Web*, May, 20-24, 2003, Budapest, Hungary (pp. 48-57). New York: ACM.

[Gruber, T.R. 1993]

Gruber, T.R. (1993). A translation Approach to portable ontology specifications. *Knowledge acquisition*, *5*(2): 199-220.

[Gruber, T.R. 1995]

Gruber T.R. (1995). Toward principles for the design of ontologies used for knowledge sharing?[J]. *International journal of human-computer studies*, *43*(5): 907-928.

[Gruber, T. & Olsen, G. 1994]

Gruber, T. & Olsen, G. (1994). An Ontology for Engineering Mathematics, in J. Doyle, P. Torasso & E. Sandewall Ed., *Proceedings of KD1994, the Fourth International Conference in Principles of Knowledge Representation and Reasoning* (pp. 258-259). Bonn (Germany): Morgan Kaufmann.

[Gruenfeld, D.H. & Tiedens, L.Z. 2010]

Gruenfeld, D.H. & Tiedens, L.Z. (2010). Organizational preferences and their consequences. *Handbook of social psychology*. Hoboken (N.J.): J. Wiley.

[Gruninger, M. & Fox, M.S. 1995]

Gruninger, M. & Fox, M.S. (1995). Methodology for the Design and Evaluation of Ontologies, in *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, 1995, August, 20-25, Montréal (Québec), pp. 1-10.

[Guarino, N. & al. 1994]

Guarino, N., Carrara, M. & Giaretta, P. (1994). An Ontology of Meta-Level Categories, in *Knowledge Representation* (pp. 270-280). Burlington (Massachusetts): Morgan Kaufmann.

[Guarino, N. & al. 1994]

Guarino, N., Carrara, M. & Giaretta, P. (1994) Formalizing Ontological Commitments. *12th National Conference on Artificial Intelligence*. *AAAI-94*, 1994, July 31-August 4, Seattle, Washington (pp. 560-567). California (USA): AAAI Press.

[Guarino, N. 1998]

Guarino, N. (Ed.). (1998). Formal Onthology in Information Systems: *Proceedings of the First International Conference* (FIOS'98), June 6-8, Trento (Italy): IOS Press.

[Gupta, U.G. 1994]

Gupta, U.G. (1994). How case-based reasoning solves new problems. *Interfaces*, *24*(6): 110-119.

[Haimowitz, I.J. 1988]

Haimowitz, I.J. (1988). *Using NIKL in a large medical knowledge base* (Doctoral dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science).

[Hallissey, M.T. & al. 1994]

Hallissey, M.T., Dunn, J.A., Ward, L.C., Allum, W.H. & Group, B.S.C. (1994). The second British Stomach Cancer Group trial of adjuvant radiotherapy or chemotherapy in resectable gastric cancer: five-year follow-up. *The Lancet*, *343*(8909): 1309-1312.

[Hamie, A. & al. 1998]

Hamie, A., Howse, J. & Kent, S. (1998). Interpreting the object constraint language, in *Software Engineering Conference, 1998. Proceedings.* 1998, August 2, Tapei, Taiwan (pp. 288-295). New York (USA): IEEE.

[Hendler, J. & al. 2002]

Hendler, J., Berners-Lee, T. & Miller, E. (2002). Integrating Applications on the Semantic Web, *Journal-Institute of electrical engineers of Japan*, *122*(10): 676-680.

[Hewitt, C. & al. 1973]

Hewitt, C., Bishop, P. & Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. *In Proceedings of the 3rd international joint conference on Artificial intelligence*, 1973, August 20-23, Stanford, California (pp. 235-245). Burlington (Mass): Morgan Kaufmann Publishers Inc.

[Hobbs, J.R. & al. 1997]

Hobbs, J.R., Appelt, D., Bear. J. & al. (1997). 13 FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text[J], *Finite-State Language Processing* (pp. 383-406). Cambridge (Mass): MIT Press.

[Horrocks, I. 2002]

Horrocks, I. (2002). DAML+OIL: A Logic Description for the Semantic Web. *IEEE Data Engineering Bulletin, 25*(1): 4-9.

[Horrocks, I. & al. 2004]

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. & Dean, M. (2004). SWR: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, *21*. Retrieved from www.w3.org/Submission/SWRL/

[IBM 2003]

IBM, Ontology Definition Metamodel, Fourth Revised Submission to OMG/RFP ad/2003-03-40 Submitted by IBM.

[IEEE Standard 830-1998]

IEEE Standard 830-1998. IEEE Recommended Practice for Software Requirements Specifications.

[IEEE Standard 1016-1998]

IEEE Standard 1016-1998. IEEE Recommended Practice for Software Design Descriptions.

[Jennings, N.R. 1993]

Jennings, N.R. (1993). Specification and implementation of a belief-desire-joint-intention architecture for collaborative solving", *Intern. Journal of Intelligent and Cooperative Information Systemes. 2*(3): 289-318.

[Jennings, N.R. 2000]

Jennings, N.R. (2000). On agent-based software engineering. *Artificial intelligence*, *117*(2) : 277-296.

[Kassel, G. et al. 2000]

Kassel, G., Abel, M.H., Barry, C., Boulitreau, P., Irastorza, C. et Perpette, S. (2000). Construction et exploitation d'une ontologie pour la gestion des connaissances d'une équipe de recherche, in *IC2000, Journées francophones d'ingénierie des connaissances*, 10-12 mai 2000, Toulouse, France (pp. 251-259). Toulouse : IRIT.

[Kassel, G. 2002]

Kassel, G. (2002). OntoSpec: une méthode de spécification semi-informelle d'ontologies, in *IC2002, Actes des 13^{e} journées francophones d'Ingénierie des Connaissances*, 28-30 mai 2002. Rouen, France (pp. 75-87). Toulouse : IRIT.

[Kawazoe, Y. & Ohe, K. 2008]

Kawazoe, Y. & Ohe, K. (2008). An ontology-based mediator of clinical information for decision support systems: a prototype of a clinical alert system for prescription. *Methods of Information in Medicine*, *47*(6): 549-559.

[Kinny, D. & al. 1996]

Kinny, D., Georgeff, M. & Rao, A. (1996). *A methodology and modelling technique for systems of BDI agents*. Berlin, Heidelberg: Springer.

[Labrou, Y. & Finin, T. 1997]

Labrou, Y. & Finin, T. (1997). *A Proposal for a new KQML Specification*. Technical Report TR CS-97-03, Baltimore: University of Maryland.

[Labrou, Y. & al. 1999]

Labrou, Y., Finin, T. & Peng, Y. (1999). The current landscape of Agent Communication Languages. *IEEE Intelligent Systems*. *14*(2): 45-52.

[Laleci, G.B. & al. 2008]

Laleci, G.B., Dogac, A., Olduz, M., Tasyurt, I., Yuksel, M. & Okcan, A. (2008). SAPHIRE: A Multi-Agent System for Remote Healthcare Monitoring through Computerized Clinical Guidelines. *Agent Technology and e-Health* (pp. 25-44). New York: Springer.

[Lamy F. & al. 2011]

Lamy F. & al. (2011). SimARC: An Ontology-driven Behavioural Model of Alcohol Abuse. *SIMUL 2011: The Third International Conference on Advances in System Simulation*. Barcelona, Spain (pp.128-133). Barcelone : IARIA.

[Lefèvre, P. 2000]

Lefèvre, P. (2000). *La recherche d'informations: du texte intégral au thésaurus*. Paris : Hermès Science.

[Le Ny, J.F. 1975]

Le Ny, J.F. (1975). Sémantique et psychologie. *Langages, 40* : 3-29.

[Lenski, G.E. 1954]

Lenski, G.E. (1954). Status crystallization: a non-vertical dimension of social status. *American sociological review*, *19*(4): 405-413. Washington (USA): American Sociological Association.

[Lenz, M. & al. 1998]

Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S. (1998). *Case-Based Reasoning Technology, from foundations to Applications*. Lecture Notes in Artificial Intelligence 1400, Heidelberg: Springer Verlag.

[Li, S.T. & al. 2010]

Li, S.T., Kuo, S.C. & Tsai, F.C. (2010) An intelligent decision-support model using FSOM and rule extraction for crime prevention[J]. *Expert Systems with Applications*, *37*(10): 7108-7119.

[Li, S.T. & Tsai, F.C. 2010]

Li, S.T. & Tsai, F.C. (2010). Constructing tree-based knowledge structures from text corpus. *Artificial Intelligence*, *33*(1) : 67-78.

[Lindberg, D.A. & al. 1993]

Lindberg, D.A., Humphreys, B.L. & McCray, A.T. (1993). The Unified Medical Language System. *Methods of information in medicine*, *32*(4) : 281-291.

[Lovins, J.B. 1968]

Lovins, J.B. (1968). Development of a stemming algorithm, in *Mechanical Translation and Computational Linguistics*, *11*(1-2): 22-31.

[Maedche, A. & Staab, S. 2002]

Maedche, A. & Staab, S. (2002). Measuring similarity between ontologies. In *Knowledge engineering and knowledge management: Ontologies and the semantic web* (pp. 251-263). Berlin, Heidelberg: Springer.

[Malaisé, V. 2005]

Malaisé, V. (2005). *Méthodologie linguistique et terminologique pour la structuration d'ontologies différentielles à partir de corpus textuels*. Université Paris-Diderot, Paris 7, 2005. French. <tel-00162575>

[Magee, J.C. & Galinsky, A.D. 2008]

Magee, J.C. & Galinsky, A.D. (2008). Social Hierarchy: The Self-Reinforcing Nature of Power and Status. *The Academy of Management Annals*, *2*(1): 351-398.

[Manning, C.D. & Schütze, H. 1999]

Manning, C.D. & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge (Mass): MIT Press.

[Marling, C. & al. 2009]

Marling, C., Shubrook, J. & Schwartz, F. (2009). Toward Case-Based Reasoning for Diabetes Management: A preliminary clinical study and decision support system prototype, *Computational Intelligence*, *25*(3): 165-179.

[Martin-Clouaire, R. & Rellier, J. P. 2009]

Martin-Clouaire, R. & Rellier, J.P. (2009). Modeling and simulating work practices in agriculture. *International Journal of Metadata, Semantics and Ontologies*, *4*(1) : 42-53.

[MASK 2001]

MASK (2001). Découvrez les possibilités de stages avec les étudiants de l'ENSC [online]. [Access : 03 janvier 2014].
Available at: http://aries.serge.free.fr/index.php?page=content/MASK/SA32.

[Martin, J. & Odell, J.J. (1994)

Martin, J. & Odell, J.J. (1994). *Object-oriented methods*. New Jersey (USA): Prentice hall PTR.

[McGuinness, D.L. & Wright, J.R. 1998]

McGuinness, D.L. & Wright, J.R. (1998). Conceptual modeling for configuration: A description logic-based approach. *AI EDAM*, *12*(4): 333-344.

[Miksch, S. & al. 1997]

Miksch, S., Shahar, Y. & Johnson, P. (1997). Asbru: A task-specific, intention-based, and time-oriented language for representing skeletal plans, in *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*, 22-24 January 1997 (pp. 9-19). Milton Keynes (UK): The Open University.
[Minsky, M. 1974]

Minsky, M. (1974). *A framework for representing knowledge*. in *Memo n° 306,* June, 1974.-Massachusetts (USA): MIT Press.

[Mizoguchi, R. & al. 1995]

Mizoguchi, R., Vanwelkenhuysen, J. & Ikeda, M. (1995). Task Ontology for Reuse of Problem Solving Knowledge. Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing (pp. 46-59). Amsterdam, Washington, DC: IOS Press.

[Montani S. & al. 2001]

Montani S. & al. (2001). *Integrating Different Methodologies for Insulin Therapy Support in Type 1 Diabetic Patients.* Lecture Notes in *Computer Science, 2101*: 121-130.

[Mots invariables 2013]

Mots invariables (2013). [Access : 11 janvier 2014]. Consultable:
http://jeanmichel.saus.free.fr/mots_invariables.htm/

[Van Bemmel, Y. & al. 1997]
Van Bemmel, Y., Musen, M.A. & Helder, J.M. Ed. (1999). *Handbook of medical informatics*. Houten (Netherlands): Bohn Stafleu van Loghum Springer.

[Müller, J.P. 1996]
Müller, J.P. (1996). *The Design of Intelligent Agents A layered Approach*. Berlin, Heidelberg (Germany): Springer Science & Business Media Verlag,

[Nebel, B. 1990]
Nebel, B. (1990). *Reasoning and revision in hybrid representation systems*, vol. 422. Heidelberg (Germany): Springer-Verlag,

[Neches, R. & al. 1991]
Neches, R., Fikes, R.E., Finin, T., Gruber, T., Patil, R., Senator, T. & Swartout, W.R. (1991). Enabling technology for knowledge sharing. *AI magazine*, *12*(3): 36-56.

[Noy, N.F. et McGuinness, D.L. 2000]
Noy, N.F. et McGuinness, D.L. (2000). *Développement d'une ontologie 101 : Guide pour la création de votre première ontologie.* Rapport technique, Université de Stanford. Retrieved from
http://www.bnf.fr/pages/infopro/normes/pdf/no-DevOnto.pdf

[Noy, N.F. & McGuinness, D.L. 2001]
Noy, N.F. & McGuinness, D.L. (2001). *Ontology development 101: A guide to creating your first ontology*. A Stanford Medical Informatics Technical Report. Retrieved from
http://protege.stanford.edu/publications/ontology_development/ontology101.pdf

[O'Leary, D.E. 1998]
O'Leary, D.E. (1998). Using AI in knowledge management: Knowledge bases and ontologies. *Intelligent Systems and Their Applications*, *IEEE Intelligent Systems*, *13*(3): 34-39.

[CIM-10 1993]
CIM-10 (1993). Classification statistique internationale des maladies, et des problèmes de santé connexes : CIM-10. *Organisation mondiale de la santé*, 10e éd. Genève : OMS.

[Pender, T. & al. 2003]
Pender, T., McSheffrey, E. & Varveris, L. (2003). *UML bible*. Chichester: Wiley.

[Porter, M.F. 1980]
Porter, M.F. (1980). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, *14*(3): 130-137.

[Rao, A.S. & Georgeff, M.P. 1995]

Rao, A.S. & Georgeff, M.P. (1995). BDI Agents: From Theory to Practice, in *ICMAS.* **95**: *1rst International Conference on Multi-Agent Systems*, 1995, June 12-14, San Francisco, California (pp. 312-319). San Francisco: AAAI Press.

[Raoult, P.D. et al. 2011]

Raoult, P.D., Brouqui, P.P., Drancourt, P.M., Rovery, C. & Fenollar, F. (2011). Recommandations thérapeutiques. Retrieved from http://www.mediterranee-infection.com/arkotheque/client/ihumed/_depot_arko/articles/88/recommandations-therapeutiques-antibio-guide_doc.pdf

[Rector, A.L. & al. 1993]

Rector, A.L., Nowlan, W.A. & Glowinski, A. (1993). Goals for concept representation in the GALEN project. *In Proceedings of the Annual Symposium on Computer Application in Medical Care*. 1993, 30 October - 1 November, Washington, USA (pp. 414-418). New York (USA): IEEE Computer Society.

[Rector, A.L. 1998]

Rector, A.L. (1998). Thesauri and formal classifications: Terminologies for people and machines. *Methods of Information in Medicine*, *37*(4-5): 501-509.

[Richters, M. & Gogolla, M. 1998]

Richters, M. & Gogolla, M. (1998). On formalizing the UML object constraint language OCL. In *Proceedings of ER'98*: *17th International Conference on Conceptual Modeling,* 1998, November 16-19, Singapore (pp. 449-464). Berlin, Heidelberg (Germany): Springer.

[Roche, C. 2005]

Roche, C. (2005). Terminologie et ontologie. *Langages*, *157* : 48-62.

[Rochfeld, A. & Bouzeghoub, M. 1993]

Rochfeld, A. & Bouzeghoub, M. (1993). From Merise to OOM. *Ingénierie des systèmes d'information*, *1*(2) : 151-176.

[Roussey, C. et al. 2002]

Roussey, C., Calabretto, S. & Pinon, J.M. (2002). Le thésaurus sémantique : contribution à l'ingénierie des connaissances documentaires, in B. Bachimont, Ed., *Actes des 6e Journées Ingénierie des Connaissances*, 28-30 mai 2002, Rouen, France (pp. 209-220). Toulouse : IRIT.

[Rumbaugh, J. & al. 1999]

Rumbaugh, J., Jacobson, I. & Booch, G. (1999). *The unified modeling language reference manua*l. Reading (Mass): Addison-Weley.

[Sánchez, D. 2010]

Sánchez, D. (2010). A methodology to learn ontological attributes from the Web. *Data Knowledge Engineering*, *69*(6): 573-597.

[Sayyad-Shirabad J. & al. 2012]

Sayyad-Shirabad, J., Wilk, S., Michalowski, W. & Farion, K. (2012). Implementing an Integrative Multi-Agent Clinical Decision Support System with Open Source Software, *Journal of Medical Systems*, *36(*1): 123-137.

[Schuster, A. & al. 1997]

Schuster, A., Dubitzky, W., Adamson, K., Bell, D.A. & Hughes, J.G. (1997). Processing Similarity between a Mix of Crisply and fuzzily Defined Case Properties, in *Proceedings of 2nd International ICSC Symposium on Fuzzy Logic and Applications*, 1997, February 12-14. Zurich, Switzerland (pp. 247-253). Zurich: ICSC Academic Press.

[Sebban, M. & al. 1997]

Sebban, M., Rabaseda, S. & Zighed, D.A. (1997). Construction of a gait identification model by a statistical learning. *In Proceedings of AIDRI'97. LEARNING: From natural principles to artificial methods*, June, 23-26. Genève, C.H. (pp. 49-53). Genève: (AIDRI) International Association for the Development of Interdisciplinary Research.

[Siegel, J. 2000]

Siegel, J. (2000). *CORBA 3 fundamentals and programming, II*. Chichester: John Wiley & Sons.

[Shen, Y. & al. 2012]

Shen, Y., Jacquet-Andrieu, A. & Colloc, J. (2012). Un système multi-agents d'aide à la décision Clinique fondé sur des ontologies. *In Proceedings of the Applications médicales de l'informatique : nouvelles approches* (AMINA 2012), 17-19 décembre, Mahdia, Tunisie (pp. 136-145). Presses Université de Monastir.

[Sommerville, I. 2004]

Sommerville, I. (2004). *Software Engineering*, 7th Edition (3 June 2004). Boston: Addison Wesley.

[Soley, R.M. 1992]

Soley, R.M. (1992). *Object Management Architecture Guide: Revision 2.0*. Object Management Group. New York: Wiley-QED, cop.

[Sowa, J. F. 1983]

Sowa, J.F. (1983). *Conceptual structures: information processing in mind and machine*. Readind (Mass): Addison Wesley.

[Sowa, J.F. 2000]

Sowa, J.F. (2000). *Knowledge representation: logical, philosophical, and computational foundations* (Vol. 13). Pacific Grove: Brooks/Cole.

[Stanley Y.W. & al. 1995]

Stanley Y.W. & al. (1995). An Extensible Knowledge Base Management System for Supporting Rule-based Interoperability among Heterogeneous Systems. *Conference*

on Information and Knowledge Management CIKM. Baltimore, Maryland (pp. 1-10). Baltimore: ACM.

[Swartout, B. & al. 1996]

Swartout, B., Patil, R., Knight, K. & Russ, T. (1996). Toward distributed use of large-scale ontologies, in *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System*. *Workshop (KAW'96)*, Nov.9-14, 1996. Banff (pp. 138-148). Calgary (Canada): SRDG Publications.

[Sybord C. & Colloc J. 1997]

Sybord C. & Colloc J. (1997). A cognitive approach of the decision process to develop health decision support systems. *ECIS'97, 5th European Conf on Information Systems*. June, 19-21, 1997, Cork, Ireland (pp. 396-412). Cork: Cork Publishing.

[Tardieu, H. et al. 1983]

Tardieu, H., Rochfeld, A., Colletti, R. et Lesourne, J. (1983). *La méthode MERISE: principes et outils*. Editions d'organisation. Retrieved from http://www.lsis.org/dea/M6optionD/Exp-GL5Merise.pdf

[Tesauro, G.J. & Kephart, J.O. 2000]

Tesauro, G.J. & Kephart, J.O. (2000). Foresight-based pricing algorithms in agent economies, *Decision Support Systems*, *28*: 49-60.

[Todirascu, A. & al. 2000]

Todirascu, A., Beuvron, F.B. & Gâlea, D. (2000). Using Description Logics for Ontology Extraction[C]. *ECAI Workshop on Ontology Learning*, August, 20-25 2000. Berlin, Germany. Netherland: The Netherland IOS Press

[UML 2001]

UML (2001), Retrieved May 17, 2013 from the Wikipedia : http://en.wikipedia.org/wiki/Unified_Modeling_Language.

[Uschold, M. & Gruninger, M. 1996]

Uschold, M. & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Engineering Review, 11*(2): 93-136.

[Van Heist, G. & al. 1997]

Van Heist, G., Schreiber, T. & Wielinga, B. (1997). Using Explicit Ontologies in KBS *International Journal of Human-Computer Studies*, *46*(2-3): 183-292.

[Vogel, C. 1988]

Vogel, C. (1988). *Génie cognitif*. Paris : Masson.

[Vogel, C. 1989]

Vogel, C. (1989). Kod : Une méthode inspirée par l'anthropologie. Est-elle applicable au droit. *Supplément de Lilliade*, *2* : 53.

[Von Bertalanffy, L. 1968]

Von Bertalanffy, L. (1968). *General system theory: Foundations, development, applications* (Revised Ed.). New York: George Braziller.

[Waksman, S.A. & Woodruff, H.B. 1941]

Waksman, S.A. & Woodruff, H.B. (1941) Actinomyces antibioticus, a new soil organism antagonistic to pathogenic and non-pathogenic bacteria[J]. *Journal of bacteriology*, ***42***(2): 231-249.

[Wang, X. & Chan, C.W. 2001]

Wang, X. & Chan, C.W. (2001). Ontology modeling using UML, in *OOIS 2001: 7th International Conference on Object-Oriented Information Systems*, 2001, August 1, 27-29. Calgary, Canada, (pp. 59-68). London: Springer London.

[Warmer, J. & Kleppe, A. 2003]

Warmer, J. & Kleppe, A. (2003). *The object constraint language: getting your models ready for MDA*. Addison-Wesley Longman Publishing Co., Inc..

[Whorf, B.L. 1956]

Whorf, B.L. (1956). *Language, thought and reality*. Selected writings of Benjamin Lee Whorf. Préf. by John B. Carroll, foreword by Stuart Chase. [Cambridge] : Technology Press of Massachussets Institute of Technology. New York : John Wiley & Sons ; London: Chapman et Hall.

[Wielinga, B.J. & al. 1992]

Wielinga, B.J., Schreiber, A.T. & Breuker, J.A. (1992). KADS: A modelling approach to knowledge engineering. *Knowledge acquisition*, *4*(1): 5-53.

[Wooldridge, M. & Jennings, N.R. 1995]

Wooldridge, M. & Jennings, N.R. (1995) Intelligent agents: Theory and practice [J]. *The knowledge engineering review*, *10*(2): 115-152.

[Wooldridge, M. & al. 2000]

Wooldridge, M., Jennings, N.R. & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*. *3*(3): 285-312.

[Zighed D.A. et al. 1992]

Zighed D.A., Auray J.P. et Duru G. (1992). *SIPINA : Méthode et Logiciel*. Lyon: Ed. A. Lacassagne.

[Zweigenbaum, P. 1994]

Zweigenbaum, P. (1994). MENELAS: an access system for medical records using natural language. *Computer methods and programs in Biomedicine*, *45*(1): 117-120.

Littérature grise

[Bourigault, D. 1994]

Bourigault, D. (1994). *Lexter : un Logiciel d'EXtraction de TERminologie : application à l'acquisition des connaissances à partir de textes*, sous la dir. de Jean-Pierre Desclès, Thèse de doctorat, EHESS Paris : [s.n.], publiée en 1995. Lille : Atelier de reproduction des thèses.

[Charlet J. 2002]

Charlet J. (2002). *L'Ingénierie des connaissances : développements, résultats et perspectives pour la gestion des connaissances médicales*. Habilitation à diriger des recherches (HDR), Université Paris 6.
http://archivesic.ccsd.cnrs.fr/sic_00001064/document

[Colloc, J. 1985]

Colloc, J. (1985). *Informatique Médicale (SIAMED) Application : Logiciel original d'antibiothérapie médicale* (Unpublished doctoral dissertation). Université Lyon 1, Lyon.
https://owl.english.purdue.edu/owl/resource/560/09/

[Daille, B. 1994]

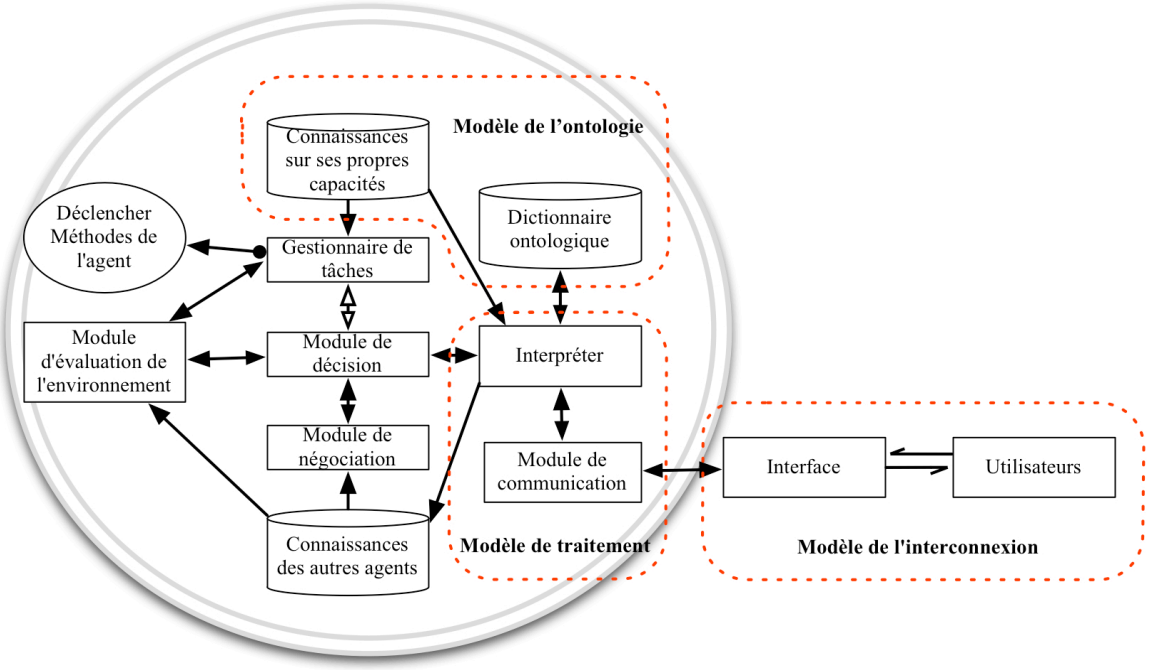
Daille, B. (1994). *Approche mixte pour l'extraction de terminologie : statistique lexicale et filtres linguistiques,* sous la dir. de Laurence Danlos, Doctorat d'informatique fondamentale, Université Paris VII. Paris : [s.n.], 1994. Retrieved from INIST-CNRS. (T 122233)
http://cat.inist.fr/?aModele=afficheN&cpsidt=1815057/

[Gandon, F. 2002]

Gandon, F.(2002). *Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web*, sous la dir. de Rose Dieng Kuntz, Doctorat d'informatique, Nice [S.l.] : [s.n.].

ANNEXE

Annexe 1. Opération



Annexe figure 1 Architecture du Type d'Agent Clinique Général (TACG)

Fonctions de modèle

« Modèle de l'Interconnexion »

Après avoir reçu les requêtes des utilisateurs finaux, le « Module d'Interface » transfère les requêtes au modèle de traitement et au modèle de l'ontologie pour analyses et extraction. En fin de compte, il renvoie les réponses au format texte ou aux documents aux utilisateurs.

« Modèle de Traitement »

Le Modèle de Traitement comprenant le « Module de Communication » qui fonctionne comme une boîte aux lettres et le « Module Interpréteur », le contrôleur de l'algorithme contient tous les algorithmes et programmations pour l'analyse et l'extraction des termes et des associations.

« Modèle de l'Ontologie »

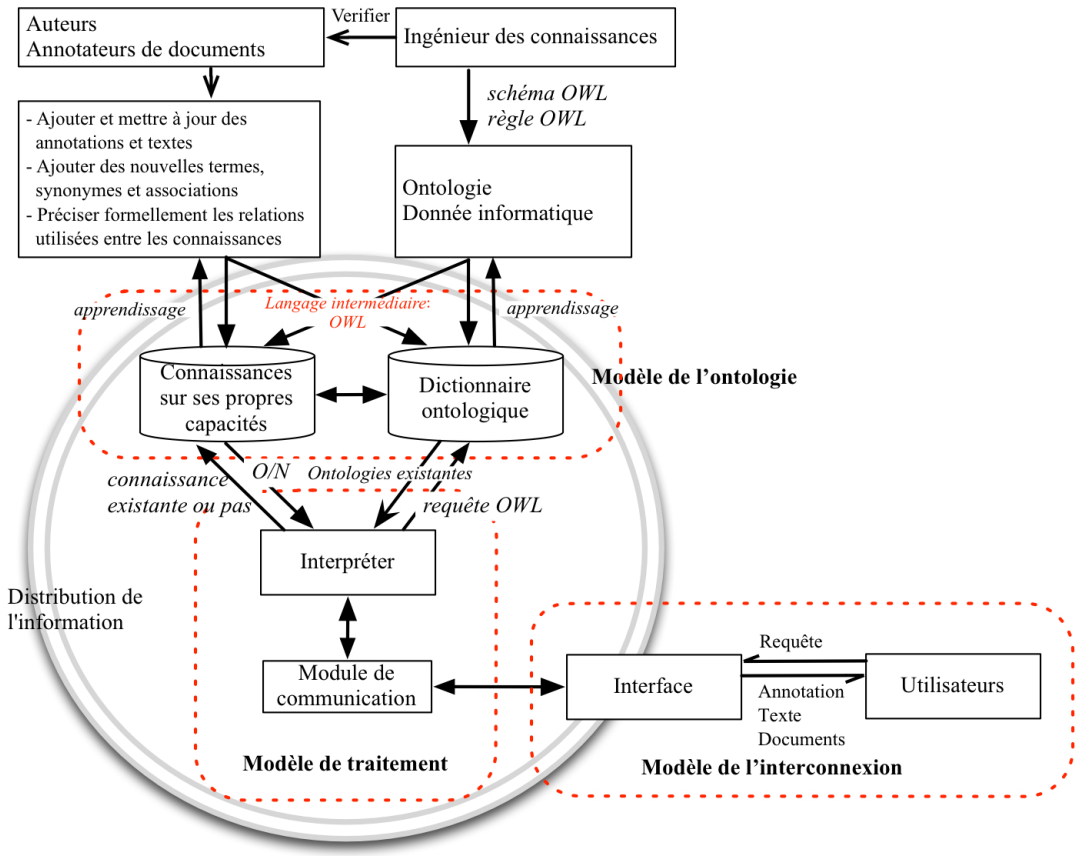
Ce module est responsable du traitement d'extraction de termes et d'associations à partir d'un corpus de textes enregistrés, ensuite, dans un « fichier texte » avec le nom du corpus utilisé pour l'extraction.

Il faut que le modèle « Connaissance sur ses propres capacités » vérifie si la connaissance correspond et que la requête d'entrée est disponible, puis il faut transférer le signal « Oui » ou « Non » au « Modèle de traitement », pour chercher les textes relatifs, ou bien terminer et revenir au résultat vide.

Fonctions d'ingénierie des Connaissances

- Entrer et vérifier les nouveaux termes, associations, synonymes, schémas et règles OWL ;
- Vérifier et autoriser les modes d'action des auteurs et annotateurs de documents ;
- Fournir des modèles de représentation permettant la re-formulation des connaissances ;
- Fournir des modèles d'implantation informatiques ;
- Proposer des méthodes décrivant les étapes de la démarche de modélisation ;
- Construire des outils permettant la programmation des bases de connaissances.

1.1 Le processus dans l'ensemble du système multi-agents



Annexe figure 2 Système multi-agents de la distribution de l'information

OPERATION-1. Processus de requête réponse dans le système multi-agents

Titre	Processus de requête réponse dans le système multi-agents
But de la fonction	La coopération des différents agents dans notre système doit être capable de répondre aux requêtes des utilisateurs finaux en passant la requête au format OWL ; il faut vérifier les propriétés sémantiques et syntaxiques des requêtes, et chercher les réponses correspondantes dans notre corpus existant (« Dictionnaire ontologique »).
Les formats de données	Requêtes
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les réponses aux requêtes des utilisateurs finaux sont des textes ou des documents.

Séquence de réponse et stimulus

Utilisateur / Ingénieur des connaissances	**Système multi-agents**
1. **L'utilisateur final** entre une requête	
	2. Le modèle « **Interface** » du «**Modèle de l'Interconnexion**» transfère les requêtes au « **Modèle de traitement** » pour l'analyse et l'extraction.
	3. Le Modèle de Traitement comprend le « **Module de Communication** », qui fonctionne comme un boîte aux lettres, et le module « **Interpréter** », le contrôleur de l'algorithme contient toutes les programmations pour l'analyse et l'extraction des termes et des associations. Programmation.1 Analyseur morphologique-lemmatisation Programmation.2 Synonyme Programmation.3 POS Tagger Programmation.4 Algorithme de l'alignement Programmation.8 Les relations remarquables Programmation.9 Extraction des relations des mots cibles
	4. Déclencher le module « **Interpréter** » pour passer la requête au format OWL. (détaillé dans OPERATION-14)
	5. Le modèle **« Connaissance sur ses propres capacités »** vérifie si la connaissance correspond la requête d'entrée est disponible, puis il transfert le signal « Oui » ou « Non » au «**Modèle de traitement**» pour chercher les textes relatifs, ou pour terminer et revenir au résultat vide.

	6. Sélectionner les textes correspondants dans le «**Dictionnaire ontologique** » (détaillé dans l'OPERATION-16)
7. **L'ingénieur des connaissances** supervise l'extraction de texte. (Optionnel)	
	8. Le modèle « **Interface** » du «**Modèle de l'Interconnexion**» renvoie les réponses au format texte ou les documents aux utilisateurs selon la pertinence et la fréquence voulue, avec l'aide de la «Programmation.10 Compteur de mots»
9. L'**utilisateur final** fournit les commentaires	
10. **L'ingénieur de connaissances** surveille et améliorer ces processus	
	11. Le modèle « **Dictionnaire ontologique** » renouvelle le corpus (détaillé dans l'OPERATION-2)

OPERATION-2. Apprendre les nouveaux termes à travers des requêtes de l'utilisateur

Titre	Apprendre les nouveaux termes et relations à travers des requêtes de l'utilisateur
But de la fonction	Notre système doit permettre d'apprendre de nouveaux concepts de la requête de l'utilisateur final
Les formats de données	Requête
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les nouveaux concepts sont tirés de la requête de l'utilisateur final, et sont enregistrés dans le « **Dictionnaire ontologiqu**e ».

Séquence de réponse et stimulus

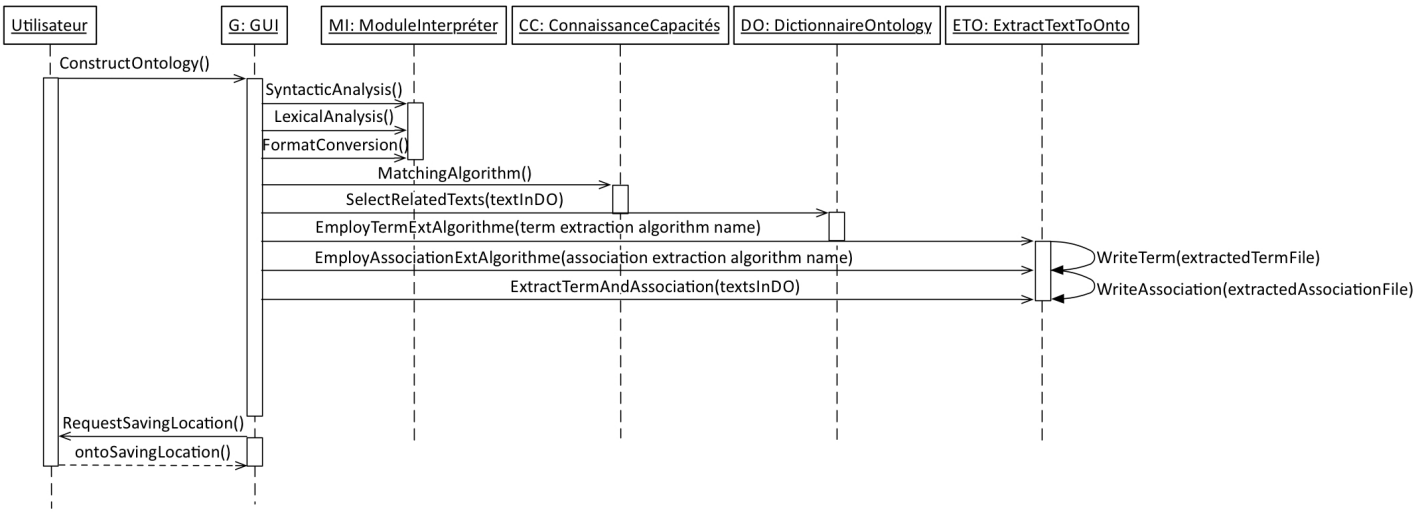
Utilisateur	**Système multi-agents**
1. **L'utilisateur final** entre une requête	
	2. Le modèle «**Interface**» du « **Modèle de l'Interconnexion** » transfère les requêtes au «**Modèle de traitement**» pour les analyses et les extractions.
	3. Le Modèle de Traitement comprenant le « **Module de Communication** » fonctionne comme un boîte aux lettres et le module « **Interpréter** », le contrôleur de l'algorithme contiennent toutes les programmations pour l'analyse et l'extraction des termes et des associations. Programmation.1 Analyseur morphologique-lemmatisation Programmation.2 Synonyme Programmation.3 POS Tagger Programmation.4 Algorithme de l'alignement Programmation.8 Les relations remarquables Programmation.9 Extraire des relations des mots cibles
	4. Déclencher le module « **Interpréter** » pour changer le format de requête à OWL. (détaillé dans OPERATION-14)
	5. Le concept existe dans le « **Dictionnaire ontologique** » ? Oui - appartient au dictionnaire - FIN Non - ne fait pas partie du dictionnaire
	6. Si le concept n'existe pas, il est alors classé dans la hiérarchie de l'ontologie existante en fonction de ses propriétés lexicales et syntaxiques
	7. « **Dictionnaire ontologique** » classe les mots de contenu (nouveaux concepts ciblés) et les contextes/instances correspondants, selon la fréquence (avec «Programmation.10 Compteur de mots»), puis il faut construire les descriptions des nouveaux concepts.
8. **L'ingénieur de connaissances** surveille et améliorer ces processus afin d'éditer le concept et le description du concept	
	9. Les nouveaux concepts sont ajoutés à l'ontologie avec ses caractéristiques linguistiques : lexique, syntaxe etc.
10. Demander à enregistrer les nouveaux concepts	
	11. Demander la confirmation d'enregistrement
12. Confirmer l'enregistrement	
	13. Mettre à jour le « **Dictionnaire ontologique** » avec les nouveaux concepts.

OPERATION-3. Générer des associations à partir d'une liste de termes extraits

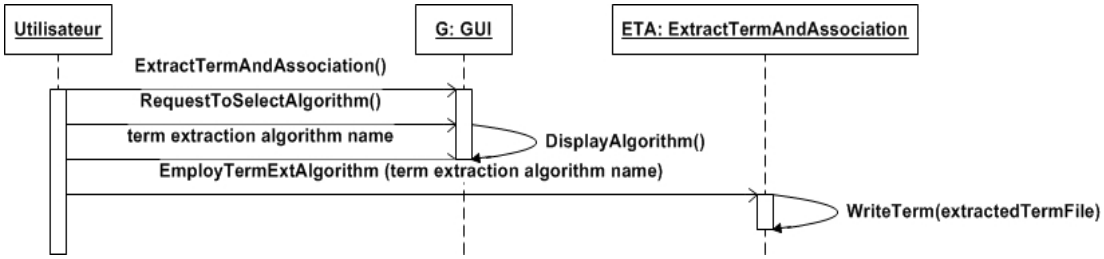
Titre	Générer des associations à partir d'une liste de termes extraits
But de la fonction	Préciser formellement les relations utilisées entre les connaissances
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les nouvelles associations sont obtenues par extraction à partir d'une liste de termes et sont enregistrées dans le «**Dictionnaire ontologique**»

Séquence de réponse et stimulus

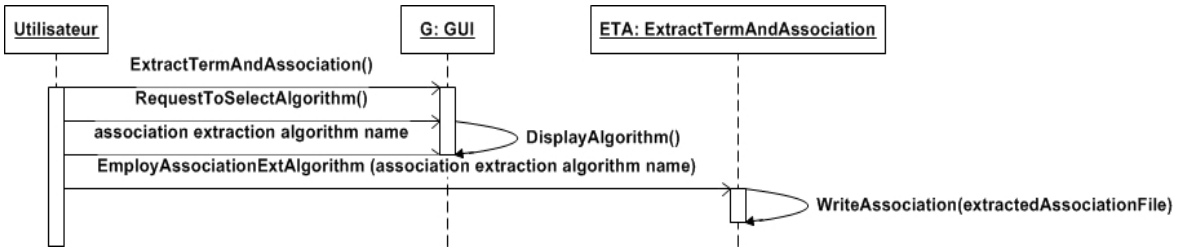
Utilisateur / Auteur / Ingénieur des connaissances	**Prototype de construction d'ontologie automatique**
1. Demander à ajouter un nouveau fichier texte à la «**Modèle de l'Ontologie**»	
	2. Lister des noms des modules des agents - «**Connaissances sur ses propres capacités**» - «**Dictionnaire ontologique**»
3. Sélectionner le dossier correspondant à l'emplacement d'un fichier texte	
4. Confirmer la sélection de corpus	
	5. Définir les fichiers sélectionnés comme corpus
	6. Extraire les associations du corpus selon la «Programmation.8 Les relations remarquables» et la «Programmation.9 Extraire des relations des mots cibles», qui sont détaillés dans l'annexe.4.
	8. Enregistrer les associations extraits



Annexe figure 3 Diagramme de séquence : L'extraction des termes et associations

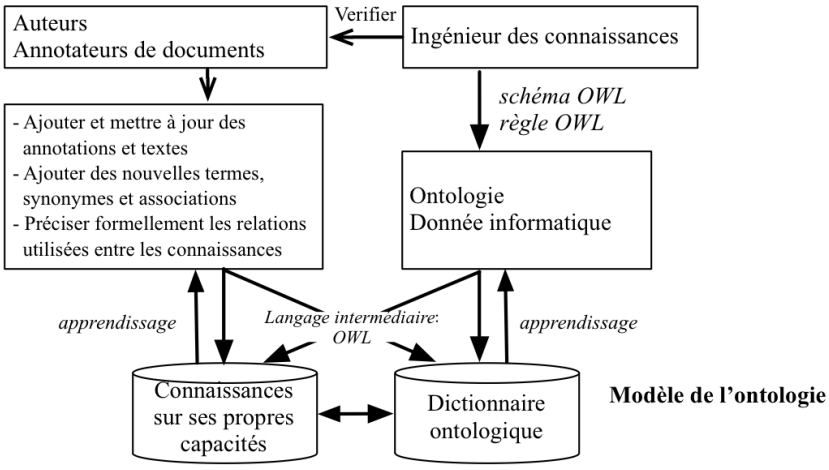


Annexe figure 4 Diagramme de séquence : Employer l'algorithme d'extraction des termes



Annexe figure 5 Diagramme de séquence : Employer l'algorithme d'extraction des associations

1.2 Opération dans le Modèle de l'Ontologie



Annexe figure 6 Modèle de l'Ontologie

OPERATION-4. Ajouter les nouveaux termes au « Modèle de l'Ontologie » par l'auteurs et l'ingénieur de connaissances

Titre	Ajouter les nouveaux termes au « Modèle de l'Ontologie » par les auteurs, annotateurs de documents ou les ingénieurs des connaissances
But de la fonction	Notre prototype doit permettre à l'auteur et l'ingénieur de connaissances d'ajouter de nouveaux concepts au « Modèle de l'Ontologie »
Les formats de données	Documents / annotation / terme d'ontologie / données informatiques
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les nouveaux concepts sont ajoutés dans le module avec l'autorisation de l'ingénieur de connaissances

Séquence de réponse et stimulus

Auteurs / Ingénieur de connaissances	**Modèle de l'Ontologie**
1.a L'auteur ou l'annotateur des documents entre les nouvelles annotations ou les documents ; 1.b L'ingénieur des connaissances entre des nouveaux termes de l'ontologie ou données informatique	
	2. Le « **Module Interpréteur** » utilise les approches basées sur l'apprentissage, elles sont détaillées au chapitre.5 et dans l'annexe.4. Programmation.1 Analyseur morphologique-lemmatisation Programmation.2 Synonyme Programmation.3 POS Tagger Programmation.8 Relations remarquables Programmation.9 Extraire des relations des mots cibles Programmation.10 Compteur de mots
	3. Déclencher « [l']Algorithme de l'alignement »
	4. Le concept existe dans le « **Dictionnaire ontologique** » ? Oui - appartient au dictionnaire - FIN Non - ne fait pas partie du dictionnaire
	5. Pour que le concept existe, il est alors classé dans la hiérarchie de l'ontologie existante selon ses propriétés lexicales et syntaxiques
	7. «**Dictionnaire ontologique**» : il classe les mots de contenu (nouveaux concepts ciblés) et les contextes/instances correspondantes selon la fréquence (avec «Programmation.10 Compteur de mots»), puis il faut construire les descriptions des nouveaux concepts.
7. L'auteur ou l'annotateur des documents édite le concept et la description du concept (optionnel)	
8. **L'ingénieur de connaissances** surveille et améliore ces processus	
	9. Les nouveaux concepts sont ajoutés à l'ontologie avec les caractéristiques linguistiques : lexique, syntaxe etc.
10. Demander à enregistrer les nouveaux concepts	
	11. Demander la confirmation d'enregistrement
12. Confirmer l'enregistrement	
	13. Mettre à jour le «**Dictionnaire ontologique**» avec les nouveaux concepts

Exigence fonctionnelle associée

OPERATION-5: Ajouter les synonymes des termes au « Modèle de l'Ontologie »

OPERATION-6: Ajouter des synonymes propres à l'utilisateur au concept dans le « Modèle de l'Ontologie »

OPERATION-5. Ajouter les synonymes des termes au « Modèle de l'Ontologie »

Titre	Ajouter les synonymes des termes au «Modèle de l'Ontologie»
But de la fonction	Les synonymes sont générés automatiquement dans notre système. Après que l'auteur les a sélectionnés ; l'annotateur des documents ou l'ingénieur de connaissances les enregistre dans la hiérarchie correspondante de l'ontologie
Les formats de données	Mots / documents / annotations / termes de l'ontologie / données informatiques
Stabilité	Stable
Degré de nécessité	Essentiel
Inspection	Vérifier si les nouveaux synonymes existent déjà dans la liste des synonymes du concept
Erreur	Une erreur surgit si le synonyme est ajouté plus d'une fois pour le même concept.
Message de fin	Un nouveau synonyme est ajouté au module

Séquence de réponse et stimulus

Auteurs / Annotateur de documents / Ingénieur de connaissances	**Modèle de l'Ontologie**
1. Sélectionner un concept cible	
2. Demander d'ajouter des synonymes au concept cible	
	3. Construire une liste de synonymes dans le «**Module Interpréteur** »
	4. Vérifier si les synonymes présentés dans la liste sont déjà dans la liste des synonymes du concept ciblé du « **Dictionnaire ontologique** » Oui - les synonymes sont existent déjà - FIN Non - les synonymes sont une nouvelle
	5. Afficher la nouvelle liste de synonymes proposés
6. L'auteur, l'annotateur des documents ou l'ingénieur de connaissances sélectionne les synonymes pour les ajouter dans la liste proposée	
7. Confirmer l'ajout des synonymes	
	8. Enregistrer les synonymes dans la hiérarchie correspondante de l'ontologie

OPERATION-6. Ajouter des synonymes spécifiques à l'utilisateur au concept dans le « Modèle de l'Ontologie »

Titre	Ajouter des synonymes spécifiques à l'utilisateur du concept dans le « Modèle de l'Ontologie »
But de la fonction	Notre système doit permettre à l'utilisateur d'ajouter les synonymes aux concepts dans le «Modèle de l'Ontologie»
Les formats de données	Mots
Stabilité	Stable
Degré de nécessité	Essentiel
Inspection	Vérifier si les synonymes entrés par l'utilisateur existent déjà existants dans la liste des synonymes du concept
Erreur	Une erreur surgit si le synonyme est ajouté plus d'une fois pour le même concept (doublon).
Message de fin	Un nouveau synonyme est ajouté au module

Séquence de réponse et stimulus

Utilisateur	**Modèle de l'Ontologie**
1. Sélectionner un concept	
2. Ajouter une liste de synonymes au concept ciblé	
	3. Le modèle «**Module Interpréteur**» du « **Modèle de Traitement** » analyse les synonymes sur les plans lexical et syntaxique.
	4. Présenter la classe, domaine, propriétés lexicales et syntaxe, et la hiérarchie de l'ontologie des synonymes d'entrée
	5. Vérifier si les synonymes entrés par l'utilisateur sont déjà dans la liste des synonymes existant du concept ciblé Oui - les synonymes existent déjà - FIN Non - les synonymes sont nouveaux
	6. Afficher la liste proposée des nouveaux synonymes
7. Confirmer la liste proposée	
	8. Enregistrer les synonymes

OPERATION-7. Ajouter les nouvelles associations au « Modèle de l'Ontologie »

Titre	L'auteur l'annotateur de documents ou l'ingénieur des connaissances ajoutent les nouvelles relations au « Modèle de l'Ontologie »
But de la fonction	Notre prototype doit permettre à l'auteur, l'annotateur des documents et l'ingénieur de connaissances d'ajouter de nouvelles associations au « Modèle de l'Ontologie »
Les formats de données	Document / annotation / terme de l'ontologie / données informatique
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les nouvelles associations sont ajoutées dans le module avec l'autorisation de l'ingénieur de connaissances

Séquence de réponse et stimulus

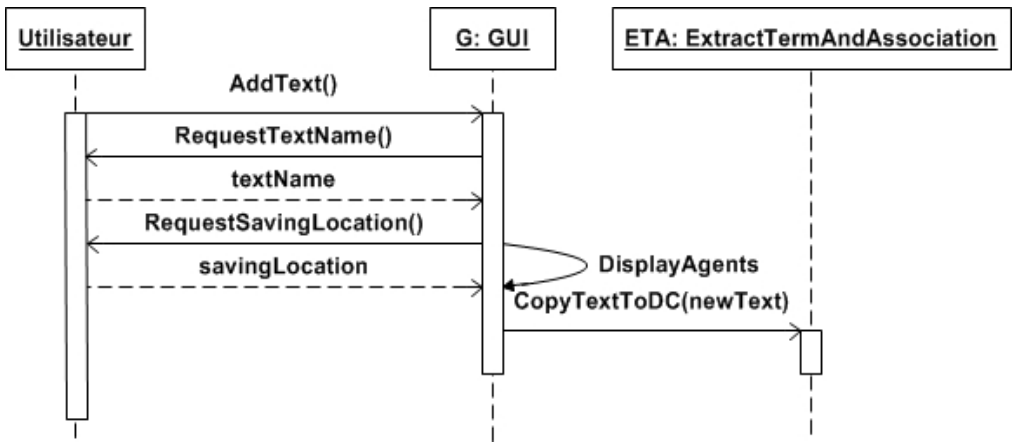
Auteurs / Ingénieur de connaissances	**Modèle de l'Ontologie**
1. Demander d'ajouter associations d'ontologies	
	2. Demander d'entrer une association
3. Entrer une association	
	4. Déclencher le «Programmation.4 Algorithme de l'alignement» du module «**Interpréter**».
	6. L'association existe-t-elle dans le « **Dictionnaire ontologique** » ? Oui - appartient au dictionnaire - FIN Non - ne fait pas partie du dictionnaire
	7. Si l'association n'existe pas, notre système demande de sélectionner les concepts liés à cette association
8. Sélecter les concepts	
	9. Demander d'ajouter le type d'association
10. Sélecter le type d'association	
11. Valider la création de l'association	
	12. Ajouter une association au « **Modèle de l'Ontologie** » et afficher le nom de l'association

OPERATION-8. Ajouter les fichiers texte au « Dictionnaire ontologique »

Titre	Ajouter les fichiers texte au « **Dictionnaire ontologique** »
But de la fonction	Notre prototype doit permettre à l'utilisateur, l'auteur et l'ingénieur de connaissances d'entrer de nouveaux fichiers texte au corpus
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Nouveaux fichiers texte sont enregistré au « **Dictionnaire ontologique** »

Séquence de réponse et stimulus

Utilisateur / Auteur / Ingénieur de connaissances	**Système de construction de l'ontologie automatique**
1. Demander d'ajouter un nouveau fichier texte au « **Modèle de l'Ontologie** »	
	2. Lister des noms des modules des agents - «**Connaissances sur ses propres capacités**» - «**Dictionnaire ontologique**»
3. Sélectionner le dossier correspondant à l'emplacement d'un fichier texte	
4. Sélectionner les fichiers correspondant au corpus de texte	
5. Confirmer la sélection du corpus	
	6. Demander le nom de fichier
7. Entrer le nom de fichier	
	8. Les nouveaux fichiers texte sont enregistrés au « **Dictionnaire ontologique** »



Annexe figure 7 Diagramme Séquence: Ajouter des fichiers texte au «Dictionnaire ontologique»

OPERATION-9. Sélectionner les heuristiques pour la construction d'ontologies

Titre	Sélectionner les heuristiques pour la construction d'ontologies
But de la fonction	Le système doit permettre à l'utilisateur, l'auteur et l'ingénieur de connaissances de sélectionner les heuristiques pour la construction d'ontologies
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les heuristiques nécessaires sont retenues pour la construction d'ontologies

Séquence de réponse et stimulus

Utilisateur / Auteur / Ingénieur de connaissances	**Système de construction de l'ontologie automatique**
1. Demander de sélectionner les heuristiques pour la construction d'ontologies	
	2. Le «**Module de communication**» affiche une liste d'heuristiques pour la construction d'ontologies
3. Sélectionner les heuristiques dans la liste proposée	
4. Confirmer la sélection des heuristiques	
	5. Construire l'ontologie selon les heuristiques sélectionnés

OPERATION-10. Ingénieur de connaissances entre et vérifie les schémas et les règles OWL

Titre	L'ingénieur de connaissances entre et vérifie les schémas et les règles OWL
But de la fonction	Le système doit permettre à l'ingénieur de connaissances d'entrer les schémas et les règles OWL dans le «**Dictionnaire ontologique**»
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les schémas et les règles d'OWL sont entrées et vérifiées

Séquence de réponse et stimulus

Ingénieur de connaissances	**Prototype de construction d'ontologie automatique**
1. Demander de créer d'un nouveau schéma ou règle d'OWL	
	2. Ouvrir et afficher le «**Dictionnaire ontologique**»
3. Ajouter un nouveau schéma ou une règle d'OWL	
	4. Demander un nom pour le nouveau schéma ou la nouvelle règle
5. Entrer un nom	
6. Valider le nouveau schéma ou la nouvelle règle	
	7. Ajouter un schéma ou une règle au «**Dictionnaire ontologique**» et afficher ses noms

OPERATION-11. L'Ingénieur de connaissances vérifie et autorise les procédures des auteurs et annotateurs de documents

Titre	L'ingénieur de connaissances vérifie et autorise les procédures des auteurs et des annotateurs de documents
But de la fonction	Les actions des auteurs et des annotateurs de documents doivent être vérifiées et autorisées par les ingénieurs de connaissances
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les actions des auteurs et des annotateurs sont autorisées

Séquence de réponse et stimulus

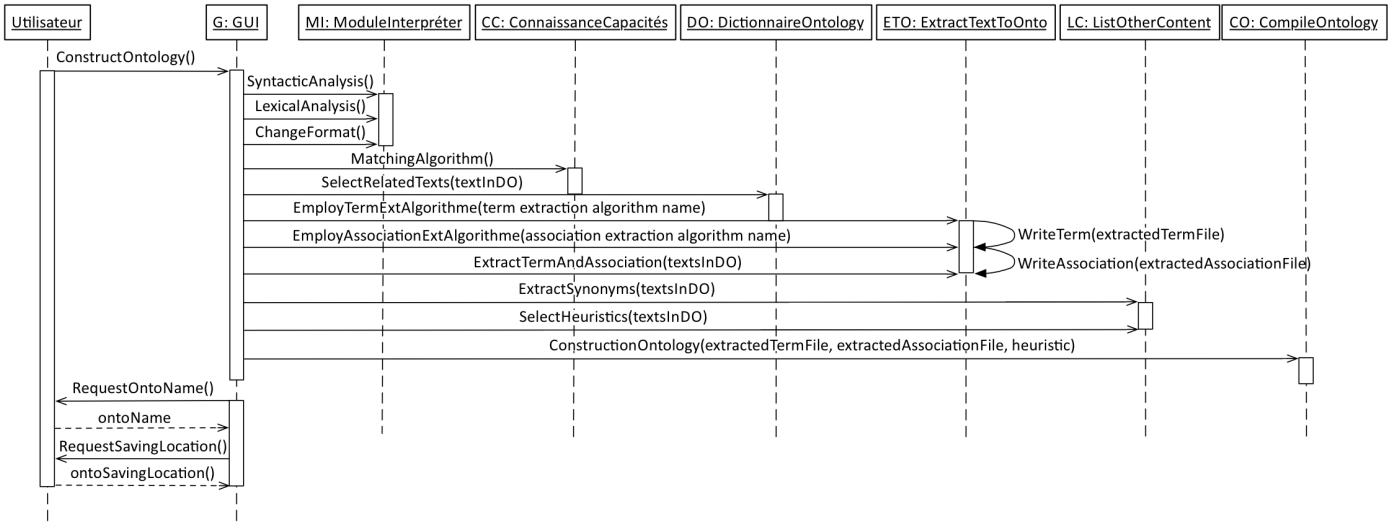
Ingénieur de connaissances	**Prototype de construction d'ontologie automatique**
1. Demander la présentation des actions à l'auteur ou à l'annotateur de documents	
	2. Afficher des actions d'auteur ou d'annotateurs de documents dans le module de «**Connaissances sur ses propres capacités »**
3. Demander d'ajouter / supprimer / modifier une action	
	4. Permettre le fonctionnement
5. Éditer les détails d'une action (les concepts, les associations et les synonymes ajoutés, heuristiques sélectionnées)	
	6. Mettre à jour les modules avec les nouvelles actions d'auteur ou d'annotateurs de documents

OPERATION-12. Construction des ontologies

Titre	Construction des ontologies
But de la fonction	Le système construit une ontologie semi-automatique à partir des termes et associations extraits, synonymes entrée et heuristiques sélectionnées
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Le but de la construction de l'ontologie c'est le semi-automatisme

Séquence de réponse et stimulus

Ingénieur de connaissances	Prototype de construction d'ontologie automatique
1. L'ingénieur de connaissances démarre le « **Modèle de l'Ontologie** » pour la construction de l'ontologie	
	2. Demander de choisir un emplacement pour enregistrer la nouvelle ontologie dans le « **Dictionnaire ontologique** »
3. Sélectionner un emplacement pour enregistrer l'ontologie	
	4. Demander d'entrer un nom dans la nouvelle ontologie
	5. Construire l'ontologie avec – Les termes des OPERATION-2 and 4, – Les synonymes des OPERATION-5 and 6, – Les associations des OPERATION-3 and 7, – Les heuristiques d'OPERATION-9, – Les schémas et les règles OWL d'OPER-10.
6. Amélioration et l'Autorisation de **l'ingénieur de connaissances**	
	7. Après avoir établi une ontologie avec les termes, les associations et les synonymes, le « **Modèle de l'Ontologie** » l'enregistre avec son nom.



Annexe figure 8 Diagramme séquence : Construction d'ontologies

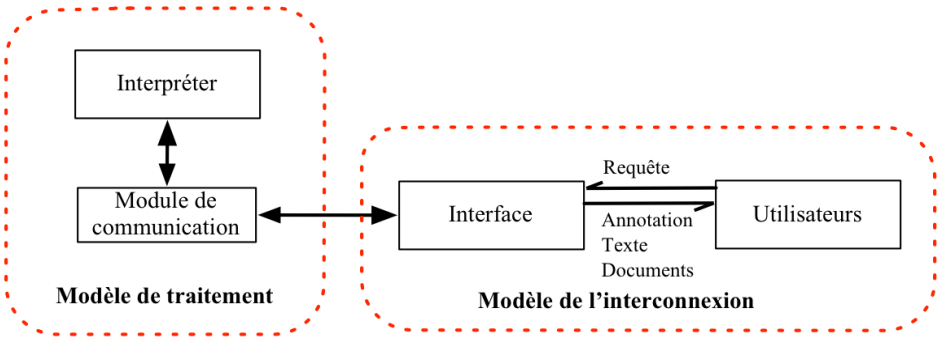
OPERATION-13. Mettre à jour le « Modèle de l'Ontologie »

Titre	**Mettre à jour le « Modèle de l'Ontologie »**
But de la fonction	Notre prototype doit permettre la modification des systèmes multi-agents et la construction d'un nouveau module d'ontologie. Les changements incluent d'ajouter, supprimer ou mettre à jour les concepts, les synonymes, ou les associations.
Les formats de données	Les données doivent inclure les concepts, les associations, les synonymes ajoutés au module pendant le processus de construction.
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Description de la fonctionnalité	L'ingénieur de connaissances ou l'utilisateur démarre l'environnement de Java et ouvre le « Modèle de l'Ontologie ». La liste des noms des modules dans les agents est ainsi présentée. L'ingénieur de connaissances ou l'utilisateur choisit l'agent(s) qu'il veut modifier et valide son choix. Par conséquent, tous les agents et les modules sélectionnés apparaissent dans l'environnement Java et l'ingénieur de connaissances ou l'utilisateur peut modifier, selon les facilités offertes par Java. Lorsque toutes les modifications ont été apportées sur le «Modèle de l'Ontologie», l'ingénieur de connaissances ou l'utilisateur peut enregistrer les modifications dans l'environnement Java.
Message de fin	Le module de l'ontologie est mis à jour avec succès

Séquence de réponse et stimulus

Ingénieur de connaissances / Utilisateur	**Prototype de construction d'ontologie automatique**
1. L'ingénieur de connaissances ou l'utilisateur démarre l'environnement de Java et ouvre le «**Modèle de l'Ontologie**»	
	2. Lister des noms des modules des agents - « **Connaissances sur ses propres capacités** » - « **Dictionnaire ontologique** »
3. Sélectionner le module pour l'édition d'ontologies	
4. Valider la sélection	
	5. Afficher les termes et les associations du «**Modèle de l'Ontologie** »
6. Éditer les associations, les termes et les synonymes du « **Modèle de l'Ontologie** »	
7. Demander d'enregistrer le « **Modèle de l'Ontologie** » mise à jour	
	8. Demander la confirmation d'enregistrement
9. Confirmer l'enregistrement	
	10. Mettre à jour des ontologies dans le «**Modèle de l'Ontologie**»

1.3 Opération entre le Modèle de Traitement et le Modèle de l'Interconnexion

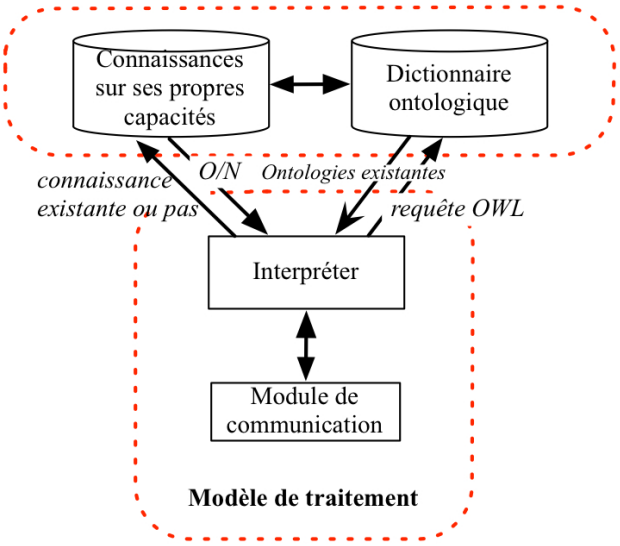


Annexe figure 9 le Modèle de Traitement et le Modèle de l'Interconnexion

OPERATION-14. Change le format de requête à OWL pour «[l']Interpréter»

Système	**Le module « Interpréter »**
1. Le système demande de passer le format de la requête à OWL, c'est réalisé dans le module «**Interpréter**»	
	2. Les concepts et les relations, analysés et filtrés par les processus « [l']analyse syntaxique» et « [l']analyse lexicale» sont classées en catégories «owlTerm» et «owlProperty» respectivement.
	3. Les «owlTerm» et «owlProperty» sélectionnés correspondent à «ontoClass» et «ontoProperty»
	4. «ontoClass» est suivi par «termLabel» qui contient le résultat des marquages de sens.
	5. « ontoProperty » est suivi par «associationLabel» qui contient les caractéristiques linguistiques
	6. Le «owlModel», est le résultat de ChangeFormat(), présente le résultat avec les termes et les associations en format owl.

1.4 Opération entre le Modèle de l'Ontologie et le Modèle de Traitement



Annexe figure 10 le Modèle de l'Ontologie et le Modèle de Traitement

OPERATION-15. Transfert de données entre le «Modèle de l'Ontologie» et le «Modèle de Traitement»

Titre	Transfert de données entre le « **Modèle de l'Ontologie** » et le « **Modèle de Traitement** »
But de la fonction	Le « **Modèle de Traitement** » doit être capable d'extraire des textes du «**Dictionnaire ontologique**» à partir des mots-clés et des relations des requêtes.
Stabilité	Stable
La plage valide de précision	Nul
Unités de mesure	Nul
Format des commandes	Nul
Formats d'écran	Nul
Degré de nécessité	Essentiel
Message de fin	Les textes sont extraits du « **Modèle de l'Ontologie** » et du «**Modèle de Traitement**»

Séquence de réponse et stimulus

Système	**Modèle de l'Ontologie et Modèle de Traitement**
1. Demander de transférer des textes du «**Modèle de l'Ontologie**» au «**Modèle de Traitement**» à partir des mots-clés et des relations des requêtes de l'utilisateur	
2. Demander l'ouverture des textes avec «**Modèle de l'Ontologie**»	
	3. Lister des noms des modules des agents - «**Connaissances sur ses propres capacités**» - « **Dictionnaire ontologique** »
4. Sélectionner le dossier correspondant à l'emplacement d'un fichier texte	
5. Sélectionner les fichiers texte correspondant au corpus de texte	
6. Confirmer la sélection du corpus de texte	
	7. Déclencher le «Programmation.4 Algorithme de l'alignement» pour extraire les textes correspondant au «**Dictionnaire ontologique**» à partir des mots clés et des relations des requête de l'utilisateur
	8. Transférer les textes au modèle « **Interface** » du « **Modèle de l'Interconnexion** » pour présenter les résultats de l'extraction en fonction de sa pertinence et de sa fréquence avec le «Programmation.10 Compteur de mots».

AUTRES OPERATIONS

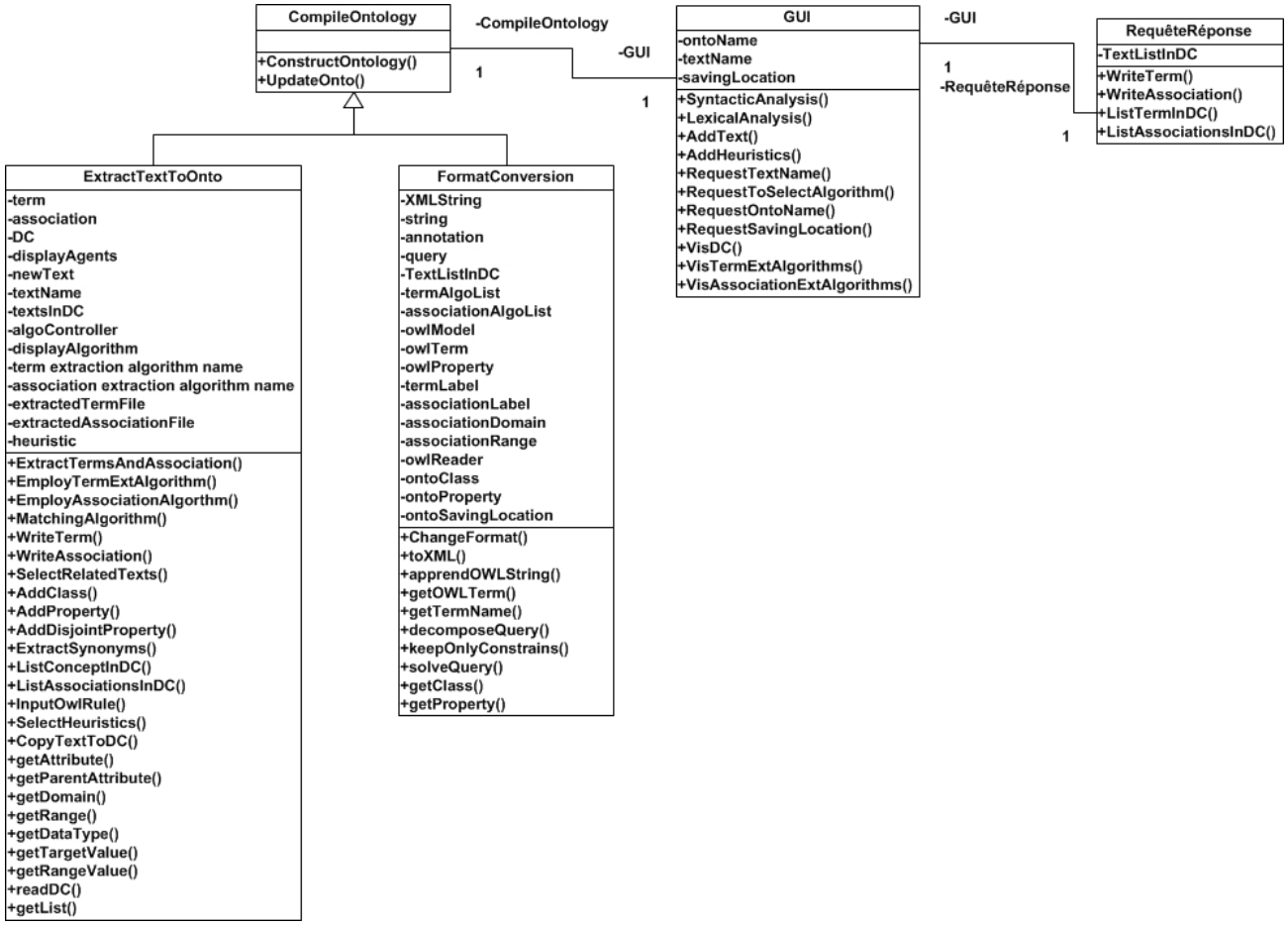
OPERATION-16. Exigence sur la maintenabilité du système de prototype

Le prototype de la construction de l'ontologie automatique doit pouvoir être agrandie en branchant nouveaux composants. En outre, les fonctionnalités du prototype doivent être adaptables avec un minimum d'effort de programmation. Tous les codes mis en œuvre pour le système de prototype doivent être écrits en Java.

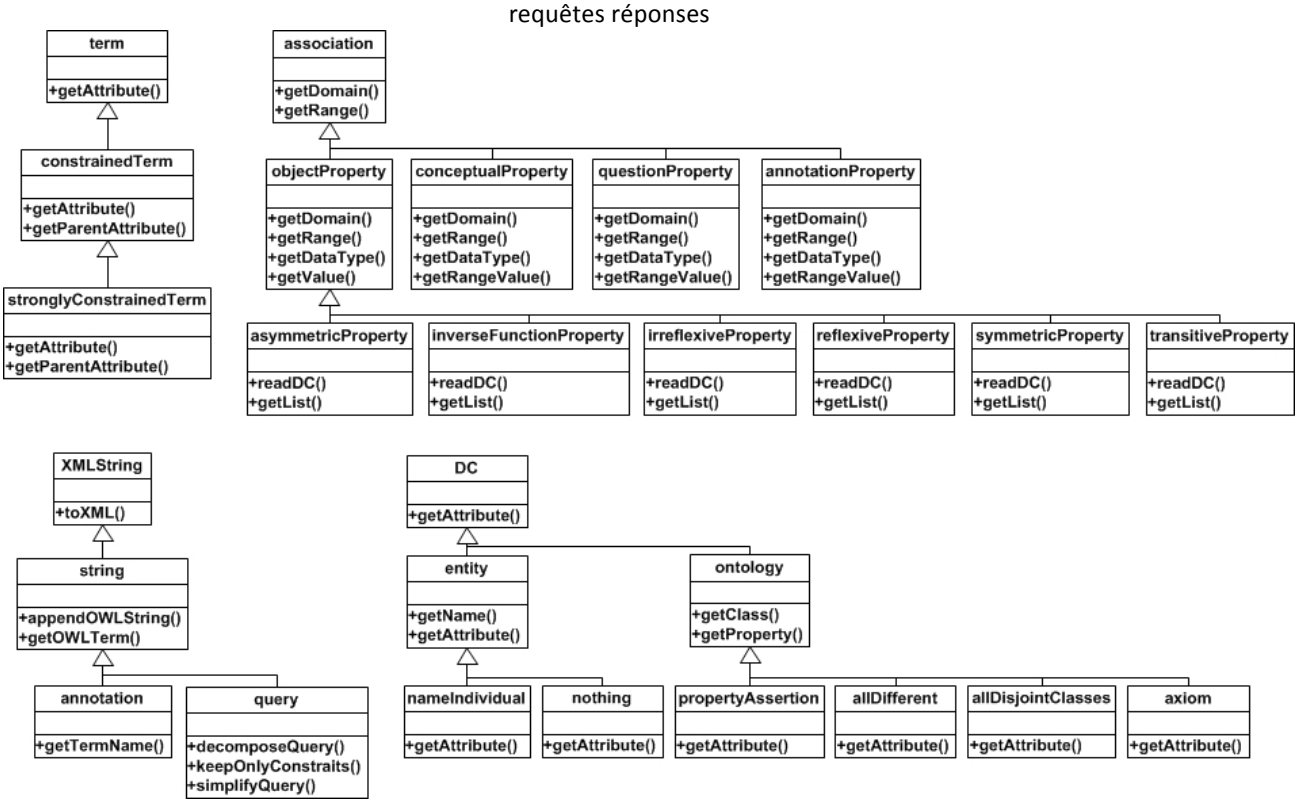
OPERATION-17. Description de GUI (*Graphical User Interface*)

L'interface utilisateur doit être facile à utiliser et satisfaire toutes les exigences spécifiées dans cette thèse. L'utilisateur doit pouvoir utiliser toutes les fonctionnalités du prototype de construction d'ontologies automatiquement, à travers l'interface graphique.

Annexe 2. Interface



Annexe figure 11 Diagramme de classe: Système prototype pour la construction d'ontologies et les requêtes réponses



Annexe figure 12 Diagramme de classe : variables et interfaces des requêtes

Interface-1. Modèle de l'ontologie

VARIABLES - ExtractTextToOnto

Variables	Sémantique des données
displayAlgorithm	Afficher tous les programmes et algorithmes disponibles : les utilisateurs peuvent effectuer des sélections.
term extraction algorithm name	Représenter le nom de l'algorithme utilisé pour l'extraction de termes.
association extraction algorithm name	Représenter le nom de l'algorithme utilisé pour l'extraction d'associations.
extractedTermFile	Présenter une liste contenant tous les termes extraits du corpus de texte.
extractedAssociationFile	Présenter une liste contenant toutes les associations extraites du corpus de texte.
heuristic	Heuristiques sélectionnées par l'ingénieur de connaissances pour la construction d'ontologies

INTERFACE-1.1. Extraction des termes et associations

Les interfaces travaillées dans le «**Modèle de l'ontologie**» permettent d'extraire des termes et des associations à partir d'un corpus de textes. Les fonctionnalités de l'interface sont appelées par le module GUI lors de la construction d'une nouvelle ontologie.

Interface	Fonction
ExtractTermsAndAssociation()	Extraire les termes et les associations des termes du corpus «**Dictionnaire ontologique**».
getAttribute()	Obtenir des attributs de la classe cible dans la hiérarchie
getParentAttribute()	Obtenir des attributs de ses classes parentes dans la hiérarchie
getDomain()	Demander et obtenir le « domaine » de la propriété cible
getRange()	Demander et obtenir le « range » de la propriété ciblée
getDataType()	Obtenir le « type de données » pour analyser la propriété cible
getValue()	Obtenir la valeur de la propriété de l'objet
getRangeValue()	Obtenir la valeur du « range » de la propriété cible
readDO()	L'accès à toutes les données et les textes du corpus «**Dictionnaire ontologique**».
getList()	Demander au corpus d'afficher toutes les donnés, les commentaires et les notes explicatives de la propriété cible.
EmployTermExtAlgorithm()	Employer l'algorithme d'extraction de termes pour extraire les termes du corpus de texte. Les programmations et algorithmes disponibles : Programmation.1 Analyseur morphologique-

	lemmatisation Programmation.2 Synonyme Programmation.3 POS Tagger Programmation.4 Algorithme de l'alignement Programmation.8 Les relations remarquables Programmation.9 Extraire des relations des mots cibles Programmation.10 Compteur de mots
EmployAssociationAlgorthm()	Employer les algorithmes d'extraction d'associations pour extraire les associations du corpus de texte. Les programmations et algorithmes disponibles : Programmation.2 Synonyme Programmation.3 POS Tagger Programmation.4 Algorithme de l'alignement Programmation.8 Les relations remarquables Programmation.9 Extraire des relations des mots cibles Programmation.10 Compteur de mots
MatchingAlgorithm()	Déclencher la Programmation.4 Algorithme de l'alignement
SelectRelatedTexts()	Sélectionner textes relatifs selon les termes et les associations extraits.

INTERFACE-1.2. Ajouter d'autres éléments nécessaires

Ces interfaces servent à ajouter des termes et des associations au « Dictionnaire ontologique », et à ajouter les synonymes aux concepts DC en utilisant les facilités Protégé-OWL. En outre, les interfaces sont responsables de l'enregistrement des termes et des associations DC en fichiers texte.

Interface	**Fonction**
AddClass()	Ajouter un terme au corpus de textes «**Dictionnaire ontologique**».
AddProperty()	Ajouter une association entre deux termes de l'ontologie.
AddDisjointProperty()	Ajouter une propriété disjointe parmi un ensemble de termes de l'ontologie.
ExtractSynonyms()	Extraire ou ajouter des synonymes au «**Dictionnaire ontologique**».
InputOwlRule()	L'utilisateur, l'auteur ou l'ingénieur de connaissances entrent les règles et les schémas d'OWL
SelectHeuristics()	Sélectionner les heuristiques pour la construction d'ontologies
CopyTextToDO()	Copier les textes sélectionnés par le «**Dictionnaire ontologique**»

Interface-2. Le Modèle de Traitement

VARIABLES-2. FormatConversion

Variables	Sémantique des données
XMLString	Le «string» en forme XML
string	Séquence de symboles ou de chiffres dans la programmation informatique
annotation	Les annotations sont les métadonnées (par exemple, un commentaire, une explication) attachées au texte, à l'image, etc. Elles font également référence à une partie spécifique des données d'origine.
query	Une requête précise pour la récupération de l'information des systèmes informatiques et des bases de données
TextListInDO	Une liste qui contient le fichier texte dans le corpus «**Dictionnaire ontologique**»
termAlgoList	Une liste qui contient les algorithmes d'extraction de termes à partir d'un corpus de texte.
associationAlgoList	Une liste qui contient les algorithmes d'extraction d'associations à partir d'un corpus de texte.
owlModel	Représente le modèle OWL contenant les termes et les associations.
owlReader	Représente le modèle OWL utilisé pour lire le contenu de « **Dictionnaire ontologique** ».
owlTerm	Représente un terme dans le modèle OWL.
owlProperty	Représente une association dans le modèle OWL.
ontoClass	Représente une liste contenant la classe de l'ontologie.
ontoProperty	Représente une liste contenant l'association de «**Dictionnaire ontologique**».
termLabel	Représente le label d'un terme pour l'ajouter à la nouvelle ontologie.
associationLabel	Représente le label d'une association pour l'ajouter à la nouvelle ontologie
associationDomain	Représente le label de domaine d'une association pour l'ajouter à la nouvelle ontologie.
associationRange	Représente le label de range d'une association pour l'ajouter à la nouvelle ontologie.

INTERFACE-2. Conversion format de texte à OWL

Interface	Fonction
ChangeFormat()	Passe le format de requête ou le texte à OWL
toXML()	Passe le format de string à XML
apprendOWLString()	Apprend les titres, les notes explicatives et tous les autres aspects du string OWL
getOWLTerm()	Extrait le terme du string OWL
getTermName()	Obtient le nom du terme dans l'annotation
decomposeQuery()	Décompose la requête entrée
keepOnlyConstrains()	Conserve uniquement les « Constrains words » de requête entrée
simplifyQuery()	Ne garde que les termes et les relations utiles de la requête entrée
getClass()	Obtient la classe de l'ontologie
getProperty()	Obtient la propriété de l'ontologie

INTERFACE-3. Modèle de l'Interconnexion – GUI

VARIABLES-3. GUI

Variables	Sémantique des données
ontoName	L'utilisateur donne un nom à la nouvelle ontologie.
textName	Représente le nom du texte pour confirmer son emplacement sur le disque dur ou dans le «**Dictionnaire ontologique** ».
savingLocation	L'utilisateur décide et répond de l'emplacement de l'enregistrement pour le nouvel élément (terme, association, ontologie...)

INTERFACE-3. Interconnexion entre l'utilisateur et les modules de SMA

L'interface utilisateur graphique (GUI) permet à l'utilisateur d'interagir avec les fonctions des modules mentionnés, et de spécifier les valeurs de paramètres différentes.

Interface	Fonction
SyntacticAnalysis()	Déclenche la «Programmation.3 POS Tagger» pour analyses syntactique.
LexicalAnalysis()	Déclenche la «Programmation.1 Analyseur morphologique » et la «Programmation.2 Synonyme» pour l'analyse lexicale.
AddText()	Ajoute le fichier(s) texte au corpus de textes.
AddHeuristics()	Sélectionne et ajoute les heuristiques pour la construction d'ontologies.
RequestTextName()	Demande à l'utilisateur de nommer le nouveau texte entré.
RequestToSelectAlgorithm()	Demande à l'utilisateur de sélectionner l'algorithme.
RequestOntoName()	Demande à l'utilisateur de nommer la nouvelle ontologie établie avant de l'enregistrer.
RequestSavingLocation()	Demande l'emplacement de l'enregistrement de la nouvelle ontologie établie ou du nouveau texte entré.
VisDO()	Permet de visualiser le contenu du corpus de texte (« Dictionnaire ontologique »).
VisTermExtAlgorithms()	Permet de visualiser les algorithmes du processus d'extraction des termes.
VisAssociationExtAlgorithms()	Permet de visualiser les algorithmes du processus d'extraction des associations.

INTERFACE-4. Construction de l'ontologie

Ce module fournit des méthodes pour ajouter des termes et des associations à l'ontologie générée. Les fonctionnalités du module sont appelées par module GUI lors de la construction d'une nouvelle ontologie.

Interface	Fonction
ConstructOntology()	Démarre le processus de construction d'ontologies pour l'établir avec la liste des termes et associations appariés, les heuristiques sélectionnées et les schémas et règles OWL entrés. L'ontologie construite est enregistrée dans un fichier OWL.
UpdateOnto()	Initialise tous les paramètres à mettre à jour pour l'ontologie existante

INTERFACE-5. Requête réponse

Interface	**Fonction**
WriteTerm()	Présente le terme(s) extrait par GUI.
WriteAssociation()	Présente l'association(s) extraite par GUI.
ListTermInDO()	Récupère les labels de tous les termes du « **Dictionnaire ontologique** » pour les écrire dans un nouveau fichier texte.
ListAssociationsInDO()	Récupère les labels de toutes les associations, ses domaines et « ranges » dans le «**Dictionnaire ontologique**» et les écrire dans un nouveau fichier texte.

Annexe 3. Programmation

PACKAGE-1. Morphologie

Programmation-1. Analyseur morphologique-Lemmatisation

```
package com.smadd;

import java.util.*;
import com.smadd.Dictionnaire;


public class Morphologie {

   public static String[][] getCombinaisons(List<Dictionnaire> etiquettes)
   {
     int compt=1;
     int nbreMots=etiquettes.size();
     for(Dictionnaire tags:etiquettes)
     {
         compt*= tags.size();
     }


     String[][] matrice = new String[compt][nbreMots];

     for(int i=0;i<matrice.length-1;i++)
         for(int j=0;j<matrice[i].length-1;j++)
         {
             List<?> et= (List<?>) etiquettes.get(j);
             int l= et.size();
             System.out.println(l);
             System.out.println(et);
             matrice[i][j]= (String) et.get((int)(i/l));
         }

     return matrice;
  }


   public static void main(String[] args) throws Exception
   {
     Syntagme phrase= new Syntagme("Le cas typique est illustré pour expliquer
la présentation d'une maladie incluant diagnostic, pronostic et traitement
d'un cancer gastrique.");
     Segmenteur seg= Segmentation.getInstance();
     List<Syntagme> syntagmes=seg.getMots(phrase);
     System.out.println("Découpage : "+syntagmes);

     Dictionnaire dico= Dictionnaire.getInstance();

     Collection<String> etiquettes;
     List<Dictionnaire> lemmes= new Vector<Dictionnaire>();
     for(Syntagme mot:syntagmes)
     {
         etiquettes= new Vector<String>();
         etiquettes.addAll(dico.getTags(mot.toString()));
         lemmes.add((Dictionnaire) etiquettes);
     }
     System.out.println("Lemmatisation : "+lemmes);

  }

}
```

Programmation 1 L'analyseur morphologique-Lemmatisation

```
package com.SMAAD;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

public class Segmentation extends Segmenteur {

   protected Collection<Segmenteur> segmentations;

   public static Segmentation instance;

   private List<Syntagme> temp;

   private Segmentation() {
     segmentations = new ArrayList<Segmenteur>();
     segmentations.add(new Segmentation1());
     segmentations.add(new Segmentation2());
  }

   public static Segmentation getInstance() {
     if (instance == null)
         instance = new Segmentation();
     return instance;
   }

   public List<Syntagme> getMots(Syntagme s) {

     List<Syntagme> avant = new ArrayList<Syntagme>();
     List<Syntagme> apres = new ArrayList<Syntagme>();
     setTemp(new ArrayList<Syntagme>());

     avant.add(s);

     for (Segmenteur segmentation : segmentations) {
         for (Syntagme current : avant) {
             setTemp(segmentation.getMots(current));
         }
         avant.clear();
         avant.addAll(apres);
         apres.clear();
     }
     return avant;
   }

   public List<Syntagme> getTemp() {
     return temp;
   }

   public void setTemp(List<Syntagme> temp) {
     this.temp = temp;
   }

}
```

Programmation 1.1 Segmentation

```
package com.SMAAD;

import java.util.List;

public class Segmentation1 extends Segmenteur {

  @Override
   public List<Syntagme> getMots(Syntagme texte) {
     // TODO Auto-generated method stub
     return null;
   }

}
```

Programmation 1.2 Segmentation1

```
package com.SMAAD;

import java.util.List;

public class Segmentation2 extends Segmenteur {

  @Override
   public List<Syntagme> getMots(Syntagme texte) {
     // TODO Auto-generated method stub
     return null;
   }

}
```

Programmation 1.3 Segmentation2

```
package com.SMAAD;

import java.util.List;

public abstract class Segmenteur {

   public abstract List<Syntagme> getMots(Syntagme phrase);
}
```

Programmation 1.4 Segmenteur

PACKAGE-2. TaggedToken

Programmation-2. Synonyme

```
package com.smadd;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
import java.util.StringTokenizer;


public class Synonyms {

   private static final Map<String, Set<Set<String>>> WORD_TO_SYNOYMYM_SETS
= new HashMap<String, Set<Set<String>>>(15000);
   private static final Set<Set<String>> EMPTY_SET =
Collections.unmodifiableSet(new HashSet<Set<String>>());

//   static {
//      synchronized (WORD_TO_SYNOYMYM_SETS) {
//         String projectDir = System.getProperty("user.dir");
//         load(projectDir + "com/WordNet-3.0/dict/data.adj");
//         load(projectDir + "com/WordNet-3.0/dict/data.adv");
//         load(projectDir + "com/WordNet-3.0/dict/data.noun");
//         load(projectDir + "com/WordNet-3.0/dict/data.verb");
//      }
//   }

   public static void load(String[] paths) {
      for (String path : paths) {
         load(path);
      }
   }

   private static void load(String path) {

      InputStream inputStream = null;
      BufferedReader reader = null;
      try {
         inputStream = new FileInputStream(path);
         reader = new BufferedReader(new InputStreamReader(inputStream));

         String line = null;
         while ((line = reader.readLine()) != null) {
//            System.out.println(line);
            processLine(line);
         }
      } catch (IOException ioe) {
         throw new RuntimeException(ioe);
      } finally {
         if (reader != null) {
            try {
               reader.close();
            } catch (IOException ioe) {
            }
         }

         if (inputStream != null) {
            try {
               inputStream.close();
            } catch (IOException ioe) {
            }
         }
      }
```

```
   }

   private static void processLine(String line) {

    if ((line.length() > 17)){
//      if ((line.length() > 17) && (line.charAt(0) == '0')) {
         line = line.substring(17);

         Set<String> synonymSet = new HashSet<String>();
         StringTokenizer st = new StringTokenizer(line, " ");
         while (st.hasMoreElements()) {
            String token = st.nextToken();

            if (token.startsWith("00")) {
               break;
            }

            if (token.length() > 2 && Character.isLetter(token.charAt(0))) {
                System.out.println("word:  " + token);
               synonymSet.add(token);
            }
         }

         if (synonymSet.size() > 1) {
            synonymSet = Collections.unmodifiableSet(synonymSet);

            for (String word : synonymSet) {
               Set<Set<String>> synonymSetsForThisWord =
WORD_TO_SYNOYMYM_SETS.get(word);
               if (synonymSetsForThisWord == null) {
                  synonymSetsForThisWord = new HashSet<Set<String>>();
                  WORD_TO_SYNOYMYM_SETS.put(word, synonymSetsForThisWord);
               }
               synonymSetsForThisWord.add(synonymSet);
            }
         }
      }
   }

   public static Set<Set<String>> getSynonymSets(String word) {
      synchronized (WORD_TO_SYNOYMYM_SETS) {
         Set<Set<String>> synonymSets = WORD_TO_SYNOYMYM_SETS.get(word);
         return ((synonymSets != null) ?
Collections.unmodifiableSet(synonymSets) : EMPTY_SET);
      }
   }

   public static void main(String[] args) {
    load("com/smadd/corpus.txt");
   }
}
```

Programmation 2 Synonyme

Programmation-3. POS Tagger

```
package com.smadd;

import java.io.*;
import java.util.List;
import java.util.Set;


public class TaggedToken {

    private static String token;
    private Tag tag;

    public class Tag implements Serializable {

 private static final long serialVersionUID = 1L;
 private String name;
 private int pictureCount;
 public Tag(String name, int pictureCount) {
     this.name = name;
     this.pictureCount = pictureCount;
 }

 public String getName() {
     return this.name;
 }

 public int getPictureCount() {
     return this.pictureCount;
 }
    }


    public TaggedToken(String token) {
        super();
        TaggedToken.token = token;
        this.tag = null;
    }

    public TaggedToken(String token, Tag tag)
    {
        super();
        TaggedToken.token = token;
        this.tag = tag;
    }

    public String getToken()
    {
        return token;
    }
    public void setToken(String token)
    {
        TaggedToken.token = token;
    }
    public Tag getTag()
    {
        return tag;
    }
    public void setTag(Tag tag)
    {
        this.tag = tag;
    }

    @Override
    public String toString()
    {
     return token + " (" + tag + ")";
    }

    @Override
    public int hashCode()
    {
        final int prime = 31;
```

```
        int result = 1;
        result = prime * result + ((tag == null) ? 0 : tag.hashCode());
        result = prime * result + ((token == null) ? 0 : token.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj)
    {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        TaggedToken other = (TaggedToken) obj;
        if (tag == null) {
            if (other.tag != null) {
                return false;
            }
        }
        else if (!tag.equals(other.tag)) {
            return false;
        }
        if (token == null) {
            if (TaggedToken.token != null) {
                return false;
            }
        }
        else if (!token.equals(TaggedToken.token)) {
            return false;
        }
        return true;
    }


   public interface PredicateArgument {
      public String getFunction();
      public boolean isOptional();
      public Set<String> getRealisations();
      public int getIndex();
   }

   public interface LexicalEntry {
      public String getWord();
      public String getLemma();
      public String getLemmaComplement();
      public String getCategory();
      public String getSubCategory();
      public String getPredicate();
      public List<PredicateArgument> getPredicateArguments();
      public PredicateArgument getPredicateArgument(String functionName);
      public Set<String> getPredicateMacros();
      public List<String> getGender();
      public List<String> getNumber();
      public List<String> getTense();
      public List<String> getPerson();
      public List<String> getPossessorNumber();
      public String getMorphology();
      public String getMorphologyForCoNLL();
      public LexicalEntry getLexicalEntry();
      public List<LexicalEntry> getLexicalEntries();
   }

   public static void main(String[] args) throws Exception {
    TaggedToken taggedToken = new TaggedToken("Gender");
    taggedToken.setTag(taggedToken.new Tag("femail", 1));
    System.out.println("word : "+token);
    }
}
```

Programmation 3 POS Tagger

PACKAGE-3. Corpus

Programmation-4. Algorithme de l'alignement

```
package com.SMAAD;

public class Concept {
   private String concept;
   public Concept(String concept) {
      this.concept = concept;
   }

   public String getConcept() {
      return this.concept;
   }

   public void setConcept(String concept) {
      this.concept = concept;
   }

   public boolean equals(Object object) {
      return (object instanceof Concept ? ((Concept)
object).concept.equals(this.concept)
            : this.concept.equals(object.toString()));
   }

   public int hashCode() {
      return this.concept.hashCode();
   }

   public String toString() {
      return this.concept;
   }
}
```

Programmation 4 Algorithme de l'alignement

Programmation-5. Dictionnaire

```
package com.SMAAD;

import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.util.Collection;


public class Dictionnaire {
 private static RandomAccessFile randomAccessFile;

 public static void main(String[] args) throws Exception {
     int bufSize = 1024;
     byte[] bs = new byte[bufSize];
     ByteBuffer byteBuf = ByteBuffer.allocate(1024);
     randomAccessFile = new
RandomAccessFile("/Users/yingshen/Desktop/corpus.txt","r");
     FileChannel channel = randomAccessFile.getChannel();
     while(channel.read(byteBuf) != -1) {
         int size = byteBuf.position();
         byteBuf.rewind();
         byteBuf.get(bs);
         System.out.print(new String(bs, 0, size));
         byteBuf.clear();
     }
 }

    public static Dictionnaire getInstance() {
 // TODO Auto-generated method stub
        return null;
    }

    public Collection<? extends String> getTags(String string) {
 // TODO Auto-generated method stub
        return null;
    }

 public int size() {
     // TODO Auto-generated method stub
     return 0;
 }

}
```

Programmation 5 Dictionnaire

Programmation-6. Hierarchy

```
package com.SMAAD;

import java.util.*;

public interface Hierarchy {

    Set <LogicalExpression> listConcept();

    void addConcept(LogicalExpression concept);

    void removeConcept(LogicalExpression le);

    abstract class LogicalExpression {

      public abstract String toString();

      public abstract int hashCode();

         public boolean isComparator() {
           return false;
         }
    }
}
```

Programmation 6 Hierarchy

Programmation-7. Instance

```
package com.SMAAD;

import java.util.*;

public interface Instance {

    Set <Concept> listConcepts();

    void addConcept(Concept memberOf);

    void removeConcept(Concept memberOf);

    void removeAttributeValues(Identifier id);

}
```

Programmation 7 Instance

PACKAGE-4. ExtractRelationForWord

Programmation-8. Les relations remarquables – RelationExample

```
package com.SMAAD;

import java.util.HashSet;
import java.util.Set;

// Cette programmation vise à ajouter les relations remarquables
// à la classe RelationExample


public class RelationExample {
   Set<String> unaryLabels = new HashSet<String>() {

      private static final long serialVersionUID = 1L;

      {
         add("est-un");
         add("a-un");
         add("est-partie-de");
         add("est-composé-de");
         add("est-Propriétaire");
         add("et");
         add("est-synonyme");
         add("lié-fonctionnellement-à");
         add("lié-conceptuellement-à");
         add("précède");
         add("suit");
         add("est-dessus");
         add("est-dessous");
         add("est-à-proximité-de");
         add("est-situé");
         add("est-associé-à");
         add("est-voisin-de");
         add("produire");
         add("provoquer");
         add("causer");
         add("évoquer");
         add("acheter");
         add("vendre");
         add("prendre");
      }
   };

   String label;

   boolean binary;

   String word1;

   String word2;

   public RelationExample(String word1, String word2, String label) {
      this.word1 = word1.trim();
      this.word2 = word2.trim();
      this.label = label.trim();
   }

   public RelationExample(String word1, String type, String label,
      boolean binary) {
      this.word1 = word1.trim();
      this.word2 = type.trim();
      this.label = label.trim();
      this.binary = binary;
   }

   public String getLabel() {
      return label;
   }
```

```
   public void setLabel(String label) {
      this.label = label;
   }

   public boolean isBinary() {
      if (unaryLabels.contains(this.label))
         return false;
      return true;
   }

   public boolean getBinary(){
      return binary;
   }

   public void setBinary(boolean binary) {
      this.binary = binary;
   }

   public String getWord1() {
      return word1;
   }

   public void setWord1(String word1) {
      this.word1 = word1.trim();
   }

   public String getWord2() {
      return word2;
   }

   public void setWord2(String word2) {
      this.word2 = word2.trim();
   }

   public String toString() {
      String relationExample = "";
      relationExample += this.getLabel() + "(";
      relationExample += this.getWord1() + ",";
      relationExample += this.getWord2() + ")";
      return relationExample;
   }
}
```

Programmation 8 Les relations remarquables - RelationExample

Programmation-9. Extraire des relations des mots cibles

```
package com.smadd;

import java.util.HashSet;
import java.util.LinkedList;
import java.util.Set;
import java.util.Vector;
import java.util.List;


public class ExtractRelationForWord {

   public class RelationExample {
      Set<String> unaryLabels = new HashSet<String>() {

         private static final long serialVersionUID = 1L;

         {
            add("est-un");
            add("a-un");
            add("est-partie-de");
            add("est-composé-de");
            add("est-Propriétaire");
            add("et");
            add("est-synonyme");
            add("lié-fonctionnellement-à");
            add("lié-conceptuellement-à");
            add("précède");
            add("suit");
            add("est-dessus");
            add("est-dessous");
            add("est-à-proximité-de");
            add("est-situé");
            add("est-associé-à");
            add("est-voisin-de");
            add("produire");
            add("provoquer");
            add("causer");
            add("évoquer");
            add("acheter");
            add("vendre");
            add("prendre");
         }
      };

      String label;

      boolean binary;

      String word1;

      String word2;

      public RelationExample(String word1, String word2, String label) {
         this.word1 = word1.trim();
         this.word2 = word2.trim();
         this.label = label.trim();
      }

      public RelationExample(String word1, String type, String label,
         boolean binary) {
         this.word1 = word1.trim();
         this.word2 = type.trim();
         this.label = label.trim();
         this.binary = binary;
      }

      public String getLabel() {
         return label;
      }

      public void setLabel(String label) {
         this.label = label;
```

```
      }

      public boolean isBinary() {
         if (unaryLabels.contains(this.label))
            return false;
         return true;
      }

      public boolean getBinary(){
         return binary;
      }

      public void setBinary(boolean binary) {
         this.binary = binary;
      }

      public String getWord1() {
         return word1;
      }

      public void setWord1(String word1) {
         this.word1 = word1.trim();
      }

      public String getWord2() {
         return word2;
      }

      public void setWord2(String word2) {
         this.word2 = word2.trim();
      }

      public String toString() {
         String relationExample = "";
         relationExample += this.getLabel() + "(";
         relationExample += this.getWord1() + ",";
         relationExample += this.getWord2() + ")";
         return relationExample;
      }
   }


   public static Set<String> Need2Change = new HashSet<String>() {
     private static final long serialVersionUID = 1L;
     {
         add("pos");
         add("noun_number");
         add("tense");
         add("gender");
         add("DEFINITE-FLAG");
         add("PRONOUN-FLAG");
         add("IDIOM-FLAG");
      }
   };


// extraire tous les mots qui forment la phrase
   public Vector<String> extractWord(Vector<String> allRelations) {
      Vector<String> allWords = new Vector<String>();
      for (String str : allRelations) {
         if (str.startsWith("pos")&& !str.contains("punctuation")) {
            allWords.add(str.substring(str.indexOf("(")+1, str.indexOf(",")));
         }
      }
      return allWords;
   }

// extraire le prédicat d'une phrase
   public String extractVerb(Vector<String> allRelations) {
      for (String str : allRelations) {
         if (str.startsWith("_subj")) {
            return str.substring(str.indexOf("("), str.indexOf(","));
         }
      }
      return null;
   }
```

```
// extraire les relations d'un mot cible
   public Vector<String> extract(Vector<String> allRelations, String word) {
       Vector<String> wordRelation = new Vector<String>();
       for (String str : allRelations) {
          if (str.contains(word)) {
             wordRelation.add(str);
          }
       }
       return wordRelation;
   }

// changer la relation du type «String» au type «RelationExample»
   public Vector<RelationExample> relationFormalization(
      Vector<String> relations) {
      Vector<RelationExample> rel = new Vector<RelationExample>();
      for (String str : relations) {
         int i = str.indexOf("(");
         int j = str.indexOf(",");
         int k = str.indexOf(")");
         if (!Need2Change.contains(str.substring(0, str.indexOf('')))) {
            RelationExample temp = new RelationExample(str.substring(i + 1,
               j), str.substring(j + 1, k), str.substring(0, i),true);
            rel.add(temp);
         } else {
            RelationExample temp = new RelationExample(str.substring(i + 1,
               j), str.substring(j + 1, k), str.substring(0, i), false);
            rel.add(temp);
         }
      }
      return rel;
   }


   public class WordNode {

      public String originalWord;
      public String word;
      public String pos;
      public boolean unaryProcessed;
      public List<RelationExample> unaryRelations;
      public List<RelationExample> binaryRelations;
      public boolean isUnaryProcessed() {
         return unaryProcessed;
      }

      public void setUnaryProcessed(boolean unaryProcessed) {
         this.unaryProcessed = unaryProcessed;
      }

      public WordNode() {

      }

      public WordNode(String word) {
         this.setOriginWord(word);
      }

      public String getOriginWord() {
         return originalWord;
      }

      public void setOriginWord(String originWord) {
         this.originalWord = originWord;
         this.word = originWord;
      }

      public String getWord() {
         return word;
      }

      public void setWord(String word) {
         this.word = word;
      }

      public String getPOS() {
         return pos;
      }
```

```
    public boolean containsRelation(String label) {
       return containsUnary(label) || containsBinary(label);
    }

    public boolean containsUnary(String label) {
       for (RelationExample re : unaryRelations) {
          if (re.getLabel().equals("pos") || re.getLabel().equals("tense")) {
             if (re.getWord2().equals(label)) {
                return true;
             }
          } else if (re.getLabel().equals(label)) {
             return true;
          }
       }
       return false;
    }

    public boolean containsBinary(String label) {
       for (RelationExample re : binaryRelations) {
          if (re.getLabel().equals(label)) {
             return true;
          }
       }
       return false;
    }

    public Object clone() {
       WordNode word = new WordNode(originalWord);
       word.getUnaryRelations().addAll(unaryRelations);
       word.getBinaryRelations().addAll(binaryRelations);
       return word;
    }

    public List<RelationExample> getUnaryRelations() {
       return unaryRelations;
    }

    public void setUnaryRelations(List<RelationExample> unaryRelations) {
       this.unaryRelations = unaryRelations;
    }

    public List<RelationExample> getBinaryRelations() {
       return binaryRelations;
    }

    public void setBinaryRelations(List<RelationExample> binaryRelations) {
       this.binaryRelations = binaryRelations;
    }

    public String toString() {
       String str = getWord() + " " + getPOS() + "\n";
       for (RelationExample temp : unaryRelations) {
          str += temp.toString() + "\n";
       }
       for (RelationExample temp : binaryRelations) {
          str += temp.toString() + "\n";
       }
       return str;
    }

    public void removeNumber() {
       if (originalWord.matches(".*_\\d+")) {
          word = originalWord.substring(0, originalWord.lastIndexOf("_"));
       }
}

    public WordNode extractWordNode(Vector<RelationExample> relations,
       String word) {
       WordNode wordNode = new WordNode(word);
       List<RelationExample> unaryRelations = new
  LinkedList<RelationExample>();
       List<RelationExample> binaryRelations = new
  LinkedList<RelationExample>();
       for (RelationExample temp : relations) {
          if (temp.isBinary() == false) {
             unaryRelations.add(temp);
```

```
            if (temp.getLabel().equals("pos")) {
               wordNode.pos = temp.getWord2();
            }
         } else {
            binaryRelations.add(temp);
         }
      }
      if (unaryRelations.size() > 1) {
         wordNode.unaryProcessed = true;
      }
      wordNode.binaryRelations=binaryRelations;
      wordNode.unaryRelations=unaryRelations;
      return wordNode;
   }
}

// changer la forme binaire de certaines relations à la forme unaire
   public static Vector<String> changeForm(Vector<String> original) {
      Vector<String> newForm = new Vector<String>();
      for (String temp : original) {
         if (!temp.equals("")) {
            if (Need2Change.contains(temp.substring(0, temp.indexOf('')))) {
               temp = temp.replace('(', ' ');
               temp = temp.replace(')', ' ');
               temp = temp.replace(",", "");
               String[] tempArray = temp.split(" ");
               if (tempArray.length == 4) {
                  temp = tempArray[3] + "(" + tempArray[0] + tempArray[2] + ")";
               } else {
                  temp = tempArray[2] + "(" + tempArray[1] + ")";
               }
            } else if (temp.contains("-FLAG")) {
               temp = temp.replace("-FLAG", "");
               temp = temp.substring(0, temp.indexOf("(")).toLowerCase()
                  + temp.substring(temp.indexOf("("), temp.indexOf(","))
                  + ")";
            }
            newForm.add(temp);
         } else {
            newForm.add(temp);
         }
      }
      return newForm;
   }



//Test
   public class Test {
      public void main(String [] args){
         new ExtractRelationForWord(); {
            System.out.println("test");
         }
      }
   }
}
```

Programmation 9 Extraire des relations des mots cibles

PACKAGE-5. WordCounter

Programmation-10. Compteur de mots

```
package com.SMAAD;

import java.util.*;
import java.io.*;

public class WordCounter
{
   private static final String RESOURCE = "resource";
   private static Scanner scanner;

   public static void main(String[ ] args)
   {
      TreeMap<String, Integer> frequencyData = new TreeMap<String, Integer>( );

      readWordFile(frequencyData);
      printAllCounts(frequencyData);
   }

   public static int getCount
   (String word, TreeMap<String, Integer> frequencyData)
   {
      if (frequencyData.containsKey(word))
      {
         return frequencyData.get(word);
      }
      else
      {
         return 0;
      }
   }

   public static void printAllCounts(TreeMap<String, Integer> frequencyData)
   {
      System.out.println("-----------------");
      System.out.println("Occurrences    Word");

      for(String word : frequencyData.keySet( ))
      {
         System.out.printf("%15d    %s\n", frequencyData.get(word), word);
      }

      System.out.println("-----------------");
   }

   public static void readWordFile(TreeMap<String, Integer> frequencyData)
   {
      Scanner wordFile;
      String word;
      Integer count;

      try {
         setScanner(wordFile = new Scanner(new
    FileReader("/Users/yingshen/Desktop/corpus.txt")));
      }
      catch (FileNotFoundException e) {
         System.err.println(e);
         return;
      }

      while (wordFile.hasNext( ))
      {
         word = wordFile.next( );

         count = getCount(word, frequencyData) + 1;
         frequencyData.put(word, count);
      }
   }

   public static Scanner getScanner() {
       return scanner;
```

```
    }

    public static void setScanner(Scanner scanner) {
        WordCounter.scanner = scanner;
    }

    public static String getResource() {
        return RESOURCE;
    }

}
```

Programmation 10 Compteur de mots

PACKAGE-6. BuildPathologieOntologie

Programmation-11. La construction d'ontologie pathologique

```
package java.SMAAD.BuildPathologieOntologie;

import java.io.*;
import java.util.*;

import java.SMAAD.BuildPathologieOntologie.*;


public class BuildPathologieOntologie {

   Lexicon lexicon = new Lexicon();
   Ontology ontology = new Ontology();
   Identifier identifier = new Identifier();


   class Identifier {
      private static final long serialVersionUID = 3688507692658669620L;
      protected String name;
      private Identifier() {
      name = null;
      }

      public Identifier(String s)  {
      name = s;
      }

      public String getName() {
      return name;
      }

      public void setName(String newName) {
      name = newName;
      }

      public Object clone() {
         try {
            return super.clone();
         }
         catch (CloneNotSupportedException e) {
            return null;
         }
      }

     public Identifier copy() {
         return  (Identifier) clone();
      }

      public String toString() {
         return name;
      }
   }


   public BuildPathologieOntologie() {
      Identifier rootIdentifier = new Identifier("universel");
      Identifier rootRelIdentifier = new Identifier("relation");
      Identifier mainLanguage = new Identifier("Français");
      Object[] objs;


      try {


////////////////// universel ////////////////////////////

         Identifier attribut = new Identifier("attribut");
         Identifier objet = new Identifier("objet");
         Identifier action = new Identifier("action");
```

```
        Identifier relation = new Identifier("relation");
        Identifier patient = new Identifier("patient");
        Identifier diagnostic = new Identifier("diagnostic");
        Identifier pronostic = new Identifier("pronostic");
        Identifier traitement = new Identifier("traitement");
        Identifier suiveThérapeutique = new Identifier("suiveThérapeutique");
        Identifier pathologie = new Identifier("pathologie");
        Identifier signe = new Identifier("signe");
        Identifier signePathologique = new Identifier("signePathologique");
        Identifier médicament = new Identifier("médicament");
        Identifier traitementChirurgicale = new
   Identifier("traitementChirurgicale");
        Identifier autre = new Identifier("autre");
        objs = new Object[] { objet, action, attribut, patient, diagnostic,
   pronostic, traitement, suiveThérapeutique, pathologie, signe, médicament,
   traitementChirurgicale, autre };
        lexicon.linkSubTypesToType(objs, rootIdentifier);


////////////////// objet ///////////////////////////////////

       Identifier partie_du_corps = new Identifier("partie_du_corps");
        objs = new Object[] { partie_du_corps };
        lexicon.linkSubTypesToType(objs, objet);


////////////////// action //////////////////////////////////

        Identifier causer = new Identifier("causer");
        Identifier témoigner = new Identifier("témoigner");
        Identifier affecter = new Identifier("affecter");
        Identifier développer = new Identifier("développer");
        Identifier réduire = new Identifier("réduire");
        Identifier évoluer = new Identifier("évoluer");
        objs = new Object[] { causer, témoigner, affecter, développer, réduire,
   évoluer };
        lexicon.linkSubTypesToType(objs, action);


////////////////// attribut ////////////////////////////////

        Identifier transmissible = new Identifier("transmissible");
        Identifier infectieux = new Identifier("infectieux");
        Identifier maligne = new Identifier("maligne");
        Identifier sévère = new Identifier("sévère");
        Identifier résistant = new Identifier("résistant");
        Identifier positif = new Identifier("positif");
        Identifier progressif = new Identifier("progressif");
        Identifier viral = new Identifier("viral");
        Identifier mortel = new Identifier("mortel");
        Identifier chronique = new Identifier("chronique");
        Identifier aigu = new Identifier("aigu");
        Identifier soudaine = new Identifier("soudaine");
        Identifier rapidement = new Identifier("rapidement");
        Identifier occasionnel = new Identifier("occasionnel");
        Identifier microscopique = new Identifier("microscopique");
        Identifier gangreneux = new Identifier("gangreneux");
        Identifier bactérienne = new Identifier("bactérienne");
        Identifier septicémique = new Identifier("septicémique");
        Identifier caséeux = new Identifier("caséeux");
       Identifier osseux = new Identifier("osseux");
        Identifier immunodépressif = new Identifier("immunodépressif");
        Identifier nécrotique = new Identifier("nécrotique");
        Identifier aplasique = new Identifier("aplasique");
        Identifier tumoral = new Identifier("tumoral");
        Identifier nerveux = new Identifier("nerveux");
        Identifier rénal = new Identifier("rénal");
        Identifier respiratoire = new Identifier("respiratoire");
        Identifier hématopoïèse = new Identifier("hématopoïèse");
        Identifier cardiovasculaire = new Identifier("cardiovasculaire");
        Identifier abdominal = new Identifier("abdominal");
        Identifier interne = new Identifier("interne");
        Identifier ventral = new Identifier("ventral");
        Identifier digestif = new Identifier("digestif");
        Identifier buccal = new Identifier("buccal");
        Identifier occuloNasal = new Identifier("occuloNasal");
        Identifier plusieursMoyens = new Identifier("plusieursMoyens");
```

```
        Identifier multiFactoriel = new Identifier("multiFactoriel");
       objs = new Object[] { transmissible, infectieux, maligne, sévère,
   résistant, positif, progressif, viral, mortel, chronique, aigu, soudaine,
   rapidement, occasionnel, microscopique, gangreneux, bactérienne,
   septicémique, caséeux, osseux, immunodépressif, nécrotique, aplasique,
   tumoral, nerveux, rénal, respiratoire, hématopoïèse, cardiovasculaire,
   abdominal, interne, ventral, digestif, buccal, occuloNasal, plusieursMoyens,
   multiFactoriel };
        lexicon.linkSubTypesToType(objs, attribut);


////////////////// relation ////////////////////////////////

        Identifier estUn = new Identifier("estUn");
        Identifier aUn = new Identifier("aUn");
        Identifier estPartieDe = new Identifier("estPartieDe");
        Identifier estComposéDe = new Identifier("estComposéDe");
        Identifier et = new Identifier("et");
        Identifier estSynonyme = new Identifier("estSynonyme");
        Identifier estPropriétaire = new Identifier("estPropriétaire");
        Identifier liéFonctionnellementA = new
   Identifier("liéFonctionnellementA");
        Identifier liéConceptuellementA = new
   Identifier("liéConceptuellementA");
        Identifier précède = new Identifier("précède");
        Identifier suit = new Identifier("suit");
       Identifier estDessus= new Identifier("estDessus");
        Identifier estDessous = new Identifier("estDessous");
        Identifier estAProximitéDe = new Identifier("estAProximitéDe ");
        Identifier estSitué = new Identifier("estSitué");
        Identifier estAssociéA = new Identifier("estAssociéA");
        Identifier estVoisinDe = new Identifier("estVoisinDe ");
        Identifier produire = new Identifier("produire");
        Identifier provoquer = new Identifier("provoquer");
        Identifier évoquer = new Identifier("évoquer");
        Identifier acheter = new Identifier("acheter");
        Identifier vendre = new Identifier("vendre");
        Identifier prendre = new Identifier("prendre");
        objs = new Object[] { estUn, aUn, estPartieDe, estComposéDe,
   estPropriétaire, et, estSynonyme, liéFonctionnellementA,
   liéConceptuellementA, précède, suit, estDessus, estDessous, estAProximitéDe,
   estSitué, estAssociéA, estVoisinDe, produire, provoquer, causer, évoquer,
   acheter, vendre, prendre };
        lexicon.linkSubTypesToType(objs, relation);


////////////////// patient ////////////////////////////////

        Identifier id = new Identifier("id");
        Identifier historique = new Identifier("historique");
        Identifier condition = new Identifier("condition");
        Identifier peau = new Identifier("peau");
        Identifier os = new Identifier("os");
        Identifier aorte = new Identifier("aorte");
        Identifier organe = new Identifier("organe");
        Identifier tête = new Identifier("tête");
        Identifier cartilage = new Identifier("cartilage");
        Identifier ventricule = new Identifier("ventricule");
        objs = new Object[] { id, historique, condition, peau, tête, os, aorte,
   organe, cartilage, ventricule };
        lexicon.linkSubTypesToType(objs, patient);


///////////////// pathologie/////////////////////////////////

        Identifier syndrome = new Identifier("syndrome");
        Identifier cause = new Identifier("cause");
        Identifier maladie = new Identifier("maladie");

       Identifier pathologieSelonEtiologie = new
   Identifier("pathologieSelonEtiologie");
        Identifier pathologieDeAàZ = new Identifier("pathologieDeAàZ");
        Identifier groupePathologique = new Identifier("groupePathologique");
        Identifier dominantesPathologiques = new
   Identifier("dominantesPathologiques");
        objs = new Object[] { pathologieSelonEtiologie, pathologieDeAàZ,
   groupePathologique, dominantesPathologiques };
```

```
        lexicon.linkSubTypesToType(objs, pathologie);


////////////////// signe ////////////////////////////////////

        Identifier perte = new Identifier("perte");
        Identifier diminution = new Identifier("diminution");
        Identifier production = new Identifier("production");
        Identifier infection = new Identifier("infection");
        Identifier apparition = new Identifier("apparition");
        Identifier lésion = new Identifier("lésion");
        Identifier affection = new Identifier("affection");
        Identifier défaillance = new Identifier("défaillance");
        Identifier rupture = new Identifier("rupture");
        Identifier accumulation = new Identifier("accumulation");
        Identifier inflammation = new Identifier("inflammation");
        Identifier évolution = new Identifier("évolution");
        Identifier présence = new Identifier("présence");
        Identifier ulcération = new Identifier("ulcération");
        Identifier tremblement = new Identifier("tremblement");
        Identifier prolifération = new Identifier("prolifération");
        Identifier trouble = new Identifier("trouble");
        Identifier tropisme = new Identifier("tropisme");
        Identifier salpingite = new Identifier("salpingite");
        objs = new Object[] { perte, diminution, production, infection,
   apparition, lésion, affection, défaillance, rupture, accumulation,
   inflammation, évolution, présence, ulcération, tremblement, prolifération,
   trouble, tropisme, salpingite };
        lexicon.linkSubTypesToType(objs, signe);


/////////////// signePathologique ////////////////////////////

        Identifier diarrhée = new Identifier("diarrhée");
        Identifier hémorragie = new Identifier("hémorragie");
        Identifier cellulite = new Identifier("cellulite");
        Identifier tumeur = new Identifier("tumeur");
        Identifier anémie = new Identifier("anémie");
        Identifier entérite = new Identifier("entérite");
        Identifier toux = new Identifier("toux");
        Identifier abcès = new Identifier("abcès");
        objs = new Object[] { diarrhée, hémorragie, cellulite, tumeur, anémie,
   entérite, toux, abcès };
        lexicon.linkSubTypesToType(objs, signePathologique);


/////////////// médicament ////////////////////////////////

        Identifier prescription = new Identifier("prescription");
        Identifier dosage = new Identifier("dosage");
        Identifier contreIndicationsMédicamenteuses = new
   Identifier("contreIndicationsMédicamenteuses");
        Identifier propriétéMédicament = new
   Identifier("propriétéMédicament");
        Identifier effectSecondaireMédicament = new
   Identifier("effectSecondaireMédicament");
        Identifier duréePrendreMédicament = new
   Identifier("duréePrendreMédicament");
        objs = new Object[] { contreIndicationsMédicamenteuses, prescription,
   dosage, propriétéMédicament, effectSecondaireMédicament,
duréePrendreMédicament };
        lexicon.linkSubTypesToType(objs, partie_du_corps);


////////////// traitementChirurgicale ////////////////////////

        Identifier duréeChirurgicale = new Identifier("duréeChirurgicale");
        Identifier DispositifMédical = new Identifier("DispositifMédical");
        objs = new Object[] { duréeChirurgicale, DispositifMédical };
        lexicon.linkSubTypesToType(objs, traitementChirurgicale);


///////////////// autre ////////////////////////////////////

        Identifier désinfectant = new Identifier("désinfectant");
        Identifier gram = new Identifier("gram");
        Identifier spore = new Identifier("spore");
```

```
        Identifier poids = new Identifier("poids");
        Identifier segment = new Identifier("segment");
        Identifier proposition = new Identifier("proposition");
        Identifier composants = new Identifier("composants");
        Identifier salmonella = new Identifier("salmonella");
        Identifier transmission = new Identifier("transmission");
        Identifier large_groupe = new Identifier("large_groupe");
        Identifier caracteristique = new Identifier("caracteristique");
        Identifier milieu = new Identifier("milieu");
        Identifier extérieur = new Identifier("extérieur");
        Identifier jeu = new Identifier("jeu");
        Identifier zone = new Identifier("zone");
        Identifier groupe = new Identifier("groupe");
        Identifier cavité = new Identifier("cavité");
        Identifier anneau = new Identifier("anneau");
        Identifier plat = new Identifier("plat");
        Identifier vers = new Identifier("vers");
        Identifier lymphoïde = new Identifier("lymphoïde");
        Identifier lymphocytaire = new Identifier("lymphocytaire");
        Identifier barbillon = new Identifier("barbillon");
        Identifier ponte = new Identifier("ponte");
        Identifier troupeau = new Identifier("troupeau");
        Identifier leucose_lymphoède = new Identifier("leucose_lymphoède");
        Identifier système = new Identifier("système");
        Identifier droit = new Identifier("droit");
        Identifier transsudat = new Identifier("transsudat");
        Identifier face = new Identifier("face");
        Identifier conjugaison = new Identifier("conjugaison");
        Identifier longueur = new Identifier("longueur");
        Identifier symptôme = new Identifier("symptôme");
        Identifier rapteur = new Identifier("rapteur");
        objs = new Object[] { désinfectant, gram, spore, poids, segment,
   proposition, composants, salmonella, transmission, large_groupe,
   caracteristique, milieu, extérieur, jeu, zone, groupe, cavité, anneau, plat,
   vers, lymphoïde, lymphocytaire, barbillon, ponte, troupeau,
   leucose_lymphoède, système, droit, transsudat, face, conjugaison, longueur,
   symptôme, rapteur };
        lexicon.linkSubTypesToType(objs, autre);


     // Enregistrer la nouvelle ontologie en formes XML
        try {
           ontology.store("PathologieOntology.ont");
           ontology.storeInXML("PathologieOntology.xml");
        }
        catch (Exception ex){
           ex.printStackTrace();
        }
     }

     catch (Exception ex) {
        System.out.println(ex.getMessage());
        ex.printStackTrace();
     }
  }

  public static void main(String[] args) {
     new BuildPathologieOntologie();
  }
}
```

Programmation 11 La construction d'ontologie pathologique

Programmation-12. Ajouter, supprimer ou modifier les concepts, relations, hiérarchies et instances d'ontologie

```
package com.SMAAD;

import java.util.*;

public interface RenouvelleOntologie {

   Exception exception = new Exception();

// Ajouter un nouveau concept à l'ontologie
   void addConcept(Concept concept);

// Supprimer un concept de l'ontologie
   void removeConcept(Concept concept);
   void removeConcept(Identifier id);

// Énumère les concepts définis par l'ontologie
   Set <Concept> listConcepts();

// Chercher un concept dans l'ensemble des concepts définis par cette ontologie
// Retour <i>null</i> si aucun trouvé
    Concept findConcept(Identifier id);

// Ajouter une nouvelle relation à l'ontologie
   void addRelation(RelationExample relationExample);

// Supprimer une relation de l'ontologie
   void removeRelation(RelationExample relationExample);
   void removeRelation(Identifier id);

// Énumère les relations définis par l'ontologie
   Set <RelationExample> listRelationExample();

// Chercher une relation dans l'ensemble des relations définis par l'ontologie
// Retour NULL si aucun trouvé
   RelationExample findRelationExample(Identifier id);

// Ajouter une nouvelle hiérarchie à l'ontologie
   void addHierarchy(Hierarchy hierarchy);

// Supprimer une hiérarchie de l'ontologie
   void removeHierarchy(Hierarchy hierarchy);
   void removeHierarchy(Identifier id);

// Énumère les hiérarchies définis par l'ontologie
   Set <Hierarchy> listHierarchy();

// Chercher une hiérarchie dans l'ensemble des hiérarchies définis par
// l'ontologie. Retour NULL si aucun trouvé
   Hierarchy findHierarchy(Identifier id);

// Ajouter une nouvelle instance à l'ontologie
   void addInstance(Instance instance);

// Supprimer une instance de l'ontologie
   void removeInstance(Instance instance);
   void removeInstance(Identifier id);

// Énumère les instances définis par l'ontologie
   Set <Instance> listInstances();

// Chercher une instance dans l'ensemble des instances définis par l'ontologie
// Retour NULL si aucun trouvé
   Instance findInstance(Identifier id);

// Chercher un objet (concept, relation, hiérarchie, instance, etc.)
// Retour NULL si aucun trouvé
   Set <Ontology> findOntology(Identifier id);
}
```

Programmation 12 Ajouter, supprimer ou modifier les concepts, relations, hiérarchies et instances d'ontologie

Autres programmations dans le «PACKAGE-6. BuildPathologieOntologie» pour réaliser le programmation-11 et 12.

```
package com.SMAAD;

public class Identifier {
   protected String name;
   public Identifier(String s)  {
      name = s;
   }

   public String getName() {
      return name;
   }

   public void setName(String newName) {
      name = newName;
   }

   public static boolean isIdentifier(Object obj) {
      return (obj instanceof Identifier);
   }

   public Object clone() {
      try {
         return super.clone();
      }
      catch (CloneNotSupportedException e) {
         return null;
      }
   }

   public Identifier copy() {
      return  (Identifier) clone();
   }

   public String toString() {
      return name;
   }
}
```

Programmation 12.1 Identifier

```
package com.SMAAD;


public class LinkSubTypesToType extends Lexicon {
   Lexicon lexicon = new Lexicon();
   LinkSubTypesToType linkSubTypesToType = new LinkSubTypesToType();
   public void linkSubTypesToType(Object[] subTypeIdentifiers, Identifier
  rootIdentifier) {
      return;
   }
}
```

Programmation 12.4 LinkSubTypesToType

```
package com.SMAAD;

import java.io.*;

public class Ontology {

   public static boolean toXML = false;
   private File file;

   public void store(String filePath) throws Exception {
      FileOutputStream fileOutStrm = new FileOutputStream(new File(filePath));
      ObjectOutputStream objOutStream = new ObjectOutputStream(fileOutStrm);
      objOutStream.writeObject(this);
      fileOutStrm.close();
   }

   public void storeInXML(String filePath) throws Exception {
      toXML = true;
      setFile(new File ("/Users/yingshen/Desktop"));
   }

   public File getFile() {
     return file;
   }

   public void setFile(File file) {
     this.file = file;
   }

}
```

Programmation 12.2 Ontology

```
package com.SMAAD;


public class Lexicon {

   private Identifier language;

   Ontology ontology = new Ontology();

   public Identifier getLanguage() {
     return language;
   }


   class CS {
      CS cs = new CS();
      public CS getCS(Identifier identifier) {
         return cs;
      }
   }


   class Type {
      Type type = new Type();
   }

   class LinkSubTypesToType extends Lexicon {
      public void linkSubTypesToType(Object[] subTypeIdentifiers, Identifier
  rootIdentifier) {
         return;
      }
   }

   public void linkSubTypesToType(Object[] objs,
  com.SMAAD.BuildPathologieOntologie.Identifier rootIdentifier) {
  // TODO Auto-generated method stub
   }
}
```

Programmation 12.3 Lexicon

PACKAGE-7. OntologyGui

Programmation-13. Visualisation d'ontologie à travers de GUI

```
package com.SMAAD;

import javax.swing.*;
import java.awt.BorderLayout;


public class OntologyGui extends javax.swing.JFrame {

   private static final long serialVersionUID = 1L;
   private JPanel root;
   private JPanel decode;
   private JPanel jPanel1;
   private JTextPane jTextPane1;
   private JTextPane encodeText;
   private JLabel jLabel1;
   private JScrollPane encodeScroll;
   private JLabel encodeName;
   private JScrollPane deodeScroll;

   public static void main(String[] args) {
      OntologyGui inst = new OntologyGui();
      inst.setVisible(true);
      inst.addEncodedMessage("Testtest\ntest\n");
      inst.addDecodedMessage("test\ntest\n");
   }

   public OntologyGui() {
      super();
      initGUI();
   }

   private void initGUI() {
      try {
         this.setSize(484, 500);
         setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
         {
            root = new JPanel();
            BoxLayout rootLayout = new BoxLayout(root,
  javax.swing.BoxLayout.Y_AXIS);
            root.setLayout(rootLayout);
            this.getContentPane().add(getRoot(), BorderLayout.CENTER);
            {
               decode = new JPanel();
               root.add(getDecode());
               decode.setPreferredSize(new java.awt.Dimension());
               {
                  encodeName = new JLabel();
                  decode.add(getEncodeName());
                  encodeName.setText("Decoded Message:");
               }
               {
                  deodeScroll = new JScrollPane();
                  decode.add(getDeodeScroll());
                  deodeScroll.setPreferredSize(new java.awt.Dimension());
                  {
                     jTextPane1 = new JTextPane();
                     deodeScroll.setViewportView(getJTextPane1());
                     jTextPane1.setText("");
                  }
               }
            }
            {
               jPanel1 = new JPanel();
               root.add(getJPanel1());
               jPanel1.setPreferredSize(new java.awt.Dimension());
               {
                  jLabel1 = new JLabel();
                  jPanel1.add(getJLabel1());
```

```
                jLabel1.setText("Encoded Messages:");
             }
             {
                encodeScroll = new JScrollPane();
                jPanel1.add(getEncodeScroll());
                encodeScroll.setPreferredSize(new java.awt.Dimension());
                {
                   encodeText = new JTextPane();
                   encodeScroll.setViewportView(getEncodeText());
                   encodeText.setText("");
                }
             }
          }
       }
    } catch (Exception e) {
       e.printStackTrace();
    }
 }

 public JPanel getRoot() {
    return root;
 }

 public JPanel getDecode() {
    return decode;
 }

 public JPanel getJPanel1() {
    return jPanel1;
 }

 public JScrollPane getDeodeScroll() {
    return deodeScroll;
 }

 public JLabel getEncodeName() {
    return encodeName;
 }

 public JScrollPane getEncodeScroll() {
    return encodeScroll;
 }

 public JLabel getJLabel1() {
    return jLabel1;
 }

 public JTextPane getEncodeText() {
    return encodeText;
 }

 public JTextPane getJTextPane1() {
    return jTextPane1;
 }

 public void addDecodedMessage(String s){
    getJTextPane1().setText( getJTextPane1().getText() + s);
 }

 public void addEncodedMessage(String s){
    getEncodeText().setText( getEncodeText().getText() + s);
 }

}
```

Programmation 13 Visualisation d'ontologie à travers de GUI

PACKAGE-8. Action d'un antibiotique sur une maladie

Programmation-14. Calculer l'action d'un antibiotique et comparer les scores de différents antibiotiques

```
public class WriteExcelB {

   public static void main(String[] args)
   {
      XSSFWorkbook workbook = new XSSFWorkbook();
      XSSFSheet sheet = workbook.createSheet("EnsembleDesGermesDuneMaladieM");
Map<String, Object[]> data = new TreeMap<String, Object[]>();

      Row header = sheet.createRow(0);
      header.createCell(0).setCellValue("Angine erythémato pultacée");
      header.createCell(1).setCellValue("poidsGerme");
      header.createCell(2).setCellValue("ampicilline");
      header.createCell(3).setCellValue("Score1");
      header.createCell(4).setCellValue("Score2");
      header.createCell(5).setCellValue("erythrocine");
      header.createCell(6).setCellValue("Score1");
      header.createCell(7).setCellValue("Score2");
      header.createCell(8).setCellValue("Score max");
      header.createCell(9).setCellValue("PeniG");
      header.createCell(10).setCellValue("Score1");
      header.createCell(11).setCellValue("Score2");

      Row dataRow1 = sheet.createRow(1);
      dataRow1.createCell(0).setCellValue("Streptocoque");
      dataRow1.createCell(1).setCellValue(3);
      dataRow1.createCell(2).setCellValue(2);
      dataRow1.createCell(3).setCellValue(0.5);
      dataRow1.createCell(4).setCellValue(32);
      dataRow1.createCell(5).setCellValue(3);
      dataRow1.createCell(6).setCellValue(1);
      dataRow1.createCell(7).setCellValue(64);
      dataRow1.createCell(8).setCellValue(64);
      dataRow1.createCell(9).setCellValue(3);
      dataRow1.createCell(10).setCellValue(1);
      dataRow1.createCell(11).setCellValue(64);

      Row dataRow2 = sheet.createRow(2);
      dataRow2.createCell(0).setCellValue("Staphilocoque");
      dataRow2.createCell(1).setCellValue(2);
      dataRow2.createCell(2).setCellValue(1);
      dataRow2.createCell(3).setCellValue(0.125);
      dataRow2.createCell(4).setCellValue(8);
      dataRow2.createCell(5).setCellValue(2);
      dataRow2.createCell(6).setCellValue(0.25);
      dataRow2.createCell(7).setCellValue(16);
      dataRow2.createCell(8).setCellValue(32);
      dataRow2.createCell(9).setCellValue(1);
      dataRow2.createCell(10).setCellValue(0.125);
      dataRow2.createCell(11).setCellValue(8);

      Row dataRow3 = sheet.createRow(3);
      dataRow3.createCell(0).setCellValue("Méningocoque");
      dataRow3.createCell(1).setCellValue(1);
      dataRow3.createCell(2).setCellValue(2);
      dataRow3.createCell(3).setCellValue(0.125);
      dataRow3.createCell(4).setCellValue(8);
      dataRow3.createCell(5).setCellValue(3);
      dataRow3.createCell(6).setCellValue(0.25);
      dataRow3.createCell(7).setCellValue(16);
      dataRow3.createCell(8).setCellValue(16);
      dataRow3.createCell(9).setCellValue(3);
      dataRow3.createCell(10).setCellValue(0.25);
      dataRow3.createCell(11).setCellValue(16);

      Row dataRow4 = sheet.createRow(4);
```

```
      dataRow4.createCell(0).setCellValue("Hémophylus");
      dataRow4.createCell(1).setCellValue(2);
      dataRow4.createCell(2).setCellValue(2);
      dataRow4.createCell(3).setCellValue(0.25);
      dataRow4.createCell(4).setCellValue(16);
      dataRow4.createCell(5).setCellValue(2);
      dataRow4.createCell(6).setCellValue(0.25);
      dataRow4.createCell(7).setCellValue(16);
      dataRow4.createCell(8).setCellValue(32);
      dataRow4.createCell(9).setCellValue(1);
      dataRow4.createCell(10).setCellValue(0.125);
      dataRow4.createCell(11).setCellValue(8);

      Row dataRow5 = sheet.createRow(5);
      dataRow5.createCell(0).setCellValue("Gonocoque");
      dataRow5.createCell(1).setCellValue(2);
      dataRow5.createCell(2).setCellValue(2);
      dataRow5.createCell(3).setCellValue(0.25);
      dataRow5.createCell(4).setCellValue(16);
      dataRow5.createCell(5).setCellValue(2);
      dataRow5.createCell(6).setCellValue(0.25);
      dataRow5.createCell(7).setCellValue(16);
      dataRow5.createCell(8).setCellValue(32);
      dataRow5.createCell(9).setCellValue(2);
      dataRow5.createCell(10).setCellValue(0.25);
      dataRow5.createCell(11).setCellValue(16);

      Row dataRow6 = sheet.createRow(6);
      dataRow6.createCell(4).setCellValue("=E2+E3+E4+E5+E6");
      dataRow6.createCell(7).setCellValue("=H2+H3+H4+H5+H6");
      dataRow6.createCell(8).setCellValue("=I2+I3+I4+I5+I6");
      dataRow6.createCell(11).setCellValue("=L2+L3+L4+L5+L6");

      Row dataRow7 = sheet.createRow(7);
      dataRow7.createCell(4).setCellValue("=E7/I7");
      dataRow7.createCell(7).setCellValue("=H7/I7");
      dataRow7.createCell(11).setCellValue("=L7/I7")

      //Iterate over data and write to sheet
      Set<String> keyset = data.keySet();
      int rownum = 0;
      for (String key : keyset)
      {
         Row row = sheet.createRow(rownum++);
         Object [] objArr = data.get(key);
         int cellnum = 0;
         for (Object obj : objArr)
         {
            Cell cell = row.createCell(cellnum++);
            if(obj instanceof String)
               cell.setCellValue((String)obj);
            else if(obj instanceof Integer)
               cell.setCellValue((Integer)obj);
          }
       }
       try
       {
           //Write the workbook in file system
           FileOutputStream out = new FileOutputStream(new
  File("/Users/yingshen/Desktop/writeExcelB.xlsx"));
           workbook.write(out);
           out.close();
           System.out.println("writeExcelB.xlsx written successfully on disk.");
       }
       catch (Exception e) {
           e.printStackTrace();
       }
    }
}
```

Programmation 14 Calculer l'action d'un antibiotique et comparer les scores de différents antibiotiques

PACKAGE-9. Coder l'ontologie à OWL

Code OWL-1. Présenter l'ontologie des agents infectieux à travers de langage OWL

```
<owl:Class rdf:ID="Virus">
    <rdfs:subClassOf rdf:resource="#AgentInfectieux" />
</owl:Class>

<owl:Class rdf:ID="Parasites">
    <rdfs:subClassOf rdf:resource="#AgentInfectieux" />
</owl:Class>

<owl:Class rdf:ID="Bactéries">
    <rdfs:subClassOf rdf:resource="#AgentInfectieux" />
</owl:Class>

<owl:Class rdf:ID="CocciGram+">
    <rdfs:subClassOf rdf:resource="#Bactéries" />
</owl:Class>

<owl:Class rdf:ID="CocciGram-">
    <rdfs:subClassOf rdf:resource="#Bactéries" />
</owl:Class>

<owl:Class rdf:ID="BactGram-">
    <rdfs:subClassOf rdf:resource="#Bactéries" />
</owl:Class>

<owl:ObjectProperty rdf:about="#estUn">
```

Code OWL 1 Présenter l'ontologie des agents infectieux à travers de langage OWL

Code OWL-2. Présenter l'ontologie des pathologies à travers de langage OWL

```
<owl:Class rdf:ID="Maladies">
    <rdfs:subClassOf rdf:resource="#MaladiesInfectieuses" />
</owl:Class>

<owl:Class rdf:ID="SignesPathognomoniques">
    <rdfs:subClassOf rdf:resource="#SignesCliniques" />
</owl:Class>

<owl:Class rdf:ID="SignesEvocateurs">
    <rdfs:subClassOf rdf:resource="#SignesCliniques" />
</owl:Class>

<owl:Class rdf:ID="SignesPathognomoniques">
    <rdfs:subClassOf rdf:resource="#SignesCliniques" />
</owl:Class>

<owl:Class rdf:ID="SignesAccessoires">
    <rdfs:subClassOf rdf:resource="#SignesCliniques" />
</owl:Class>

<owl:Class rdf:about="#SignesCliniques">
    <rdfs:subClassOf>
        <owl:Class rdf:about="#Syndromes"/>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Syndromes">
    <rdfs:subClassOf>
        <owl:Class rdf:about="#TableauxClinique"/>
    </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:about="#TableauxClinique">
    <rdfs:subClassOf>
        <owl:Class rdf:about="#Maladies"/>
    </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:about="#estPartieDe">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <owl:inverseOf>
        <owl:ObjectProperty rdf:about="#estCollection"/>
    </owl:inverseOf>
    <rdfs:domain>
        <owl:Class rdf:about="#Maladie"/>
    </rdfs:domain>
</owl:ObjectProperty>
```

Code OWL 2 Présenter l'ontologie des pathologies à travers de langage OWL

Code OWL-3. Présenter l'ontologie des antibiotiques à travers de langage OWL

```
<owl:Class rdf:ID="Antimycosiques">
    <rdfs:subClassOf rdf:resource="#PrincipesActifs" />
</owl:Class>  //Antimycosiques est une sous classe de PrincipesActifs

<owl:Class rdf:ID="Antiviraux">
    <rdfs:subClassOf rdf:resource="#PrincipesActifs" />
</owl:Class>

<owl:Class rdf:ID="Antibiotiques">
    <rdfs:subClassOf rdf:resource="#PrincipesActifs" />
</owl:Class>

<owl:Class rdf:ID="Antiparasitaires">
    <rdfs:subClassOf rdf:resource="#PrincipesActifs" />
</owl:Class>

<owl:Class rdf:ID="Phénicolés">
    <rdfs:subClassOf rdf:resource="#Antibiotiques" />
</owl:Class> // Phénicolés est une sous classe de Antibiotiques

<owl:Class rdf:ID="Nitrofuranes">
    <rdfs:subClassOf rdf:resource="#Antibiotiques" />
</owl:Class>

<owl:Class rdf:ID="Aminosides">
    <rdfs:subClassOf rdf:resource="#Antibiotiques" />
</owl:Class>

<owl:Class rdf:ID="βLactamines">
    <rdfs:subClassOf rdf:resource="#Antibiotiques" />
</owl:Class>

<owl:Class rdf:ID="Pénicillines">
    <rdfs:subClassOf rdf:resource="#βLactamines" />
</owl:Class> // Pénicillines est une sous classe de βLactamines

<owl:Class rdf:ID="Céphalosporines">
    <rdfs:subClassOf rdf:resource="#βLactamines" />
</owl:Class>

<owl:Class rdf:ID="CéphalosporinesI">
    <rdfs:subClassOf rdf:resource="#Céphalosporines" />
</owl:Class>

<owl:Class rdf:ID="CéphalosporinesII">
    <rdfs:subClassOf rdf:resource="#Céphalosporines" />
</owl:Class>

<owl:Class rdf:ID="CéphalosporinesIII">
    <rdfs:subClassOf rdf:resource="#Céphalosporines" />
</owl:Class>
```

Code OWL 3 Présenter l'ontologie des antibiotiques à travers de langage OWL

Code OWL-4. Présenter l'ontologie du Dossier Médical à travers de langage OWL

```
    <owl:Class rdf:about="#AntécédentsFamiliaux">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Patient"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#Traitements">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Patient"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#EtatGénéral">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Patient"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#AnécédentsPersonnels">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Patient"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#EtatPhysiologique">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Patient"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#Patient">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#estAtteint"/>
                <owl:someValuesFrom rdf:resource="#MaladiesInfectieuses"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#Béctérie">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#provoquer"/>
                <owl:someValuesFrom rdf:resource="#MaladiesInfectieuses"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#Antibiotique">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#aPouvoirBactéricideSur"/>
                <owl:someValuesFrom rdf:resource="#Béctérie"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#Patient">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="#aContreIndicationLieeAuxAntécédents"/>
                <owl:someValuesFrom rdf:resource="#Antibiotique"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
```

Code OWL 4 Présenter l'ontologie du Dossier Médical à travers de langage OWL

Code OWL-5. Présenter l'ontologie des posologies à travers de langage OWL

```
    <owl:Class rdf:about="#Administration">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Posologie"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#DoseDeCharge">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Posologie"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#Rythme">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Posologie"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#MonoDoseJournalière">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Posologie"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#MultiDoseJournalière">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Posologie"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#PerfusionContinue">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Posologie"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#SousDosage">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#provoquer"/>
                <owl:someValuesFrom
rdf:resource="#PathologiesIatrogènes"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#DosageSériqueDesAntibiotiques">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Soulager"/>
                <owl:someValuesFrom
rdf:resource="#PathologiesIatrogènes"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#DosageSériqueDesAntibiotiques">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#aUnPouvoirSur "/>
                <owl:someValuesFrom rdf:resource="#CertainesMolecules"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#CertainesMolecules">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Affecter"/>
                <owl:someValuesFrom rdf:resource="#Posologie"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
```

Code OWL 5 Présenter l'ontologie des posologies à travers de langage OWL

Code OWL-6. Présenter l'ontologie de monothérapie antibiotique et associations d'antibiotiques à travers de langage OWL

```
    <owl:Class rdf:about="#monothérapieAntibiotique">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Thérapeutique"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#associationsAntibiotiques">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Thérapeutique"/>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#monothérapieAntibiotique">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#aUnPouvoirSur"/>
                <owl:someValuesFrom rdf:resource="#laPlupartDesInfections"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#associationsAntibiotiques">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#aUnPouvoirSur"/>
                <owl:someValuesFrom rdf:resource="#InfectionsSévères"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#associationsAntibiotiques">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#aUnPouvoirSur"/>
                <owl:someValuesFrom
rdf:resource="#MicrobiologiquementNonDocumentées"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#associationsAntibiotiques">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#aUnPouvoirSur"/>
                <owl:someValuesFrom
rdf:resource="#InfectionsAPseudomonasAeruginosa"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#EntérobactériesDuGroupe3">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Antibiotique"/>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#CéphalosporinesDe3eGénération"/>
    </owl:Class>

    <owl:Class rdf:about="#AcideFusidique">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Antibiotique"/>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#StaphylococcusAureus"/>
    </owl:Class>

    <owl:Class rdf:about="#Fosfomycine">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Antibiotique"/>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#StaphylococcusAureus"/>
    </owl:Class>

    <owl:Class rdf:about="#Fluoroquinolones">
```

```
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Antibiotique"/>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#StaphylococcusAureus"/>
    </owl:Class>

    <owl:Class rdf:about="#Rifampicine">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Antibiotique"/>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#StaphylococcusAureus"/>
    </owl:Class>

    <owl:Class rdf:about="#fluoroquinolones">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Antibiotique"/>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#Entérobactéries"/>
    </owl:Class>

    <owl:Class rdf:about="#AcideNalidixique">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Antibiotique"/>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="#Entérobactéries"/>
    </owl:Class>
```

Code OWL 6 Présenter l'ontologie de monothérapie antibiotique et associations d'antibiotiques à travers de langage OWL

Code OWL-7. Présenter l'ontologie concernant la durée d'utilisation d'antibiotique à travers de langage OWL

```
    <owl:Class rdf:ID="TempsUtilisationAntibiothérapieCurative">
        <rdfs:subClassOf rdf:resource="#Temps" />
    </owl:Class>

    <owl:Class rdf:ID="TempsUtilisationAntibioprophylaxieChirurgicale">
        <rdfs:subClassOf rdf:resource="#Temps" />
    </owl:Class>

    <owl:Class rdf:ID="TempsRéévaluationAntibiothérapie">
        <rdfs:subClassOf rdf:resource="#Temps" />
    </owl:Class>

    <owl:Class rdf:about="#TempsUtilisationAntibiothérapieCurative">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConstraint"/>
                <owl:someValuesFrom rdf:resource="#Temps"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConstraint"/>
                <owl:allValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Verification">
                            <rdf:Description
rdf:about="#MaintienAssociationAntibiotiques"/>
                            <rdf:Description rdf:about="#PlusPetitOuEgalA"/>
                            <rdf:Description rdf:about="#3jours"/> // ≤3jours
                        </owl:unionOf>
                    </owl:Class>
                </owl:allValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConstraint"/>
                <owl:allValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Verification">
                            <rdf:Description rdf:about="#AntibiothérapieCurative"/>
                            <rdf:Description rdf:about="#PlusPetitOuEgalA"/>
                            <rdf:Description rdf:about="#7jours"/> // ≤7jours
                        </owl:unionOf>
                    </owl:Class>
                </owl:allValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#TempsUtilisationAntibioprophylaxieChirurgicale">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConstraint"/>
                <owl:someValuesFrom rdf:resource="#Temps"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConstraint"/>
                <owl:allValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Verification">
                            <rdf:Description rdf:about="#inductionAnesthésique"/>
                            <rdf:Description rdf:about="#injectionIntraveineuse"/>
                            <rdf:Description rdf:about="#maximum"/>
                            <rdf:Description rdf:about="#1h"/>
                            <rdf:Description rdf:about="#avant"/>
                            <rdf:Description rdf:about="#incisionCutanée"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:allValuesFrom>
```

```
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConstraint"/>
                <owl:allValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Verification">
                            <rdf:Description rdf:about="#acteOpératoire"/>
                            <rdf:Description rdf:about="#PlusPetitQue"/>
                            <rdf:Description rdf:about="#24h"/> //<24h
                        </owl:unionOf>
                    </owl:Class>
                </owl:allValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#TempsRéévaluationAntibiothérapie">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConstraint"/>
                <owl:someValuesFrom rdf:resource="#Temps"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConstraint"/>
                <owl:allValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Verification">
                            <rdf:Description rdf:about="#RéévaluationAntibiothérapie
"/>
                            <rdf:Description rdf:about="#entre"/>
                            <rdf:Description rdf:about="#24e h"/>
                            <rdf:Description rdf:about="#et"/>
                               <rdf:Description rdf:about="#72e h"/>
                                                //24e h<temps<72e h
                        </owl:unionOf>
                    </owl:Class>
                </owl:allValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
```

Code OWL 7 Présenter l'ontologie concernant la durée d'utilisation d'antibiotique à travers de langage OWL