Un système multi-agent adaptatif pour la construction d'ontologies à partir de textes

Kévin Ottens

THÈSE

présentée devant

l'Université Paul Sabatier de Toulouse III

U.F.R. MATHÉMATIQUES, INFORMATIQUE ET GESTION

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ PAUL SABATIER

Mention INFORMATIQUE

par

KÉVIN OTTENS

École doctorale : Informatique et Télécommunication
Laboratoire d'accueil : Institut de Recherche en Informatique de Toulouse
Équipe d'accueil : Systèmes Multi-Agents Coopératifs

Un système multi-agent adaptatif pour la construction d'ontologies à partir de textes

---

JURY

Marie-Pierre GLEIZES	*Professeure, Université de Toulouse III*	(présidente du jury)
Anne NICOLLE	*Professeure, Université de Caen*	(rapporteure)
Jean CHARLET	*Chargé de mission AP-HP, INSERM, HdR*	(rapporteur)
Joël QUINQUETON	*Professeur, Université de Montpellier III*	(examinateur)
Nathalie AUSSENAC-GILLES	*Chargée de recherche CNRS, HdR*	(co-directrice de thèse)
Pierre GLIZE	*Ingénieur d'études CNRS, HdR*	(directeur de thèse)

INVITÉE

Valérie CAMPS	*MCF, Université de Toulouse III*	(co-encadrante)

Kévin Ottens

UN SYSTÈME MULTI-AGENT ADAPTATIF POUR LA CONSTRUCTION D'ONTOLOGIES À PARTIR DE TEXTES

Directeur de thèse :
Pierre Glize, IE CNRS, HdR
Université Paul Sabatier

Résumé

Le Web sémantique désigne un ensemble de technologies visant à rendre le contenu des ressources du World Wide Web accessible et utilisable par les programmes et agents logiciels. Ainsi, il doit faciliter l'accès à l'information pour les utilisateurs. Or, un des enjeux du succès du Web sémantique est la disponibilité d'ontologies qui sont des représentations de connaissances formalisées et exploitables par des systèmes informatiques pour leur communication. Malheureusement leur construction est généralement longue et coûteuse, et leur maintenance soulève des problèmes jusqu'ici sous-estimés. S'appuyer sur des textes pour la conception d'ontologies est vu comme une issue possible à leur coût, malgré les difficultés inhérentes à l'exploration d'analyses textuelles.

Parce que l'ontologie doit être maintenue, et parce qu'elle peut-être vue comme un système complexe constitué de concepts, nous proposons d'utiliser les systèmes multi-agents adaptatifs pour semi-automatiser le processus de construction des ontologies à partir de texte. L'état stable de ces systèmes résulte des interactions coopératives entre les agents logiciels qui les constituent. Dans notre cas, les agents utilisent des algorithmes distribués d'analyse statistique pour trouver la structure la plus satisfaisante d'après une analyse syntaxique et distributionnelle des textes. L'utilisateur peut alors valider, critiquer ou modifier des parties de cette structure d'agents, qui est la base de l'ontologie en devenir, pour la rendre conforme à ses objectifs et à sa vision du domaine modélisé. En retour, les agents se réorganisent pour satisfaire les nouvelles contraintes introduites. Les ontologies habituellement fixées deviennent ici dynamiques, leur conception devient « vivante ». Ce sont les principes sous-jacents de notre système nommé Dynamo.

La pertinence de cette approche a été mise à l'épreuve par des expérimentations visant à évaluer la complexité algorithmique de notre système, et par son utilisation en conditions réelles. Dans ce mémoire, nous présentons et analysons les résultats obtenus.

Institut de Recherche en Informatique de Toulouse - UMR 5505
Université Paul Sabatier, 118 route de Narbonne, 31062 TOULOUSE cedex 4

Kévin Ottens

AN ADAPTIVE MULTI-AGENT SYSTEM FOR ONTOLOGY BUILDING FROM TEXTS

Supervisor :
Pierre Glize, IE CNRS, HdR
Université Paul Sabatier

Abstract

Semantic Web refers to a set of technologies trying to make the World Wide Web resources content available and usable by programs and software agents. Then, it should ease users access to information. Its raising requires the availability of ontologies which are formalized knowledge models usable by a software systems for their communication. Unfortunately their construction is generally slow and costly, and their maintenance raises issues generally underestimated. The usage of texts for designing ontologies is considered as a possible solution to their cost, despite the fact that this task is still difficult.

Since ontologies must be maintained, and because they can be seen as complex systems of concepts, we propose to use adaptive multi-agent systems to semi-automate the process of building ontologies from texts. The stable state of those systems comes from the cooperative interactions between the constituting software agents. In our case, the agents use distributed algorithms of statistical analysis to find the most satisfying structure from a syntactical and distributional analysis of texts. Then, the user can validate, criticize or modify parts of the agent structure, which is the base of the ontology in progress, to make it fulfil his objectives and his own vision of the modeled domain. In return, the agents reorganize themselves to satisfy the newly introduced constraints. Ontologies which are generally fixed become here dynamic, they become a "living design". Those are the underlying principles used in our system named Dynamo.

The relevance of this approach has been tested thanks to experiments aiming at evaluating the performances of our system, and by its use in real operations. In this thesis, we present and analyse the obtained results.

Institut de Recherche en Informatique de Toulouse - UMR 5505
Université Paul Sabatier, 118 route de Narbonne, 31062 TOULOUSE cedex 4

Je dédie cette thèse
à la petite Camille.

JE tiens à remercier...

... Pierre GLIZE, ingénieur CNRS à l'IRIT, mon directeur de thèse, pour m'avoir dirigé, guidé et encouragé du DEA à la thèse. Grâce à son dynamisme et à sa capacité à toujours aller de l'avant même lorsque les difficultés semblaient insurmontables j'ai pu progresser sur le plan scientifique. Sans cette pression permanente de mon environnement le petit système que je suis chercherait encore une solution.

... Nathalie AUSSENAC-GILLES, chargée de recherche CNRS à l'IRIT, responsable de l'équipe IC3, ma co-directrice de thèse, pour avoir suivi mes pas depuis déjà plus de quatre ans ! Nos discussions et remises en cause incessantes sur la marche à suivre pour mes travaux m'ont permis d'en être là où j'en suis aujourd'hui.

... Marie-Pierre GLEIZES, professeure à l'Université Paul Sabatier, Toulouse III, responsable de l'équipe SMAC, pour avoir toujours gardé un œil sur l'avancée de mes travaux, pour avoir cru en moi, et pour avoir présidé mon jury de thèse.

... Anne NICOLLE, professeure à l'Université de Caen, et Jean CHARLET, chargé de mission Assistance Publique - Hôpitaux de Paris, pour avoir accepté d'évaluer ma thèse avec intégrité et rigueur. Les questions et commentaires reçus ont été la preuve de l'intérêt porté à mon travail, ce en quoi je leur suis reconnaissant.

... Joël QUINQUETON, professeur à l'Université de Montpellier III, pour sa participation à mon jury de thèse, ses questions lors de la soutenance, et sa bonne humeur.

... Valérie CAMPS, maître de conférences à l'Université Paul Sabatier, Toulouse III, pour la qualité des relations de travail que nous avons entretenues, pour ses encouragements, et pour ses questions qui ont eu un grand impact sur la qualité de mes écrits.

... l'équipe SMAC de l'IRIT, mon équipe d'accueil depuis quatre ans pour m'avoir intégré et offert d'excellentes conditions de travail. Merci aux permanents, André MACHONIN, Carole BERNON, Christine RÉGIS, et Jean-Pierre GEORGÉ, ainsi qu'aux autres thésards et post-doctorants, Davy CAPÉRA, Jean-Baptiste WELCOMME et Jean-Pierre MANO.

... l'équipe IC3 de l'IRIT, équipe avec laquelle j'ai collaborée pour ces travaux. Merci en particulier à Guy CAMILLERI, Jacques VIRBEL, Jean-Luc SOUBIE, Nathalie HERNANDEZ et Pascale ZARATÉ.

... tous les personnels qui font vivre le laboratoire, sans eux de tels travaux ne verraient simplement pas le jour. Merci en particulier à Jean-Pierre BARITAUD et Maryse CAILLOUX.

... la secrétaire de l'école doctorale, Martine LABRUYÈRE qui sait se rendre disponible pour les doctorants, et qui fait preuve de patience pour gérer nos dossiers au cas par cas.

... l'équipe pédagogique de l'IUP ISI pour m'avoir permis d'enseigner dans de bonnes conditions et m'offrir ainsi un contrepoint intéressant à mes activités de recherche. Je suis particulièrement reconnaissant envers Henri MASSIÉ, directeur de l'IUP ISI, pour sa confiance et son soutien pour les propositions de formations que j'ai pu faire ces quatre dernières années.

... mes étudiants pour avoir énormément appris à leur contact. J'espère qu'ils ont retiré de mes interventions au moins autant que ce qu'ils m'ont apporté.

... Gauthier PICARD, ancien SMACker, pour son soutien, sa culture et ses sensibilités artistiques. Il a eu un impact sur mon approche de la science et de la vie plus grand qu'il ne voudra jamais l'admettre.

... Aaron SEIGO pour son énergie et sa motivation, pour m'avoir reçu chez lui lorsque j'ai eu besoin de changer d'air pour finir le dernier chapitre de ma thèse, et pour m'avoir fait découvrir le *Bar Named Sue*.

... la *clique des thésards toulousains*, Élise, Axel, Florian et Sylvain, pour leur soutien qui a permis de rendre mon travail plus agréable lors des périodes parfois difficiles. Bon courage pour la suite, ce n'est qu'un bon moment à passer.

... les *insomniaques* – toulousains ou non – Lionel, Ludovic et Mario pour avoir gardé contact depuis toutes ces années et pour être la source de mes meilleurs souvenirs d'étudiant.

... Mes parents pour avoir toujours cru en moi, même lors de mes pires moments de doutes, et pour m'avoir toujours poussé à aller vers mes rêves. Merci d'avoir fait de moi l'être humain que je suis aujourd'hui.

... Mais surtout, Virginie qui partage ma vie et sait me comprendre comme personne. Elle est le phare qui donne une direction à ma vie chaque jour.

Table des matières

Introduction **1**

I Contexte et état de l'art **5**

1 Vers les ontologies dynamiques **7**

1.1 Des ontologies 8

1.1.1 Définition 8

1.1.2 Motivations 9

1.1.3 Diversité des types d'ontologies 10

1.1.4 Construction et maintenance des ontologies à partir de texte 10

1.2 Un point de départ : le corpus textuel 11

1.2.1 Définition 12

1.2.2 Linguistique de corpus et ingénierie des connaissances 12

1.3 Des termes et des concepts 13

1.3.1 Définitions 13

1.3.2 Comment lier terme et concept ? 14

1.4 Traitement automatique des langues pour la construction d'ontologies à partir de textes 15

1.4.1 Place des outils et point de vue sémantique 15

1.4.2 Importance de l'interprétation 19

1.5 Limites de la construction d'ontologies à partir de textes 20

1.5.1 Efforts et raffinement du produit résultant 20

1.5.2 La maintenance : un problème mal géré 21

1.6 Bilan 22

2 Systèmes multi-agents et construction d'ontologies **23**

2.1 Systèmes multi-agents et ontologies 23

2.2 La théorie des *Adaptive Multi-Agent Systems* (AMAS) 26
2.3 Une théorie appliquée aux systèmes multi-agents 28
2.4 Quelques problèmes traités avec des AMAS 29
2.5 Le processus de la méthode ADELFE 31
2.5.1 Survol du processus 32
2.5.2 Spécificités 32
2.6 Bilan 33

II Conception d'un système multi-agent pour la construction d'ontologies dynamiques 35

3 Ingénierie agent basée sur les modèles 37
3.1 Un modèle d'agent pour l'auto-organisation coopérative 38
3.1.1 Les différents modules 38
3.1.2 Fonctionnement interne 39
3.2 Utilisation du paradigme objet 40
3.3 Notations et stéréotypage 41
3.3.1 Le stéréotype de classe « cooperative agent » 42
3.3.2 Les stéréotypes d'attributs et de méthodes 43
3.4 De la conception à l'implémentation 45
3.4.1 Positionnement MDA 45
3.4.2 Du modèle statique à la structure objet de l'agent 46
3.5 Bilan 49

4 Dynamo : système multi-agent pour les ontologies dynamiques 51
4.1 L'ontologie en tant que système multi-agent 52
4.2 Vue d'ensemble de Dynamo 53
4.2.1 Application d'ADELFE et choix fondamentaux 53
4.2.2 Architecture proposée 54
4.2.3 Structure interne des agents-concepts 56
4.3 Hiérarchie basée sur un procédé de classification 58
4.3.1 Rappels de classification 58
4.3.2 Un algorithme distribué de classification 60
4.3.3 Constat qualitatif 62
4.3.4 Exemple 63
4.4 Hiérarchie multi-critère 64

4.4.1 Ajouter les règles de « couverture en tête » . . . 64
4.4.2 De l'utilisation de plusieurs critères . . . 66
4.5 Simplification de l'arborescence . . . 67
4.5.1 Elimination des branches dégradées . . . 67
4.5.2 S'éloigner des arbres binaires . . . 68
4.6 Bilan . . . 70

III Expérimentations et Analyses **71**

5 Evaluation du système **73**
5.1 Performances du système . . . 73
5.1.1 Complexité de l'algorithme distribué de classification . . . 74
5.1.2 Impact des règles de « couverture en tête » . . . 75
5.1.3 Impact de l'élimination des branches dégradées . . . 76
5.1.4 Impact du passage aux arbres n-aires . . . 77
5.2 Application . . . 79
5.2.1 Protocole expérimental . . . 79
5.2.2 Exécution automatique . . . 80
5.2.3 Intervention manuelle . . . 82
5.3 Bilan . . . 85

6 Perspectives **87**
6.1 Relations transverses . . . 87
6.2 Apprendre sur les règles de calcul . . . 89
6.3 Evaluer l'apport des ontologies formelles et la réutilisation . . . 90
6.4 Améliorer l'algorithme de classification . . . 91
6.5 Enrichir l'interaction avec l'utilisateur . . . 92
6.6 Bilan . . . 92

Conclusion **95**

IV Annexes **99**

A Corpus **101**
A.1 REACHIR . . . 101
A.2 Astronomy & Astrophysics (Astro) . . . 103

B Ontologies **105**

B.1 Ontologie de référence pour le corpus Astro 105

B.2 Brouillon d'ontologie obtenue sur le corpus Astro par Dynamo 107

B.3 Ontologie obtenue sur Astro après la première modification 109

B.4 Ontologie obtenue sur Astro après la deuxième modification 112

B.5 Ontologie obtenue sur Astro après la troisième modification 114

B.6 Ontologie obtenue sur Astro après la quatrième modification 116

C Mesure de similitude entre taxonomies **119**

C.1 Définitions 119

C.2 Comparaison de taxonomies 119

Bibliographie **121**

Liste des figures **129**

Liste des exemples **131**

Introduction

« Lorsque l'on ne comprend pas quelque chose, on le considère comme "vide" de sens pour soi, mais ce n'est pas un vrai "vide". Tout cela n'est qu'égarement. »

Miyamoto Musashi – Traité des cinq roues

« Construction d'ontologies à partir de textes à l'aide de systèmes multi-agents adaptatifs » ; derrière ce titre, se cache la volonté de faire découvrir les possibilités offertes par une problématique liée à trois domaines différents : l'ingénierie des connaissances, le traitement automatique des langues et l'ingénierie des systèmes multi-agents.

Motivations

Depuis plusieurs années, le Web sémantique est considéré comme la prochaine évolution possible pour l'accès à la connaissance en réseau. Toutefois, une des difficultés majeures pour le développement du Web sémantique est son besoin en connaissances structurées. C'est pourquoi une grande part des recherches se concentrent sur la formalisation des ontologies ce qui a, entre autre, donné naissance à des langages *ad hoc* comme OIL[1] [Horrocks *et al.*, 2000], DAML[2] [Horrocks, 2002] et plus récemment OWL[3] [Dean et Schreiber, 2004] proposé comme un standard par le World Wide Web Consortium (W3C). Bien entendu, il y a bien d'autres recherches sur les ontologies, visant à faciliter leur utilisation, comme le stockage, l'alignement d'ontologies différentes pour un même domaine, les réflexions sur la réutilisation et leur intégration dans des applications. Ces axes étant pour la plupart parvenus à une certaine maturité, nous nous concentrons dans ces pages essentiellement sur la construction des ontologies elles-mêmes. Cette problématique fait l'objet de recherches depuis près de dix ans mais elle reste pourtant un des verrous majeurs pour le développement du Web sémantique.

[1] Ontology Inference Layer
[2] DARPA Agent Markup Language
[3] Web Ontology Language

De la construction d'ontologies aux ontologies dynamiques

Depuis des années, la communauté scientifique a bien identifié les difficultés et le coût inhérents à l'acquisition des connaissances en général et à la construction d'ontologies en particulier. Ce travail est réalisé par un analyste. Outre ces contraintes à prendre en compte, l'analyste doit toujours s'assurer que le modèle qu'il est en train de produire est conforme aux objectifs d'*adéquation* visés du point de vue de l'application qui exploitera l'ontologie mais aussi du domaine couvert.

Pour se donner ce type de garanties, l'intérêt de travailler avec des experts, mais aussi avec des textes, est important. Or, il est aujourd'hui avéré que le travail à partir de *corpus textuels* permet de réduire l'effort de construction d'ontologies [Bourigault et Aussenac-Gilles, 2003; Szulman *et al.*, 2002; Maedche et Staab, 2000b; Daille, 1994]. Pour cela, il convient d'utiliser un outillage adapté afin de trouver les informations pertinentes, en provenance des textes, pour la tâche de construction. Toutefois, l'analyste est noyé par la masse d'informations qui doit être examinée, et les systèmes actuels ne sont pas capables de les discriminer automatiquement pour sélectionner ce qui est réellement exploitable.

C'est pourquoi on se repose encore essentiellement sur l'analyste qui apporte sa propre vision, critique, et sélectionne. Les travaux réalisés à ce jour ont donc très probablement augmenté la pertinence du produit résultant, et réduit les coûts de construction en aidant l'analyste, mais il reste beaucoup à faire. En particulier, les coûts devraient pouvoir être encore réduits en s'attaquant aux difficultés citées ci-dessus : en présentant seulement les éléments les plus pertinents tirés des textes et en cherchant automatiquement des informations plus précises. C'est pourquoi, depuis peu, un nouveau courant a émergé dans la communauté, celui des *ontologies dynamiques*.

Vers un système adapté : Dynamo

Le but avoué du courant des ontologies dynamiques est d'automatiser au maximum le processus de construction à partir de textes [Heflin et Hendler, 2000]. Ainsi, on introduit la dynamique dans le procédé de fabrication, et non dans le produit lui-même qui reste inerte, en quelque sorte un consommable du Web sémantique. Pourquoi alors ne pas aller plus loin en ayant une *structure elle-même dynamique* ? Il s'agit là d'un but séduisant, en rupture avec les approches classiques où une ontologie est une référence non révisable, contrairement à la connaissance qui est en évolution constante. Ce statut consacré et figé de l'ontologie provient principalement de la limitation des outils actuels qui ne sont pas conçus pour gérer cette évolution.

En général, les agents utilisent les ontologies pour fonctionner. A notre connaissance, aucun travail n'utilise les systèmes multi-agents pour les créer. Mais en cherchant à matérialiser ces ontologies dynamiques, nous nous sommes concentrés sur l'utilisation de systèmes multi-agents adaptatifs comme moyen pour atteindre ce nouveau but. En effet, par leur biais, nous pouvons construire des systèmes *à fonctionnalité émergente* dont les entités sont en interaction constante[4]. Ainsi, la construction d'ontologies redevient une tâche de concep-

[4]On parle de système à fonctionnalité émergente lorsque la fonctionnalité du système n'est pas explicitement exprimé dans les entités constituantes du système. Le comportement observé est obtenu par l'interaction de ces

tion, mais en se plaçant dans un contexte de *conception vivante* et de structure maléable en fonction des interactions avec l'utilisateur. C'est le point de vue adopté pour notre système nommé Dynamo (DYNAMic Ontologies).

Cette idée originale n'a jamais été expérimentée à notre connaissance. Tout est à définir et s'inscrit dans un courant utilisant le traitement automatique de la langue (TAL) pour parvenir à une ontologie. Une première étape de la recherche a donc consisté à définir la part d'analyse des textes prise en charge par les agents et la part traitée par des logiciels de TAL existants. Nous avions déjà cherché à évaluer la faisabilité d'une chaîne de TAL basée sur un système multi-agent [Ottens, 2004] et les résultats obtenus nous ont poussé à faire intervenir le système multi-agent adaptatif en aval d'outils de TAL plus classiques. Ainsi, chaque agent de notre système représente un concept de notre ontologie et cherche sa place dans l'organisation en utilisant des analyses statistiques basées sur un réseau terminologique créé par un extracteur de termes. Pour cela, nous nous reposons principalement sur un algorithme distribué de classification et des comportements complémentaires implémentés dans nos agents pour construire la composante taxonomique de l'ontologie.

Toutefois, cette approche soulève des difficultés inhérentes à la création de tels systèmes adaptatifs. C'est pourquoi notre contribution comporte également un volet concernant l'*ingénierie agent*. Le domaine d'application étant complexe, notre système doit être construit avec rigueur, et notre apport pour l'ingénierie des systèmes multi-agents se place dans cette optique méthodologique, proche du génie logiciel.

Structure du mémoire

Ce mémoire, structuré en trois parties, présente les résultats des recherches que nous avons effectuées pour la conception d'ontologies dynamiques, et pour faciliter l'implémentation de systèmes multi-agents adaptatifs pour des domaines complexes.

La première partie est composée de deux chapitres. Le chapitre 1 décrit les développements de l'ingénierie des connaissances partant de la construction d'ontologies à partir de textes jusqu'à l'idée récente des ontologies dynamiques. Le chapitre 2 se concentre sur l'ingénierie des systèmes multi-agents ; il présente tout d'abord leurs liens traditionnels avec les ontologies, puis il détaille la classe des systèmes multi-agents adaptatifs qui nous concerne plus particulièrement, notamment sur un plan plus méthodologique.

La seconde partie est dédiée aux travaux qui ont conduis à l'élaboration de notre système. Tout d'abord, le chapitre 3 expose les développements méthodologiques qui ont été nécessaires pour faciliter l'implémentation de systèmes multi-agents adaptatifs dans des domaines complexes. Pour cela, nous définissons un pont possible entre conception et implémentation qui est souvent sous-estimé. Le chapitre 4 montre les points sur lesquels nous avons dû nous déterminer pour concevoir notre système, puis il détaille l'architecture de notre système supportant nos choix, enfin il se concentre sur la structure interne de nos agents ainsi que leur comportement pour obtenir une aide à la construction d'ontologies à partir de textes.

La troisième partie dresse un bilan des possibilités et limites de notre système. Le cha-

entités.

pitre 5 met en lumière ses forces et faiblesses, à la fois du point de vue des performances brutes, de la qualité des résultats obtenus et de l'interaction avec l'utilisateur. Enfin, le chapitre 6 présente un bilan sur l'état actuel de Dynamo et les perspectives envisageables à plus ou moins long terme.

Première partie

Contexte et état de l'art

1 Vers les ontologies dynamiques

« Rien n'est statique. Même La Joconde tombe en morceaux. »
Chuck Palahniuk – Fight Club

DEPUIS une quinzaine d'années, les recherches sur la construction d'ontologies à partir de textes sont considérées comme une issue possible aux problèmes du coût et de la difficulté de construction de ces ontologies, qui conditionnent leur utilisation, en particulier dans le cadre du Web sémantique. Notre recherche se situe dans ce cadre. A l'heure actuelle, ce domaine a sélectionné les outils de Traitement Automatique des Langues (TAL) les plus pertinents. Il a établi des propositions méthodologiques pour passer des textes aux ontologies et pour assurer leur maintenance, qui ont nécessité une réflexion sur l'articulation entre termes et concepts. Enfin, depuis cinq ans, plusieurs plateformes d'ingénierie ont vu le jour intégrant certains des outils et éléments méthodologiques dégagés, les plus récents allant vers une automatisation complète du processus sous le nom d'*ontology learning* [Maedche, 2002].

Dans ce chapitre, nous présentons les travaux d'ingénierie des connaissances qui ont nourri notre réflexion pour cette thèse. Toutefois, il ne vise pas à couvrir le domaine de manière exhaustive car la littérature offre ce type de panorama [Charlet, 2003]. Notre contribution se démarque des approches traditionnelles dont elle reprend finalement peu d'éléments. Cependant, nous insisterons sur les besoins et résultats en matière d'analyse de textes car notre objectif est bien de construire et maintenir des ontologies à partir de textes.

Ce chapitre introduit dans la section 1.1 ce que sont les ontologies, et dresse un rapide état de l'art sur leur construction et leur maintenance. Dans la section 1.2, nous présentons ce qu'est la notion de corpus textuel ainsi que son utilisation en ingénierie des connaissances. Puis, nous mettons en avant, dans la section 1.3, les liens entre les notions de termes et de concepts. Dans la section 1.4, nous donnons un aperçu de l'usage du traitement automatique des langues pour la construction d'ontologies à partir de textes. La section 1.5 permet d'examiner les limites de l'approche par corpus pour la construction et la maintenance d'ontologies. Enfin, la section 1.6 synthétise les principes retenus pour Dynamo.

1.1 Des ontologies

La notion d'ontologie intéresse à la fois l'ingénierie des connaissances, la linguistique et la philosophie. Initialement, l'Ontologie était un domaine de la philosophie concernant « l'étude de l'être en tant qu'être, c'est-à-dire l'étude des propriétés générales de ce qui existe »[1]. Or, progressivement les objets produits par cette discipline ont pris le nom d'ontologies. Pour plus d'informations sur les liens entre philosophie et ingénierie des connaissances on pourra se reporter à [Bachimont, 2004].

1.1.1 Définition

En ingénierie des connaissances, une ontologie est un modèle de connaissances qu'il convient de construire. Cette tâche de construction est dévolue à un analyste que nous appellerons ontologue – nous aurions pû aussi l'appeler cogniticien dans la tradition de l'ingénierie des connaissances. Il effectue généralement ce travail de construction en se reposant sur des textes et en interrogeant des expert du domaine à modéliser.

Une ontologie est définie comme une description rigoureuse et structurée du vocabulaire, au sens de la logique, d'un domaine spécialisé [Bachimont, 2004]. Elle est constituée d'un ensemble de concepts structuré par des relations sémantiques (décrivant les propriétés des concepts) parmi lesquelles la relation « est-un » joue le rôle privilégié de relation d'héritage de propriétés entre les concepts [Charlet, 2003]. Cette structuration est à rapprocher des réseaux sémantiques qui permettent de représenter les concepts de l'ontologie (les nœuds du réseau) et les relations entre eux (les arcs du réseau). La figure 1.1 donne un extrait d'un tel réseau composé de deux parties de la hiérarchie conceptuelle et disposant de deux relations : « est-un » notée ici « is-a », et une relation sémantique « produces ». Toutefois, un tel réseau n'est, en principe, une ontologie que si les concepts sont définis de manière consensuelle et s'ils ne disposent que des propriétés nécessaires et suffisantes pour les caractériser. En pratique, les concepts et relations sont retenus en fonction du point de vue adopté sur le domaine modélisé [Charlet, 2003].

Une ontologie ayant vocation à fournir un modèle de connaissances permettant de raisonner à l'aide de la logique, plusieurs langages ont été définis pour formaliser et opérationnaliser des ontologies. Aujourd'hui, nous disposons d'un standard récent nommé OWL[2]. Ces langages permettent de manipuler et de spécifier des classes, des propriétés de ces classes et des restrictions sur ces propriétés. OWL-DL et OWL-Full deux variantes de OWL, permettent de définir des axiomes [Gandon, 2002; Baget *et al.*, 2004].

Une fois l'ontologie d'un domaine constituée, on peut définir des instances des classes qui devront vérifier les propriétés, les contraintes de leur classe ontologique et satisfaire les axiomes. Lorsque l'on considère le couple ontologie et ensemble d'instances, on peut parler d'une ontologie lourde qui pourra être complétée en base de connaissances si on y ajoute des règles d'inférences utilisant les concepts de l'ontologie.

[1] cf. l'article de Wikipédia : `http://fr.wikipedia.org/wiki/Ontologie_(philosophie)`
[2] Recommandation du W3C, cf. `http ://www.w3.org/2001/sw/WebOnt/`

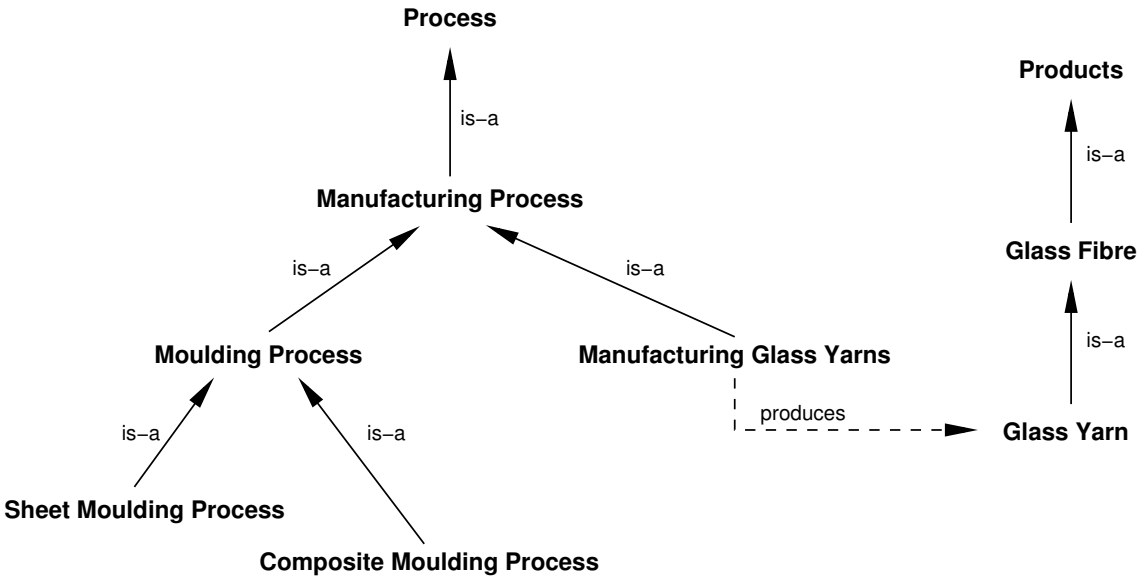


Figure 1.1 — Extrait d'une ontologie du domaine de la fabrication de la fibre de verre [Aussenac-Gilles et Busnel, 2002]

1.1.2 Motivations

Historiquement, la notion d'ontologie est apparue pour satisfaire des besoins d'interopérabilité dans les systèmes informatiques et de réutilisation. On attend d'elles qu'elles améliorent la communication non seulement entre machines, mais aussi entre humains et machines ou encore entre humains par le biais de logiciels. Les propriétés de ce type de structure de données ont permis de diversifier leur utilisation à différentes applications, en particulier la gestion des connaissances et le Web sémantique. Elles sont utilisées pour :

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

[Bergenti *et al.*, 2002] ; l'ontologie offre alors une description commune des objets utilisés et tient lieu de connaissance partagée ; la standardisation des formats est alors une clef pour assurer leur interprétation par les agents ;
– annoter des ressources à l'aide de méta-données ; cette annotation est automatisée dans un système comme Magpie [Domingue *et al.*, 2003] ; elle permet d'associer des objets formels et consensuels reflétant le sens de l'information qui se trouve dans la ressource annotée en vue d'offrir des services (navigation, recherche, etc.) sur ces ressources ; ainsi, on rend compte d'un point de vue sur les ressources ;
– améliorer les processus de recherche d'informations [Baziz, 2005] grâce à une indexation plus riche ou encore en introduisant une meilleure interprétation des requêtes utilisateur en langage naturel ; on peut ainsi dépasser les approches par mots-clefs en s'intéressant aux concepts manipulés par l'utilisateur ou en fournissant des réponses dégradées là où un système classique ne donnerait pas de résultat.

1.1.3 Diversité des types d'ontologies

Paradoxalement, le terme « ontologie » est associé à une grande variété de types d'objets. La plupart des ontologies sont au moins structurées à l'aide de la relation « est-un » qui est la relation de généralisation entre concepts ; ainsi on peut définir des hiérarchies de concepts. L'autre relation très utilisée est la relation tout-partie (« est composé de »). Ainsi, on peut considérer les taxonomies comme des variétés d'ontologies n'utilisant que la relation de généralisation. D'ailleurs, tout comme les taxonomies, la hiérarchie d'une ontologie est fortement influencée par son but [Charlet, 2003].

Elles peuvent aussi varier par rapport à d'autres métriques comme le nombre d'axiomes contraignant le sens des concepts (en opposition à une simple hiérarchie de concepts sans autre définition), ou la taille qui peut aller de quelques dizaines de concepts à des milliers. Enfin, elles peuvent être générales ou ne couvrir qu'un domaine particulier. Pour les ontologies couvrant un domaine particulier, la construction à partir de corpus textuels est souvent utilisée ; nous allons à présent aborder cet aspect.

1.1.4 Construction et maintenance des ontologies à partir de texte

Les premières initiatives en matière de construction et de maintenance d'ontologies ont eu pour objectif de définir un processus systématique selon les mêmes critères que le génie logiciel. Parmi les méthodes les plus significatives citons METHONTOLOGY [Fernandez-Lopez *et al.*, 1997] et On-To-Knowledge [Sure *et al.*, 2003]. Ces méthodes fournissent alors peu d'aide sur les modalités d'extraction des connaissances et sur la manière de choisir le contenu de l'ontologie. D'autres approches comme OntoClean [Guarino et Welty, 2003] font référence à des travaux théoriques sur la nature des ontologies et, par là même, orientent le choix du contenu.

Or, s'intéresser au contenu requiert de s'intéresser à l'expression des connaissances. Il devient alors impossible d'ignorer la langue, vecteur principal de la transmission de connaissances. De plus, depuis quelques années, avec l'essor du Web sémantique, l'articulation entre langue et connaissance est la source d'un nouvel enjeu : celui de l'utilisation de mo-

dèles de connaissances pour annoter les ressources du Web qui sont essentiellement textuelles.

C'est pourquoi, une des facettes de l'ingénierie des connaissances est d'étudier l'extraction de connaissances à partir de textes. Les premiers travaux qui se sont intéressés à la construction d'ontologies à partir de textes datent de 1995. De manière complémentaire aux approches formelles, ils abordent la question d'un point de vue plus pragmatique faisant appel à des outils de TAL pour déboucher sur des propositions méthodologiques [Bourigault et Jacquemin, 2000; Bourigault et Aussenac-Gilles, 2003].

Par exemple, la méthode Terminae [Szulman *et al.*, 2002] identifie quatre phases principales dans le processus de construction :

1. constituer le corpus, en sélectionnant les textes pertinents du domaine pour l'application ;
2. effectuer une analyse terminologique à l'aide d'outils de TAL pour identifier des indices linguistiques de connaissance (termes, relations lexicales, classes sémantiques, etc.) ;
3. normaliser l'ontologie, en organisant les concepts au sein d'un modèle non formel ;
4. formaliser l'ontologie, afin de répondre aux exigences d'une interprétation logique.

Aujourd'hui, dans la continuité de ces démarches d'ingénierie des connaissances, les travaux portent sur l'intégration des outils de TAL et des méthodes pour constituer des plateformes comme Text2Onto [Maedche et Staab, 2000b]. Une autre tendance, liée à la pression du Web sémantique qui exige de disposer rapidement d'ontologies et de les peupler, est d'introduire plus d'automatisation via des techniques d'apprentissage [Buitelaar *et al.*, 2005].

Quelle que soit l'approche retenue, la constante reste le point de départ du processus de construction : le corpus textuel. Ainsi les sections suivantes s'intéressent au corpus ainsi qu'aux outils d'analyse de textes et aux problèmes qu'ils soulèvent.

1.2 Un point de départ : le corpus textuel

Le *corpus* est devenu un objet d'étude à part entière. Cette notion présente en revanche de nombreux visages dans la littérature. En effet, son statut et sa définition sont variables en fonction de la population considérée [Pearson, 1998]. C'est pourquoi nous nous attacherons en premier lieu à présenter l'objet que nous plaçons derrière ce nom. En second lieu, nous montrerons les usages et enjeux tournant autour de cette notion, à la lumière de l'ingénierie des connaissances sur laquelle se focalise nos travaux avec, en particulier, la construction de ressources terminologiques ou ontologiques.

Dans ce contexte nous associerons ressources termino-ontologiques et ontologies formelles sous le même nom générique d'ontologies. Cette relative imprécision dans les termes utilisés est justifiée selon nous par la nature du travail à partir de corpus. En effet, les objets bruts obtenus depuis les textes sont de nature termino-ontologique – c'est-à-dire présentant à la fois des informations d'ordre terminologique et d'ordre conceptuel. De ces ressources il est possible d'obtenir des ontologies formelles en passant par des phases de normalisation et de formalisation que nous ne présentons pas ici [Bachimont, 2004]. S'agissant donc

d'un matériau brut destiné à devenir une véritable ontologie, il nous parait légitime de parler uniquement d'ontologies dans la suite de ce document pour les ressources termino-ontologiques.

1.2.1 Définition

Selon nous, la meilleure définition d'un corpus textuel est celle donnée dans [Condamines, 2003] : « Collection de textes (éventuellement un seul texte) constituée à partir de critères linguistiques ou extra-linguistiques pour évaluer une hypothèse linguistique ou répondre à un besoin applicatif ». Elle est intéressante à plus d'un titre. En effet, cette définition dispose d'un point de vue favorisant l'utilisation de textes réels, et, par ce biais, intègre l'existence d'une variation au sein de la langue. De plus, elle présente l'avantage de tenir compte d'études sur corpus dont le but n'est pas une exploration purement linguistique. Elle est alors suffisamment souple pour être partagée par différentes disciplines dont l'ingénierie des connaissances, qui nous intéresse plus particulièrement. Enfin il s'agit d'une définition ayant une volonté unificatrice de la notion même de corpus, recouvrant notamment les définitions de [Atkins *et al.*, 1992] et [Sinclair, 1996].

1.2.2 Linguistique de corpus et ingénierie des connaissances

Il est alors légitime de se poser la question des spécificités de l'ingénierie des connaissances en tant que domaine utilisateur de la linguistique de corpus. Un point important provient du fait qu'il s'agit d'une ingénierie. En tant qu'ingénierie, ce domaine définit des travaux de conception (construction d'ontologies, par exemple) laissant une place importante aux choix de la personne en charge de la modélisation. Elle n'utilise donc pas les résultats des analyses pour constater des phénomènes linguistiques particuliers, mais pour aller plus vite dans la tâche de conception. Ce domaine laisse alors une plus grande part à l'interprétation que la linguistique : le corpus est un appui nécessaire mais insuffisant.

Dans ce cadre particulier, nous nous intéressons à dégager certains indices dénotant la présence de connaissances au sein du corpus. La littérature [Hearst, 1992; Séguéla, 2001; Assadi, 1998; Szulman *et al.*, 2002; Bourigault et Jacquemin, 2000; Maedche et Staab, 2000b; Daille, 1994] présente de nombreux critères, statistiques ou non, utilisés en ingénierie des connaissances ou en terminologie. Nous ne ferons pas une liste exhaustive de ces critères par soucis de concision, mais ils peuvent aller de métriques simples comme la co-occurence de termes, ou leur fréquence à des dispositifs plus complexes comme l'utilisation de patrons morpho-syntaxiques permettant d'identifier des régularités dans les constructions syntaxiques du corpus.

Il s'agit bien évidemment d'un aspect très actif du traitement automatique de la langue directement utilisé en ingénierie des connaissances. Le but principal de cette utilisation des corpus est de pouvoir se baser le plus possible sur les usages des termes pour réaliser un travail de modélisation de qualité, mais aussi, pour une grande part, de réduire les coûts (ne serait-ce que simplement le temps passé à la réalisation).

En vue de la construction d'une ontologie couvrant un domaine technique précis, se pose

la question de constituer un corpus textuel adapté. Le choix des textes se portera alors, suivant leur disponibilité, sur des textes didactiques ou descriptifs sur les notions et procédés utilisés dans le domaine à couvrir [Condamines, 2003; Bourigault *et al.*, 2004].

L'application ciblée contraint le contenu ontologique du résultat, ce qui influence le choix des textes du corpus à étudier. Le corpus peut être ajusté en fonction des premiers résultats de son analyse par des outils de TAL. Par exemple, malgré une sélection stricte des textes du domaine lors de la construction du corpus, les ontologues constatent parfois des phénomènes de *sous-corpus* [Aussenac-Gilles et Busnel, 2002]. Il s'agit de sous-ensembles de textes du corpus initial correspondant à un sous-domaine encore plus spécialisé et parasitant les analyses. Il convient alors de scinder le corpus initial pour éviter un biais au moment d'interpréter les données extraites des textes. Enfin, pour certaines applications, il existe peu de textes sur le domaine ce qui contraint les choix pour la construction du corpus. Dans un tel cas, il paraît indiqué de garder cette contrainte à l'esprit et de faire preuve de précautions supplémentaires pendant la modélisation, en faisant d'autant plus appel à des experts du domaine par exemple. Pour plus d'informations, on pourra se reporter à [Condamines, 2003].

1.3 Des termes et des concepts

Dans le cadre des interactions constatées entre la terminologie et l'ingénierie des connaissances via les corpus, il apparaît nécessaire de se pencher plus en détail sur certaines notions capitales, notamment la distinction faite entre *terme* et *concept*. En effet, en travaillant à partir de corpus, l'ontologue est en présence d'un nombre important de termes du domaine. Or, il cherche à construire une ontologie et donc à manipuler et créer des concepts. Pour faire ce passage entre corpus – domaine des termes – et ontologie – domaine des concepts – dans nos travaux, il nous faut nous attarder sur les termes et les concepts.

1.3.1 Définitions

Le problème de définir de façon rigoureuse et de distinguer avec précision la frontière entre les notions de terme et de concept est une des questions au cœur de la linguistique de corpus.

Préalablement, il semble essentiel d'étudier l'opposition ontologique entre *syntagme* et *terme*. Un syntagme peut-être défini comme un ensemble de mots formant une seule unité fonctionnelle ou catégorielle, tandis qu'un terme est un élément de vocabulaire de spécialité lié à un domaine. Il y a alors opposition ontologique dans le sens où le syntagme peut-être vu comme une association libre de mots, alors que le terme peut-être vu comme un syntagme renvoyant à une signification spécifique au domaine. Le terme a alors un statut particulier du point de vue d'un domaine donné.

Le courant terminologique actuel introduit un « item tiers » pour briser la dichotomie. On rencontre deux cas : soit cet item tiers est vu comme un terme (et dans ce cas, les termes sont découpés en termes hautement techniques, termes techniques et termes sous-techniques, comme le préconisent Hoffman ou Trimble cités par [Pearson, 1998]), soit

cet item tiers se situe dans un continuum entre syntagme et terme (Godman et Payne, également cités par [Pearson, 1998]). [Slodzian, 2000] critique ces deux approches, tout en soulignant la perméabilité des deux notions. En effet, ses travaux, portant sur l'étude du fonctionnement réel des unités en discours selon une approche textuelle, l'ont amenée à définir la notion de « candidat terme » qui ne deviendra terme qu'à l'expresse condition d'être accepté par un expert du domaine. Cette position découle d'une école de pensées qui souligne le caractère contextuel d'une telle construction liée aux occurrences observées en corpus. À l'opposé, une démarche purement logicienne, initiée par Wüster à la fin des années 30, définit le terme comme le représentant linguistique d'un concept dans un domaine de connaissances [Wüster, 1976], le concept étant considéré comme stable et existant indépendamment de la langue.

De façon informelle, on pourrait définir un concept comme ce qui est mobilisé dans notre esprit lorsque nous utilisons une forme linguistique, en production ou en réception. Plus formellement, un concept correspond à une classe d'objets concrets ou abstraits que l'on va chercher à définir en intension, par des caractéristiques qui sont le plus souvent leurs relations avec d'autres classes (par exemple, le concept de vin est lié au concept de cépage). Il sert alors à constituer une abstraction du monde dans un système formel du point de vue d'un domaine donné. Au sein d'une ontologie, on peut voir le concept comme essence : le concept d'un objet est le noyau des propriétés nécessaires vérifiées par un objet, indépendamment des variations qu'il peut subir dans les différents contextes où il se trouve [Bachimont, 2004].

Une fois définies les notions de terme et de concept, on constate l'émergence de deux points de vue opposés sur la façon de mettre en relation un concept et un (ou plusieurs) terme(s). Nous allons donc exposer de façon plus détaillée ces deux positions, débattre de leur bien-fondé dans le cadre de nos recherches, pour ensuite nous positionner.

1.3.2 Comment lier terme et concept ?

On a pu voir précédemment la difficulté à donner des définitions objectives de ce que représentent réellement le terme et le concept en linguistique, en partie à cause du fait que ces deux notions sont fortement corrélées et que leur mise en relation influe fortement sur la vision globale de la construction ontologique. Examinons à présent les deux points de vue qui s'opposent sur la question.

Historiquement la plus ancienne, la *Théorie Générale de la Terminologie* a été fondée par Eugen Wüster à la fin des années trente, dans la mouvance du Cercle de Vienne [Wüster, 1976]. Cette doctrine défend l'idée selon laquelle le monde de la connaissance est découpé en domaines stables, dont chacun est équivalent à un réseau fixe de concepts, les termes en étant les simples représentants linguistiques. Dans cette optique normative, le concept pré-existe donc aux termes. Les concepts d'un domaine sont ainsi organisés selon des relations non linguistiques. De plus, cette école de pensée fait l'hypothèse de la monosémie des termes pour un domaine de spécialité. Le but de la terminologie devient alors de rassembler,

selon une *démarche onomasiologique*[3], les termes correspondant à un concept donné par sa définition. Les plus puristes vont même jusqu'à reprocher à certains spécialistes la polysémie qui peut exister dans leur domaine, les accusant de manquer de « créativité néologique ».

Toutefois, on se rend bien compte de la portée limitée que peut avoir un tel point de vue étant donnée l'évolution qu'a connu la théorie terminologique. En effet, la multiplication des types de ressources terminologiques (notamment avec l'apparition de la notion de Web sémantique) et l'essor de la polysémie dû à l'entremêlement des domaines de spécialité ou le passage de termes issus d'un domaine vers la langue générale [Meyer et Mackintosh, 2000], mettent à mal le principe de l'unicité et de la fixité du réseau conceptuel. La position wüstérienne, qui consiste, au final, à nier la forte corrélation entre une terminologie et son domaine d'application, s'avère désormais bel et bien dépassée.

Nous préférons donc adopter une *démarche sémasiologique*[4], qui prône de partir des manifestations linguistiques dans le but d'identifier des termes et leurs relations. Une telle vision de la terminologie avance que les connaissances pertinentes pour un domaine se trouvent dans les textes rédigés par une communauté scientifique ou technique, autant (voire plus) que chez l'auteur du terme ou dans les définitions qu'il a produites. Ce point de vue novateur a inspiré de nombreuses approches qui commencent par une analyse lexicale, syntaxique et/ou sémantique du corpus d'étude pour en déduire un ensemble de termes, les rassembler au sein de concepts, puis étudier leurs relations éventuelles. Par exemple, [Harris, 1968] aborde le problème en développant l'idée suivante : il est possible de regrouper certains termes d'un corpus si leurs occurrences apparaissent dans des contextes similaires. En pratique, il pose qu'étant donné les relations de dépendances syntaxiques entre syntagmes, ceux ayant un sens proche se caractérisent par des dépendances similaires. La mise en œuvre se découpe en deux parties : collecte des dépendances syntaxiques puis analyse de leur distribution. Il donne ensuite le moyen de comparer ces distributions pour regrouper les syntagmes proches en signification [Slodzian, 2000]. Idéalement, l'ontologue pourra obtenir des concepts en examinant les résultats de l'analyse syntaxique et lexicale, qui sont situés à un niveau terminologique.

1.4 Traitement automatique des langues pour la construction d'ontologies à partir de textes

Aujourd'hui, la contribution du traitement automatique des langues à la construction d'ontologies prend la forme de plusieurs types de logiciels utilisés pour repérer, dans des textes, des formes linguistiques révélatrices de connaissances. Notre but n'est pas ici d'en dresser une liste exhaustive, mais plutôt d'en dégager les tendances fortes et les principales caractéristiques.

[3] qui part d'une idée vers ses expressions dans une langue, vers des mots
[4] qui part des manifestations langagières vers les idées

1.4.1 Place des outils et point de vue sémantique

Parmi les systèmes les plus utilisés, on trouve des concordanciers, des classifieurs, des extracteurs de termes ou de relations et, des outils plus sophistiqués de regroupements sémantique ou conceptuel [Bourigault et Aussenac-Gilles, 2003]. Soit ces systèmes s'appuient sur des techniques purement linguistiques, soit ils combinent critères statistiques et analyses linguistiques. Nous allons passer en revue quelques-uns des outils de traitement automatique des langues utilisables pour la construction d'ontologies à partir de textes.

Identification de termes

Pour permettre l'identification de termes dans le corpus, on utilise des extracteurs de terme dont le travail est de détecter au sein du corpus les groupes nominaux intéressants. Pour cela les extracteurs de termes se reposent généralement – mais pas tous – sur des analyses syntaxiques leur permettant de découper les groupes nominaux. Nous présentons à présent trois extracteurs de termes de conception différente.

Syntex [Bourigault, 2007] est utilisé comme une aide à l'identification de termes pour le français ou l'anglais. Pour cela, il se repose sur un étiquetage morpho-syntaxique préalable du corpus (associant chaque instance d'un mot à une catégorie grammaticale et à sa forme canonique), mais aussi sur un ensemble de règles d'analyse grammaticale, syntaxique et lexicale. En entrée, il reçoit donc un texte étiqueté grammaticalement comme la phrase de l'exemple 1.1. En sortie, il permet d'obtenir des propositions de dépendances syntaxiques (voir exemple 1.2) sur l'ensemble des phrases du corpus ainsi que les termes utilisés. La caractéristique la plus intéressante de cet outil est la création automatique d'un réseau de termes basé sur les contextes d'usage des termes extraits. En effet, il ne retourne pas simplement une liste à plat de termes mais un réseau complet, dans lequel les termes sont reliés soit par des relations syntaxiques, soit par des proximités sémantiques calculées à partir de l'analyse distributionnelle. De plus, chaque terme dispose de critères numériques correspondant à sa fréquence ou à sa productivité (son utilisation dans des termes plus complexes).

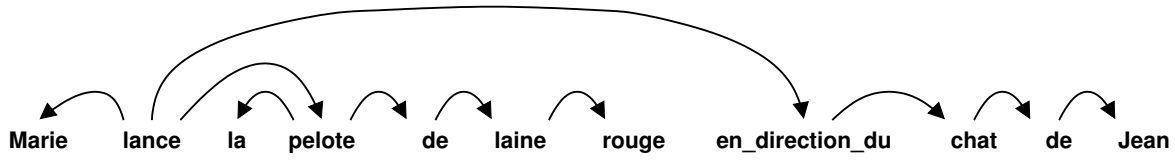
Un autre outil performant d'extraction de termes est *Termostat* [Lemay *et al.*, 2005]. Il repose aussi sur des connaissances linguistiques et effectue la comparaison entre l'utilisation d'un terme dans un corpus spécialisé et son utilisation dans un corpus de langue générale pour déterminer s'il semble pertinent. Toutefois, il ne retourne qu'une simple liste de termes donnant le terme identifié, sa forme lemmatisée, sa fréquence et le poids indiquant sa pertinence probable pour le domaine du corpus.

À ce type de systèmes, on pourra opposer des extracteurs de termes comme *ANA* [Enguehard et Pantera, 1995] qui s'affranchissent totalement de connaissances linguistiques. ANA est conçu pour apprendre des formes permettant d'identifier des termes directement à partir des régularités du corpus. Il utilise pour cela une liste de mots fonctionnels utilisés pour la composition des termes et effectue les comparaisons en utilisant la distance d'édition. De fait, il présente le double avantage de s'adapter à différentes langues et de pouvoir traiter des textes grammaticalement incorrects. En entrée, il reçoit une première liste de termes du domaine ainsi qu'une liste de mots vides (qui n'apportent pas d'information supplémentaire sur le domaine). En sortie, la liste de termes du domaine est augmentée. Il s'agit

Exemple 1.1 Une phrase en français étiquetée par TreeTagger

mot	catégorie	lemme
Marie	NAM	Marie
lance	VER :pres	lancer
la	DET :ART	le
pelote	NOM	pelote
de	PRP	de
laine	NOM	laine
rouge	ADJ	rouge
en	PRP	en
direction	NOM	direction
du	PRP :det	du
chat	NOM	chat
de	PRP	de
Jean	NAM	Jean
.	SENT	.

Exemple 1.2 Proposition de dépendances syntaxiques d'une phrase en français



d'un processus itératif qui peut être répété autant de fois que l'utilisateur le désire, jusqu'à ce qu'il soit satisfait de la liste obtenue. Toutefois, les résultats sont moins riches puisque, nous venons de le voir, aucune analyse syntaxique du corpus ou d'indice sur les contextes d'usage des termes extraits n'est fournie par ANA. Il permet uniquement d'augmenter une liste des syntagmes potentiellement intéressants dans le domaine (voir exemple 1.3).

Identification de classes sémantiques

Lexiclass [Assadi, 1998] et *ASIUM* [Faure, 2000] sont deux exemples d'outils permettant l'identification de classes sémantiques. Tous deux sont basés sur des algorithmes de classification hiérarchique ascendante. Ainsi, leur approche n'est pas dépendante d'une langue particulière. Toutefois, ils présupposent un ensemble de traitements en amont sur le corpus.

Par exemple, ASIUM suppose une analyse syntaxique complète du corpus. Il reçoit alors ce corpus analysé comme entrée pour ensuite utiliser plus particulièrement les relations sujet-verbe. Grâce à ces dernières, il forme les clusters initiaux des termes utilisant les mêmes verbes après la même préposition. Il construit ensuite la hiérarchie en rapprochant les clusters les plus similaires. On obtient alors en sortie une hiérarchie de classes.

En revanche, Lexiclass travaille à partir du réseau de termes obtenu en sortie de Lexter (ancêtre de Syntex). Il reçoit ce réseau de termes en entrée et par son exploration détermine

Exemple 1.3 Liste partielle des termes extraits par ANA sur un corpus d'acoustique

```
- ACOUSTIC ACTIVITY
- ACOUSTIC AMPLITUDES
- ACOUSTIC BOILING NOISE DETECTION
- ACOUSTIC LEAK DETECTION
- ACOUSTIC LEAK DETECTION SYSTEM
- ACOUSTIC PULSE
- ACOUSTIC SIGNAL AMPLITUDE
- ACOUSTIC SOURCE LOCATION
- ACOUSTIC TRANSMISSION IN SGUS PLANT AND LABORATORY MEASUREMENTS
- ACOUSTIC SURVEILLANCE TECHNIQUES FOR SGU LEAK
- ACOUSTIC SURVEILLANCE
- ADVANCED SIGNAL PROCESSING
- AMPLITUDE
- ATTENUATION
- ATTENUATION IN X CELL
- ATTENUATION OF ACOUSTIC SIGNAL
- BACKGROUND NOISE IN A SGU
- BEEN SET
```

Exemple 1.4 Format utilisé par Lexiclass pour la classification des individus [Assadi, 1998]

	ADMISSIBLE	NOMINAL	HT	THT	63KV
CÂBLE	0	0	1	0	1
INTENSITÉ	1	1	0	0	0
LIGNE	0	0	1	1	1
TENSION	1	1	0	0	0

les contextes possibles des individus. L'exemple 1.4 montre les contextes utilisables pour un terme donné ; on voit que « CÂBLE » peut être utilisé avec « HT » mais pas avec « NOMINAL ». Lexiclass utilise alors des mesures de similitudes entre deux termes en fonction des contextes partagés ou non entre ces deux individus. Ensuite, il utilise cette mesure pour effectuer la classification. On obtient donc aussi une hiérarchie de classes en sortie.

Cette hiérarchie de classes regroupant des termes est alors une base intéressante pour produire – après examen de l'ontologue – une hiérarchie de concepts constituant la colonne vertébrale d'une ontologie. En effet, les regroupement obtenus par la classification des termes peuvent servir à construire des concepts. Cette technique ne permet toutefois pas d'obtenir les relations transverses d'une ontologie, elle n'est pertinente que pour aider à construire la composante taxonomique.

Identification de relations lexicales et sémantiques

Prométhée [Morin, 1999] ou *Caméléon* [Séguéla, 2001] permettent de compléter les identifications précédentes. Il se reposent sur des textes étiquetés par des analyseurs grammaticaux pour identifier des relations lexicales ou sémantiques. Pour cela, reprenant l'idée de [Hearst, 1992], ils utilisent des patrons morpho-syntaxiques, décrivant les contraintes qu'une phrase doit suivre pour être identifiée (voir exemple 1.5). Cela constitue donc une ressource dépen-

Exemple 1.5 Un patron morpho-syntaxique

```
Pronom "prendre" Article Nom Adjectif
```

dante de la langue. Lorsqu'un patron est identifié dans les textes entre deux termes, cela constitue un indice sur la présence de la relation correspondante entre ces deux termes. En accumulant de tels indices, l'utilisateur peut choisir de valider ou non la présence de la relation entre les couples de termes. De plus, on notera que Prométhée est capable d'apprendre de nouveaux patrons. Pour cela, il extrait les phrases où un couple de termes connus pour être en relation est présent, puis il cherche un environnement commun à toutes ces phrases. Cet environnement constitue alors le nouveau patron.

Pour l'extraction de relation il est aussi possible d'utiliser GATE [Maynard *et al.*, 2005] qui est une plateforme d'extraction d'informations à partir de textes. Elle permet de rechercher des entités nommées, de définir des patrons de fouille de texte, d'annoter sémantiquement les textes et même d'enrichir une ontologie. Toutefois, l'outil présuppose des compétences en TAL, et l'utilisateur doit entièrement le paramétrer pour pouvoir l'adapter à son projet. En effet, en tant que plateforme, GATE fournit les procédés, mais pas les ressources nécessaires à l'extraction.

1.4.2 Importance de l'interprétation

Comme nous venons de le voir, les outils de traitement automatique de la langue sont une aide pour la tâche de construction d'ontologies. Comme tous les outils de TAL, ils produisent des analyses du langage, des descriptions de phrase ou de discours, parfois des caractérisations sémantiques.

Les systèmes de traitement automatique des langues pour construire des ressources ontologiques que nous venons de voir reposent sur une hypothèse commune. En s'appuyant sur des informations au niveau syntaxique, il sera possible de suggérer à un analyste des regroupements ou des structurations sémantiques ou conceptuelles. Ces principes renvoient aux travaux de Z. Harris, s'appliquant à des langages de spécialité dans des domaines bien délimités. Selon Harris, « Le sens donné à des entités et le sens des relations grammaticales entre celles-ci sont liés à la restriction des combinaisons possibles de ces entités entre elles » [Harris, 1968]. Autrement dit, dans un domaine particulier, la distribution des mots et les configurations syntaxiques dans lesquelles ils se trouvent, ne sont pas illimitées et sont révélatrices du sens de ces mots.

Cependant, ces outils n'ont pas d'autre prétention que de se situer au niveau linguistique. Or, construire une ontologie nécessite de viser une représentation conceptuelle. Ce passage ne peut être que le fruit d'un ensemble de choix et d'un travail d'interprétation. A ce titre, les outils de TAL ne peuvent être considérés comme capables de résultats directement intégrables dans une ontologie. La plupart repose sur des ressources linguistiques ou statistiques et, dans les deux cas, les informations extraites doivent être sélectionnées par l'utilisateur. Par exemple, un syntagme identifié par un outil peut très bien ne pas être pertinent pour l'ontologie. C'est cette idée que véhicule la notion de « candidat terme » vue à la

section 1.3.

En imposant cette distinction entre termes et candidat termes, l'interprétation par un opérateur humain est *de facto* au centre du travail d'ingénierie des connaissances. En plus de sa connaissance du domaine et des objectifs de l'application, l'ontologue s'appuie sur des critères statistiques [Assadi, 1998; Bourigault et Aussenac-Gilles, 2003] pour dégager un maximum d'indices potentiellement pertinents en provenance des textes. Il est alors amené à exclure, après examen, les éléments qui lui paraissent superflus pour la modélisation en cours. Après avoir constitué la première version du modèle, il lui faut faire un retour aux textes [Aussenac-Gilles et Busnel, 2002] pour vérifier si les éléments retenus conservent leur pertinence en contexte et ensuite ajuster l'ontologie en conséquence [Séguéla, 2001].

1.5 Limites de la construction d'ontologies à partir de textes

Comme nous l'avons vu dans les sections précédentes, les approches à partir de textes fournissent une grande quantité d'informations différentes, pas nécessairement pertinentes ou structurées. Le prix à payer des approches à partir de textes est de définir des moyens de sélectionner et d'exploiter ce volume d'informations jusqu'à obtenir une ontologie. De plus, l'enchaînement nécessaire de ces outils est encore mal défini et produit un « effet tunnel » qui rend le processus difficilement itératif. La maintenance des ontologies construites à partir de textes pose alors un problème de cohérence entre les besoins applicatifs courants et les modèles.

1.5.1 Efforts et raffinement du produit résultant

A cause du contenu des corpus, les ontologies construites à partir de textes ne constituent pas un produit fini et raffiné. En effet, les structures obtenues sont beaucoup plus riches sur le plan terminologique, mais malheureusement moins bien formalisées. On parle alors de brouillon d'ontologie ou d'*ontology kick-off* [Sure *et al.*, 2003]. Cela justifie la présence de phases de normalisation (où l'on sélectionne les éléments pertinents pour l'application et améliore la structuration) et de formalisation (où l'on traduit le réseau sémantique obtenu dans un langage formel comme OWL) [Aussenac-Gilles et Condamines, 2003] en aval du travail d'extraction de connaissances à proprement parler.

De plus, même si une quantité intéressante de connaissances peut être retrouvée dans les textes, ils ne contiennent pas l'ensemble des connaissances manipulées dans un domaine. Par exemple, beaucoup de définitions sont implicites, et la connaissance générale n'est habituellement pas présente dans les documents. Cette connaissance peut être intégrée dans la phase de formalisation, en raccrochant les connaissances apprises des textes à une ontologie formelle générale. Il s'agit là d'un des principaux atouts des ontologies formelles telles que *DOLCE* [Gangemi *et al.*, 2002]. En effet, ces ontologies disposent de concepts d'un très haut niveau d'abstraction qui ne sont généralement pas présents dans les textes, et donc dans une ontologie construite à partir de textes. Les utiliser lors de la formalisation permet de corriger cet aspect et d'améliorer la réutilisation.

Enfin, les imperfections précédentes justifient aussi de travailler avec des experts du

domaine qui pourront critiquer ce qui a été produit ou aider à interpréter les informations tirées des textes. Ainsi, l'intervention d'un expert permet d'enrichir, corriger et améliorer la structure de connaissances. Il est donc important d'adopter une approche incrémentale même pendant l'extraction de connaissances afin de pouvoir compléter progressivement la structure avant même les phases de normalisation et de formalisation.

1.5.2 La maintenance : un problème mal géré

Un des arguments en faveur de la construction d'ontologies à partir de textes est d'améliorer la maintenance des modèles obtenus [Aussenac-Gilles et Condamines, 2003]. En effet, lorsque le modèle conserve un lien vers les textes utilisés, ceux-ci fournissent des informations qui justifient la manière dont les connaissances ont été modélisées. Ils servent de documentation du modèle et, *a priori*, facilitent la maintenance. D'autres éléments importants à conserver sont les critères et principes d'organisation des concepts ainsi que la justification de la sémantique des relations définies.

Même lorsque le modèle est documenté, la maintenance n'en reste pas moins délicate [Aussenac-Gilles et Condamines, 2003] : il est souvent difficile de retrouver la logique de l'analyste qui a choisi d'organiser les concepts d'une certaine façon, ou encore de savoir comment une évolution du domaine ou des besoins liés à l'application doivent se répercuter sur l'ontologie. Si on se réfère aux textes comme indicateurs de l'évolution des connaissances ou de la terminologie du domaine, le problème revient à savoir comment prendre en compte de nouveaux textes, comment les traiter (séparément ou avec les premiers textes), puis comment intégrer les connaissances extraites de ces textes dans la structure existante.

Aujourd'hui, les ontologies sont vues comme des structures figées, relativement stables, décrivant des définitions consensuelles et fixées. Implicitement, on s'attendrait donc à ce qu'elles nécessitent une maintenance réduite. Or pratiquement, en particulier dans les cas de domaines spécialisés innovants et évoluant rapidement, certaines ontologies doivent régulièrement être revues, surtout au niveau terminologique. On peut aussi expliquer cette nécessité par le fait que beaucoup d'ontologues ne respectent pas de critères ontologiques et construisent plutôt des réseaux sémantiques hiérarchisés. Dans ce contexte, le scénario de construction de l'ontologie et celui de maintenance gagneraient à être envisagés de manière cohérente [Horrocks, 2005]. Ainsi, on peut se demander si les mêmes outils sont utilisables pour ces deux phases, et le prévoir dès la construction de l'ontologie.

Dans ce même objectif de cohérence, une vision alternative commence à émerger en s'appuyant sur des logiciels de traitement automatique des langues et de techniques d'apprentissage. Cette tendance vise à construire le plus automatiquement possible des structures plus proches de réseaux terminologiques et donc moins complexes à élaborer [Maedche et Staab, 2000a; Maedche, 2002]. Ces structures, moins formelles, sont souvent suffisantes pour des applications comme la recherche d'information, l'indexation de documents ou la gestion des connaissances (contrairement à la communication entre agents, par exemple). Dans ce cas, la construction devrait sûrement être envisagée comme un cas particulier de maintenance. En effet, cela revient à effectuer la maintenance d'une structure initialement vide. À

notre connaissance, aucun outil n'a été développé dans cette optique.

1.6 Bilan

Dans ce chapitre, nous avons vu ce qu'est une ontologie, les approches méthodologiques pour sa construction, la place des corpus textuels et les apports des outils de TAL dans ces méthodes. Nous retenons de cet état de l'art les points suivants :

- les corpus textuels en tant que source d'information sont très riches mais ne permettent pas de définir entièrement une ontologie ; au mieux, ils permettent de constituer un brouillon ;
- la structure hiérarchique est l'ossature principale d'une ontologie et des ressources termino-ontologiques assimilées ;
- les outils de TAL répondent aux principales attentes en matière de construction d'ontologies grâce à la diversité et à la complémentarité de leurs résultats ;
- l'enchaînement des analyses de TAL et de l'examen des résultats est linéaire, rigide et laisse peu de place à l'incrémentalité ;
- l'ontologue joue un rôle capital en interprétant les résultats des outils de TAL pour définir des éléments d'ontologie ; pour cela il doit sélectionner les termes qui donneront lieu à la création de concepts ;
- la construction et la maintenance d'ontologies sont vues comme des tâches dissociées.

A partir de ce diagnostic nous avons retenu un certain nombre de choix pour définir un système multi-agent qui assisterait l'ontologue dans cette tâche :

- la source première d'information de notre système est le corpus textuel ;
- ce système ne prendra pas en charge le traitement automatique du langage ; en revanche, il devra être capable de s'appuyer sur des dépendances syntaxiques données en entrée, des régularités et relations associées à l'usage qui seront utilisées par l'ontologue pour dégager des éléments d'ordre conceptuel ;
- l'importance de la relation hiérarchique, et la disponibilité d'informations statistiques sur les termes en provenance des textes nous poussent vers une solution proche de celle présentée dans LexiClass basée sur un système de classification utilisant le réseau « Tête-Expansion » de Syntex en entrée ;
- l'importance de l'interprétation par l'ontologue nécessite de mettre au point un système capable de faire des propositions à l'utilisateur et de prendre en compte ses critiques pour faire évoluer l'ontologie ;
- le même système doit permettre ainsi de construire l'ontologie et de la faire évoluer pour sa maintenance.

La définition d'un tel système multi-agent requiert d'examiner les liens possibles entre ontologies et systèmes multi-agents. Cela nécessite aussi d'opter pour une approche de conception qui conditionnera la nature et le comportement des agents. Il s'agit là du propos des chapitres suivants.

2 Systèmes multi-agents et construction d'ontologies

> « Il y avait quelque chose d'à la fois comique et terrifiant à l'idée d'une épidémie de monolithes noirs. »
>
> *Arthur C. Clarke – 2010 : Odyssée deux*

COMME nous l'avons vu au chapitre précédent, le domaine de la construction d'ontologies à partir de textes a besoin d'aller vers plus de dynamique. Notre thèse a pour objectif d'évaluer si l'idée de distribuer le contrôle et la connaissance du système dans un ensemble d'entités autonomes pourrait être une approche viable pour atteindre ce but. En effet, nous espérons semi-automatiser la construction d'ontologies par le recours à un système multi-agents adaptatif. Cette classe de système est utilisable pour résoudre des problèmes rarement résolus efficacement et pour la mise au point interactive de systèmes. Ces deux aspects correspondent à notre avis à la construction d'ontologies à partir de texte.

Ce chapitre présente les travaux qui ont influencé cette thèse dans le domaine des systèmes multi-agents. Nous nous sommes intéressés au rapport entre les ontologies et cette catégorie de systèmes. Après un panorama de différents approches consacrées au lien existant entre systèmes multi-agents et ontologies, une part importante de ce chapitre sera consacrée à la théorie des *Adaptive Multi-Agent Systems* qui a plus particulièrement guidé notre réflexion. Enfin, nous présentons la méthode d'ingénierie multi-agent *ADELFE*.

2.1 Systèmes multi-agents et ontologies

Depuis quelques années, les termes « systèmes multi-agents » et « ontologies » sont souvent mentionnés ensemble dans la littérature. Sans y prêter attention, nous pourrions simplement supposer qu'il s'agit là de deux domaines en vue pour expliquer cette présence conjointe. Toutefois, en y regardant de plus près, il s'agit bien d'éléments complémentaires.

L'usage veut que les ontologies soient utilisées pour permettre le fonctionnement des systèmes multi-agents. En effet, les systèmes multi-agents, en particulier s'ils sont ouverts, soulèvent de nombreuses questions liées à la communication. Or, les ontologies permettent

de s'attaquer à des problèmes communs de langage et de représentation du monde. De plus, comme nous l'avons vu au chapitre précédent, les ontologies ont un coût de construction et de maintenance qu'il convient de réduire au maximum. Des systèmes décentralisés comme les systèmes multi-agents sont alors une possibilité pour atteindre ce but, tout comme l'utilisation de techniques d'apprentissage.

Nous allons donc examiner les différentes relations présentes entre les systèmes multi-agents et les ontologies, et plus particulièrement, l'utilisation des systèmes multi-agents pour la construction des ontologies qui est le but recherché dans nos travaux.

Ontologies pour les systèmes multi-agents

Le lien le plus répandu entre systèmes multi-agents et ontologies est l'utilisation d'ontologies pour faire fonctionner les systèmes multi-agents [Gandon, 2002], en particulier pour ce qui concerne les agents dans le Web sémantique et les Web services [Greenwood *et al.*, 2007]. Dans ce cas, les agents se réfèrent à une ontologie connue pour mettre en œuvre un langage commun et être capable d'interagir. Les ontologies sont alors un moyen de médiation entre agents logiciels, ce qui est une des raisons de leur définition.

Il existe également d'autres types d'usage des ontologies à l'échelle du système. Par exemple, un ensemble fixe d'ontologies donné initialement au système sera utilisé par tous les agents pour effectuer un traitement sur un ensemble de données [Elmore *et al.*, 2003]. Dans ce cas le langage des agents est commun, et les ontologies leur permettent uniquement d'interpréter un flux de données en provenance de l'extérieur du système.

Un autre système utilisant des ontologies est COMMA [Bergenti *et al.*, 2002]. Dans ce dernier, chaque ontologie est encapsulée dans un agent et, en collaboration avec un agent interface, aide l'utilisateur à ajouter des documents dans une base de connaissance, ou à retrouver des documents. Pour cela, l'agent ontologie mis en œuvre est une aide pour construire correctement les méta-données du nouveau document, ou la requête de recherche.

Ontologies pour les agents

Parfois, chaque agent est porteur de sa propre ontologie et est capable de la modifier en fonction des interactions avec d'autres agents [Siebes et van Harmelen, 2002]. Dans ce cas, nous sommes plus proches d'ontologies utilisées comme représentation du monde pour chacun des agents d'un système ouvert. Les formalisations employées sont généralement basées sur XML, comme OWL, aujourd'hui la plus répandue. L'inconvénient principal de leur utilisation provient de l'impossibilité d'avoir un point de vue unique sur un domaine. Ainsi, pour un système ouvert, il est tout à fait possible d'avoir plusieurs ontologies d'un même domaine donné. On retrouve alors des problèmes de communication entre ces agents, ce qui justifie le besoin de système d'alignements d'ontologies. Aujourd'hui, la majorité de ces systèmes est centralisée, mais des systèmes décentralisés commencent à apparaître pour effectuer cette tâche [Laera *et al.*, 2007]. L'intérêt pour les agents de s'appuyer sur des ontologies explicites et au même format réside justement dans cette possibilité de comparer les représentations du monde auxquelles ils font référence.

Systèmes multi-agents pour les ontologies

Comme nous venons de le voir, les systèmes multi-agents gagnent à s'appuyer sur des ontologies pour fonctionner et l'alignement entre ontologies[1] est une des conditions de leur bonne communication. On trouve donc des systèmes multi-agents prenant en charge des calculs sur des ontologies différentes afin de les aligner [Lister *et al.*, 2006]. Ces systèmes jouent alors un rôle de médiateur entre des ontologies en identifiant leurs points communs, en utilisant par exemple des métriques appropriées [Hernandez, 2005] sans qu'eux-mêmes ne disposent de connaissance sémantique.

En revanche, il est un peu plus rare de trouver, dans la littérature, des exemples où les systèmes multi-agents sont une aide pour la construction de ressources ontologiques. Dans ce cas, ils sont le plus souvent utilisés comme support pour des ateliers de construction d'ontologies collaboratifs et n'exploitent pas de résultat en provenance de textes [Bao et Honavar, 2004]. Ils sont essentiellement là pour assurer la cohérence du produit modélisé en suivant les modifications concurrentes des utilisateurs.

Nos choix pour la construction d'ontologies

Nous venons de voir différents liens possibles entre ontologies et systèmes multi-agents. Mais, à notre connaissance il n'existe pas de système multi-agent pour construire des ontologies à partir de textes où le système lui-même serait l'ontologie, et les composants du système (les agents) seraient alors des entités de l'ontologie (concepts, relations, etc.). Pourtant, cette approche peut se justifier par les liens entre la modélisation objet et la construction d'ontologies. Il existe dans la littérature des tentatives d'utilisation d'UML pour une modélisation au moins partielle d'ontologies [Cranefield et Purvis, 1999]. Ce point est particulièrement intéressant lorsque l'on tient compte de [Odell *et al.*, 2000] qui considère les agents comme une évolution des objets. Ainsi, il paraît évident qu'un lien peut être fait entre construction d'ontologies et systèmes multi-agents et nous nous proposons de persévérer dans cette direction.

De plus, si nous considérons l'utilisation fructueuse de la classification hiérarchique pour la construction d'ontologies [Assadi, 1998; Faure, 2000], il devient intéressant de considérer la classification hiérarchique distribuée.

Alors, [Parunak *et al.*, 2006] présente un algorithme utilisé dans un système multi-agent pour construire dynamiquement une hiérarchie de classes depuis un flux de données en entrée. Dans ce système, chaque nœud de la hiérarchie est un agent. Quand une nouvelle donnée est rencontrée, un nouvel agent est créé en tant que feuille de la hiérarchie. Ensuite, les agents prennent des décisions en fonction d'une mesure d'homogénéité des données calculée par chaque agent en fonction de ses fils. En fonction de cette mesure, un agent choisira soit de fusionner avec un de ses frères, soit de promouvoir un de ses fils pour le faire remonter dans la hiérarchie.

Comme nous le voyons, un aspect intéressant de cette approche est que la classification

[1] C'est-à-dire construire la correspondance entre deux ontologies différentes pour un même domaine.

résultante n'est pas extérieure au système multi-agent. Il est lui-même la hiérarchie en sortie. Le système prend alors les deux rôles : porteur du processus de classification et résultat de la classification.

Cet algorithme est appliqué pour de la fouille de données, mais pourrait probablement être transposé à la classification de termes. Toutefois, même si l'approche générale est séduisante, l'algorithme lui-même ne semble pas favoriser l'utilisation de plusieurs critères pour la construction de la hiérarchie, ni l'application de simplifications à la hiérarchie. Ce sont deux points importants à garder à l'esprit pour permettre le passage d'un arbre binaire tel que produit par un processus de classification hiérarchique à une taxonomie, puis à une ontologie.

Ces considérations sur la classification distribuée nous permettent de préparer nos choix en vue d'un système multi-agent de construction d'ontologies à partir de textes. En effet, il convient de se demander quel type de tâches nous voulons déléguer au système multi-agent pour soulager l'utilisateur, et quel type de connaissances les agents pourront utiliser. Le chapitre précédent nous a permis de constater que l'utilisateur passe une grande partie de son temps à dépouiller des résultats en provenance des textes pour isoler ce qui lui semble pertinent. Ensuite, il peut enfin se concentrer sur des décisions de conception mais doit revenir aux textes pour vérifier la cohérence de ses choix personnels.

Il nous paraît donc important de déléguer au système multi-agent, le plus possible, les tâches éloignant l'utilisateur de la conception à proprement parler, c'est-à-dire l'examen des statistiques en provenance des textes. La classification est alors un excellent point de départ puisqu'elle fournit un premier support hiérarchique pour nos concepts et est visiblement envisageable de manière distribuée comme [Parunak *et al.*, 2006] le prouve. Les connaissances utilisées par les agents seraient alors les statistiques en provenance des analyses de textes ou de calculs complémentaires faits par les agents eux-mêmes.

2.2 La théorie des *Adaptive Multi-Agent Systems* (AMAS)

Comme nous avons envisagé d'aborder la construction et la maintenance d'ontologies à partir de textes à l'aide de systèmes multi-agents, nous nous sommes demandés quel outil pourrait être utilisé pour concevoir un système multi-agent supportant l'utilisateur dans cette tâche. En constatant qu'il s'agit d'une tâche de conception, et en considérant certains travaux proposant l'idée de *Living Design* [Georgé *et al.*, 2003b], une approche permettant de mettre en place ce type de conception nous paraissait requise. En cela, la théorie des *Adaptive Multi-Agent Systems* (AMAS) nous semble comporter des qualités adaptées.

Le but de cette théorie est de permettre à un système, utilisant des critères de réorganisation locale au niveau de ses entités, d'atteindre l'adéquation fonctionnelle. Par « adéquation fonctionnelle », nous nous référons au comportement global du système qui doit être adapté à la tâche courante. Pour vérifier cette adéquation, le comportement doit être jugé par un observateur extérieur au système qui connait sa finalité. La théorie des AMAS se base sur le théorème suivant démontré dans [Camps *et al.*, 1998] :

Théorème 1. *Pour tout système fonctionnellement adéquat, il existe au moins un système à milieu intérieur coopératif qui réalise une fonction équivalente dans le même environnement.*

Un système possède un « milieu intérieur coopératif » lorsqu'il n'existe plus en son sein d'incompréhension, d'ambiguïté, d'incompétence, de conflit, de concurrence ou d'inutilité entre les entités composant ce système. La démonstration de ce théorème repose sur un axiome et quatre lemmes. Notre but n'est pas de refaire cette démonstration en détail, mais de présenter succintement cet axiome, ces quatre lemmes et leurs conséquences.

Axiome 1. *Un système fonctionnellement adéquat n'a aucune activité antinomique sur son environnement.*

Une activité antinomique sur l'environnement est une activité allant à l'encontre des intérêts de l'environnement.

Lemme 1. *Tout système coopératif est fonctionnellement adéquat.*

Car par définition, un système coopératif n'a pas d'activité antinomique ou indifférente. Donc l'ensemble des systèmes coopératifs est inclus dans l'ensemble des systèmes fonctionnellement adéquats.

Lemme 2. *Pour tout système S fonctionnellement adéquat, il existe au moins un système coopératif S^* qui soit fonctionnellement adéquat dans le même environnement.*

Ce lemme lève l'incertitude restante vis-à-vis de l'existence d'un système coopératif équivalent pour chaque système fonctionnellement adéquat dans le même environnement. Sans ce lemme, l'intérêt porté aux systèmes coopératifs serait amoindri puisque pour certains systèmes fonctionnellement adéquats, il ne serait plus possible de trouver un système coopératif équivalent.

Lemme 3. *Tout système à milieu intérieur coopératif est un système coopératif.*

Donc l'ensemble des systèmes à milieu intérieur coopératif est inclus dans l'ensemble des systèmes coopératifs. Il reste donc comme précédemment, à s'assurer que chaque système coopératif dispose d'un système à milieu coopératif équivalent.

Lemme 4. *Pour tout système coopératif, il existe au moins un système à milieu intérieur coopératif avec une fonction équivalente dans le même environnement.*

La dernière incertitude étant levée, il est alors possible de définir une application surjective de l'ensemble des systèmes fonctionnellement adéquats vers l'ensemble des systèmes à milieu intérieur coopératif. Ces derniers systèmes constituant un sous-ensemble de l'ensemble des systèmes fonctionnellement adéquats.

Ainsi, le théorème central de la théorie des AMAS permet de se focaliser uniquement sur une classe de systèmes pour obtenir des systèmes fonctionnellement adéquats dans un environnement donné (voir figure 2.1).

Le milieu intérieur coopératif d'un système est la notion centrale de la théorie des AMAS énoncée dans [Camps *et al.*, 1998]. Cette théorie s'intéresse donc en particulier aux relations

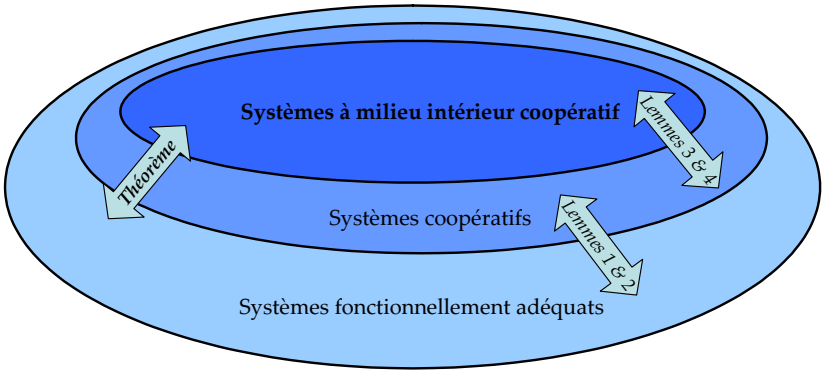


Figure 2.1 — Relation entre les systèmes fonctionnellement adéquats et les systèmes à milieu intérieur coopératif

entre les entités constituantes des systèmes. Il devient alors clair qu'elle est aisément applicable aux systèmes multi-agents. La figure 2.2 présente comment doit s'adapter un tel système [Georgé *et al.*, 2003a].

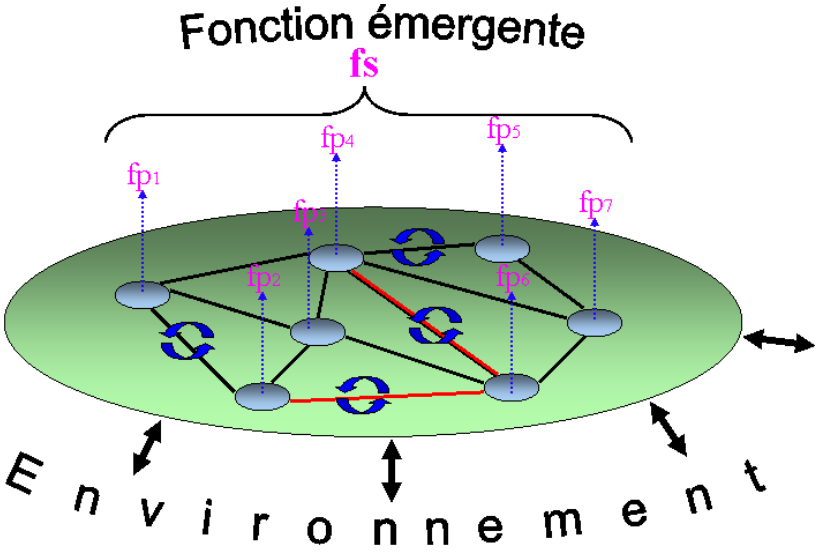


Figure 2.2 — Adaptation par auto-organisation

Le concepteur s'intéresse uniquement aux agents P_i qui produisent chacun leur fonction partielle f_{P_i} et leur donne le moyen de décider de changer les liens les unissant. Ce sont ces liens qui règlent la combinaison des fonctions f_{P_i} engendrant la fonction globale f_S. Ainsi, en fonction des interactions du système avec son environnement, l'organisation des agents se modifie pour faire face aux perturbations de l'environnement. Le système s'adapte à tout moment pour produire une nouvelle fonction f'_S.

2.3 Une théorie appliquée aux systèmes multi-agents

Malgré son nom, il est tout à fait envisageable d'appliquer la théorie des AMAS basée sur la coopération pour d'autres paradigmes que celui du multi-agent. D'ailleurs, nous avons

entrepris de ne mentionner les systèmes multi-agents que vers la fin de la section précédente. Il s'avère pourtant que ce paradigme est aujourd'hui le mieux adapté à une telle théorie car elle requiert des capacités décisionnelles dans chacune des parties du système.

Pour pouvoir l'appliquer, il existe une spécification générale de l'architecture d'un agent AMAS [Picard, 2004]. Un tel agent dispose en général des éléments suivants pour mener son travail à bien :

- des connaissances d'un domaine particulier permettant à l'agent de réaliser sa fonction partielle (compétences) ;
- d'une représentation de lui-même, des autres et de l'environnement utilisant la notion de confiance sur ses croyances (représentations) ;
- d'une attitude sociale coopérative, autrement dit de critères locaux lui permettant de savoir s'il est ou non coopératif et d'agir en conséquence (coopération) ;
- d'un langage d'interaction par envois de messages directs ou indirects par l'environnement (interaction) ;
- de capacités à raisonner sur ses connaissances et ses représentations (aptitudes).

L'élément le plus important pour la coopération est l'attitude sociale de l'agent. Un agent doit avant tout anticiper et réparer une situation qu'il considère non coopérative. Si l'on prend un point de vue d'assez haut-niveau, la détection des situations non-coopératives (SNC) est exprimable en utilisant l'équation suivante [Camps *et al.*, 1998] :

$$\textbf{NON COOPERATION} = \neg C_1 \vee \neg C_2 \vee \neg C_3 \tag{2.1}$$

- $\neg C_1$ **Incompréhension** : un signal est incompris ou comporte des ambiguïtés. Dans ce cas, l'agent va tenter de transmettre l'information à d'autres agents potentiellement plus compétents ou de se faire aider par autrui afin de lever les ambiguïtés.
- $\neg C_2$ **Défaut de raisonnement** : une information reçue est déjà connue ou n'a aucune conséquence logique. L'agent ne peut en tirer profit et va donc chercher d'autres agents pouvant en bénéficier.
- $\neg C_3$ **Inutilité de l'action** : d'après ses croyances courantes, l'agent considère que la transformation qu'il peut opérer n'est pas bénéfique à autrui (conflit ou concurrence).

Un système plongé dans un environnement dynamique va forcément devoir traiter des situations imprévues du point de vue local de l'agent. Cet imprévu sera pris en compte comme une SNC au niveau de l'agent qui va chercher à modifier ses relations avec les autres. Les fonctions partielles du système se réorganisent, changeant la fonction globale, pour réagir à la situation imprévue et la prendre en charge. Ainsi, un agent doit non seulement réaliser sa fonction partielle, mais aussi agir sur l'organisation du système pour résoudre les SNC. Ces décisions sur la coopération sont locales aux agents et ne doivent pas être dictées par la fonction globale (qui évolue au cours du temps et en fonction des imprévus).

2.4 Quelques problèmes traités avec des AMAS

La théorie des AMAS nécessite d'être expérimentée, d'autant plus que l'adéquation fonctionnelle est un concept subjectif qui ne peut être validé que par un observateur dans un contexte donné. C'est pourquoi, nous présentons plusieurs systèmes pour lesquels la

théorie des AMAS a été appliquée avec succès. Nous n'avons pas ici pour but de détailler avec précision chacune des solutions retenues, mais simplement de présenter quelques problèmes résolus dans des domaines d'application différents pour montrer la polyvalence de cette approche.

Tout d'abord, FORSIC [Gleizes *et al.*, 2000a] est un système de recherche d'informations dans un cadre éducatif. À partir des formateurs inscrits, le système a pour but de répondre aux requêtes d'un utilisateur cherchant des cours dans un domaine donné. L'intérêt d'une approche AMAS dans ce cas est la possibilité de tenir compte de la dynamique des besoins des utilisateurs et des services offerts. Les descriptions de formations et les requêtes utilisateurs sont saisies en texte libre. Les co-occurrences sont alors exploitées car elles sont souvent un critère statistique suffisant pour la recherche d'informations. De plus, le système effectue des réajustements par petits incréments ou décréments en fonction des situations non coopératives détectées.

ABROSE est un système de courtage électronique de services sur internet. Chaque agent représente un fournisseur de services ou un consommateur de services. Ils travaillent sur leurs accointances exploitées pour faire suivre les requêtes utilisateur à des agents jugés plus compétents lorsqu'ils ne peuvent pas satisfaire complètement les besoins. L'utilisateur a la possibilité de critiquer les résultats du système en les notant « bon », « mauvais » ou « peu importe ». Cette critique est alors utilisée par les agents pour modifier leurs accointances si nécessaire [Gleizes et Glize, 2002]. Ces accointances sont un autre sous-système multi-agent où chaque agent représente un mot utilisé dans les requêtes utilisateur ou les descriptions de services. Chaque agent de courtage possède un tel sous-système. Son mode de fonctionnement simple lui permet de fournir des résultats sans connaissance linguistique.

Un troisième exemple est le système de prévision des crues STAFF [Régis *et al.*, 2002]. Il s'agit là d'un problème très complexe dépendant de nombreux paramètres : topographie, superficie, pluviométrie, etc. De plus, ces caractéristiques ont tendance à évoluer avec le temps. Le système, développé en tirant parti de la coopération, est entraîné en utilisant des historiques de crues des stations sur lesquels il est installé. Chaque agent est responsable d'une station et effectue un réajustement permanent relativement aux mesures d'entrée, aux résultats des autres agents et à l'erreur de prévision commise. Cette réorganisation se fait par petites variations sur les coefficients d'une somme pondérée utilisée par chaque agent pour conclure d'une prévision.

SYNAMEC est un projet de conception de mécanismes aéronautiques assisté par un AMAS. Dans ce dernier, les composantes d'un mécanisme vont s'auto-organiser dans l'espace pour respecter la fonctionnalité (une trajectoire, par exemple) et les contraintes posées par le concepteur [Capera *et al.*, 2004; Picard *et al.*, 2005]. Pour cela, les agents, représentant des sous-parties du mécanisme à construire, cherchent la position idéale pour respecter la trajectoire. Ils disposent de plusieurs moyens pour parvenir à leurs fins : changement de partenaires, instanciation de nouveaux agents, et disparition des agents superflus.

Enfin, dans [Ottens, 2004], nous avons développé un système multi-agent d'étiquetage morpho-syntaxique. En entrée, le système traite un texte brut, et nous obtenons en sortie le texte étiqueté. À noter qu'il s'agit d'un étiquetage simplifié afin d'éviter d'avoir recours à des

ressources linguistiques (lexique ou règles). Les résultats obtenus sont très mitigés puisque le système obtenu n'est que 10% meilleur qu'un tirage aléatoire. Toutefois, il a permis de faire ressortir des difficultés liées à un domaine complexe comme le traitement automatique des langues.

Les résultats obtenus dans le cadre de ces projets ont montré la pertinence de l'approche par AMAS pour la résolution de problèmes complexes n'admettant pas de solution algorithmique connue *a priori*. Ces travaux ont également confirmé la nécessité d'introduire plus de guides (méthodologiques et techniques) afin de construire des systèmes pour ce type de domaines.

2.5 Le processus de la méthode ADELFE

Afin de rester dans cette obligation de confrontation de la théorie des AMAS à des domaines variés, il paraît naturel de tenter de l'appliquer sur le plus grand nombre de domaines possibles. Dans cette optique, la méthode ADELFE[2] [Picard, 2004] a été développée afin de guider la construction de systèmes multi-agents adaptatifs.

La méthode ADELFE, dédiée au développement de systèmes multi-agents adaptatifs, est décrite suivant trois axes fondamentaux : le processus, les notations et les outils. Dans ce chapitre, nous discutons principalement le processus. Les notations sont discutées au chapitre suivant en tenant compte des modifications que nous y avons apportées. Pour plus d'information concernant les outils, nous invitons le lecteur à aller consulter le site du projet ADELFE, où ils sont téléchargeables librement : http://www.irit.fr/ADELFE.

2.5.1 Survol du processus

Proposer une méthode de développement relève principalement d'une démarche à suivre pour atteindre des objectifs donnés. Le processus fourni doit permettre à tout concepteur d'effectuer un suivi pas-à-pas des étapes de développement de logiciels et des méthodes mises en jeu. Définir un processus consiste non seulement en la définition de ses étapes (ou phases), mais aussi en leur ordonnancement et en l'expression des modèles et artefacts produits et nécessaires. Le processus d'ADELFE, comme l'illustre la figure 2.3 [Picard, 2004], se divise en cinq *définitions de travaux* : les besoins préliminaires, les besoins finals, l'analyse, la conception et le codage. Ces travaux s'intègrent dans le RUP (*Rational Unified Process*) et ont été étendus afin de répondre au métier agent [Jacobson *et al.*, 2000]. Chaque ensemble de travaux ainsi que ses artefacts sont définis en utilisant la notation issue de SPEM (pour *Software Process Engineering Metamodel*) permettant de définir des processus de manière simple et compréhensible pour les utilisateurs d'UML [OMG, 2005].

2.5.2 Spécificités

ADELFE est spécifique à la conception d'AMAS ; aussi, certaines activités (marquées en gras dans la figure 2.3) ont été ajoutées au RUP pour l'adapter à cette technologie.

[2] Atelier de DEveloppement de Logiciels à Fonctionnalité Emergente

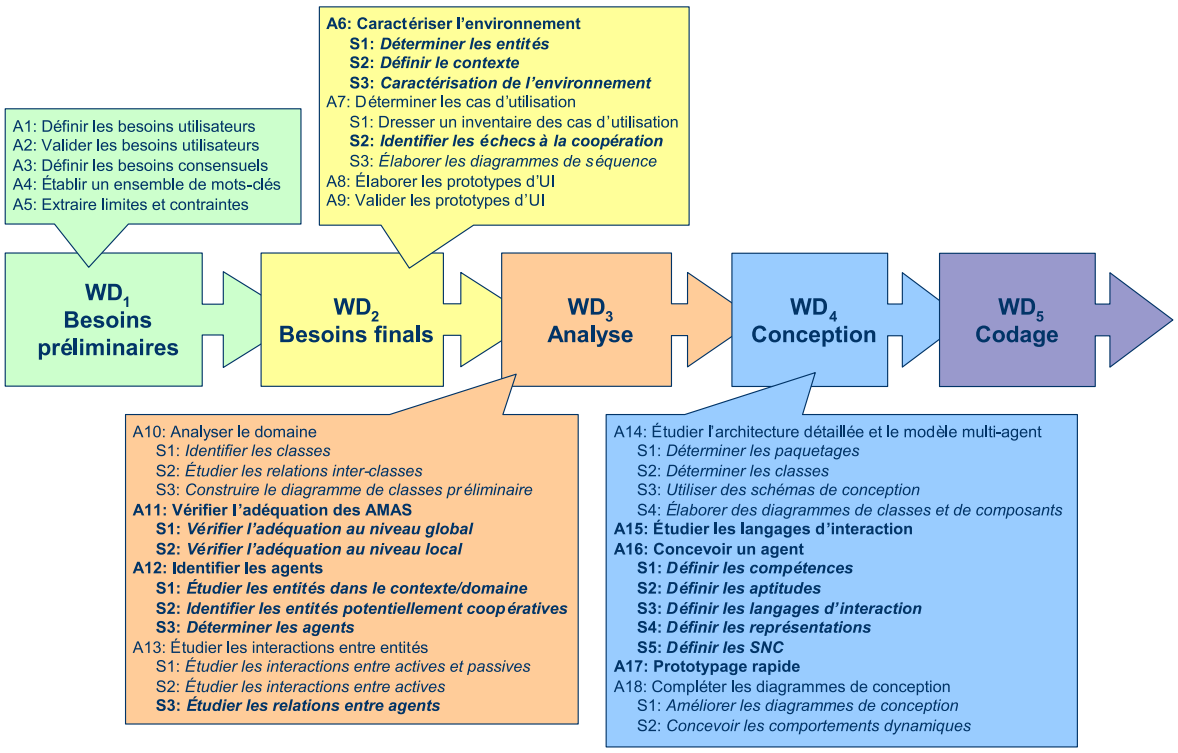


Figure 2.3 — Les cinq premières définitions de travaux du processus d'ADELFE

Durant l'expression des besoins finals (WD_2) :

A_6 : *caractérisation de l'environnement*. La méthode propose de se concentrer, dès les premières analyses des interactions entre le système et son environnement, sur la nature de ces interactions car ce sont elles qui guideront l'auto-organisation du système et qui influeront sur la définition des règles de coopération ;

A_7-S_2 : *identification des échecs à la coopération*. Cette étape est une suite logique du point précédent. Une fois que les propriétés de l'environnement ont été identifiées, il faut définir quelles sont les interactions entre acteurs (environnement) et cas d'utilisations (système) par lesquelles les problèmes d'inadéquation entre système et environnement passeront.

Durant l'analyse (WD_3) :

A_{11} : *vérification de l'adéquation aux AMAS*. Certes, ADELFE propose de développer des systèmes dont le fonctionnement adaptatif repose sur l'auto-organisation par coopération, mais tous les problèmes n'ont pas forcément besoin d'être résolus par cette approche. De plus, même si l'utilisation de tels systèmes semble convenir, l'analyse ne sait pas forcément où placer des agents coopératifs, ou bien quelles sont les entités du domaine qui devront/pourront être décomposées en agents ;

A_{12} : *identification des agents impliqués dans le système à construire*. Les agents ne sont par forcément évidents à identifier pour des systèmes non orientés utilisateurs. En effet, dans des systèmes de courtage en ligne, par exemple, les agents sont directement associés aux utilisateurs. Mais dans un problème comme la prévision de crue, la présence d'agents horaires, devant produire des prévisions pour une

heure, n'est pas une identification directement extractible de l'analyse des entités du domaine. C'est pourquoi ADELFE présente une telle activité d'identification des agents à partir de l'analyse des interactions inter et extra système ;

A_{13}-S_3 : *étude des relations entre ces agents*. Comme pour les objets, les interactions entre agents doivent être étudiées, avec les diagrammes de séquences ou de collaboration. Cependant, les agents peuvent faire preuve de plus de richesse dans leurs protocoles de communication, ce qui nécessite l'ajout d'une activité spécifique devant produire des modèles d'interaction entre agents (protocoles AUML).

Durant la conception (WD_4) :

A_{15} : *étude des langages d'interaction* qui permettent aux agents d'échanger de l'information. Tous les agents ne sont par forcément capables de communiquer avec tous les autres. Il faut définir des langages d'interaction entre agents portant sur des aspects de résolution de problèmes particuliers. Ces langages d'interaction peuvent eux aussi être spécifiés à partir de diagrammes de protocole. Un langage correspond alors à un ensemble de méthodes et d'attributs nécessaires pour comprendre les autres agents, et à une ou plusieurs machines à états décrivant la bonne utilisation du langage ;

A_{16} : *conception complète de ces agents*. Un agent qui intervient dans un AMAS est composé de différentes parties qui forment son comportement : des compétences, des aptitudes, un langage d'interaction, des représentations du monde et des situations non coopératives (SNC). L'ajout de cette activité est essentiel à la conception de systèmes multi-agents adaptatifs, puisqu'elle consiste à attribuer un comportement nominal, pour résoudre une tâche courante, et un comportement coopératif aux agents afin de résoudre le problème voulu. C'est donc dans cette activité que les règles d'auto-organisation coopérative vont être définies et attribuées aux agents ;

A_{17} : *prototypage rapide*. Afin de vérifier que le comportement des agents est bien celui désiré, cette activité permet de simuler les agents non finalisés. Cette étape peut notamment servir de prévalidation ou à trouver des SNC non encore identifiées par l'observation et l'établissement de théories *a posteriori*.

La description complète des activités s'arrête actuellement à la fin de la conception. Dans le chapitre suivant, nous étudions les ponts possibles entre la conception et le codage, en prenant pour acquis que la programmation des systèmes obtenus sera effectuée en programmation par objets. Cette réflexion se situe dans l'évolution naturelle de cette méthode, et nous semble nécessaire pour pouvoir l'appliquer à des domaines complexes comme la construction d'ontologies à partir de textes ou le traitement automatique des langues.

2.6 Bilan

Dans ce chapitre, nous avons présenté les interactions possibles entres systèmes multi-agents et ontologies. Il ressort qu'il n'existe à notre connaissance aucune tentative pour concevoir un système multi-agent d'aide à la construction d'ontologies à partir de textes,

en particulier si le système représente directement le système en fonctionnement. Toutefois, nous remarquons des pistes concernant un système multi-agent de classification, mais non appliqué à la construction de ressources termino-ontologiques. Cela confirme notre intérêt pour la classification déjà souligné dans le bilan du chapitre précédent.

Dans la deuxième moitié du chapitre, nous nous sommes concentrés sur les caractéristiques des AMAS en tant que classe de systèmes. Les développements récents de la méthode ADELFE nous paraissent intéressants. Toutefois, notre expérience de l'approche AMAS, pour des manipulations abstraites comme le traitement automatique des langues [Ottens, 2004], nous pousse à la vigilance et à rechercher plus de rigueur dans l'application de l'approche préalablement à nos développements. Ainsi, nous nous proposons d'étendre ADELFE à la phase d'implémentation dans le chapitre suivant, avant d'appliquer ces extensions pour concevoir notre système d'ontologies dynamiques.

Deuxième partie

Conception d'un système multi-agent pour la construction d'ontologies dynamiques

3 Ingénierie agent basée sur les modèles

« Vous passez votre temps à bâtir des modèles. Des cercles de pierre. Des cathédrales. Des orgues à tuyaux. Des machines à additionner. »

William Gibson – Neuromancien

LE chapitre précédent nous a permis de voir, entre autres, quelques applications de la technologie des AMAS. Toutefois, dans les projets présentés, rien ne garantissait que les implémentations suivaient bien l'architecture d'un agent AMAS. Pour cela, davantage de méthode est requise pour garantir certaines propriétés dans les systèmes implémentés, en particulier lorsque la structure interne des agents atteint une taille et une complexité importante. Comme nous le verrons dans les chapitres suivants, ce besoin devient incontournable pour les agents du système Dynamo à cause de la difficulté du domaine de la construction d'ontologies à partir de textes.

Or, depuis quelques années, les méthodes orientées agent fleurissent dans le domaine académique [Arlabosse *et al.*, 2004; Bergenti *et al.*, 2004; Henderson-Sellers *et al.*, 2005] – chacune ayant ses spécificités : modèle d'agent, formalisme, paradigme de programmation, domaine d'application, etc. Cet effort méthodologique se limite souvent aux phases préliminaires du cycle de vie du logiciel : l'analyse et la conception. Face à ce constat, nous proposons une réflexion sur la base de la méthode ADELFE pour le développement de systèmes multi-agents adaptatifs [Picard, 2004; Picard et Gleizes, 2004; Georgé *et al.*, 2003a]. Dans la continuité du processus de développement existant – s'arrêtant à la conception, nous proposons des outils à mettre en œuvre pour faciliter la génération de code à partir de modèles d'agents donnés – comme par exemple, le modèle d'agent coopératif dans ADELFE. Cette approche est positionnée dans le cadre du MDA (*Model Driven Architecture*) de l'OMG (*Object Management Group*), dont le but est de raisonner sur les modèles et leurs règles de transformation tout au long du développement logiciel, jusqu'à l'implémentation, voire la maintenance [Kleppe *et al.*, 2003]. À notre connaissance, les premières tentatives d'utilisation du MDA dans le cadre d'une ingénierie des agents sont établies dans les projets Méta-DIMA [Thiefaine *et al.*, 2003], Tropos [Perini et Susi, 2005], CaFne [Jayatilleke *et al.*, 2005] et INGENIAS [Pavón, 2006].

L'intérêt de cette démarche est une amélioration de la structuration du système afin de dégager des composants logiciels de granularités différentes et réutilisables (agents, constituants des agents, protocoles, etc.). Nous nous sommes intéressés à cette approche à cause de la complexité du domaine à couvrir dans nos travaux. En effet, l'approche AMAS a été utilisée pour plusieurs applications qui ont toutes abouti à des implémentations différentes du modèle d'agent sans garantie que les caractéristiques du modèle aient été rigoureusement respectées. Ceci nous pousse alors à définir – ce qui est le but de ce chapitre – des règles de transformation permettant le passage de la conception à l'implémentation dans le processus ADELFE, pour une meilleure lisibilité et modularité de l'approche, mais aussi pour un meilleur respect de l'utilisation du modèle d'agent coopératif et de ses différents modules. Ainsi, nous pouvons nous prémunir, au moins partiellement, des écueils rencontrés par nos prédecesseurs et mieux gérer la complexité du but à atteindre.

Dans la section 3.1, nous présentons le modèle d'agent coopératif, propre à la méthode ADELFE, grâce auquel nous illustrons notre propos. Dans cette méthode, l'utilisation du modèle d'agent est contrainte par une extension du métamodèle *via* un stéréotypage spécifique. Nous avons entièrement révisé le système de stéréotypes présenté dans [Picard, 2004], les résultats obtenus sont présentés dans la section 3.3. Dans la section 3.4, nous précisons ensuite les méthodes et techniques que nous avons développées pour assurer le passage de la conception (diagrammes de classes) à l'implémentation de manière semi-automatique. Nous montrons aussi comment nous les avons mises en œuvre, ainsi que leur positionnement par rapport à l'approche MDA.

3.1 Un modèle d'agent pour l'auto-organisation coopérative

Cette section expose le modèle d'agent utilisant la coopération comme moteur de l'auto-organisation du système multi-agent afin d'atteindre l'adéquation fonctionnelle [Picard, 2004]. Les agents coopératifs sont composés de différents modules représentant une partition de leurs capacités physiques, cognitives ou sociales (voir figure 3.1). Chaque module représente une ressource spécifique pour l'agent, et est utilisé lors de son cycle de vie composé des trois phases « perception-décision-action ».

3.1.1 Les différents modules

Les interactions entre agents sont régies par deux modules. Le *module de perception* représente les données en entrée que l'agent reçoit de son environnement. Ces stimulis peuvent être de nature et de type différents : entiers, booléens ou même des messages structurés de haut niveau (dans le cas d'une boîte aux lettres). Le *module d'actions* représente les sorties et le moyen pour l'agent d'agir sur son environnement physique, social ou sur lui-même dans le cas d'une action d'apprentissage. Comme pour les perceptions, les actions peuvent être de granularité variable : activation simple d'effecteurs pour un robot ou envoi de messages à fort contenu sémantique pour des agents sociaux.

Le *module de compétences* concerne les tâches des agents. Même si les agents coopératifs

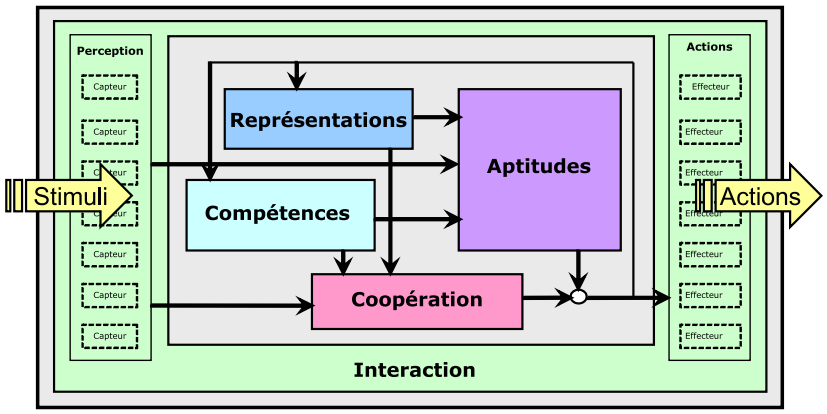


Figure 3.1 — Les différents modules d'un agent coopératif et leurs dépendances

essaient en priorité d'éviter les situations non coopératives, ils ont une ou plusieurs tâches à remplir correspondant à leurs objectifs individuels. Les compétences représentent un champ de connaissances permettant à l'agent de réaliser sa fonction partielle – en tant que partie d'un système réalisant une fonction globale. Aucune contrainte technique n'est requise pour concevoir et développer ce module. Par exemple, les compétences peuvent être implémentées comme une base de faits et de règles sur un domaine particulier. Dans ce module, nous pouvons aussi intégrer les caractéristiques intrinsèques aux agents, telles ques leur poids, leur couleur, etc.

Comme pour les compétences, le *module de représentations* peut être implémenté sous forme d'une base de faits et de règles, mais il porte sur la connaissance locale et subjective de l'environnement (physique ou social) et sur l'agent lui-même. Les croyances sur les autres agents et sur lui-même sont considérées comme des représentations. Les représentations peuvent être décomposées en systèmes multi-agents de plus bas niveau pour obtenir des représentations dynamiques par le biais de l'auto-organisation [Gleizes *et al.*, 2000b].

Les aptitudes représentent les capacités de l'agent à raisonner sur ses perceptions, ses compétences et ses représentations – pour interpréter des messages, par exemple. Le *module d'aptitudes* peut être un moteur d'inférences raisonnant sur des bases de compétences ou de représentations. Pour un état donné de compétences, de représentations et de perceptions, ce module doit en déduire une action à exécuter. Les cas pour lesquels le module est incapable de proposer une unique action, correspondent à des situations non coopératives devant être prises en compte.

L'attitude coopérative de l'agent est implantée dans le *module de coopération*. Comme le module d'aptitudes, ce module doit fournir une action en fonction d'un état courant de perceptions, de compétences et de représentations, mais seulement si l'agent est en situation non coopérative. Par conséquent, les agents doivent posséder des règles de détection de situations non coopératives et y associer des actions de résolution ou de prévention.

3.1.2 Fonctionnement interne

Lors de la phase de perception, le module de perception met à jour les valeurs des entrées. Ces données peuvent impliquer des changements directs dans les compétences ou les représentations. Une fois ces connaissances mises à jour, la phase de décision doit conduire au choix d'une action. Durant cette phase, les modules d'aptitudes et de coopération fonctionnent en parallèle. Le premier doit fournir une action correspondant à un comportement nominal. Le second doit détecter si l'agent est en situation non coopérative et s'il doit agir en conséquence. Dans ce cas, l'action corrective subsume l'action nominale. Si aucune action coopérative n'est proposée, l'agent est dans un état coopératif et peut donc agir de façon normale. Une fois une action choisie, l'agent agit lors de la phase d'action afin d'utiliser ses effecteurs (comprenant entre autre l'envoi de messages) ou de changer ses connaissances.

3.2 Utilisation du paradigme objet

De nombreuses approches considèrent l'agent comme une extension de l'objet. [Parunak, 1997] et [Odell, 2002] expliquent l'évolution des approches de programmation, de la programmation monolithique à la programmation par agent. Dans la programmation monolithique, les unités de logiciel représentent le programme dans sa globalité, entièrement spécifié par le programmeur et contrôlé par le système. La programmation modulaire répond à l'augmentation des besoins en matière de complexité, de mémoire des applications, ainsi que de réutilisabilité. Par contre, l'état des unités (modules) reste toujours sous contrôle externe par instructions d'appel. L'évolution logique est la décomposition des programmes en objets. Ici, l'état des objets est encapsulé grâce à des méthodes d'accès ou d'instanciation. En programmation par objet classique, les objets sont considérés comme passifs et donc soumis au contrôle par envoi de messages et appel de méthodes. Enfin, la programmation par agent se propose de concevoir les agents comme des « objets actifs avec initiative » – un peu comme les acteurs en conception objet – qui ont leur propre « thread » de contrôle et sont mus par des règles, des buts ou des intentions. Cependant, des approches récentes, provenant directement de l'ingénierie des objets, sont très proches des besoins de l'ingénierie des agents. Par exemple, la librairie Java ProActive pour le développement de grilles repose sur un schéma d'objets actifs et mobiles rigoureux [Baude *et al.*, 2000] utilisant le modèle de composants Fractal [Baude *et al.*, 2000; David, 2005]. Un autre modèle de composants, ACEEL, propose, *via* le schéma de conception *Strategy*, des capacités d'auto-adaptation des objets [Chefrour et André, 2003]. Malgré cela, ces approches restent loin des problématiques décisionnelles attachées aux agents, même si ACEEL propose un protocole d'adaptation nécessitant une capacité de décision des stratégies à mettre en œuvre.

[Odell, 2002] soulève la question de l'autonomie des agents suivant deux axes : l'autonomie dynamique (de passif à pro-actif en passant par réactif) et l'autonomie non-déterministe (d'attendu à inattendu). Les objets peuvent être identifiés comme des entités passives au comportement attendu alors que des agents, comme des agents de courtage, possèdent un comportement inattendu entre le réactif et le pro-actif. Les systèmes naturels, comme les colonies de fourmis, se placent en haut des deux échelles : pro-actifs et inattendus.

Les interactions entre agents, comparées aux interactions objet à objet, sont plus riches sémantiquement et ne sont pas uniquement des demandes d'exécution. Les agents peuvent bien sûr faire appel à des méthodes au sens objet du terme, mais y ajoutent un contenu plus riche, comme par exemple avec les actes de langage FIPA[1] ou KQML[2] – il n'est pas rare d'ailleurs de trouver des librairies dédiées à ces actes dans les plates-formes de développement de systèmes multi-agents. Alors que les objets voient les autres objets comme des fournisseurs de services par appel de méthodes, les notions de conversations, de communications à long terme et de réseaux de partenaires sont aussi des notions importantes en programmation par agent. En effet, les agents peuvent construire des réseaux de contacts et donc posséder des capacités cognitives correspondantes comme une mémoire ou des représentations sur les autres agents.

En définitive, les agents peuvent être vus comme des objets ayant des règles structurelles ou comportementales supplémentaires – définies par des stéréotypes ou des extensions du métamodèle UML. La plupart des concepts manipulés par la communauté agent peut être modélisée en objet (ontologie, dynamique d'états...) mais certains axes sont encore peu explorés. Il semble aussi que la caractéristique d'autonomie d'un agent [Carabelea *et al.*, 2003] reste le point difficile de la conception d'agent – comme cela l'est toujours dans le domaine de l'intelligence artificielle. Le problème est de trouver le bon niveau d'abstraction permettant de qualifier un agent d'autonome ou non. En effet, un agent reste toujours dépendant du système d'exploitation ou de la plate-forme sur laquelle il s'exécute, mais non vis-à-vis des autres agents ou des objets qui l'entourent.

Ce rapprochement entre les concepts d'objet et d'agent a bien entendu l'avantage de permettre de réutiliser/étendre des *modèles et notations*, comme UML, mais aussi des *processus*, comme le RUP (*Rational Unified Process*) ce qui est le cas dans ADELFE et MESSAGE, par exemple.

3.3 Notations et stéréotypage

Les notations sont une dimension importante dans la définition de méthodes. ADELFE, que nous suivons, a fait le choix de la notation UML compte tenu de la proximité relevée entre certains des concepts agents et le paradigme objet. Cependant, nous sommes conscients des limites d'UML, tant au niveau de l'expressivité que de la validation. En effet, les concepts UML ne sont pas suffisants pour exprimer les propriétés des systèmes multi-agents. De plus, UML propose des solutions de vérification syntaxique mais pas sémantique. Nous allons donc proposer des extensions à UML. Les extensions que nous proposons portent sur la vue statique des agents, par la définition de stéréotypes d'attributs et d'opérations, permettant un enrichissement sémantique et l'intégration des modules des agents coopératifs.

La première extension apportée à UML par ADELFE est le stéréotypage d'opérations et d'attributs en fonction de leur appartenance aux modules définis pour le modèle d'agent coopératif fourni par ADELFE (voir section 3.1). Tous ces stéréotypes s'appliquent à des traits

[1] http://www.fipa.org/repository/aclspecs.html
[2] http://www.cs.umbc.edu/kqml/

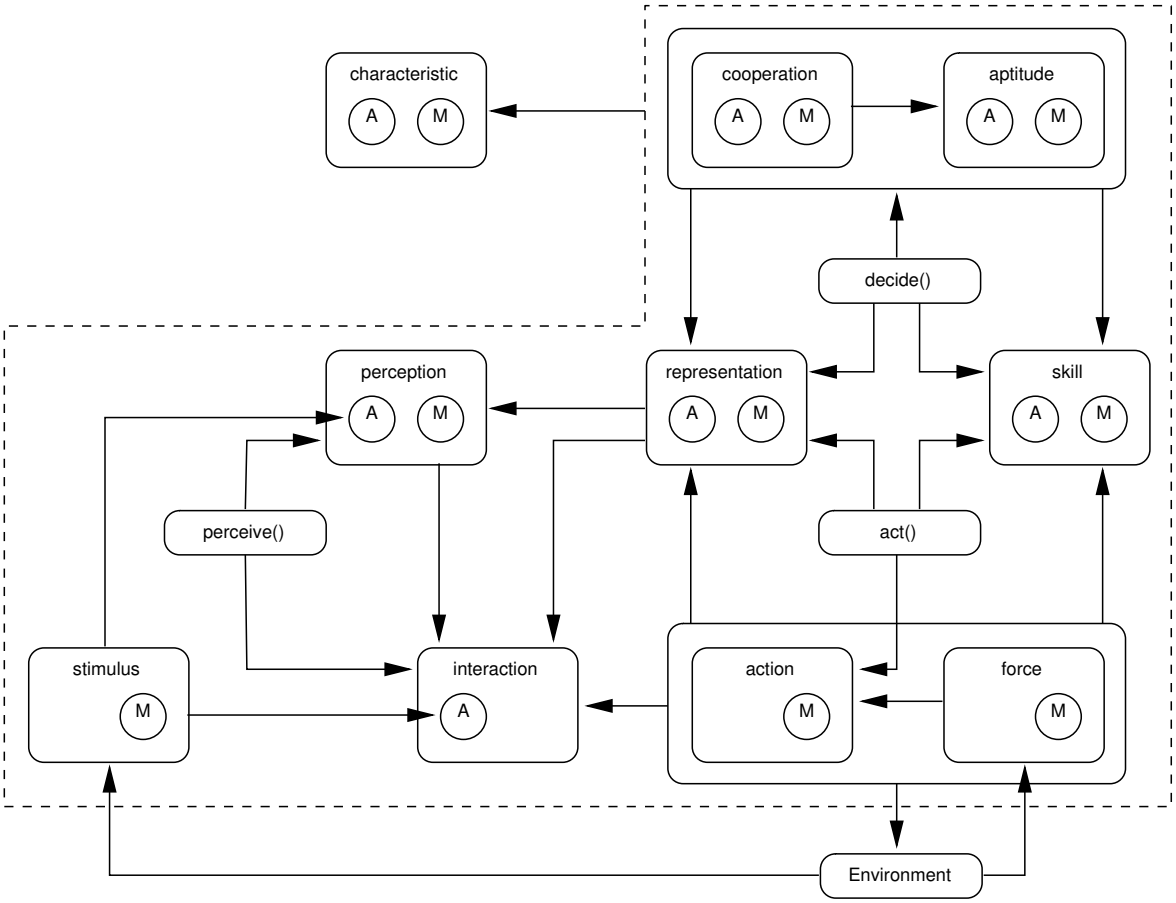


Figure 3.2 — Constituants d'un agent coopératif et visibilités associées (A pour attributs, M pour méthodes)

(de classe Feature du métamodèle UML), c'est-à-dire à des attributs ou à des opérations, excepté le stéréotype « cooperative agent » qui s'applique à des classes. Les stéréotypes forment une extension du métamodèle UML afin de contraindre l'utilisation du modèle d'agent coopératif. Toutefois, cette première proposition d'extension spécifiée dans [Picard, 2004] nous semble incomplète, nous présentons dans ces pages une nouvelle version. Ici, chaque stéréotype correspond à une sémantique particulière, mais surtout à des contraintes de visibilité entre modules. Ceci est résumé dans la figure 3.2 et détaillé dans les sections suivantes.

3.3.1 Le stéréotype de classe « cooperative agent »

Le stéréotype « cooperative agent » exprime le fait qu'une classe est un agent possédant une attitude coopérative participant à la construction d'un AMAS. Un agent sera donc modélisé comme une classe stéréotypée. Cette classe devra posséder une opération run() simulant le cycle de vie de l'agent. Pour assurer que cette opération existe, un agent doit hériter d'une superclasse abstraite appelée CooperativeAgent, fournie par ADELFE. Cette classe possède les opérations abstraites perceive(), decide() et act() et une opération run(). Ceci implique que les agents soient animés par l'opération run(), que ce soit par un ordonnanceur préétabli (dans le cas de *threads*, par exemple) ou spécifiquement développé. Ainsi, à chaque pas d'exécution,

l'ordonnanceur appellera l'opération run() de chaque agent dans un ordre préétabli ou aléatoire (dans le cas de *threads* par exemple). Cependant, c'est au développeur de remplir les opérations perceive(), decide() et act() de la classe d'agent, héritant de CooperativeAgent, qu'il code. Une classe stéréotypée « cooperative agent » doit hériter directement ou indirectement de la classe CooperativeAgent. De plus, afin de modéliser un agent coopératif « viable », une classe stéréotypée « cooperative agent » devra nécessairement posséder au moins des opérations ou attributs stéréotypés par chacun des stéréotypes suivants :

- « perception », car un agent doit forcément percevoir son environnement ;
- « action », car un agent doit agir s'il ne veut pas être inutile et donc non coopératif ;
- « aptitude », car un agent est une entité autonome dans ses décisions et raisonnements ;
- « cooperation », car un agent coopératif doit forcément avoir une attitude sociale coopérative.

3.3.2 Les stéréotypes d'attributs et de méthodes

Le stéréotype « characteristic » est utilisé pour pointer les propriétés intrinsèques ou physiques d'un agent coopératif. Un attribut ainsi stéréotypé représente la valeur de cette propriété. Une opération « characteristic » modifie ou met à jour ces propriétés. Une caractéristique peut être lue ou modifiée à n'importe quel moment du cycle de vie de l'agent. Elle peut aussi être lue ou appelée par les autres agents.

Le stéréotype « perception » exprime un moyen que l'agent a pour traiter et décoder de l'information venant de son environnement physique ou social. Les attributs représentent les données provenant de l'environnement ou décodées. Les opérations sont des moyens de mettre à jour ou de modifier les attributs « perception ». Une opération ou un attribut stéréotypé « perception » est nécessairement privé ou protégé. Un attribut « perception » peut être manipulé par l'opération perceive(), des opérations « stimulus » ou des opérations « representation ». Une opération « perception » peut être appelée par l'opération perceive() ou des opérations « representation ».

Le stéréotype « stimulus » exprime un moyen que l'agent a de recevoir de l'information venant de son environnement physique ou social (ou d'être prévenu de la disponibilité d'informations dans son environnement). Seules des opérations peuvent être stéréotypées « stimulus ». Une opération stéréotypée « stimulus » peut être appelée à n'importe quel moment du cycle de vie de l'agent. Elle peut aussi être appelée par les autres agents. Enfin, une telle opération ne peut accéder qu'à des attributs stéréotypés « perception » ou « interaction ».

Le stéréotype « action » est utilisé pour signaler un moyen d'agir sur l'environnement durant la phase d'action. Il ne peut être utilisé que pour des opérations qui sont alors les actions possibles pour un agent. Un agent est le seul à pouvoir activer ses propres actions afin d'assurer l'autonomie de contrôle. Une opération stéréotypée « action » est nécessairement privée et ne peut être appelée que lors de la phase d'action. Ceci signifie que les opérations stéréotypées « action » ne peuvent être appelées (directement ou indirectement) que par l'opération act(). Enfin, une telle opération ne peut accéder qu'à l'environnement, aux attributs stéréotypés « interaction », aux traits stéréotypés « representation » ou aux traits stéréotypés « skill ».

Le stéréotype « force » est utilisé pour modéliser des actions que l'environnement social ou physique peut réaliser sur l'agent sans que ce dernier ne puisse intervenir. Il ne peut être utilisé que pour des opérations qui sont alors accessibles uniquement à l'environnement. Il convient d'utiliser ce stéréotype avec parcimonie puisqu'il peut permettre de briser l'autonomie de contrôle. Il n'est disponible que pour modéliser des phénomènes d'ordre physique, par exemple un robot ne peut pas aller à l'encontre du déplacement engendré par un tremblement de terre. Enfin, une opération stéréotypée « force » ne peut accéder qu'à l'environnement, aux opérations stéréotypées « action », aux attributs stéréotypés « interaction », aux traits stéréotypés « representation » ou aux traits stéréotypés « skill ».

Le stéréotype « interaction » étiquette les attributs qui permettent à l'agent de communiquer ou d'agir sur son environnement. Ils peuvent donc être utilisés depuis une opération étiquetée « perception », « stimulus », « action », « force » ou « representation ».

Le stéréotype « skill » est utilisé pour étiqueter les compétences, c'est-à-dire des connaissances spécifiques à un domaine, permettant à l'agent de réaliser sa fonction partielle. Les opérations représentent les règles de raisonnement que peuvent avoir les agents. Les attributs sont les données (ou faits) sur le monde ou les paramètres des opérations stéréotypées « skill ». De tels attributs ou opérations ne sont accessibles qu'à l'agent les possédant pour exprimer son autonomie de décision. Les compétences peuvent être représentées par un système multi-agent adaptatif si les compétences ont besoin de changer au cours du temps. Une telle décomposition doit être détectée lors de l'analyse du système (voir activité A_{11} du processus ADELFE). Le système multi-agent résultant sera alors un attribut stéréotypé « skill » de la classe d'agent décomposée. Un attribut ou opération stéréotypé « skill » est nécessairement privé. De plus, de tels traits ne peuvent être uniquement appelés lors des phases de décision ou d'action de l'agent (seules les opérations decide() et act() peuvent l'appeler de manière directe ou indirecte).

Le stéréotype « aptitude » représente la capacité d'un agent à raisonner sur ses perceptions, ses connaissances et ses compétences. Les opérations expriment le raisonnement qu'un agent est capable de faire, par exemple de l'inférence sur ses compétences (qui peuvent alors être des règles de production et des faits). Les attributs représentent les données de fonctionnement ou les paramètres du raisonnement. Une opération ou attribut stéréotypé « aptitude » peut seulement être accessible à l'agent lui-même, pour exprimer son autonomie de décision. Une opération ou attribut stéréotypé « aptitude » est nécessairement privé ou protégé. Un attribut « aptitude » peut seulement être utilisé par une opération « aptitude ». Une opération « aptitude » peut uniquement être appelée lors de la phase de décision du cycle de vie de l'agent (opération decide()). Une opération « aptitude » peut uniquement faire appel à des attributs ou opérations stéréotypés « characteristic », « skill », « representation ». Cette dernière règle met en place le fonctionnement interne de l'agent, comme exprimé dans la section 3.1.

Le stéréotype « representation » est un moyen d'indiquer les représentations du monde que possède l'agent. Les attributs « representation » sont des unités de connaissances. Les opérations « representation » sont des moyens de les manipuler : accès, modification, etc. Les représentations peuvent évoluer, et, de ce fait, être modélisées par un AMAS. Une opération

(ou attribut) « representation » est nécessairement privée ou protégée. En effet, la connaissance d'un agent est locale et personnelle, à moins qu'il ne la communique *via* des messages, par exemple. Un trait « representation » peut seulement être accédé lors des phases de décision ou d'action (opérations decide() et act()). Enfin une opération « representation » peut faire appel à des traits « perception » et « interaction ».

Le stéréotype « cooperation » exprime l'attitude sociale coopérative de l'agent en implantant les règles de résolution des Situations Non Coopératives (SNC). L'agent doit avoir un ensemble de règles (ou prédicats) permettant de détecter les SNC. Ces règles sont écrites en utilisant les perceptions, les représentations et les compétences. L'agent doit aussi posséder des opérations de résolution associées aux règles de détection. Une opération « cooperation » pourra être, par exemple :

- une opération renvoyant un booléen indiquant la détection d'une SNC et possédant des paramètres provenant de perceptions, de représentations et/ou de compétences ;
- une opération de résolution associant une ou plusieurs actions possibles à chaque SNC. Le choix, dans le cas d'actions multiples, devra aussi être géré par une opération « cooperation » ou « aptitude ».

Ces opérations « cooperation » doivent subsumer les opérations « aptitude », c'est-à-dire que les actions choisies par les opérations « cooperation » prennent le pas sur les actions choisies par les opérations « aptitude ». Ceci peut être obtenu de plusieurs manières : à l'aide d'instructions conditionnées ou en utilisant des exceptions pour simuler la priorité des résolutions de situations non coopératives. Une opération ou un attribut « cooperation » est nécessairement privé ou protégé et peut seulement être manipulé durant la phase de décision (opération decide()). Une opération « cooperation » peut accéder à des traits « aptitude », « representation », « skill », ou « characteristic ».

Grâce à ce nouvel ensemble de stéréotypes, nous couvrons la totalité de la conception statique des agents. Alors, le code conditionnant l'organisation interne de nos agents peut être obtenu par de la génération automatique de code. Nous abordons donc cet aspect qui nous permet de faire la passerelle entre la conception et l'implémentation de nos agents.

3.4 De la conception à l'implémentation

3.4.1 Positionnement MDA

L'approche envisagée dans ce chapitre pour prendre en charge le passage de la conception à l'implémentation dans le processus ADELFE, s'inscrit dans la proposition de MDA (*Model Driven Architecture* ou architecture basée sur les modèles) faite par l'OMG[3] [Kleppe *et al.*, 2003]. Le MDA a pour but d'accélérer le développement d'applications en séparant les aspects domaines de l'application, des aspects liés à son implantation sur une architecture particulière. Il propose ainsi de résoudre les problèmes d'interopérabilité et de portabilité. Plus concrètement, l'approche MDA consiste à définir les fonctionnalités du système dans un modèle indépendant de la plate-forme (PIM : *Platform Independant Model*) puis, de le

[3]http://www.omg.org/mda

traduire dans un modèle spécifique à une plate-forme (PSM : *Platform Specific Model*), base pour la génération de code exécutable. L'objectif visé par l'OMG et en partie atteint, consiste à rendre toutes ces transformations de modèles automatiques. Nous désirons, dans le cadre de notre travail, aller vers la manipulation d'un modèle d'agent de haut niveau, indépendant de la plate-forme et permettant de générer au moins semi-automatiquement un ou plusieurs modèles spécifiques à une plate-forme puis du code.

Dans cette optique, il nous paraît nécessaire de tenir compte du caractère agent du système dès le PIM. Certains pourront argumenter qu'il s'agit là d'une décision de conception conditionnant la plate-forme utilisée pour le système. Il faut toutefois considérer que l'utilisation ou non d'agents est un choix de très haut niveau et indépendant d'une plate-forme agent particulière. Nous notons aussi que la distinction entre PIM et PSM se fait finalement de manière relative [Kleppe *et al.*, 2003]. Le type de modèle que nous considérons comme PIM dans la suite de ce chapitre pourra donc être considéré comme un PSM pour une personne voulant faire une modélisation de plus haut niveau.

Dans ce cadre, le PIM sera un modèle décrivant les agents du système – qui tirera bien évidemment partie des notations discutées dans la section 3.3, comme le montre la figure 3.3. Une fois transformé, le modèle disponible sera un PSM objet décrivant la structure des agents du système. À partir de ce modèle, il devient tout à fait possible de générer du code. Notons toutefois que la solution proposée est incomplète puisque nous ne couvrons pas la dynamique de nos agents. On pourra se reporter à [Ottens *et al.*, 2006] où nous proposons des dispositions supplémentaires couvrant une partie de la dynamique des agents. Pour cela, nous présentons des moyens pour décrire les protocoles utilisés par les agents ainsi que leur transformation en machines à états. La solution n'est toutefois pas complète, car aucune proposition n'est faite pour lier les différentes phases de fonctionnement des agents avec les machines à états.

3.4.2 Du modèle statique à la structure objet de l'agent

Dans la section 3.3, nous avons présenté un modèle fonctionnel à visibilités restreintes. Il s'agit bien d'un modèle de haut niveau qui nécessite de progresser vers une solution implémentée et conforme au modèle initial. En ce sens, il s'agit d'un métamodèle relativement au PIM. Dans ce cas, les règles de transformation pour passer du PIM au PSM doivent garantir ces propriétés. Pour cela, nous avons mis au point une procédure permettant de créer un modèle objet restreignant la visibilité des méthodes de l'agent conformément aux contraintes des stéréotypes ADELFE. Cette procédure s'inscrit donc dans une chaîne MDA, comme précisé dans la figure 3.3. Nous la détaillons pour un agent donné (pour cela nous utilisons l'agent décrit figure 3.4 comme exemple).

Tout d'abord, pour chaque stéréotype « s1 », nous insérons deux classes nommées S1Fields et S1MethodsBase (voir figure 3.5). Elles contiennent respectivement les attributs et les méthodes appartenant à « s1 ». Tous les traits ainsi créés sont d'accès public et les opérations sont abstraites.

Ensuite, pour chaque stéréotype « s1 » un deuxième traitement est opéré. Nous insérons une classe dont le nom est S1Namespace (voir figure 3.6). Dans cette classe nous insérons un

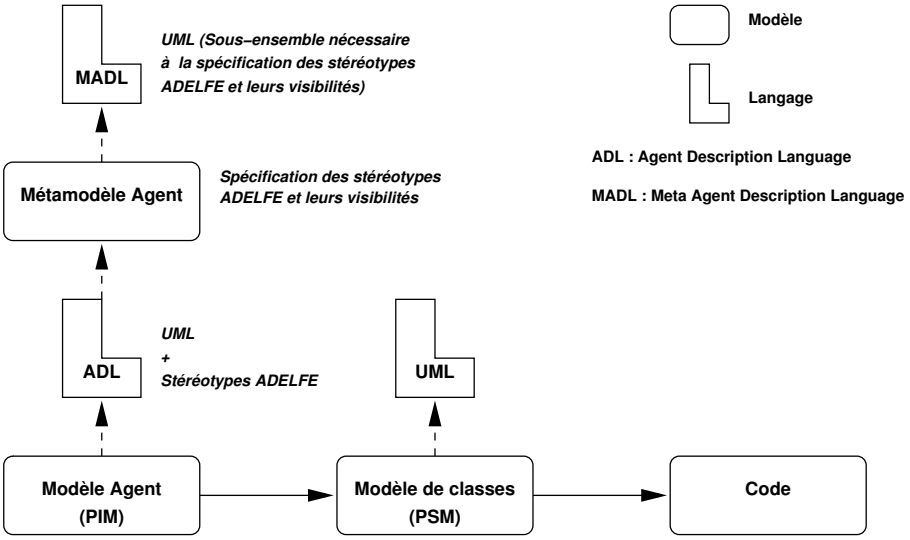


Figure 3.3 — Une chaîne de type MDA pour l'aspect statique d'un agent

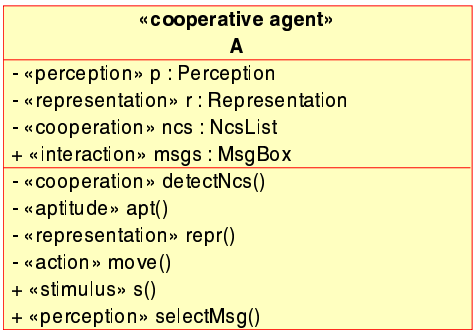


Figure 3.4 — Exemple de classe d'agent coopératif obtenue en fin de conception

attribut s1Fields de type S1Fields. Et, pour chaque stéréotype « s2 » directement accessible (sans transitivité) depuis le « s1 », on insère dans la classe S1Namespace un attribut s2Fields de type S2Fields, et un attribut s2Methods de type S2MethodsBase en fonction des accès autorisés dans le modèle. Il ne reste alors qu'à insérer dans la classe S1MethodsBase un attribut protégé nommé self et de type S1Namespace. Ainsi les méthodes du stéréotype « s1 » ont uniquement accès aux constituants de l'agent autorisés par le modèle par le biais de self.

Ensuite, une procédure similaire est utilisée pour les phases (perceive(), decide(), act()) du cycle de vie de l'agent (voir figure 3.7). Nous insérons donc pour chaque phase ph(), une classe PhPhase (contenant une unique opération ph()) et une classe PhNamespace. De même, la classe PhPhase dispose d'un attribut self de type PhNamespace.

Figure 3.5 — Etape 1 de la génération de la structure : ajout des classes d'attributs et de méthodes pour un stéréotype

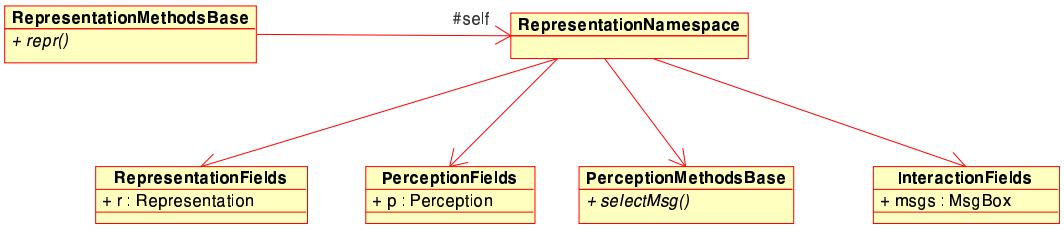


Figure 3.6 — Etape 2 de la génération de la structure : ajout de la classe d'espace de nom pour le stéréotype et de l'attribut self pour l'accès à ses méthodes

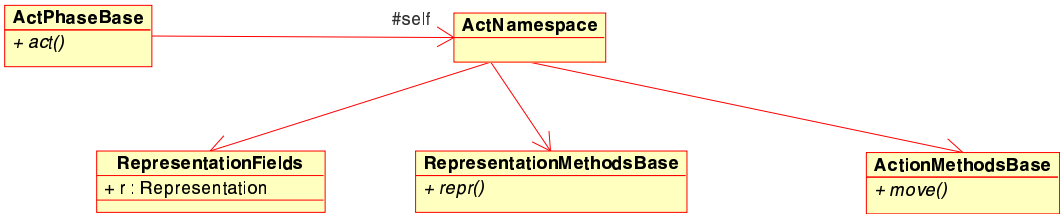


Figure 3.7 — Etape 3 de la génération de la structure : ajout de la classe d'espace de nom pour chaque phase de l'agent (ici, action) et de l'attribut self correspondant

Enfin, nous insérons une classe AgentBase contenant pour chaque stéréotype « s1 » un attribut s1Fields de type S1Fields et un attribut s1Methods de type S1MethodsBase, et, pour chaque phase ph() un attribut phPhase de type PhPhase (voir figure 3.8). Tous ces attributs sont privés. Ceci conclut le passage du PIM au PSM.

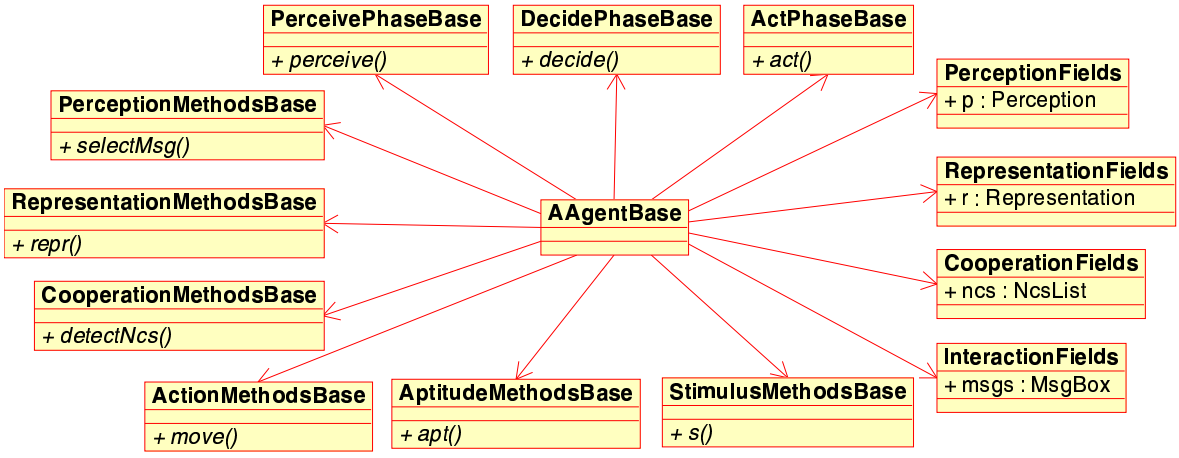


Figure 3.8 — Etape 4 de la génération de la structure : ajout de la classe de l'agent

Lors du passage du PSM au code, il convient d'obtenir une initialisation correcte des champs de la classe AgentBase. En effet, elle doit disposer d'un constructeur qui va initialiser les différents champs suffixés Base (en général avec une sous-classe non abstraite pour garantir que les méthodes sont bien implémentées) ainsi que leurs champs self.

Grâce à cette procédure il est tout à fait possible de générer le code de nos agents. C'est pourquoi nous avons spécifiés deux dialectes XML correspondants respectivements à notre

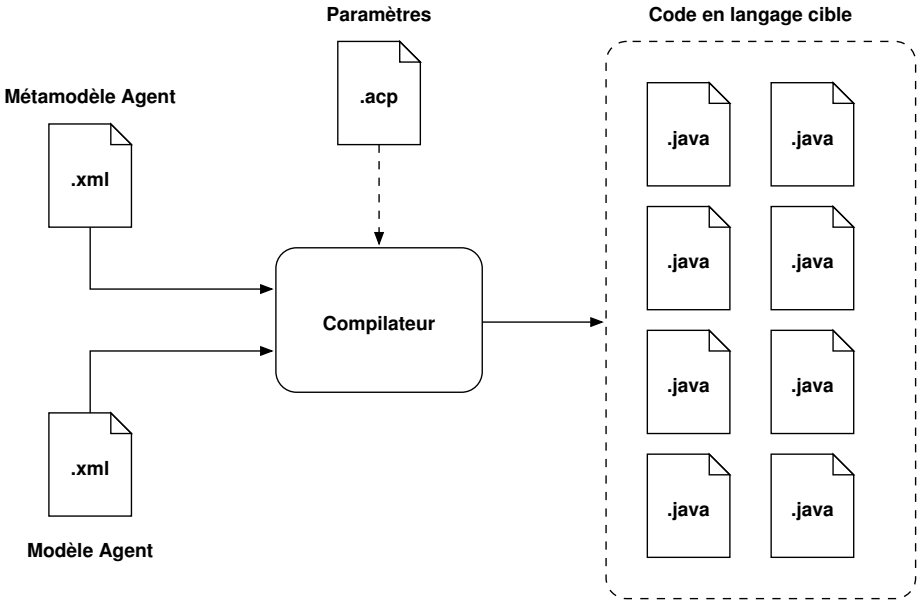


Figure 3.9 — Entrées et sorties du compilateur d'agents

méta-langage de description des agents et à notre langage de description des agents. Ainsi, nous avons développé un protoype de *compilateur d'agents* prenant en entrée deux fichiers XML correspondants au méta-modèle agent et au modèle agent dont on veut générer le code. En sortie de cet outil, on obtient le code source de nos agents conformément à la procédure décrite ci-dessus.

3.5 Bilan

Dans ce chapitre, nous avons vu comment il est possible d'automatiser l'écriture du code d'un système multi-agent. Nous disposons à présent d'un cadre semi-formel apporté par l'approche MDA pour décrire la structure interne de nos agents. Ainsi, nous disposons à présent d'un *compilateur d'agents* (voir figure 3.9) permettant de passer d'une représentation de haut niveau à un code objet classique.

Nous utilisons cette approche dans le chapitre suivant pour la conception du système Dynamo. Grâce à ce complément de la méthode ADELFE, nous pouvons garantir que le code produit suit les contraintes induites par la conception prévue. Ainsi, nous avons pu nous attaquer plus spécifiquement au problème posé de la construction d'ontologies. Bien que poussant à avoir un code plus verbeux, la rigueur introduite par l'approche MDA nous a guidés tout au long du développement de notre prototype.

4 Dynamo : système multi-agent pour les ontologies dynamiques

> « Il ne peut pas réellement nous comprendre, tu sais. Il dispose bien de ses profils mais ce ne sont jamais que des statistiques. »
>
> *William Gibson – Neuromancien*

COMME nous l'avons vu dans le chapitre 1, la construction d'ontologies à partir de textes nécessite encore d'évoluer pour devenir plus efficace. En particulier, le temps nécessaire pour dépouiller ou traiter les grandes quantités d'informations qui peuvent être tirées des textes, nous a incités à définir des logiciels pour les construire de façon automatique ou semi-automatique, et les maintenir à jour. Cela justifie l'émergence de propositions pour l'obtention d'ontologies dynamiques [Heflin et Hendler, 2000], et plus particulièrement les techniques d'apprentissage d'ontologies [Maedche et Staab, 2000b; Maedche, 2002].

Dans ce chapitre, nous présentons le cœur de ce travail, Dynamo (un acronyme pour « DYNAMic Ontologies »), un outil basé sur un système multi-agent adaptatif pour aider à construire et maintenir une ontologie depuis un corpus textuel. Il a été développé en utilisant la méthode ADELFE révisée par nos soins et plus particulièrement le modèle d'agent coopératif présenté au chapitre 3. Comme pour les outils présentés dans le chapitre 1, notre but n'est pas de construire une ontologie exhaustive, mais une ressource termino-ontologique spécifique à un domaine. Cet outil est semi-automatique car il est fondé sur un processus de construction en interaction avec l'ontologue, puisque toutes les informations requises pour faire un modèle de ce type ne sont pas présentes dans les textes.

Ce chapitre présente, dans la section 4.1 les motivations qui nous ont poussés à la création de Dynamo ainsi que les buts visés. Puis, la section 4.2 donne une vue d'ensemble du système, en expliquant comment ADELFE – complétée par les aménagements du chapitre précédent – a été mise en oeuvre et en détaillant l'architecture globale de Dynamo. Ensuite, dans la section 4.3, nous discutons l'algorithme de classification hiérarchique utilisé dans Dynamo. Nous le comparons à un algorithme centralisé de classification hiérarchique d'approche plus classique. La section 4.4 est consacrée à des améliorations du comportement des agents pour tenir compte de critères supplémentaires ignorés par la classification. Enfin, dans la section 4.5, nous présentons une autre extension sur le comportement des agents

permettant de simplifier la taxonomie.

4.1 L'ontologie en tant que système multi-agent

Dynamo vise à réduire le besoin d'intervention manuelle dans l'examen des résultats d'analyses textuelles en suggérant un brouillon de réseau conceptuel afin de construire efficacement des ontologies. Nous voulons améliorer le processus de construction d'une ontologie à partir de textes afin de rendre plus simple les modifications par l'utilisateur et pour réduire sa charge de travail. Un objectif est alors de permettre à un ontologue de construire et faire évoluer une ontologie à partir de textes de manière cohérente, en restant dans le même environnement, et de manière continue. A tout moment, il doit pouvoir soit figer le modèle, soit le faire évoluer en prenant en compte de nouveaux textes ou en effectuant des modifications manuelles.

L'approche choisie est à notre connaissance complètement originale et repose sur l'usage d'un système multi-agent adaptatif. Ce choix provient des qualités offertes par les systèmes multi-agents adaptatifs : ils peuvent faciliter la conception interactive d'un système fonctionnellement adéquat [Georgé *et al.*, 2003b] (dans notre cas, un réseau conceptuel), ils permettent sa construction incrémentale en prenant progressivement en compte les nouvelles données (tirées des analyses textuelles et de l'interaction avec l'utilisateur). Enfin, ils peuvent être facilement distribués sur un réseau informatique.

Dynamo utilise des résultats d'analyses syntaxiques et terminologiques de textes comme entrée. Il exploite différents critères basés sur des statistiques calculées sur les contextes linguistiques des termes pour créer et positionner les concepts. Le cœur du comportement des agents est inspiré de la classification. Contrairement aux approches de classification plus classiques, l'ontologue doit pouvoir réviser ses choix à tout moment en fonction du résultat courant. Les approches classiques ne permettent pas cette révision des résultats : quand un niveau de la hiérarchie est constitué par une approche classique, il ne peut pas être modifié plus tard sans perdre les résultats intermédiaires.

En sortie, Dynamo fournit à l'ontologue une organisation hiérarchique de concepts (le système multi-agent lui-même) qui peut être validée, raffinée ou modifiée, jusqu'à obtenir un état satisfaisant du réseau sémantique. Cette organisation réagit dynamiquement en fonction des pressions de l'ontologue.

Pour prendre en charge la flexibilité requise par un tel système, nous utilisons un système multi-agent auto-organisé basé sur l'approche par AMAS (voir chapitre 2), ce qui explique pourquoi nous avons conçu les règles suivies par les agents afin de maximiser la coopération au sein du système. Pour cela, nous avons suivi la méthode ADELFE complétée des aménagements présentés au chapitre 3 pour concevoir et implémenter le système Dynamo.

Dans un premier temps, nous nous concentrons uniquement sur la composante hiérar-

chique de l'ontologie, c'est-à-dire une taxonomie de concepts. Il nous paraissait judicieux de se concentrer d'abord sur cette composante centrale de l'ontologie avant de pouvoir s'attaquer aux problèmes supplémentaires que posent la création de relations transverses.

4.2 Vue d'ensemble de Dynamo

4.2.1 Application d'ADELFE et choix fondamentaux

La méthode ADELFE utilisée pour le développement d'AMAS nous a servi de guide pour poser les bases de notre prototype. Avant, de préciser les contours de l'architecture du système, ainsi que le détail de son comportement, nous présentons l'impact que les activités centrées sur les agents dans ADELFE a eu sur notre réflexion (cf. section 2.5). Nous nous attachons ici à uniquement faire ressortir les aspects spécifiques au développement d'un système multi-agent, nous ne présentons pas les activités du développement logiciel en général.

WD_2 : Expression des besoins finals

Tout d'abord, l'expression des besoins finals nous a permis de nous intéresser en particulier à l'environnement (A_6) du système. Nous avons d'abord cherché à identifier les entités du système. Bien sûr, suite aux réflexions présentées au chapitre 1, nous avons considéré les termes et les concepts comme entités essentielles de notre système.

Ainsi, il nous restait à trancher sur l'environnement lui même. Il s'agit d'un environnement principalement social, les agents ne seront pas situés. Les différentes entités proviendront soit des textes (pour les termes) soit de l'activité du système suite aux interventions de l'ontologue (pour les concepts). Autour du système dans son ensemble nous aurons la sortie Syntex comme première source de données, et l'ontologue qui exercera des pressions sur le système.

WD_3 : Analyse

Ensuite, l'analyse nous a poussés à vérifier l'adéquation des AMAS pour notre problème (A_{11}), à identifier les agents (A_{12}) et étudier les relations entre agents (A_{13}-S_3).

Vérifier l'adéquation des AMAS requiert de s'interroger selon deux points de vue, global et local. Global tout d'abord : nous devons nous demander si le problème à résoudre est complexe ou dispose d'une solution algorithmique connue. Les AMAS paraissent alors être une bonne approche pour notre problème, car d'un point de vue global, le problème de la construction d'ontologies à partir de textes est complexe et ne dispose pas de solution algorithmique connue a priori mais seulement des approches partielles. Et, les entités dégagées précédemment (termes et concepts) sont présentes en nombre important lors de la résolution, ce qui est un indice supplémentaire dénotant de l'adéquation de cette technique.

L'identification des agents du système se fait parmi les entités précédemment identifiées (A_6). Ils nous faut identifier celles qui seront « agentifiées ». Ici, seuls les concepts sont de-

venus des agents. En effet, ces derniers doivent se positionner et changer de place dans une organisation et disposent de mesures statistiques qui peuvent varier en fonction de ces positionnements. A contrario, parce que notre système n'a pas vocation à réaliser une analyse de textes, les termes ne disposent que de données figées en provenance de l'extracteur de termes. Des calculs complémentaires seraient possibles ; ils seraient d'un autre ordre. Nous avons préféré n'exploiter que les statistiques en provenance d'une analyse du corpus textuel. Nous avons donc choisi de ne retenir que des *agents-concepts*.

Enfin, dans notre cas, l'identification des agents a contraint les relations possibles. Puisque tous nos agents représentent des concepts, les seules relations disponibles entre eux sont des relations de position dans une organisation conceptuelle. Nous nous sommes arrêtés à la production de la hiérarchie pour des raisons pratiques de temps. Chaque agent devra donc déterminer son parent, ses enfants et ses frères.

WD_4 : Conception

Enfin, la conception (WD_4) nous a permis de donner les grandes lignes de notre système. Dans notre cas, une ontologie peut être vue comme une carte stable constituée d'entitées conceptuelles, ici représentées par des agents, liées par des relations étiquetées. Donc, notre approche considère une ontologie comme un équilibre dynamique entre les *agents-concepts* qui la composent, dont les forces sont définies par leurs relations potentielles. La modification de l'ontologie est alors une perturbation de l'équilibre précédent par l'apparition ou la disparition d'agents ou de relations. Dans ce cas, une ontologie dynamique est un processus auto-organisé déclenché lorsque de nouveaux textes sont inclus dans le corpus, ou quand l'ontologue interagit avec, en critiquant ou modifiant l'état courant.

Nous sommes partis de ce point de vue pour ensuite détailler l'architecture du système ainsi que la structure interne des agents et leur comportement. Ces deux aspects sont détaillés dans la suite de ce chapitre, et la phase de conception interne des agents tient compte des compléments apportés à ADELFE au chapitre 3.

4.2.2 Architecture proposée

Le système Dynamo se compose de trois grandes parties (cf. figure 4.1) :

- le réseau de termes, obtenu en sortie d'un extracteur de termes utilisé pour pré-traiter le corpus textuel ; il s'agit de syntagmes nominaux sous forme lemmatisée comme par exemple « hépatite viral » ; il s'agit de données non modifiées par le système ;
- le système multi-agent qui utilise le réseau de termes pour effectuer une classification hiérarchique afin d'obtenir une taxonomie ; les agents du système coopèrent afin de se positionner dans une hiérarchie en fonction des données et des indications de l'ontologue ; c'est l'état final du système multi-agent qui constitue la taxonomie ; puisque le système est la taxonomie et qu'il est porteur du processus de construction, cette taxonomie peut plus facilement être maintenue ;
- l'interface permettant à l'utilisateur de visualiser et contrôler le processus de classification et ses résultats.

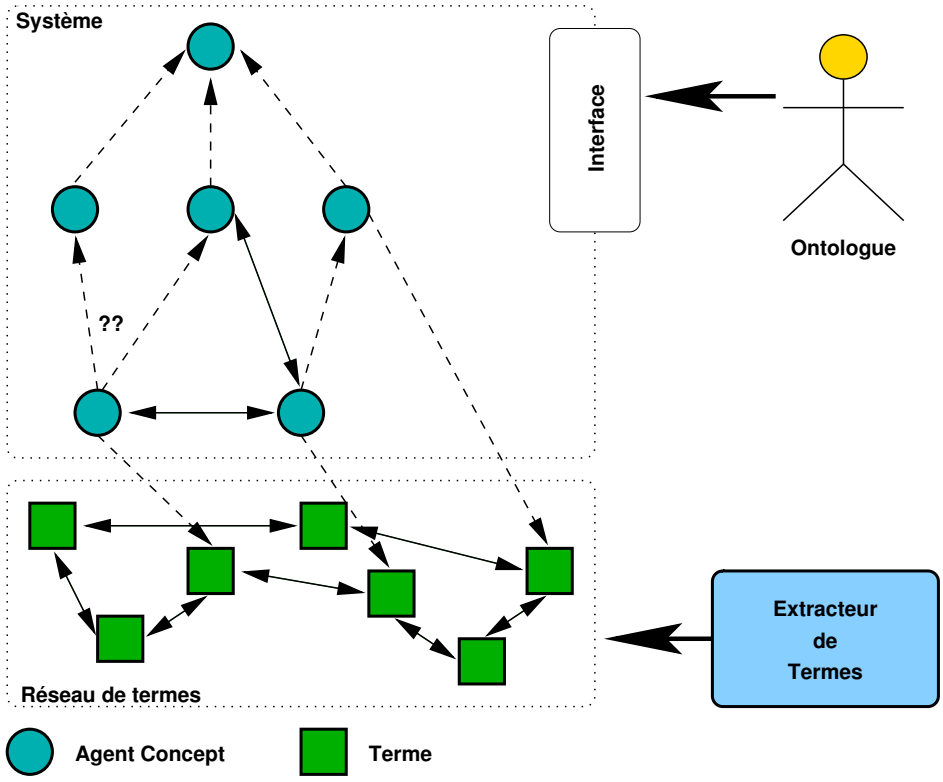


Figure 4.1 — Architecture du système Dynamo

L'extracteur de termes choisi est le logiciel Syntex, utilisé notamment pour des tâches de construction d'ontologies [Bourigault et Aussenac-Gilles, 2003]. Nous l'avons sélectionné principalement pour sa robustesse et la grande quantité d'informations extraites à la fois par les propositions de dépendances syntaxiques et par l'analyse distributionnelle. Nous nous sommes concentré sur le réseau « Tête-Expansion », créé par cet outil, qui a déjà prouvé être une structure intéressante pour un système de classification [Assadi, 1998]. Dans ce réseau, chaque terme est relié, d'une part à sa tête[1] et son expansion[2], et d'autre part à tous les termes dont il est lui-même tête ou expansion. Par exemple, « algorithme centralisé de classification » a comme tête « algorithme centralisé » et comme expansion « classification ». De même, « algorithme centralisé » est composé de « algorithme » et « centralisé ». Pour un exemple plus complet, on se reportera à la figure 4.2.

Le réseau de termes récupéré en sortie de l'extracteur est stocké dans une base de données. Pour chaque paire de termes, on suppose qu'il est possible de calculer son indice de similitude (ou similarité), en vue d'effectuer une classification [Faure, 2000] [Assadi, 1998]. Comme son nom l'indique, un tel indice permet d'évaluer à quel point deux éléments d'une population sont similaires ; il existe de nombreux indices de similitude dans la littérature. De par la nature des données, nous nous intéressons uniquement à des indices de similarité entre objets décrits par des variables binaires, c'est-à-dire qu'un individu est décrit par la présence ou l'absence d'un ensemble de caractéristiques [Saporta, 1990; Manning et Schütze, 1999]. Dans le cas de termes, il s'agit, en général, des contextes d'utilisation. Avec

[1] *i.e.* le sous-syntagme maximal situé en tête du terme
[2] *i.e.* le sous-syntagme maximal situé en queue du terme

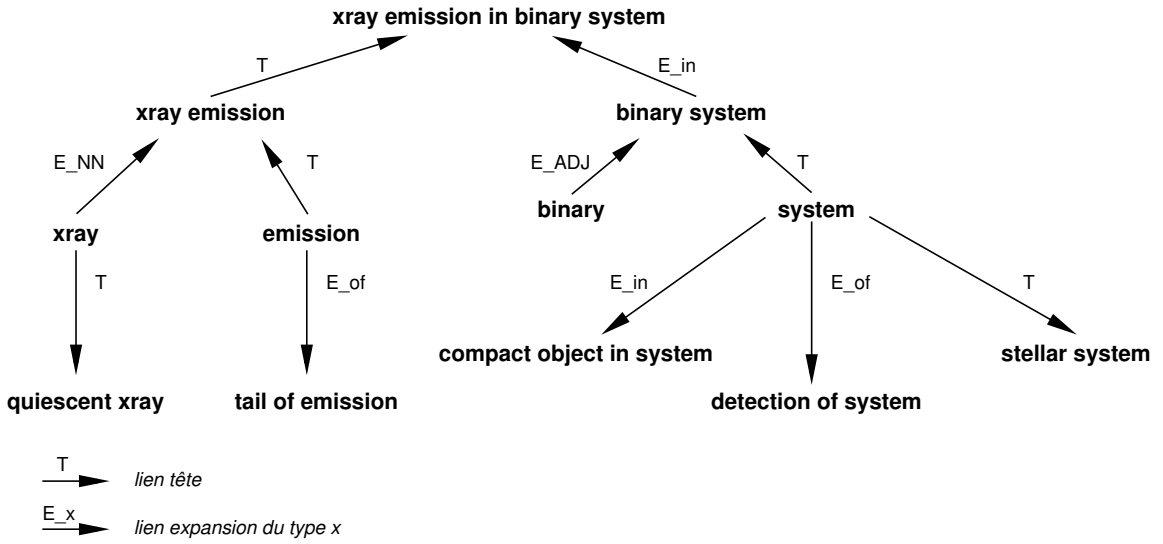


Figure 4.2 — Extrait de réseau « Tête-Expansion »

le réseau « Tête-Expansion » de Syntex, ces contextes sont identifiés par les expansions dont les termes considérés sont en tête.

Le système multi-agent implémente l'algorithme distribué de classification décrit en détail dans la section 4.3 ainsi que les règles décrites dans la section 4.4. Il est conçu pour être à la fois le système produisant la structure résultante et la structure elle-même. En ce sens, chaque agent est une entité informatique représentant un concept, dont le comportement autonome lui permet de trouver, à partir d'une vue locale de ses voisins, sa place dans l'organisation, à savoir l'ontologie. Ils disposent tous de capacités de communication et d'algorithmes leur permettant de s'approcher ou de s'éloigner selon différents critères. La sortie du système est donc l'organisation obtenue par l'interaction des agents, tout en tenant compte des pressions extérieures exercées par l'ontologue lorsqu'il effectue des modifications de la taxonomie selon les besoins de l'application et l'expertise du domaine qu'il peut recueillir auprès d'experts.

4.2.3 Structure interne des agents-concepts

Dans cette section, nous décrivons la structure interne des agents au sein de Dynamo. Ils ont été conçus en appliquant la méthode ADELFE avec les extensions décrites au chapitre 3. Ainsi, nous détaillons le contenu des composants de la figure 3.1 qui sont utilisés dans nos agents.

Composant « characteristic »

La seule caractéristique de nos agents est un identifiant (unique dans le système) utilisé pour les envois de messages.

Composant « perception »

Le composant perception dispose d'un ensemble de méthodes permettant d'extraire plusieurs types d'informations spécialisées en provenance du bus de communication (voir le composant « interaction ») :

- les mises à jour de la hiérarchie lorsqu'un agent se déplace ;
- les problèmes signalés par d'autres agents ;
- la demande la plus critique du voisinage.

Ces informations sont ensuite utilisées pour la mise à jour des représentations.

Composant « representation »

Chaque agent manipule un certain nombre de représentations sur ce qui l'entoure. Dans notre cas, ils n'ont aucune représentation sur un environnement physique, c'est pourquoi les seules représentations dont ils disposent sont sur eux-mêmes et leurs voisins.

En particulier, chaque agent gère un ensemble d'accointances et conserve aussi l'information de la position de ses pairs dans la hiérarchie : frère, parent, enfant ou autre. Pour compléter cela, il conserve une liste des agents de son voisinage étant dans une situation non coopérative. Il peut ainsi s'en servir pour leur venir en aide et entreprendre une action leur permettant de sortir de la situation non coopérative détectée.

De plus, concernant ses représentations sur lui, chaque agent possède une liste des termes le représentant dans le réseau tête-expansion fourni par Syntex. Ainsi, si cette liste est vide, notre agent est un agent de structuration de la hiérachie, il est nommé ainsi car la seule information qu'il apporte concerne la structure de l'ontologie et le positionnement relatif des concepts. Chaque agent conserve aussi une liste des actions qu'il prévoit d'exécuter dans le cycle d'exécution courant. Donc, pour chaque cycle d'exécution de l'agent, cette liste commence en étant vide, puis est remplie durant la phase de raisonnement, pour être à nouveau vidée pendant la phase d'action (où sont exécutées les actions prévues).

Enfin, on ajoutera à cela deux valeurs de tolérance sur la similitude notées ε_P et $\varepsilon_{P'}$ qui seront détaillées dans la section 4.5.

Composant « interaction »

Dans notre système, les agents sont purement communicants dans le sens où ils ne sont pas situés dans un environnement physique et peuvent uniquement communiquer entre eux. Donc le seul objet disponible dans ce composant est une boîte aux lettres permettant d'utiliser le bus de communication du système par envoi et réception de messages.

Composant « cooperation »

Ce composant est structuré selon deux axes. Premièrement, un module de gestion des situations non coopératives. En fonction des représentations, ce module détermine le niveau de non coopération courant de l'agent ainsi qu'un classement des situations non coopératives détectées en fonction de leur gravité. Deuxièmement, le composant dispose d'un

ensemble de règles condition/action. Lorsqu'une règle est applicable, elle donne une liste d'actions à exécuter avant la fin du cycle d'exécution de l'agent.

C'est ce composant qui implémente le comportement de nos agents. Il est donc détaillé dans les sections qui suivent où nous présentons l'ensemble des règles utilisées.

Composant « action »

Dans ce composant sont exécutées les actions planifiées dans le composant de représentations. Toutes les actions possibles des agents concernent l'envoi de message, la mise à jour de représentations internes et la gestion de leur durée de vie. À l'aide de ces primitives les actions suivantes sont implémentées :

- communiquer ses préférences de voisinage à son parent ;
- proposer une modification du seuil $\varepsilon_{P'}$ à ses fils ;
- proposer un nouveau parent à ses fils ;
- changer de position dans la hiérarchie ;
- créer un nouvel agent dans le système ;
- disparaître du système.

La mise en contexte de ces actions par rapport au comportement des agents est faîte dans les sections suivantes.

4.3 Hiérarchie basée sur un procédé de classification

4.3.1 Rappels de classification

Afin d'améliorer la compréhension de notre algorithme distribué de classification, et en vue de son évaluation dans la section 5.1.1, nous rappelons ici l'algorithme de base utilisé pour une classification hiérarchique ascendante dans un espace non métrique, mais dont la mesure de similitude utilisée est symétrique (ce qui est le cas avec les mesures utilisées dans notre système) [Saporta, 1990; Manning et Schütze, 1999].

Dans l'algorithme 4.1, à chaque étape de la classification, la paire des éléments les plus similaires est déterminée. Ces deux éléments sont regroupés, et la classe résultante est insérée dans la liste des éléments restants. L'algorithme s'arrête quand cette liste ne contient plus qu'un élément.

La structure hiérarchique résultante de l'algorithme 4.1 est nécessairement un arbre binaire de par le regroupement deux à deux. Le regroupement des éléments les plus similaires revient aussi à les éloigner des éléments dont ils sont les plus dissimilaires. L'algorithme distribué multi-agent que nous proposons a été conçu autour de ces deux constats. Cet algorithme distribué est exécuté de manière concurrente par chacun des agents du système.

Notons aussi que, dans la suite de ce mémoire, lors de l'évaluation des deux algorithmes, nous utilisons une stratégie d'agrégation en moyenne [Saporta, 1990]. Ce qui nous donne la formule suivante pour calculer la similitude entre deux groupes de termes (de cardinalités

Algorithme 4.1 — Algorithme centralisé de classification hiérarchique ascendante

Entrées : La liste L des individus à hiérarchiser
Sorties : La racine R de l'arbre de classification

```
1  tant que longueur(L) > 1 faire
2      max ← 0;
3      A ← nil;
4      B ← nil;
5      pour i ← 0 à longueur(L) faire
6          I ← L[i];
7          pour j ← i + 1 à longueur(L) faire
8              J ← L[j];
9              sim ← similitude(I, J);
10             si sim > max alors
11                 max ← sim;
12                 A ← I;
13                 B ← J;
14             fin
15         fin
16     fin
17     retirer(A, L);
18     retirer(B, L);
19     ajouter((A, B), L);
20 fin
21 R ← L[0];
```

N et M) $\{i_n; n \in [1, N]\}$ et $\{j_m; m \in [1, M]\}$:

$$sim(\{i_n\}, \{j_m\}) = \sum_{p=1}^{N} \sum_{q=1}^{M} \frac{sim(i_p, j_q)}{|\{i_n\}| \times |\{j_m\}|}$$

De plus, nous utilisons l'indice de similarité présenté dans [Assadi, 1998], ce qui nous donne la formule suivante pour calculer la similitude entre deux termes i et j :

$$sim(i, j) = \frac{\alpha}{2}\left(\frac{a}{a+b} + \frac{a}{a+c}\right) + \frac{1-\alpha}{2}\left(\frac{d}{d+c} + \frac{d}{d+b}\right)$$

Où, a, b, c et d sont respectivement le nombre de contextes partagés par i et j, présents uniquement chez i, présents uniquement chez j, et présents ni chez i ni chez j. Les contextes sont déterminés en explorant le réseau « Tête-Expansion » [Assadi, 1998]. Dans notre cas, les contextes d'un terme correspondent à l'ensemble des expansions des termes dont il est en tête. Par exemple, dans la figure 4.2, les contextes de « system » sont « stellar » et « binary ». Pour nos expérimentations, nous avons fixé α à 0,75. Tous ces choix conditionnent l'arbre résultat, mais n'influencent ni le déroulement général de l'algorithme ni sa complexité.

4.3.2 Un algorithme distribué de classification

Cette section présente l'algorithme distribué de classification utilisé dans le système. Nous détaillerons d'abord l'état initial au démarrage du système pour ensuite dérouler les trois étapes du procédé.

Le système est initialisé de la manière suivante :

- un agent *TOP* n'ayant aucun parent est créé, il sera la racine de la taxonomie résultante ;
- pour chaque terme donné en entrée, on crée un agent-concept dont le but est de se positionner dans la hiérarchie ; ainsi ces agents initiaux ont tous un terme du réseau comme référent ; enfin chacun de ces agents a *TOP* comme parent.

Initialement, nous créons donc autant d'agents-concepts que de termes en entrée plus un pour *TOP*. Une fois cette structure de base en place, l'algorithme se déroule en parallèle au sein de chaque agent, jusqu'à obtenir une position d'équilibre global, alors une première version de la taxonomie résultante est présentée à l'utilisateur. Par la suite, les modifications effectuées par l'utilisateur dans la taxonomie auront pour effet de réactiver les agents concernés, l'agorithme sera donc relancé localement.

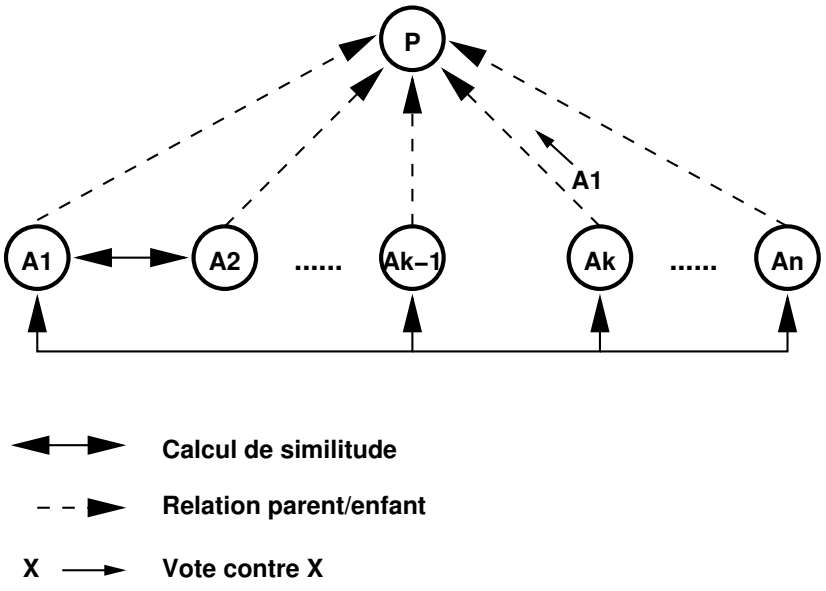


Figure 4.3 — Classification distribuée : étape 1

La première étape du procédé (figure 4.3) se déroule en parallèle dans les agents (ici, on s'intéressera uniquement à A_k) ayant plus d'un frère (puisque nous cherchons à obtenir un arbre binaire). L'agent A_k envoie alors un message à son père P donnant la liste de ses frères par ordre de dissimilarité décroissante (ici, on a représenté uniquement A_1 comme le plus dissimilaire). P reçoit donc un message du même type de chacun de ses fils. Par la suite, ce type de message sera appelé un « vote ».

Ensuite, lorsque P a reçu les messages de tous ses fils, il exécute la deuxième étape (figure 4.4). Grâce aux messages reçus indiquant les préférences de ses fils, P peut constituer trois sous-groupes parmi ses fils :

- le fils qui pose globalement le plus de problèmes à ses frères, c'est-à-dire le fils qui

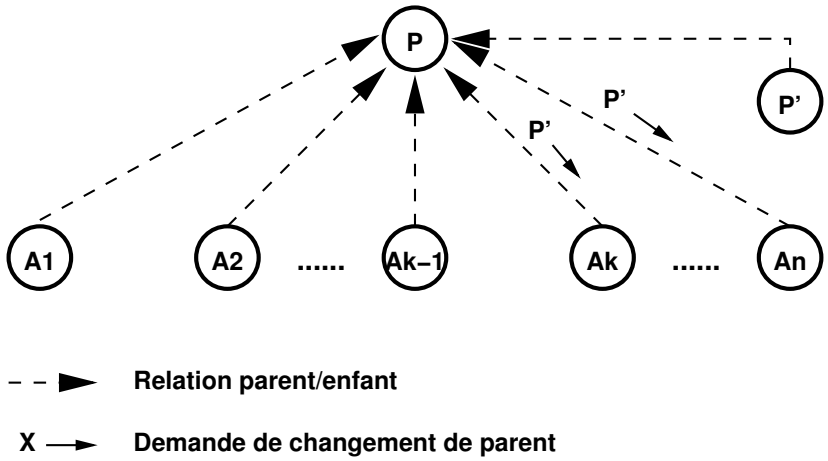


Figure 4.4 — Classification distribuée : étape 2

gagne « l'élection » lorsque les « votes » sont pris en compte avec une procédure de vote Condorcet [Condorcet, 1785]. Ainsi, doit être élu le candidat qui a gagné toutes les confrontations lors de l'expression des préférences. En cas de conflit circulaire, P cherche le candidat qui a gagné le plus de confrontations et s'il y a égalité, un des *ex aequo* est choisi au hasard (en pratique ce dernier cas n'a jamais été rencontré). Dans notre exemple, le gagnant est A_1 ;

- les fils ayant permis « l'élection » du premier groupe, c'est-à-dire les agents ayant choisi leur frère du premier groupe comme étant celui qui leur est le plus dissimilaire (ici A_k à A_n) ;
- les fils restants (ici A_2 à A_{k-1}).

P crée alors un nouvel agent-concept P' (de père P) et demande aux agents du second groupe (ici les agents A_k à A_n) d'en faire leur nouveau père.

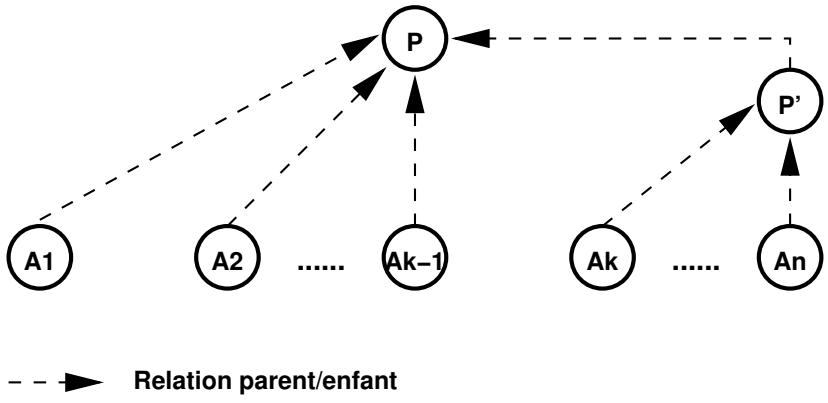


Figure 4.5 — Classification distribuée : étape 3

Enfin, l'étape 3 (figure 4.5) est assez évidente. Les fils repoussés par P (ici les agents A_k à A_n) tiennent compte de son message et choisissent P' comme nouveau père. La hiérarchie se constitue ainsi de proche en proche.

Remarquons que cet algorithme converge nécessairement, puisque le nombre de frères d'un agent va en diminuant. Lorsqu'un agent n'a plus qu'un seul frère, son activité est stop-

pée (hors traitement des messages de ses fils). À noter que cette propriété est dépendante du système de vote utilisé qui pourrait très bien ne pas permettre de déterminer un gagnant. Initialement, le gagnant était déterminé « à la majorité » et échouait dans environ trente pour cent des cas. Le passage au système Condorcet avec la procédure de résolution de conflits décrite plus haut a permis de s'affranchir de ce problème. L'expression des préférences totales par les agents a rendu l'algorithme nécessairement convergent. En utilisant un vote par adhésion simple, la quantité d'information n'est pas suffisante pour conclure dans tous les cas.

Il s'agit donc bien d'un algorithme distribué de classification hiérarchique, les décisions étant prises par des négociations au sein de groupes d'agents autonomes. Les traitements ainsi que les connaissances étant locales aux agents, et la communication se faisant par envoi de messages, un tel algorithme peut tout à fait être réparti sur un réseau de machines. De plus, la distribution à la fois du contrôle et des connaissances permet à l'ontologue d'intervenir dans le processus quand il le souhaite. Le système peut-être stoppé à tout instant pour permettre de modifier l'état courant. Dans un tel cas une partie des connaissances seront invalidées, mais le système continuera son travail. Par exemple, si l'utilisateur intervient entre l'étape 1 et l'étape 2 de l'algorithme pour déplacer un des A_k ailleurs dans la hiérarchie, lors de la reprise il est facile pour P d'ignorer le vote en provenance du fils qui est parti. Il peut tout à fait continuer la procédure de délibération pour constituer les groupes lors de l'étape 2.

4.3.3 Constat qualitatif

Dans cette section, nous nous intéressons aux traits qualitatifs du système déterminés par le comportement de nos agents. En l'analysant, nous nous attachons à faire ressortir les avantages obtenus grâce à l'utilisation d'une approche multi-agent.

Le principal avantage de l'utilisation d'un système multi-agent adaptatif pour une tâche de classification est le côté dynamique introduit par un tel système. L'ontologue peut intervenir pour effectuer des corrections et la hiérarchie s'adapte en fonction de cette demande. Cette caractéristique est particulièrement intéressante dans un contexte d'ingénierie des connaissances. En effet, la hiérarchisation fournie par le système est amenée à être modifiée par l'ontologue puisqu'elle est le résultat d'un traitement statistique. Lors de retours nécessaires au texte pour examiner les contextes d'usage des termes [Aussenac-Gilles et Condamines, 2004], l'ontologue pourra interpréter le contenu réel et ainsi réviser la proposition du système. Ceci est très difficile à réaliser avec une approche centralisée. Dans la plupart des cas, il faut trouver l'étape du raisonnement qui a engendré le résultat erroné et modifier la classe correspondante manuellement. Malheureusement, dans ce cas, toutes les étapes de raisonnement consécutives à la création de la classe modifiée sont perdues et doivent être recalculées en tenant compte de la modification. C'est pourquoi un système comme ASIUM [Faure, 2000] présente à l'utilisateur les classes créées à chaque étape du raisonnement, pour essayer de gommer ce problème grâce à une collaboration système-utilisateur. De plus, la hiérarchie complète n'est visible qu'à la fin du processus. Ainsi, l'utilisateur peut ne constater l'introduction d'une erreur que trop tard. Dans Dynamo, nous laissons l'algorithme se dérouler, bien que l'utilisateur ait l'opportunité de l'interrompre et de le relancer quand il

le souhaite. Il dispose à tout instant d'une hiérarchie complète qu'il peut modifier. Ses interventions déclencheront l'algorithme localement jusqu'à obtenir un nouvel état stable : c'est en cela que Dynamo fournit un véritable processus de construction d'ontologies.

4.3.4 Exemple

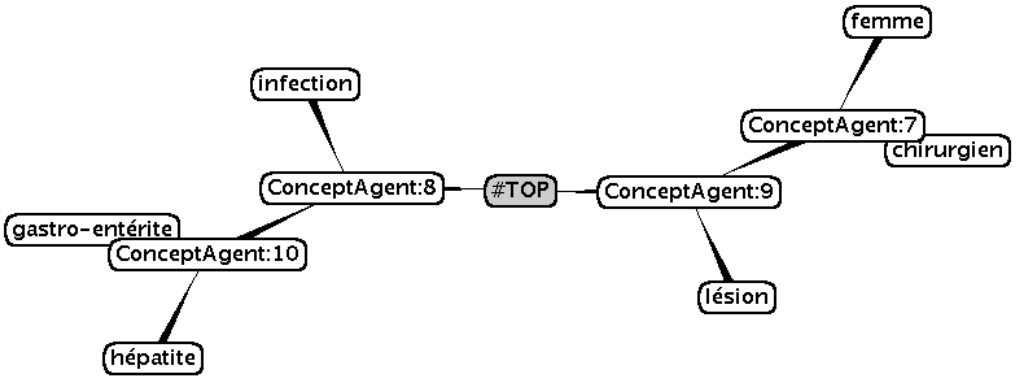


Figure 4.6 — Arbre d'agents concept après stabilisation autonome du système

Afin d'illustrer notre propos, nous présentons un exemple simple à l'aide de captures d'écran du système en fonctionnement sur le corpus REACHIR dont des extraits sont fournis en annexe A.1. En utilisant un jeu de tests et en laissant le système travailler de lui-même, on obtient l'arborescence de la figure 4.6 après stabilisation[3]. Il est clair que le concept représenté par le terme « lésion » est mal placé. Il se trouve que les calculs de similitudes le placent plus proche de « femme » et « chirurgien » que d'« infection », « gastro-entérite » et « hépatite ». Cette mauvaise position pour « lésion » s'explique alors par le fait que sans intervention de l'ontologue, le système effectue le raisonnement uniquement sur ce critère statistique.

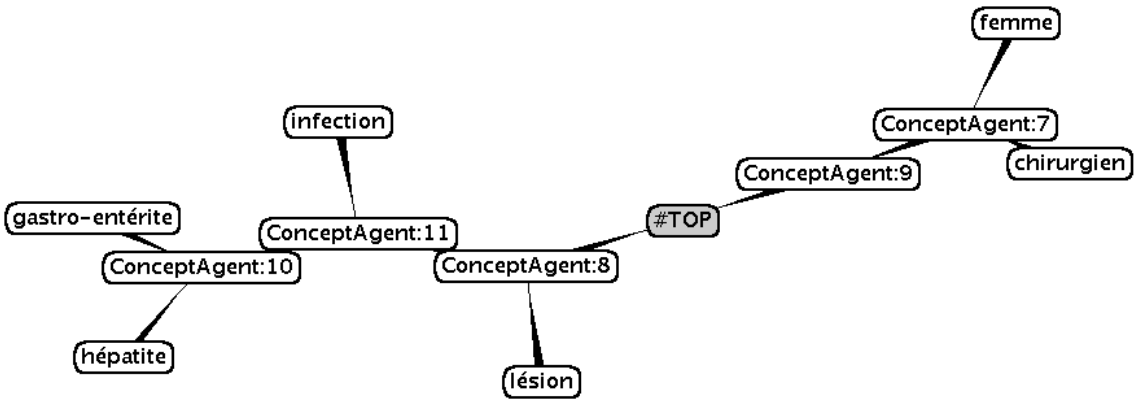


Figure 4.7 — Arbre d'agents concept après intervention de l'ontologue

L'utilisateur intervient alors pour le replacer dans la bonne branche, en lui affectant

[3]Dans la suite du chapitre, toutes les figures d'arbres sont obtenues par captures d'écrans du système en fonctionnement.

« ConceptAgent :8 »[4] comme nouveau père. Le système réagit ensuite de lui-même pour raffiner l'arbre de classification et revenir à un arbre binaire, créant donc « ConceptAgent :11 ». Le nouvel état stable est alors celui de la figure 4.7.

Ce couplage système-ontologue est indispensable pour une création d'ontologie. Pour le mettre en place, aucun ajustement particulier sur le principe de l'algorithme distribué n'a été nécessaire car chaque agent effectue un traitement local autonome et communique avec ses voisins par envoi de messages.

De plus, cet algorithme peut être *de facto* réparti sur un réseau. Il est tout à fait envisageable d'exécuter les agents sur des machines différentes. La communication entre les agents se fait par envoi de messages et chacun d'entre eux conserve son autonomie de décision. Ainsi, aucun ajustement de l'algorithme n'est nécessaire pour lui permettre de tourner sur plusieurs machines. En revanche, cela demanderait de revoir la couche de communication et la gestion de la création des agents de l'implémentation actuelle.

4.4 Hiérarchie multi-critère

Dans les sections précédentes, nous avons supposé que la similarité pouvait être calculée pour n'importe quelle paire de termes. Mais, dès que l'on utilise des données réelles, cette propriété n'est plus vérifiée. Certains termes n'ont de valeur de similarité avec aucun des autres termes extraits. De plus, pour les feuilles de la taxonomie, il est parfois intéressant d'utiliser d'autres moyens pour les positionner. En effet, pour cette structuration fine des feuilles de la taxonomie, les ontologues basent généralement leurs choix sur des heuristiques simples. Dans le réseau « Tête-Expansion », la relation tête est assez proche d'une relation générique/spécifique sur les termes. Les ontologues s'en servent alors comme indice pour positionner des concepts. Si les termes représentant deux concepts sont liés par une relation tête, alors les deux concepts peuvent être liés par une relation « est-un ». Par cette observation, nous avons conçu un nouvel ensemble de règles qui ne sont pas basées sur la similarité pour supporter cette structuration des feuilles de la taxonomie.

4.4.1 Ajouter les règles de « couverture en tête »

Dans ce cas, les agents peuvent agir avec un point de vue local simplement en examinant la relation parent/enfant. Chaque agent peut essayer de déterminer si son parent est adéquat. Il est possible de faire des hypothèses sur ce point car chaque agent concept est décrit par un ensemble de termes et grâce au réseau « Tête-Expansion ».

Dans la suite, T_X sera l'ensemble des termes décrivant un agent concept X, et $tete(T_X)$ l'ensemble de tous les termes qui sont en tête d'au moins un des éléments de T_X. Grâce à ces deux notations nous pouvons écrire la fonction d'adéquation d'un parent $a(P, E)$ entre un parent P et un enfant E :

$$a(P, E) = \frac{|T_P \cap tete(T_E)|}{|T_P \cup tete(T_E)|} \tag{4.1}$$

[4] « ConceptAgent :X » est le nom automatiquement donné à un agent concept qui n'est pas directement représenté par un terme.

Le meilleur parent pour E est l'agent P qui maximise $a(P, E)$. En effet, plus $a(P, E)$ est grand plus le recouvrement entre les termes de P (T_P) et les termes tête de E ($tete(T_E)$) est important. Or, le père d'un concept est adéquat lorsque l'ensemble de ses termes est un sur-ensemble de l'ensemble des termes têtes de son enfant, la part des termes non partagés devant être minimale. Alors, un agent insatisfait par son parent peut alors essayer d'en trouver un meilleur en évaluant l'adéquation avec un ensemble de candidats. Nous avons conçu un algorithme complémentaire pour piloter cette recherche :

Règle 1. *Quand un agent E est insatisfait de son père P, il évalue $a(F_i, E)$ avec tous ses frères (notés F_i). Celui maximisant $a(F_i, E)$ est alors choisi comme nouveau parent.*

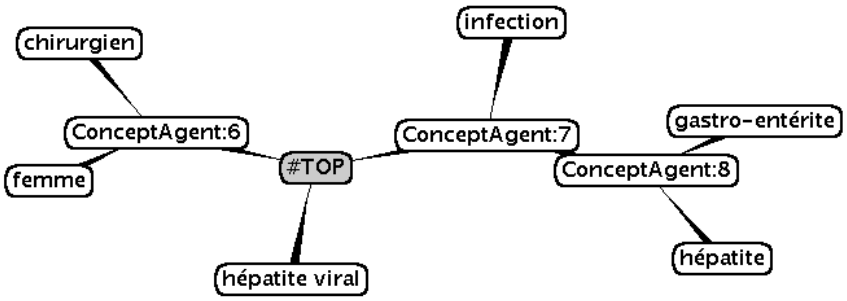


Figure 4.8 — Arbre d'agents concept après stabilisation autonome du système sans la règle de couverture en tête

Nous illustrons le comportement de cette règle avec un exemple. La figure 4.8 montre l'état du système après une stabilisation sur des données de test. Nous pouvons remarquer que « hépatite viral » est toujous lié à la racine de la taxonomie. Cela vient du fait qu'il n'y a aucune valeur de similarité entre le terme « hépatite viral » et tous les autres termes des agents concepts.

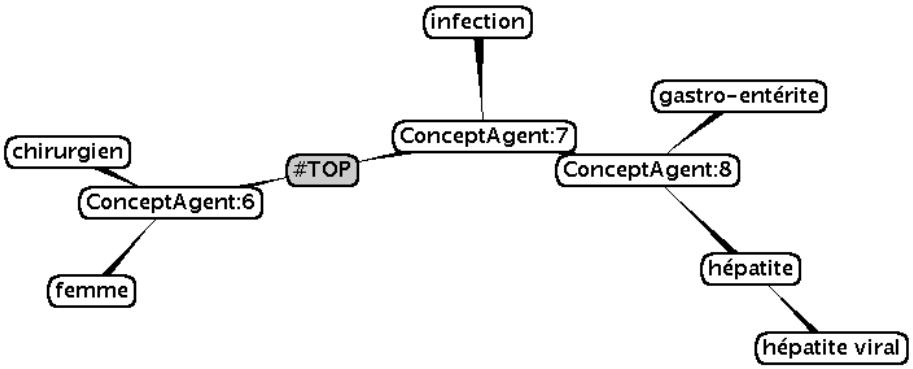


Figure 4.9 — Arbre d'agents concept après activation de la règle de couverture en tête

Après avoir activé la règle de couverture en tête et laissé le système se stabiliser à nouveau, nous obtenons la figure 4.9. Nous pouvons constater que « hépatite viral » a glissé le long de la branche conduisant à « hépatite » et l'a choisi comme son nouveau parent. Il s'agit d'un choix par défaut pertinent puisque « hépatite viral » est un terme plus spécifique que « hépatite ». Cette règle tend à pousser vers les feuilles de la taxonomie les agents décrits

par un ensemble de termes. Cela permet d'atteindre notre but d'améliorer la structure des feuilles de notre taxonomie.

4.4.2 De l'utilisation de plusieurs critères

Dans les sections et exemples précédents, nous avons utilisé uniquement un algorithme à la fois. L'algorithme distribué de classification tend à introduire de nouveaux niveaux dans la taxonomie, tandis que l'algorithme de couverture en tête tend à pousser certains des agents vers les feuilles de la taxonomie. Évidemment, cela pose la question des moyens à mettre en œuvre pour gérer ces critères multiples lors de la construction de taxonomie, et comment les agents sont censés déterminer leurs priorités à un moment donné.

La solution que nous avons choisie provient de notre souci de minimiser la non coopération dans le système en accord avec l'approche AMAS. Chaque agent calcule trois niveaux de non coopération et choisit sa priorité courante en fonction du niveau le plus élevé, qui sera utilisée lors de l'émission de messages. Ainsi, chaque message envoyé a une priorité p correspondant au niveau de non coopération de l'agent au moment de l'envoi. Pour un agent A donné de parent P, ayant un ensemble de frères F_i et qui a reçu un ensemble de messages M_k de priorité p_k, les trois niveaux de non coopération sont :

- $\mu_T(A) = 1 - a(P, A)$, est le niveau de non coopération de « couverture en tête », déterminé par la couverture en tête du parent de l'agent,
- $\mu_F(A) = max(1 - sim(A, F_i))$, est le niveau de non coopération de la « fratrie », déterminé par le pire frère de A vis-à-vis de la similarité,
- $\mu_M(A) = max(p_k)$, est le niveau de non coopération par « message », déterminé par le plus critique des messages reçus.

Alors, le niveau de non coopération $\mu(A)$ de l'agent A est :

$$\mu(A) = max(\mu_T(A), \mu_F(A), \mu_M(A)) \tag{4.2}$$

L'utilisation du max nous permet de faire ressortir le type de problème le plus critique conformément à la théorie des AMAS. Donc, nous avons trois cas déterminant quel type d'action A va choisir :

- si $\mu(A) = \mu_T(A)$ alors A va activer l'algorithme de couverture en tête que nous avons détaillé dans la section précédente ;
- si $\mu(A) = \mu_F(A)$ alors A va activer l'algorithme distribué de classification (voir section 4.3) ;
- si $\mu(A) = \mu_M(A)$ alors A va traiter le message M_k immédiatement afin d'aider son émetteur.

Ces trois cas résument l'activité de nos agents : ils doivent trouver le meilleur parent pour eux ($\mu(A) = \mu_T(A)$), améliorer la structure grâce à la classification ($\mu(A) = \mu_F(A)$) et traiter les messages provenant des autres agents ($\mu(A) = \mu_M(A)$) pour les aider à satisfaire leurs buts. Ainsi, les agents recherchent bien un équilibre entre altruisme et égoïsme [Picard et Glize, 2006].

4.5 Simplification de l'arborescence

Dans les sections précédentes, nous avons vu l'algorithme distribué de classification implémenté dans Dynamo, ainsi que des règles complémentaires pour obtenir un arbre multi-critère. Toutefois, comme nous restons proches de l'algorithme de classification, les résultats ne sont pas nécessairement optimaux sur le plan structurel pour une taxonomie. Le système nécessite donc des aménagements complémentaires que nous allons présenter dans la suite.

4.5.1 Elimination des branches dégradées

Après des modifications effectuées par l'ontologue, des concepts introduits initialement pour la structuration de l'arborescence restent et créent donc des branches *dégradées* (dont certains nœuds n'ont qu'un seul fils). Ce phénomène est, par exemple, observable sur la figure 4.7 de l'exemple donné précédemment. Bien entendu, il faut éliminer de telles branches car elles n'apportent aucune information conceptuelle supplémentaire. Pour cela, il suffit de s'intéresser à l'inutilité des agents présents dans ces branches.

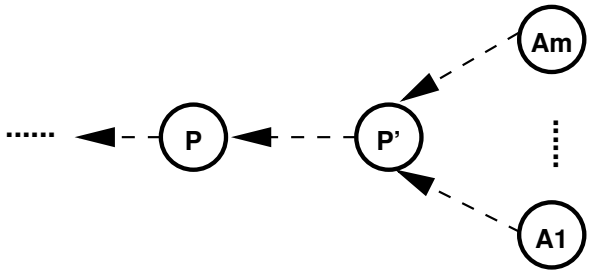


Figure 4.10 — Branche dégradée

Dans le cas d'une branche dégradée, on obtient une situation similaire à la figure 4.10. Par exemple, les frères de P' ont été déplacés à un autre endroit de la hiérarchie. Dans ce cas, P' n'apporte rien à la hiérarchie. Il a été initialement introduit pour améliorer la structuration mais, à cause des changements de condition, il est devenu superflu : ses fils pourraient aussi bien être sous P pour une structuration équivalente mais plus simple. Ce constat nous donne la première règle (règle 2) permettant l'élimination de branches dégradées :

Règle 2. *Quand un agent P' a des fils (notés A_k), mais aucun frère, alors P' propose à ses fils d'avoir son parent P comme nouveau parent.*

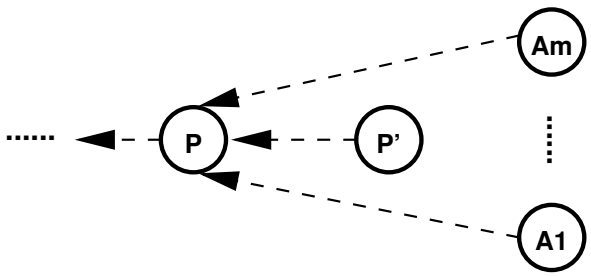


Figure 4.11 — Branche simplifiée

L'application de cette règle nous amène à la situation de la figure 4.11. La branche étudiée n'est effectivement plus dégradée, mais P' est encore là. Comme il s'agissait d'un concept de structuration, qui n'était par représenté directement par des termes, et qu'il n'a plus de fils, il devient tout à fait inutile à l'échelle du système. Il doit donc disparaître, ce qui nous donne la deuxième règle (règle 3) permettant l'élimination de branches dégradées, qui est la règle de détection de l'inutilité d'un agent :

Règle 3. *Quand un agent P' n'a aucun fils et n'est représenté par aucun terme, il doit se retirer du système.*

En appliquant cette deuxième règle, on a effectivement simplifié cette zone de la hiérarchie pour éliminer la branche dégradée.

4.5.2 S'éloigner des arbres binaires

Maintenant que notre système est capable de nettoyer les branches dégradées, il nous reste un dernier point à régler pour obtenir une solution générant des taxonomies. Avec ce que nous avons vu jusqu'à présent, nous obtenons nécessairement des arbres binaires, sauf pour le tout dernier niveau de la hiérarchie, si la règle de couverture en tête a joué. Or, notre finalité n'est pas de proposer un algorithme distribué de classification hiérarchique, auquel cas celui présenté suffirait, mais d'obtenir une taxonomie dynamique. Comme cette dernière ne sera que rarement un arbre binaire, il faut donc ajouter un critère supplémentaire permettant d'effectuer des regroupements et ainsi obtenir des nœuds n-aires.

Comme nous l'avons vu à la section 4.3, la forme de la hiérarchie est influencée par les regroupements effectués. Il nous paraît donc possible de s'éloigner des arbres binaires en modifiant la stratégie de vote des agents. En effet, c'est la façon dont a voté un agent qui conditionne dans quelle partition il se trouve et si son parent est capable de faire des partitions parmi ses agents fils. Pour rappel, si nous reprenons la dernière étape de l'algorithme, une fois que les groupes ont été constitués, nous obtenons la figure 4.12.

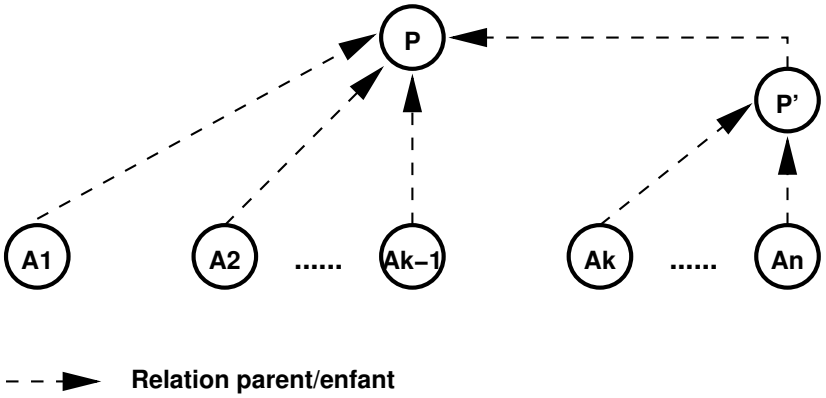


Figure 4.12 — Classification, partitions constituées

Par construction, le groupe sous l'agent P' est constitué des agents les plus dissimilaires du groupe situé sous P et qui sont le moins dissimilaires entre eux. Donc, si on note $sim(A, B)$ la mesure de similitude entre deux agents A et B, les agents doivent vérifier la

propriété suivante dans toute la hiérarchie :

$$\forall i \in [k;n], \forall j \in [k;n] \setminus \{i\}, \forall l \in [1;k-1] : sim(A_i, A_j) > sim(A_i, A_l) \tag{4.3}$$

Le profil binaire de la hiérarchie provient du fait que les agents cherchent à être éloignés dans des groupes différents dès qu'il y a une dissimilarité. Un moyen simple d'éviter cette découpe binaire est donc d'introduire une tolérance ε. Ainsi la propriété à vérifier devient :

$$\forall i \in [k;n], \forall j \in [k;n] \setminus \{i\}, \forall l \in [1;k-1] : sim(A_i, A_j) > sim(A_i, A_l) + \varepsilon \tag{4.4}$$

Pour cela, nous avons réajusté la stratégie de vote des agents. Ils prennent toujours part à un vote Condorcet. Toutefois au lieu d'exprimer un classement total de leurs préférences, ils expriment un classement partiel tenant compte de la tolérance. Ainsi, un agent A, après avoir classé ses frères F_k par ordre décroissant de dissimilarité, ne conservera dans son vote que les F_k pour lesquels $1 - sim(A, F_k) > \varepsilon$.

À présent, nous disposons donc d'un outil qui influence le facteur de branchement de la taxonomie. Il peut être utilisé de deux façons différentes, soit en fixant un ε global au système, soit en fixant un ε_A pour chaque agent A du système. Ce choix a des implications sur le type de résultat obtenu :

- soit la tolérance est fixée globalement ; auquel cas tous les agents l'utilisent, ainsi, bien que l'arbre soit n-aire, le facteur de branchement pourra varier fortement suivant la zone de la hiérarchie où l'on se trouve (certains groupes pouvant présenter moins de dissimilarités que d'autres) ;
- soit la tolérance est ajustée localement ; dans ce cas, chaque agent manipule deux valeurs de tolérances différentes : la première provient de son parent, utilisée lors des votes (par exemple, pour voter P' utilise ε_P), et la seconde est sa tolérance propre qui lui permet d'influencer le facteur de branchement.

Nous avons retenu cette dernière option. En effet, il paraît plus parlant du point de vue de l'ontologue de donner au système un intervalle pour le facteur de branchement global et de voir les agents réajuster les tolérances locales pour essayer de s'y conformer en fonction des dissimilarités avec leur voisinage. Cela implique que la tolérance n'est plus une constante pour les agents. Nous avons ainsi dégagé une nouvelle règle permettant de prendre en compte des variations de tolérance :

Règle 4. *Quand un agent P' a un de ses fils (notés A_k) qui ne respecte plus la propriété sur la tolérance $\varepsilon_{P'}$, alors P' propose à ses fils d'avoir son parent P comme nouveau parent.*

Lorsqu'elle s'applique, cette règle permet à P d'effectuer une nouvelle répartition. À noter qu'il n'y a pas de risque d'oscillation[5] puisque lors de la création du nouveau niveau intermédiaire, le remplaçant de P' aura initialement $\varepsilon_{P'} = \varepsilon_P$.

À présent, nous sommes capables de prendre en compte la tolérance durant la classification et de la faire varier localement. Il nous reste à la lier au facteur de branchement. Nous n'avons pas réussi à dégager une fonction permettant de corréler la tolérance locale

[5] P crée P' qui disparait à cause d'un seuil de tolérance plus faible que celui appliqué par P, puis le processus recommence.

et le facteur de branchement local dans notre système. Ce problème a été toutefois résolu par l'introduction d'une dernière règle poussant chacun des agents à rechercher la tolérance optimale pour obtenir un facteur de branchement dans l'intervalle spécifié par l'utilisateur :

Règle 5. *Quand un agent P' a un nombre de fils trop élevé (respectivement, trop faible), il abaisse (respectivement, augmente) sa tolérance $\varepsilon_{P'}$.*

Nous remarquerons que cette nouvelle valeur de $\varepsilon_{P'}$ sera utilisée par P' pour vérifier si ses fils sont conformes à la propriété sur la tolérance, et par les fils de P' lors des prochains votes.

4.6 Bilan

Dans ce chapitre, nous avons successivement présenté la structure globale de Dynamo, la structure interne de nos agents, ainsi que l'ensemble des règles utilisées par les agents afin de mener à bien la tâche de construction d'ontologies. Cette structuration en règles s'est initialement basée sur des considérations de classification puis a ensuite été complétée pour tenir compte de critères de coopération supplémentaires. Ainsi notre corpus de règles est complet du point de vue des modifications organisationnelles possibles :

- les règles de l'algorithme de classification forcent les agents à descendre dans la hiérarchie et à se différencier ;
- les règles de couverture en tête poussent les agents concernés à descendre ;
- les règles de tolérance sur la similitude permettant d'obtenir des arbres n-aires obligent les agents à remonter et à simplifier la structure par aggrégation ;
- les règles de simplification de branche et d'inutilité poussent les agents à remonter.

À présent, il nous faut évaluer les performances de Dynamo ainsi que son utilisation ; il s'agit de l'objet du chapitre suivant.

Troisième partie

Expérimentations et Analyses

5 Evaluation du système

« [Il] ajouta lentement et distinctement : Il y a une chance sur un million, mais ça peut marcher quand même ! »

Terry Pratchett – Au guet !

LE chapitre précédent a décrit le système Dynamo. Nous y avons tout d'abord donné un point de vue macroscopique indiquant ses grandes composantes, puis elles ont été détaillées progressivement en expliquant la structure interne des agents qui le compose, à l'aide des travaux présentés au chapitre 3, pour ensuite rendre compte de l'ensemble des règles décrivant le comportement des agents.

Ce chapitre a pour objet l'évaluation de Dynamo. Il convient de bien caractériser ses forces et ses faiblesses. Ainsi, nous allons adopter la démarche inverse du chapitre 4. Nous allons dans un premier temps nous intéresser à analyser son comportement dans le détail pour dans un deuxième temps évaluer son comportement macroscopique.

Ce chapitre est structuré autour de deux grandes sections. La section 5.1 présente les résultats des expérimentations conduites afin d'évaluer la complexité du système avec des données réelles (la qualité du résultat en sortie n'y est pas abordée). Cette section est structurée conformément aux sections 4.3, 4.4 et 4.5 du chapitre précédent. En revanche, la section 5.2 propose une vision globale de l'utilisation du système dans une application ayant pour but la construction d'une taxonomie complète. Cette expérience permet d'évaluer la capacité du système à produire une taxonomie en coopération avec l'utilisateur.

5.1 Performances du système

Dans chaque sous-section, nous abordons l'évaluation des performances d'une partie des règles du système. Comme nous l'avons vu au chapitre précédent, nous disposons de plusieurs sous-ensembles de règles qui se complètent. C'est la raison pour laquelle nous menons initialement notre réflexion avec le cœur des règles constituant notre algorithme distribué de classification pour le compléter au fur et à mesure avec des dispositifs supplémentaires et évaluer leur impact en performances [Ottens et Aussenac-Gilles, 2007; Ottens *et al.*, 2007].

Nous évaluons les performances de chaque ensemble de règles selon plusieurs points de vue lorsque cela est possible. Pour les règles permettant de tels calculs, nous avons déterminé la complexité théorique de l'algorithme résultant. De plus, nous avons conduit un ensemble de mesures expérimentales chaque fois que cela était possible. Dans le cadre de ces expérimentations, nous avons utilisé des jeux de données de différentes tailles issues d'un corpus de comptes-rendus de réanimation chirurgicale (corpus REACHIR, dont des extraits sont disponibles en annexe à la section A.1).

5.1.1 Complexité de l'algorithme distribué de classification

Pour évaluer les propriétés de notre algorithme distribué, il convient de commencer par une évaluation quantitative, basée sur sa complexité, tout en le comparant à l'algorithme 4.1 du chapitre précédent.

La complexité théorique sera calculée pour le pire cas, en considérant l'opération de calcul de la similitude entre deux éléments comme élémentaire. Pour l'algorithme distribué, le pire cas signifie qu'à chaque étape, on ne peut constituer qu'un seul groupe de deux individus. Dans ces conditions, pour un jeu de données initial de n éléments, nous pouvons déterminer le nombre de calculs de similitude effectués par chacun des algorithmes.

Pour l'algorithme 4.1, nous notons $l = longueur(L)$; la boucle « pour » la plus imbriquée est exécutée $l - i$ fois. Son corps contient le seul calcul de similarité, son coût est donc $l - i$. La deuxième boucle « pour » est exécutée $l + 1$ fois pour i allant de 0 à l. Alors son coût est $\sum_{i=0}^{l}(l - i)$ qui peut être simplifié en $\frac{l \times (l-1)}{2}$. Finalement pour chaque exécution de la boucle « tant que », l est décrémenté de n à 1, soit $t_1(n)$ comme quantité de calculs de similitude pour l'algorithme 4.1 :

$$t_1(n) = \sum_{l=1}^{n} \frac{l \times (l-1)}{2} \tag{5.1}$$

Durant une exécution de l'algorithme distribué, chacun des l agents évalue sa similarité avec ses $l - 1$ frères. Donc, chaque exécution a un coût de $l \times (l - 1)$. Alors, les groupes sont créés et un autre vote a lieu avec l décrémenté de 1 (puisque nous supposons le cas pire, seuls des groupes de taille 2 ou $l - 1$ sont construits). Puisque l est égal à n lors de la première exécution, nous obtenons $t_{dist}(n)$ comme quantité de calculs de similitudes pour l'algorithme distribué :

$$t_{dist}(n) = \sum_{l=1}^{n} l \times (l-1) \tag{5.2}$$

Les deux algorithmes ont donc une complexité en $O(n^3)$. Dans le pire cas, l'algorithme distribué effectue deux fois plus d'opérations élémentaires que l'algorithme centralisé. Cette différence de facteur 2 provient de la localité des prises de décision au sein des agents. Ainsi, les calculs de similitude sont faits deux fois pour chaque paire d'agents. On pourrait envisager qu'un agent, après avoir effectué un tel calcul, envoie son résultat à l'autre partie. Toutefois, cela reviendrait simplement à déplacer le problème en générant plus de communications au sein du système.

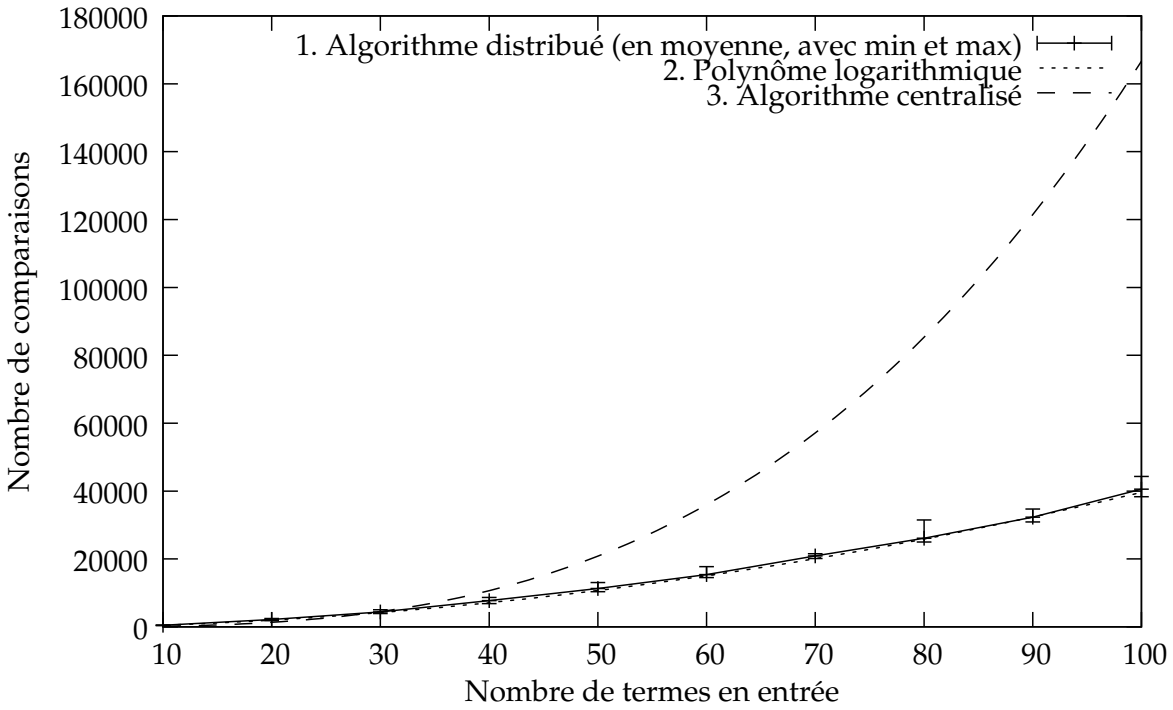


Figure 5.1 — Résultats expérimentaux de l'algorithme distribué de classification

La complexité en moyenne de l'algorithme a été déterminée expérimentalement. Pour cela, le système multi-agent a été exécuté avec des jeux de données en entrée variant de dix à cent termes. La valeur retenue est la moyenne du nombre de comparaisons effectuées pour une centaine d'exécutions sans intervention de l'utilisateur. Ceci donne les courbes de la figure 5.1. L'algorithme distribué est donc en moyenne plus efficace que l'algorithme centralisé, et sa complexité en moyenne plus réduite que dans le pire cas. Cela s'explique simplement par la faible probabilité qu'un jeu de données pousse le système à ne constituer que des groupes minimaux (deux éléments) ou maximaux ($n-1$ éléments) à chaque étape du raisonnement. La courbe 2 représente le polynôme logarithmique minimisant l'erreur avec la courbe 1. Le terme de plus fort degré de ce polynôme est en $n^2log(n)$, donc notre algorithme distribué a en moyenne une complexité en $O(n^2log(n))$. Enfin, nous remarquons l'écart réduit entre les performances en moyenne, au maximum et au minimum. Dans le pire cas, pour 100 termes, l'écart type est de 1960,75 pour une moyenne sur 100 exécutions de 40550,70 (soit environ 5%) ce qui souligne la stabilité du système.

5.1.2 Impact des règles de « couverture en tête »

Nous avons évalué la complexité expérimentale pour le système multi-agent lorsque les règles de « couverture en tête » sont activées en plus de l'algorithme distribué de classification. Dans ce cas, la métrique utilisée est le nombre de messages échangés dans le système. Encore une fois, le système a été lancé sur des ensembles de données en entrée d'une taille variant de dix à cent termes. La valeur donnée est le nombre moyen de messages envoyés dans l'ensemble du système pour une centaine d'exécutions sans interaction utilisateur. Cela nous donne les courbes de la figure 5.2.

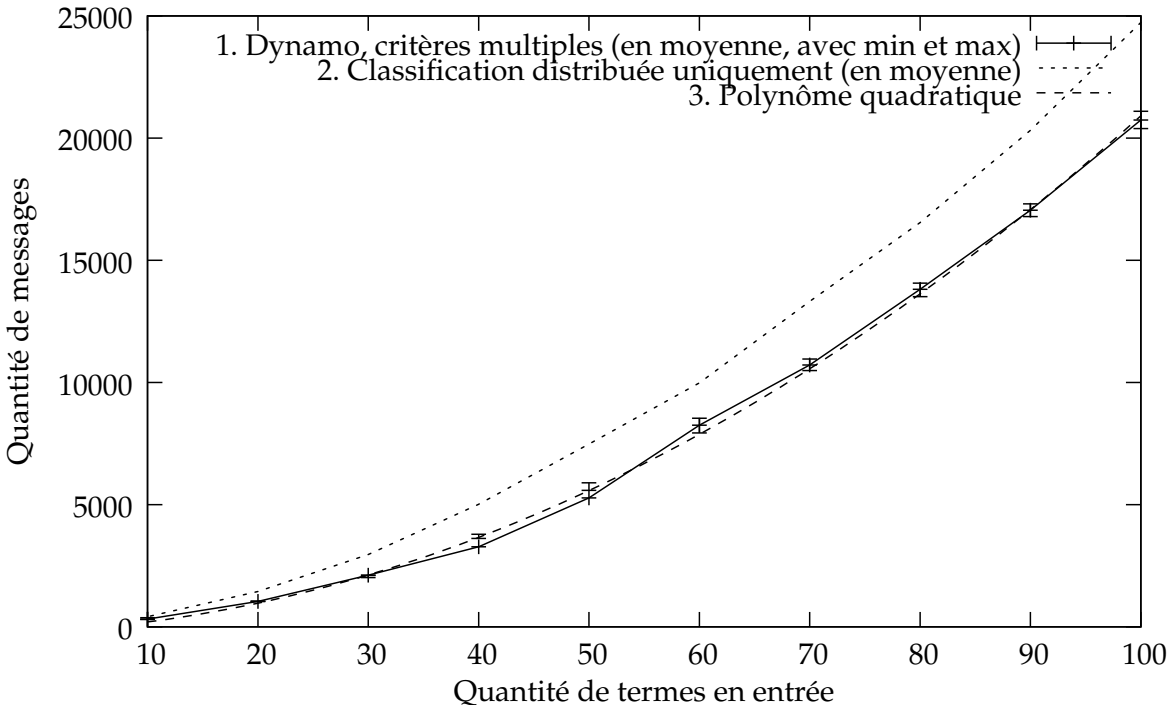


Figure 5.2 — Résultats expérimentaux avec les règles de « couverture en tête »

La courbe 1 représente la moyenne des valeurs obtenues. La courbe 2 représente la moyenne des valeurs obtenues lorsque seul l'algorithme distribué de classification est activé, mais pas l'ensemble des règles. La courbe 3 représente le polynôme minimisant l'erreur avec la courbe 1. Le plus haut degré de ce polynôme est en n^2, donc notre système multi-agent a une complexité de $O(n^2)$ en moyenne. Notons aussi la très faible variance des performances en moyenne avec le maximum et le minimum. Dans le pire cas pour 100 termes, la variance est de 126,73 pour une moyenne de 20737,03 (environ 0,6%) ce qui prouve l'excellente stabilité du système.

Finalement, les règles supplémentaires de couverture en tête sont une réelle amélioration sur l'algorithme distribué de classification seul. Elles ajoutent de nouvelles contraintes et la stabilité est obtenue avec moins d'interactions et de prises de décisions par les agents. Cela signifie que moins de messages sont échangés dans le système tout en obtenant un arbre tenant compte de critères supplémentaires pour sa construction.

5.1.3 Impact de l'élimination des branches dégradées

Pour évaluer la complexité de l'opération d'élimination des branches dégradées, une approche expérimentale est peu pertinente au vu de la simplicité de l'opération. De plus, il s'agit d'un traitement qui ne se déclenche que suite à une intervention manuelle de l'utilisateur, il est donc difficile de mettre en place un protocole expérimental générant suffisamment de données pour être exploitable. C'est pourquoi nous évaluons l'impact de ce traitement uniquement d'un point de vue théorique.

La complexité théorique est calculée en considérant comme élémentaire l'opération d'en-

voi d'un message entre deux agents. Nous pouvons déterminer le nombre de messages envoyés entre les agents concernés lors de l'élimination d'une branche dégradée. Pour visualiser cette opération, il suffit de reprendr les figures 5.3 et 5.4 du chapitre prédécent.

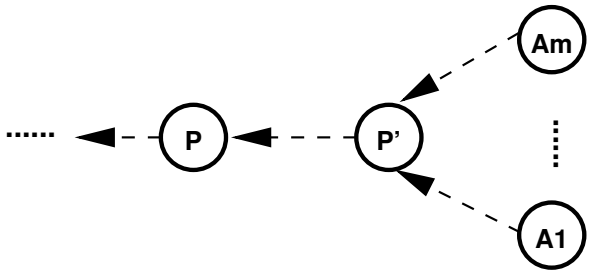


Figure 5.3 — Branche dégradée (bis)

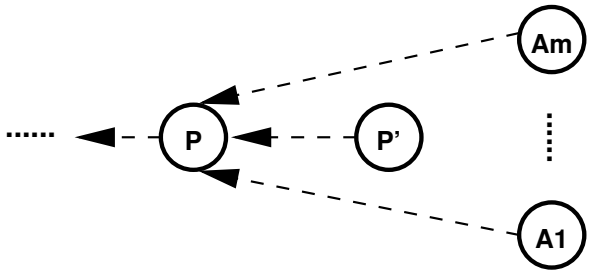


Figure 5.4 — Branche simplifiée (bis)

L'agent P' a m fils notés A_k. Pour effectuer la simplification, P' envoie un message à chacun de ses fils pour proposer P comme nouveau père, c'est-à-dire m messages. Ensuite, chacun des A_k envoie un message à P pour indiquer le changement de parent, soit à nouveau m messages. Enfin, P envoie un message de confirmation à chacun des A_k, encore une fois m messages.

Au final, le coût de l'opération est donc $3 \times m$ avec m le nombre de fils de l'agent P'. Ainsi, lors de la production d'un arbre binaire issu de notre algorithme décentralisé de classification, m est toujours égal à 2, ce qui nous donne une opération de complexité $O(1)$ à l'échelle du système. Enfin, pour une production d'arbre n-aires telles que le permettent les règles que nous évaluons dans la sous-section suivante, m peut varier mais reste borné, ne pouvant dépasser une valeur m_{max}. Dans ce cas, le coût de l'opération est au pire $3 \times m_{max}$, nous retrouvons alors une opération de complexité $O(1)$ à l'échelle du système.

5.1.4 Impact du passage aux arbres n-aires

Nous allons enfin évaluer la complexité expérimentale pour le système multi-agent complet lorsque toutes les règles sont activées. Comme dans les deux sections précédentes, la métrique utilisée est le nombre de messages échangés dans le système. Encore une fois le système a été lancé sur des ensembles de données en entrée d'une taille variant de dix à cent termes. Toutefois, le protocole expérimental est sensiblement différent. Dans les expérimentations des sections précédentes, nous avons effectué nos mesures entre le lancement du système et sa stabilisation. Ici, il s'agissait de mesurer le coût du passage d'un arbre binaire

à un arbre n-aire. La meilleure approche était alors de laisser le système se stabiliser une première fois puis de déclencher les règles permettant d'obtenir un arbre n-aire. Pour cela, après la première stabilisation, les paramètres ont été changés pour que le système cherche une structure dont le facteur de branchement de chaque nœud est de trois ou quatre. La valeur donnée est alors le nombre moyen de messages envoyés dans l'ensemble du système entre la première et la seconde stabilisation pour une centaine d'exécutions sans intervention de l'utilisateur. Cela nous donne les courbes de la figure 5.5.

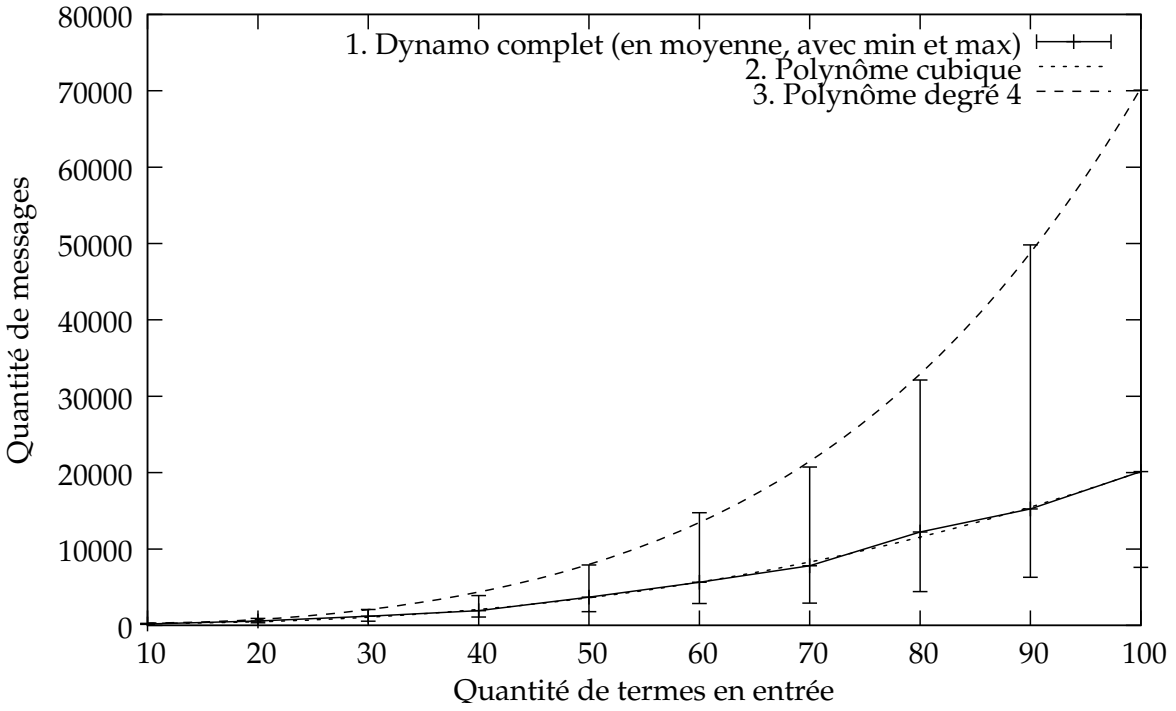


Figure 5.5 — Résultats expérimentaux du passage d'un arbre binaire à une taxonomie

La courbe 1 représente la moyenne des valeurs obtenues. La courbe 2 représente le polynôme minimisant l'erreur avec la courbe 1. Le plus haut degré de ce polynôme est en n^3, donc notre système multi-agent a une complexité de $O(n^3)$ en moyenne. Remarquons toutefois la forte variance des performances en moyenne avec le maximum et le minimum. Afin de situer la complexité maximale, nous avons ajouté la courbe 3 qui représente le polynôme minimisant l'erreur avec le maximum. Le plus haut degré de ce polynôme est en n^4, donc notre système multi-agent a une complexité de $O(n^4)$ pour le pire cas.

Dans la section 5.1.2, concernant les règles de couverture en tête, nous avons observé que l'apport de contraintes supplémentaires a permis au système de converger plus vite. Il est intéressant de constater qu'ici, nous observons le phénomène inverse. Le passage d'arbres binaires à des arbres n-aires présente une complexité élevée malgré les contraintes introduites. Selon nous, ce comportement est justifié par le fait que les règles introduites ne constituent pas des contraintes supplémentaires sur l'espace des solutions. Au contraire, le passage aux arbres n-aires implique que nous élargissions les solutions possibles. Or, cet espace élargi contient un nombre important de solutions structurellement correctes. On ne peut les dis-

criminer que par une approche interprétative, ce qui requiert de réintroduire l'utilisateur dans la boucle.

Ce dernier ensemble de règles permet d'atteindre les limites des évaluations possibles dans un déroulement entièrement automatique du système sans intervention extérieure. Il est donc important de se recentrer sur les interactions avec l'utilisateur, et évaluer le comportement de notre système dans ce cadre. La suite du présent chapitre est principalement focalisée sur cet aspect.

5.2 Application

Comme nous l'avons vu dans la première partie de ce chapitre, le système Dynamo ne peut simplement s'évaluer selon le seul critère de la complexité. Les limites du champ expérimental ne peuvent pas tenir compte de l'apport de l'utilisateur dans l'approche Dynamo. C'est pourquoi nous allons à présent nous concentrer sur une utilisation de notre système dans des conditions similaires à une utilisation réelle. Nous désirons faire ressortir de cette évaluation la qualité du modèle produit, et l'effort fourni pour obtenir la taxonomie.

5.2.1 Protocole expérimental

Dans la suite de ce chapitre, nous utilisons un corpus nommé *Astro*. Il s'agit d'une collection de résumés d'articles scientifiques publiés dans la revue « Astronomy & Astrophysics », dont un extrait est fourni en annexe dans la section A.2. Nous avons choisi ce corpus car il est accompagné d'une ontologie de référence construite à partir de résultats de Syntex. Ainsi, l'approche de notre nouvelle évaluation est d'isoler une sous-partie de l'ontologie de référence (cette sous-partie est disponible dans la section B.1) puis d'utiliser notre système pour construire une nouvelle ontologie qui pourrait être jugée équivalente. Lors de ces manipulations, nous mesurons le temps total passé par l'utilisateur et tenons compte du nombre d'interventions nécessaires pour obtenir le résultat final. Il reste alors à déterminer la qualité du modèle.

Toutefois, ce type d'évaluation est difficile. En particulier, notre ontologie référence n'a pas de portée universelle, il est donc difficile voire dangereux de caractériser le système uniquement en cherchant à obtenir exactement une réplique de l'ontologie référence. *In fine*, la qualité du résultat sera nécessairement subjective, il convient de garder ce point en mémoire et de considérer l'ontologie de référence comme un point de comparaison, et non comme le but à atteindre. À cela, il faut ajouter que le simple fait de comparer deux ontologies demande de disposer d'une métrique adaptée.

Pour tenter de faire l'évaluation en utilisation la plus claire possible, nous avons mis en place le protocole expérimental suivi dans ce chapitre. Il se déroule en deux phases. Premièrement, le système est démarré en utilisant en entrée les termes associés aux concepts présents dans l'ontologie de référence. Notre but étant d'examiner les mécanismes de structuration et le comportement du système, nous nous garantissons ainsi que le vocabulaire est le même dans les deux ontologies. Le système est ensuite lancé en mode totalement automatique sans intervention utilisateur. Cette phase permet de prendre connaissance du compor-

tement du corpus sur un plan purement statistique, et de déterminer les paramètres initiaux. En fonction de ces paramètres, le brouillon obtenu est différent, il revient alors à l'utilisateur de déterminer le résultat qui lui convient le mieux, et ainsi de figer ces paramètres initiaux. Il s'agit bien entendu de la phase la plus subjective. Deuxièmement, l'utilisateur intervient sur la structure pour la faire converger vers le résultat qu'il veut obtenir, en tenant compte du comportement du système provenant des statistiques calculées sur le texte.

Il reste alors à permettre la comparaison entre les deux ontologies. Pour cela, nous avons examiné différentes métriques pour déterminer laquelle serait la plus adaptée pour notre tâche. [Hernandez, 2005] présente plusieurs métriques permettant de vérifier l'adéquation d'une ontologie à un corpus, ou de comparer deux ontologies. Toutefois, dans les deux cas, ces métriques s'intéressent essentiellement au lexique disponible dans l'ontologie et assez peu à la structure. Or, dans notre cas, la structure nous importe plus que le lexique. En effet, d'une part c'est l'expérimentateur qui choisit un certain nombre de termes en entrée parmi ceux extraits du corpus. D'autre part, les termes de l'ontologie de référence ont aussi été tirés des textes. Nous avons trouvé une mesure s'intéressant à comparer la structure de deux taxonomies dans [Maedche et Staab, 2002] (présentée en annexe C). Nous utilisons cette mesure de *taxonomy overlapping* pour comparer l'état courant du système avec l'ontologie de référence. Il s'agit d'une mesure allant de 0 à 1, 1 étant l'égalité entre les deux taxonomies.

5.2.2 Exécution automatique

Pour cette première phase, nous avons testé plusieurs réglages pour la mesure de similitude utilisée par le système. Tout d'abord, nous avons testé la mesure Jaccard qui a donné un résultat peu intéressant sur ce corpus, les notions étant difficilement localisables dans la hiérarchie. Nous avons ensuite utilisé à nouveau la mesure proposée par [Assadi, 1998] (voir 4.3) avec différents réglages du paramètre α. Le meilleur résultat, présentant à notre avis le moins de dispersion, a été obtenu avec un α de 0,75. Nous avons donc retenu cette dernière mesure pour paramétrer notre système.

Il nous restait alors à choisir la fourchette indicative du facteur de branchement, et celle de la tolérance sur la similarité, utilisées par les règles des agents présentées en 4.5. Comme nous désirions obtenir un produit comparable à notre ontologie de référence, cette dernière nous a guidée pour faire ce choix. En effet, chaque niveau de l'ontologie référence est assez large – un grand nombre de concepts frères par niveau. Nous avons donc choisi le facteur de branchement variant entre 2 et 7 pour le facteur de branchement, la fourchette de tolérance allant de 0,1 à 0,4. Avec 90 termes en entrée, cela reste en-deçà du facteur de branchement moyen de l'ontologie référence. Nous cherchons ici à pousser le système à proposer une structure un peu plus profonde.

Lors de ces mises au point, nous avons effectivement pu obtenir des informations complémentaires sur le corpus. Cette première phase d'approche nous paraît importante pour la suite. Dans le cas du corpus *Astro*, sa forme synthétique mais bien rédigée, donne d'assez bons résultats avec la classification, mais elle est encore plus réceptive à la règle de couverture en tête, ce qui dénote un usage important de l'ellipse. Ce phénomène est probablement dû aux contraintes de places inhérentes à des résumés d'articles scientifiques.

Figure 5.6 — Première taxonomie après exécution automatique sur le corpus Astro

La figure 5.6 montre le résultat obtenu après exécution automatique avec les paramètres sus-cités. Une partie des feuilles de la taxonomie a été retirée de la capture pour essayer d'améliorer la lisibilité. Il est alors frappant, qu'avec seulement 90 termes en entrée, la lisibilité de la structure obtenue soit si limitée. Bien entendu, ce point est empiré par le fait que le système se développe dans le temps avant de stabiliser, et il est difficile de suivre le processus. En ce sens, on peut déjà considérer qu'il est trop difficile d'interpréter les résultats de notre système. En effet, les interfaces actuelles semblent inadaptées et la représentation d'un graphe en évolution quasi-permanente est un problème difficile. Dès lors, notre système ne peut, en l'état, constituer une solution complète d'ingénierie d'ontologie.

La métrique retenue pour la comparaison entre ontologies donne environ 0,78. Cette valeur est assez élevée malgré notre structure moins « à plat » que l'ontologie de référence. Cela signifie que, malgré l'existence de branches supplémentaires à plus haut niveau, le placement relatif des concepts dans les couches plus basses est comparable. À présent, en y regardant de plus près, on peut constater que la structure est assez mal équilibrée. En effet, de la racine de l'arborescence (au centre du regroupement racine, situé en bas à droite dans la figure) part une longue branche dans laquelle on peut distinguer trois sous groupes sans thématique tranchée. Cela constituera notre premier angle d'attaque pour interagir avec le système.

5.2.3 Intervention manuelle

À présent, nous intervenons sur le système pour essayer d'obtenir une structure plus satisfaisante. Cette section présente les quatre modifications apportées au système pour obtenir le résultat final.

Première modification

Comme nous l'avons vu à la section précédente, la structure obtenue automatiquement présente une branche dégradée et doit donc être améliorée. Cette branche présente trois sous-groupes. Chacun d'eux est ramené à la racine, le système réagit et se réorganise.

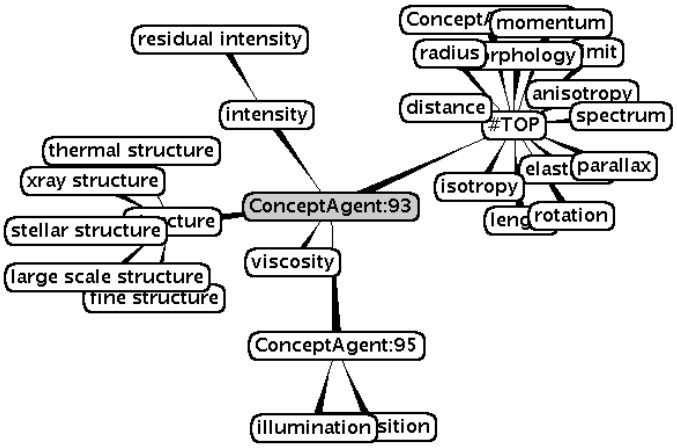


Figure 5.7 — Stabilisation du système après la première modification (extrait centré sur ConceptAgent :93)

Étant plus plate, cette nouvelle structure a une valeur de 0,80 lorsqu'elle est comparée à notre ontologie de référence. On remarque que « ConceptAgent :93 » regroupe essentiellement des propriétés mécaniques et thermodynamiques, alors que « #TOP » dispose en plus de propriétés géométriques. La figure 5.7 présente un extrait de la hiérarchie obtenue centrée sur « ConceptAgent :93 ». Nous allons à présent chercher à améliorer la séparation des différentes notions représentées.

Deuxième modification

Comme nous venons de le voir, « ConceptAgent :93 » constitue un pôle attractif pour des propriétés mécaniques et thermodynamiques. Notre seconde modification est alors de nettoyer les autres branches et déplacer les concepts concernant des propriétés mécaniques ou thermodynamiques sur « ConceptAgent :93 ». Les concepts concernés sont : *isotropy*, *morphology*, *momentum*, *anisotropy*, *spectrum*, *elasticity*, *mass*, *sensitivity*, *emission*, *density*, *entropy*, *diagram*, et *temperature*, c'est-à-dire que l'on effectue 13 déplacements en tout.

La nouvelle structure obtenue (figure 5.8) présente alors une branche plus complexe concernant uniquement des propriétés mécaniques et thermodynamiques. À cause de la structure de cette branche, nous nous sommes éloignés de l'ontologie de référence, et la mé-

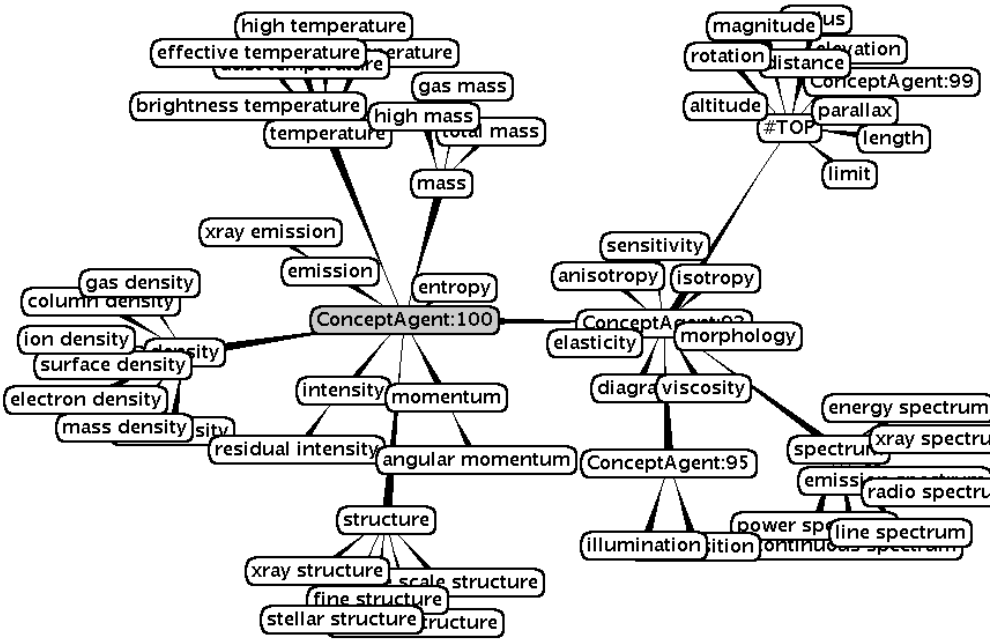


Figure 5.8 — Stabilisation du système après la seconde modification (extrait présentant la branche « mécanique et thermodynamique »)

trique de comparaison nous donne environ 0,76. Cette valeur reste stable jusqu'à la fin de l'expérimentation.

Troisième modification

À présent, nous nettoyons la branche démarrant avec « ConceptAgent :99 » qui a été ignorée jusqu'à présent. Elle contient des propriétés géométriques, ainsi qu'une sous-branche contenant des propriétés optiques. Nous déplaçons donc la sous-branche sous « #TOP », et toutes les propriétés géométriques de « #TOP » sous « ConceptAgent :99 ».

Alors le système réagit, et déplace toutes les propriétés géométriques et optiques sous « #TOP », puis crée une nouvelle branche contenant uniquement les propriétés optiques et *width* (figure 5.9). La destruction de notre branche pour les propriétés géométriques dénote d'une trop grande disparité de son contenu sur le plan statistique. De plus, les concepts en question, sont suffisamment génériques pour être gardés proches des plus spécifiques comme ceux de thermodynamique ou d'optique. Nous décidons donc de conserver cette modification.

Quatrième modification

À présent, nous sommes proche du résultat final. Nous disposons de trois ensembles cohérents de concepts sous « #TOP » (les concepts liés à la racine, un groupe sous « ConceptAgent :93 » et un groupe sous « ConceptAgent :104 ». Bien que nous ne cherchions plus à créer une troisième sous-branche sous « #TOP », il est clair que *width* n'a pas sa place dans la sous branche des propriétés optiques. Notre dernière modification est alors de déplacer

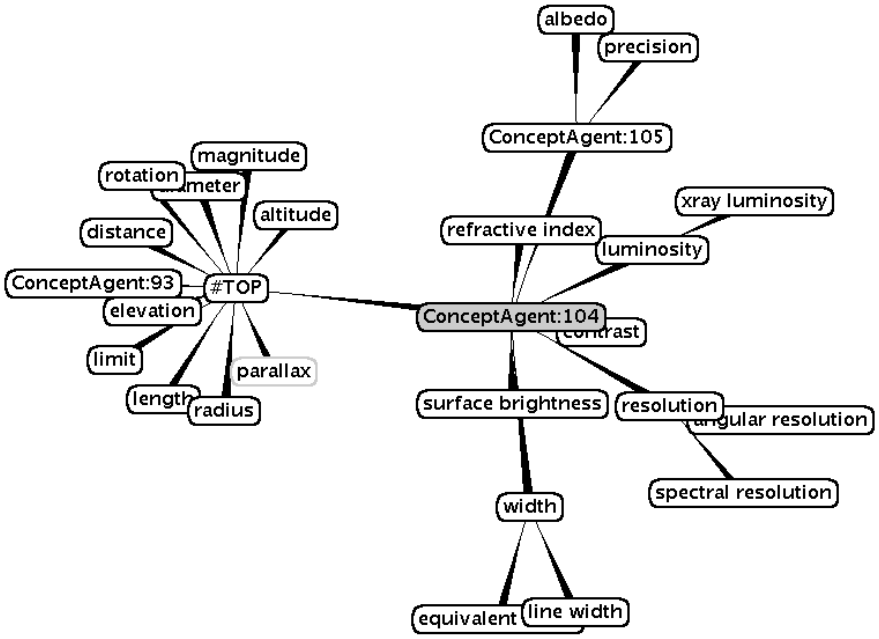


Figure 5.9 — Stabilisation du système après la troisième modification (extrait présentant la branche « optique »)

ce concept à la racine.

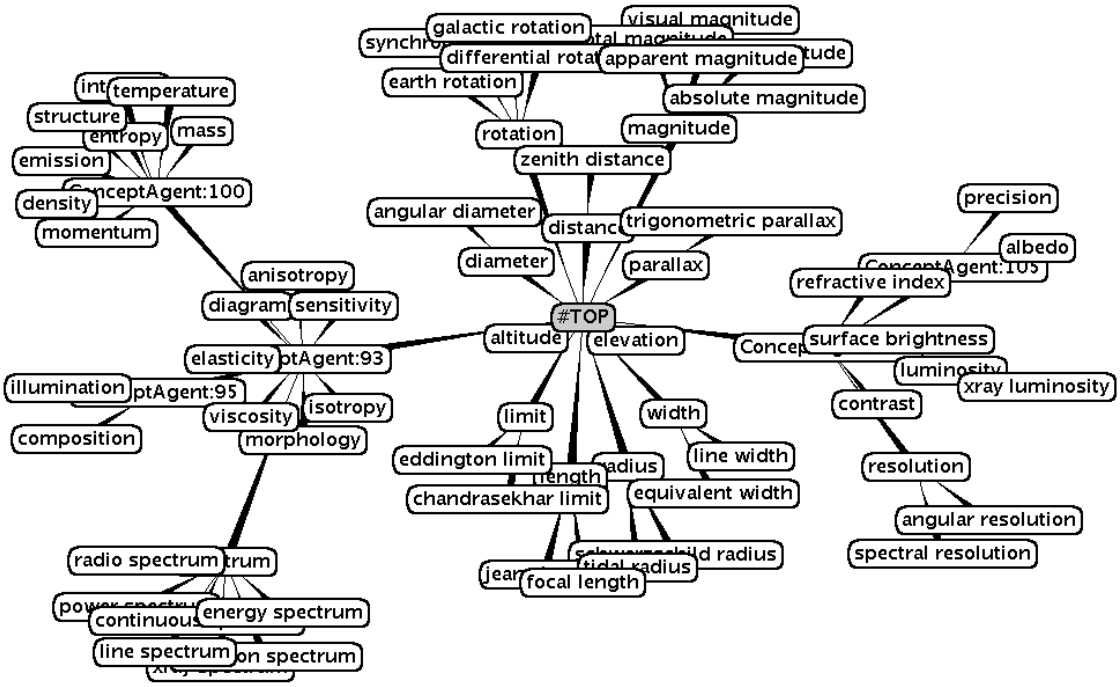


Figure 5.10 — Stabilisation du système après la quatrième modification

Ainsi, après stabilisation, nous obtenons la structure de la figure 5.10. On y distingue clairement trois sous-groupes cohérents : à gauche les propriétés mécaniques et thermodynamiques, au milieu les propriétés géométriques et à droite les propriétés optiques. Le temps total passé par l'utilisteur entre le démarrage du système et l'obtention du résultat a

été d'environ trois heures. La plupart du temps a été passé à réfléchir aux actions à entreprendre, mais surtout à observer la structure et modifier sa visualisation pour pouvoir faire ses choix.

Considérations sur l'effort de production de la taxonomie

Comme nous venons de le voir, malgré les difficultés en lisibilité du système, un résultat proche de l'ontologie de référence a été obtenu en un temps relativement court. On peut estimer que pour un ontologue ce travail nécessiterait quatre fois plus de temps avec un outil plus traditionnel pour le même résultat. La taxonomie produite est même légèrement plus raffinée ce qui explique les valeurs un peu basse de notre métrique.

Le processus en tant que tel a nécessité une intervention réduite de l'ontologue. En effet, le travail a été effectué en seulement quatre étapes. Chaque intervention de l'utilisateur a engendré des déplacements d'agents dans la hiérarchie. Pour les deux premières, environ une trentaine d'agents ont changé de parent à chaque étape, les deux dernières interventions n'ont généré qu'environ une demi-douzaine de déplacements chacunes.

Il est donc intéressant de constater que le système converge, et que la part d'intervention manuelle de l'utilisateur va en réduisant. Ainsi, l'ontologue peut se concentrer sur les grandes lignes du brouillon qu'il veut produire, le système adaptant les résultats de plus bas niveau.

5.3 Bilan

Dans ce chapitre, nous avons évalué les performances quantitatives de notre système. Pour cela, nous avons déterminé la complexité de notre algorithme distribué de classification, puis reévalué cette complexité à chaque ajout d'un comportement complémentaire. Nous avons donc constaté que la complexité en moyenne de notre algorithme distribué de classification est meilleure que celle d'un algorithme centralisé. De plus, nous avons pu vérifier que cette complexité restait raisonnable malgré des dispositifs complémentaires visant à apporter des aides supplémentaires à l'utilisateur et améliorer les résultats.

Enfin, nous avons cherché à utiliser notre système en conditions proches d'une utilisation réelle. Pour cela, nous avons utilisé un corpus disposant d'une ontologie de référence. Nous avons pu obtenir un résultat comparable à l'original avec un nombre de modifications réduit ce qui valide la pertinence de notre approche. Toutefois, il nous faut constater la difficulté à manipuler l'outil qui pose des problèmes de visualisation. Mais aussi, ce système requiert que l'utilisateur ait une connaissance non négligeable du comportement implémenté, étant donné le niveau d'abstraction auquel les manipulations sont effectuées.

Les possibilités d'évolution de notre approche et les recherches pertinentes à faire dans ce cadre sont nombreuses. Nous nous proposons de les examiner dans le chapitre suivant.

6 Perspectives

« Galilée : La science ne connaît qu'une loi : la contribution scientifique. »
Bertolt Brecht – La Vie de Galilée

LES différentes évaluations présentées dans le chapitre précédent nous ont permis de mieux caractériser notre système. En particulier, nous avons confirmé les avantages et les limites actuelles du prototype Dynamo. Bien entendu, afin de stimuler et baliser les recherches complémentaires qui seront nécessaires pour aller plus loin et obtenir une solution plus complète, nous avons porté notre réflexion sur certaines pistes à explorer.

Dans ce chapitre, nous abordons donc différents domaines et courants qui pourront apporter les pièces manquantes à un environnement complet d'ingénierie d'ontologies basé sur l'approche multi-agent présentée dans cette thèse, ainsi que les questions supplémentaires que de tels compléments soulèvent.

La section 6.1 présente les pistes envisageables pour intégrer des approches par patrons dans notre système afin d'identifier des relations transverses et ainsi ne plus se limiter à une taxonomie. Ensuite, la section 6.2 envisage la possibilité de réviser la méthode de calcul de l'indice de similarité en fonction des interactions avec l'utilisateur. Puis, la section 6.3 montre que malgré notre éloignement préalable des approches formelles, pour se concentrer sur les informations en provenance des textes, ces dernières sont séduisantes comme moyen d'améliorer nos taxonomies. En revanche, la section 6.4 concerne des aspects de plus bas niveau pour voir comment le cœur du système effectuant la classification pourrait être amélioré. Enfin, la section 6.5 examine un point qui semble aujourd'hui crucial : l'ergonomie d'un tel système.

6.1 Relations transverses

Comme nous l'avons vu au chapitre 4, notre système se concentre uniquement sur le dégagement de relations hiérarchiques de type « est un ». En ce sens, nous sommes capables de travailler uniquement sur la composante taxonomique de notre ontologie. Mais tout reste à faire concernant les autres relations nécessaires à l'obtention d'un « brouillon d'ontologie »

valable. À notre avis, il faudrait s'intéresser à nouveau à une approche par patrons comme celle présentée dans [Séguéla, 2001] pour dégager des relations supplémentaires. Pour cela, une solution possible serait une architecture étendue de Dynamo telle que celle présentée dans la figure 6.1.

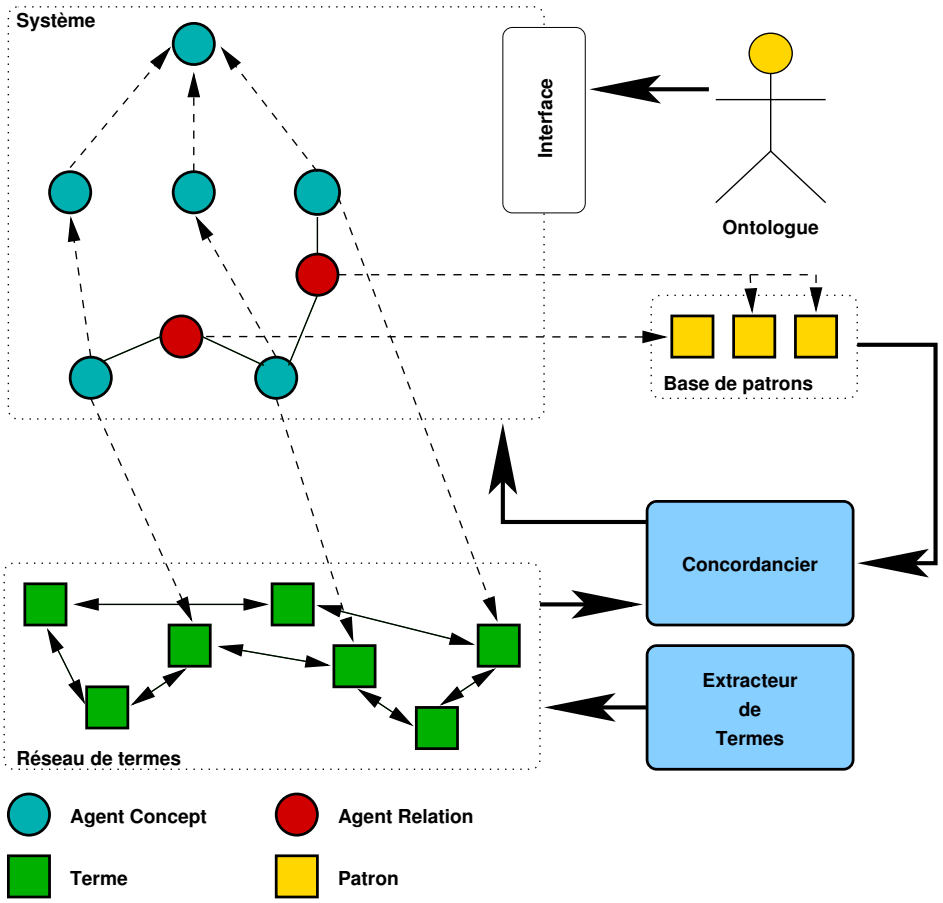


Figure 6.1 — Architecture incluant l'approche par patrons

Dans cette architecture mise à jour, le système dispose d'une base de patrons morpho-syntaxiques, par exemple similaire à celle disponible dans Caméléon [Séguéla, 2001]. Avec de tels patrons, il est possible de trouver les paires de termes utilisées dans un contexte syntaxique particulier. Un exemple simple de patron est « X "prendre" Article Y Adjectif », dans ce cas un des résultats valides renvoyés par le concordancier pourrait être la phrase « Demain, <X>Jean</X> prendra la <Y>pomme</Y> juteuse de l'arbre ». Ainsi, le concordancier permet de découvrir que « Jean » et « pomme » sont en relation syntaxique tel que décrit dans le patron.

Afin d'enrichir la structure construite par Dynamo, il faut passer de ces relations syntaxiques aux relations conceptuelles. Ainsi, les patrons seront utilisés par des agents « relation » dans le système. En ce sens, le statut des patrons sera similaire à celui des termes qui sont utilisés par des agents « concept ». Toutefois, le comportement des agents « relation » sera bien différent de celui des agents « concept ».

Chaque agent « relation » sera lié à un ou plusieurs patrons dans la base. Chaque agent

« relation », grâce à ses patrons et au concordancier, pourra essayer de déterminer si deux agents concepts sont liés par la relation représentée. Pour cela, un agent « relation » sera capable de déclencher des recherches via le concordancier. De plus, il disposera de capacités lui permettant de parcourir la structure en place pour trouver des partenaires potentiels. Ainsi des relations transverses seront établies entre des paires de concepts.

Toutefois, ce comportement suppose la pré-existence d'un ensemble de relations et de patrons. À l'heure actuelle, nous n'envisageons pas la possibilité de faire fonctionner une telle approche sans avoir une base initiale de patrons. Elle jouerait le rôle de la base des patrons génériques de l'approche Caméléon. Mais, cela n'interdit pas d'envisager que le système puisse découvrir de manière autonome de nouveaux patrons. C'est pourquoi notre architecture présente une boucle entre le système multi-agent en tant que tel et le concordancier. Il est tout à fait possible de mettre en place un apprentissage de nouveaux patrons pour un agent « relation » comme proposé par [Hearst, 1992] :

1. un agent « relation » R peu productif (il a trouvé peu de paires de concepts correspondants) a besoin d'augmenter sa productivité ;
2. il établit la liste des termes des agents « concept » qu'il relie ;
3. il recherche les endroits dans le corpus où les termes sont co-occurrents et mémorise leurs environnements ;
4. il détermine les régularités dans ces environnements et en extrait de nouveaux patrons (nous faisons ici l'hypothèse que ces régularités sont bien la base de nouveaux patrons) ;
5. l'utilisateur a alors accès à de nouveaux résultats qu'il pourra critiquer, l'agent R pourra alors décider de conserver ou non le nouveau patron.

Il est donc possible, grâce aux concepts et relations existants, d'effectuer des projections complémentaires avec le concordancier pour découvrir ces nouveaux patrons et donc raffiner les relations existantes ou en découvrir de nouvelles. Toutefois, cela pose la question de l'étiquetage lors de la découverte de nouvelles relations. Comme pour les agents « concept » de structuration dans l'algorithme de classification, un étiquetage automatique semble exclu. Ainsi, il est fort probable que cette approche requiert d'introduire une interaction supplémentaire avec l'utilisateur. Cela comporte aussi le risque de noyer l'utilisateur de demandes de critiques pour les nouveaux résultats produits si les marqueurs deviennent trop productifs trop vite. La réduction de ces échanges pourrait aussi se contrôler en exploitant les AMAS, en proposant à l'ontologue les relations actuelles les plus critiques en terme de productivité.

6.2 Apprendre sur les règles de calcul

Dans le chapitre 4, nous traitons les relations d'expansion de manière uniforme pour le calcul de similitudes dans Dynamo. Or *Syntex* présente des résultats plus fins avec un ensemble différent de relations d'expansion. Par exemple, le terme « prévoir » est une expansion des termes « intervention prévoir » et « bilan à prévoir ». Dans le premier cas, il s'agit d'une expansion de type adjectif alors que dans le second cas il s'agit d'une expansion de type « à ». Il paraît donc intéressant d'explorer l'impact qu'une pondération sur la

prise en compte de ces relations pourrait avoir. En effet, on ne peut garantir que l'apport des différents types d'expansions existants soit uniforme dans un corpus.

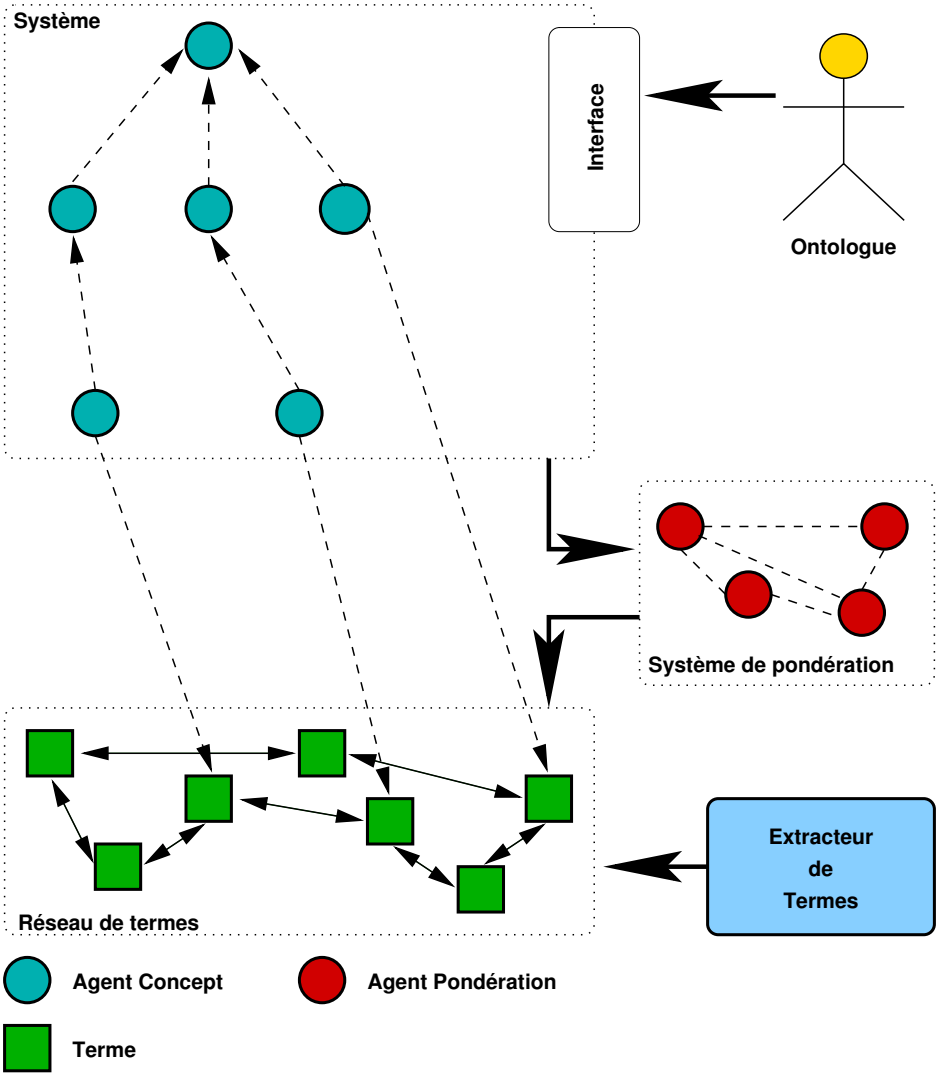


Figure 6.2 — Architecture incluant la pondération des relations du réseau tête-expansion

Avec l'architecture proposée à la figure 6.2, il devient possible d'obtenir une telle révision sur les calculs de similitude. Pour cela, nous pouvons introduire un petit système multi-agent réactif représentant les pondérations sur les types d'expansions. Les pondérations sont alors déterminées en fonction des interactions entre les agents de ce second système spécialisé. Les résultats sont alors utilisés dans les calculs de similitudes entre termes, et le système multi-agent constitué par les agents concepts peut répercuter les pressions de l'utilisateur afin d'affiner les calculs de similitudes en fonction des critiques.

6.3 Evaluer l'apport des ontologies formelles et la réutilisation

Comme nous avons pu le voir dans les chapitres 4 et 5, notre système travaille uniquement à partir de textes et des interactions avec l'utilisateur. Il produit une structure taxono-

mique faiblement formalisée. Ce point est très certainement à l'origine de la forte variance constatée à la figure 5.5 ainsi que la possibilité des résultats différents corrects structurellement mais pas forcément sémantiquement. En ce sens, l'utilisateur est livré à lui-même. Il s'agissait d'un de nos buts initiaux de ne pas trop contraindre l'utilisateur, mais à la lumière des résultats, on peut se demander s'il ne serait pas sage de le guider un peu plus dans ses choix.

Or, nous savons que par essence tout ne peut être découvert dans les textes. C'est entre autre la raison pour laquelle nous nous reposons en partie sur l'utilisateur. Il convient alors de dégager une source d'informations supplémentaire pour alléger l'utilisateur. Pour cela, une inspiration possible provient des courants de la construction d'ontologies formelles. Par exemple, donner aux agents concepts des connaissances de plus haut niveau, telles que les critères de différenciation de [Bachimont, 2004], les critères de structuration d'OntoClean ou des concepts de DOLCE [Gangemi *et al.*, 2002], est une piste intéressante. En effet, cela permettrait aux agents de se critiquer et de se repositionner pour satisfaire un maximum des critères mis en place.

Toutefois, cela nécessite de disposer d'un *bootstrap* de concepts correctement étiquetés et positionnés. Là encore, on peut envisager de se reposer sur l'utilisateur. Il serait alors amené à différencier et étiqueter lui-même les entités de l'ontologie. Mais une solution probablement meilleure est la réutilisation, en important dans le système une portion d'ontologie déjà construite, valide, bien formée et étiquetée comme DOLCE ou les core-ontologies de domaine [Hoekstra *et al.*, 2007]. L'ontologue peut ainsi y raccrocher les portions de plus bas niveau construites à partir des textes. Ainsi, des vérifications de validité supplémentaires pourront être effectuées par héritage de propriétés.

Enfin, grâce à un contenu plus formel dans Dynamo, il serait envisageable de permettre à l'ontologue d'exprimer des contraintes supplémentaires pour les concepts de l'ontologie. En effet, dans un tel contexte, il semble possible de permettre la saisie d'axiomes que la structure résultante devra satisfaire.

6.4 Améliorer l'algorithme de classification

Comme nous l'avons vu au chapitre 4, le comportement de Dynamo s'organise autour de l'algorithme distribué de classification. Après les évaluations du chapitre 5, les résultats sont encourageants mais certainement perfectibles. Il nous semble qu'il existe quatre axes d'exploration potentiels pour notre algorithme et son implémentation.

Tout d'abord, la possibilité d'améliorer la complexité. La notion en elle-même demande à être précisée ; en effet, tout dépend du type d'opération que l'on cherche à réduire. Comme nous l'avons pointé au chapitre 5, dans le pire cas, notre algorithme effectue deux fois plus de calculs de similitudes que l'algorithme centralisé auquel il est comparé. La possibilité d'éviter ces doublons est tentante mais au prix d'un plus grand nombre de messages. Or, dans un tel système, il paraît, à notre avis, plus important de limiter le nombre de messages échangés. Et c'est sur ce point que la réflexion devra porter. Actuellement, pour un agent donné, l'état entre deux votes n'est pas conservé, pourtant lorsque son voisinage change peu, il paraît raisonnable de vouloir réutiliser les données du vote précédent, ce qui permet-

trait certainement de réduire la quantité de messages échangés.

Ensuite, le problème majeur de notre algorithme est sa dépendance à l'ordre d'insertion des agents supplémentaires dans le système. Suivant l'ordre d'instanciation ou les déplacements des agents à un instant donné, la position d'équilibre peut être différente. Pour cela, une réflexion devra être conduite qui poussera très certainement à donner des actions et mesures supplémentaires aux agents afin d'avoir une insertion plus informée en amont. Pour cet aspect, la notion d'homogénéité d'un niveau présentée dans [Parunak *et al.*, 2006] constitue une première piste intéressante.

De plus, l'approche développée au chapitre 3 que nous avons employée pour construire notre prototype, a permis d'expérimenter grâce à une conception clarifiée. Et, grâce à la génération de code, nous avons garanti certaines propriétés internes aux agents pour leur implémentation. Toutefois, cette structure interne plus complexe impose *de facto* des traitements plus complexes et coûteux. En effet, un simple message est recopié et raffiné plusieurs fois dans les traitements internes de l'agent. Ainsi, le délai entre la prise en compte d'un message et l'action résultante pourrait certainement être réduit en simplifiant la structure des agents. Il conviendra bien de s'assurer que l'intérêt de ce type de micro-optimisation (un traitement plus rapide des messages) est plus élevé que son coût (une plus faible maintenabilité et réutilisabilité dans le système).

Enfin, il serait probablement intéressant d'utiliser cet algorithme pour des tâches de *data mining*. Ainsi, nous pourrions vérifier s'il est entièrement réutilisable hors d'un contexte purement terminologique comme nous le supposons. Les qualités de l'algorithme que nous avons dégagées au chapitre 5 pourraient alors être transposées à d'autres domaines.

6.5 Enrichir l'interaction avec l'utilisateur

Comme nous l'avons vu au chapitre 5, il est très difficile de manipuler simplement le prototype Dynamo et d'évaluer la qualité des résultats sur un contexte global. C'est la raison pour laquelle nous nous sommes concentrés sur l'évaluation de la qualité du comportement obtenu uniquement sur des sous-parties de la taxonomie.

Ce problème nécessite une réflexion profonde sur l'ergonomie d'un tel système. En effet, il est soumis à des contraintes classiques d'un outil permettant de naviguer dans un graphe. Plus ce graphe a une taille importante, plus il est difficile à visualiser et plus l'utilisateur est noyé sous la quantité d'information. Dans notre cas, une dimension supplémentaire doit être prise en compte : le temps. En effet, nous sommes en présence d'une structure potentiellement importante mais surtout malléable et changeante. Une pression de l'utilisateur à un endroit peut avoir des répercussions supplémentaires qui poussent le système à changer de configuration. Cette problématique doit être abordée par des spécialistes de l'ergonomie.

6.6 Bilan

Dans ce chapitre, nous avons vu cinq axes possibles pour améliorer notre approche afin de nous orienter vers un environnement plus complet et une approche plus robuste pour un

système multi-agent de construction d'ontologies à partir de texte. Toutefois, nous pensons que les axes présentés dans les sections 6.1 et 6.5 sont prioritaires.

En effet, les relations transverses sont clairement un pré-requis nécessaire à d'autres approches d'amélioration de la structuration. Par exemple, sans ces relations, la quantité d'information disponible dans le système est sûrement trop faible pour pouvoir utiliser une approche formelle en vue d'améliorer la structuration de manière pertinente et fiable. De plus, l'ergonomie nous semble très importante puisque Dynamo montre clairement que l'utilisateur doit manipuler une structure potentiellement complexe qui évolue dans le temps. Une réflexion sur ce thème est donc cruciale pour pouvoir obtenir un système utilisable sans une connaissance importante du fonctionnement interne du système.

Conclusion

« On doit exiger de moi que je cherche la vérité, mais non que je la trouve. »
Denis Diderot – Pensées philosophiques

TOUT au long de ce travail, notre objectif a été de fournir les moyens de construire des ontologies dynamiques. Des expérimentations et exemples d'utilisation de notre prototype ont été présentés afin de valider notre approche. Ces travaux font le pont entre deux domaines particuliers : l'ingénierie des connaissances et l'ingénierie des systèmes multi-agents. Certains choix fondamentaux ont été pris en compte pour satisfaire notre souci de diffusion.

Les choix scientifiques

Le premier choix est celui de l'approche méthodologique pour l'ingénierie des connaissances : la construction d'ontologies à partir de textes. Ce choix est important, d'autant plus que nous réfutons l'argument de la connaissance en tant qu'objet totalement consensuel et stable. Cette position n'est pas nouvelle, de nombreux travaux – mais pas tous – la considèrent comme acquise.

Le deuxième choix est celui de la classe de systèmes utilisée par Dynamo : les *AMAS*. Nous avons confirmé que ces systèmes multi-agents, donc distribués, peuvent très bien implémenter les règles nécessaires pour supporter des algorithmes similaires à ceux utilisés dans les approches classiques. De plus, dans ce cadre nous avons pu développer des règles supplémentaires tenant compte des manipulations généralement faites par l'utilisateur pour construire une ontologie. Nous nous sommes particulièrement concentrés sur le facteur de branchement et la structuration de bas niveau de l'ontologie. Ainsi, nous avons suivi notre volonté de revoir la construction d'ontologie sous l'angle de la *conception vivante*.

Le troisième choix est directement lié au choix des AMAS : l'ingénierie des modèles. Ce choix est probablement celui qui a le moins d'impact sur un plan scientifique. Toutefois, il reste fort judicieux pour améliorer la compréhension de la structure interne de nos agents et pour guider l'implémentation de notre prototype. Les systèmes multi-agents étant des systèmes logiciels, ce type d'approche est tout à fait adapté, d'autant plus que nous avions

fait le choix du paradigme objet via ADELFE.

Bilan et apports à l'ingénierie des connaissances

Notre travail sur Dynamo a plusieurs retombées sur l'ingénierie des connaissances. Tout d'abord, il s'agit, à notre connaissance, de la première utilisation des systèmes multi-agents dans un tel contexte. Nous avons eu le souci d'obtenir une structure qui comporte en son sein à la fois le processus et le résultat de la construction d'une ontologie à partir de textes.

Ce travail nous a poussé à avoir une réflexion sur les manipulations de l'utilisateur pour obtenir un brouillon d'ontologie à partir d'une base de termes. Ainsi, nous avons pu identifier les contraintes implicitement utilisées par l'ontologue pour la structuration de bas niveau.

Un autre apport est alors la rénovation de la classification avec une approche distribuée. En particulier, nous avons introduit une métrique supplémentaire – la fonction d'adéquation d'un parent. Mais nous avons aussi ajouté de la souplesse à l'algorithme distribué de classification par l'introduction de seuils. Certains de ces paramètres peuvent probablement être réutilisées pour des algorithmes centralisés.

Enfin, notre prototype et les expérimentations conduites, nous ont permis de valider l'approche. Ainsi, nous avons pu confirmer l'importance de conserver l'ontologue au centre du processus de construction et donc l'impossibilité d'automatiser totalement un processus d'acquisition de connaissances.

Cependant, tout n'est pas arrêté et complet. De nombreuses perspectives s'ouvrent dans le sens du développement de Dynamo, comme il a été souligné dans le chapitre 6. Le système permet d'obtenir des ébauches d'ontologies, mais il doit être complété par des dispositifs permettant de ramener plus d'informations à l'utilisateur mais aussi en le guidant davantage dans sa construction. La problématique de l'ergonomie trouvera certainement sa place dans cette nouvelle approche des ontologies dynamiques.

Bilan et apports aux AMAS

À l'instar de tous les travaux utilisant les AMAS comme modèle de conception, notre travail a permis de clarifier leur utilisation et de les confronter à un domaine supplémentaire. Les réflexions sur le modèle d'agent et ses stéréotypes ont permis de le préciser et d'y ajouter de nouvelles contraintes. Comme le domaine en question – la construction d'ontologies à partir de textes – est complexe, ces travaux ont été un véritable défi pour l'approche AMAS. En effet, les réflexions de conception se portent souvent sur des considérations anthropomorphiques totalement absentes ici. Il a donc fallu s'assurer d'une plus grande rigueur dans notre progression.

Cette rigueur nécessaire, ainsi que les réflexions sur le modèle, nous ont donc poussés à étendre ADELFE avec une approche d'ingénierie des modèles. Nous avons été directement utilisateurs de cette approche, et nous avons ainsi commencé à couvrir la phase d'implémentation d'ADELFE qui était, à ce jour, non traitée.

Des résultats prometteurs...

Ces travaux se situent dans la continuités de mes études de DEA [Ottens *et al.*, 2004; Ottens, 2004], dont les ambitions étaient moindre et les résultats moins encourageants. Ici, nous nous sommes attachés à prendre en compte des objectifs de plus haut niveau. Ainsi, nous avons pu niveler le terrain scientifique pour obtenir une réelle dynamique dans la construction d'ontologies à partir de texte. Le comportement des agents que nous avons développés ainsi que nos manipulations le prouvent. Nous sommes réellement proches d'une structure de *conception vivante*.

En plus de la validité constatée de l'approche, nos travaux ont confirmé que les performances sont au rendez-vous. En effet, les calculs et expérimentations présentés dans le chapitre 5 confirment que la complexité du système est tout à fait acceptable pour ce type de problème.

... et des progrès à faire

Toutefois, il reste un long chemin à parcourir pour obtenir un système complètement utilisable. Bien que les résultats obtenus soient encourageants, la qualité des structures obtenues peut sûrement être améliorée. Dans tous les cas, nous avons essentiellement travaillé sur la composante taxonomique de l'ontologie, et elle doit à l'avenir être complétée. Le chapitre 6 donne un éclairage sur les pistes à suivre pour atteindre ce but. En particulier, l'utilisation de patrons permettra de compléter l'arbre construit pour obtenir un réseau conceptuel plus complet.

Enfin, il nous semble que la priorité doit être donnée à l'ergonomie de Dynamo. Comme souvent, elle est sous-estimée, et pourtant s'avère être un aspect critique de notre approche. Malheureusement, tout doit avoir une fin et cet aspect ne pourra être abordé ici. Il me semble donc nécessaire avant de pouvoir continuer plus avant dans cette voie, d'étudier l'ergonomie de systèmes distribués et dynamiques – tels que les AMAS – ce qui représente une de mes perspectives de recherches futures.

Quatrième partie

Annexes

A Corpus

CETTE annexe présente des extraits des corpus utilisés pour nos expérimentations. La section A.1 est un extrait du corpus REACHIR constitué de compte-rendus de réanimation chirurgicale, il a été principalement utilisé pour nos évaluations de performance. Ce corpus a été constitué lors des travaux de DEA de Sophie Lemoigno à l'INSERM UMR_S 872, éq. 20 (anciennement U729) et a servi à la construction d'une ontologie de la réanimation chirurgicale [Le Moigno *et al.*, 2002].

La section A.2 est un extrait du corpus « Astronomy & Astrophysics » constitué de résumés d'articles parus dans le journal du même nom. Il est intéressant de constater que le style des deux corpus est très différent : prise de notes en français peu rédigées pour REACHIR, et textes descriptifs correctement rédigés en anglais pour « Astronomy & Astrophysics ». Ils ont été utilisés pour évaluer les performances de notre système ainsi que son comportement en utilisation.

A.1 REACHIR

- traumatisme crânien balistique.
- non connus.
- Amenée aux urgences de Villeneuve Saint Georges le 19 / 02 à 23 h 30 par son ami, plaie intracrânienne balistique avec orifice d'entrée temporal gauche, pas d'orifice de sortie.
- Prise en charge SAMU 94 pour transfert en Réanimation chirurgicale :
- GCS 3, intubée d'emblée, mydriase aréactive bilatérale, absence de reflexes cornéens et oculocardiaques.
- SpO2 100 % sous ventilation mécanique, PAS 80 mmHg, pouls 105, adrénaline 0,25 mg / h IV continu.
- La radiographie du crâne face profil retrouve de nombreux éclats de grenaille intracrâniens.
- Décès par état de mort encéphalique confirmé à l'électro-encéphalogramme effectué le 20 / 02 à 1 h 30.
- Bilan biologique toxicologique négatif sauf pour l'éthanolémie (0,2 en plasmatique).
- état de mort encéphalique dans les suites d'un traumatisme crânien balistique.
- Polytraumatisme.
- Appendicectomie dans l'enfance.
- Accident de la voie publique.
- Passagère avant éjectée d'un véhicule retourné, la passagère étant incarcérée sous un

deuxième véhicule.
- A l'arrivée des premiers secours, elle est consciente, agitée, pression artérielle imprenable.
- Déformation majeure des membres inférieurs qui sont retournés à 135 ° par rapport au niveau du bassin.
- Désincarcération faite.
- La patiente est agitée, polypnéique.
- Mise en place d'une voie d'abord par voie sous-clavière droite.
- Intubation oro-trachéale n° 7.
- Le premier bilan constate une fracture ouverte au niveau de la cheville droite, une plaie de 10 cm sur la face antérieure de la jambe gauche, une fracture du bassin avec une mobilité transversale, d'énormes ecchymoses au niveau du thorax gauche et un hématome.
- Il existe, en plus, une plaie du cuir chevelu frontal sans embarure visible.
- La patiente est adressée à l'hôpital Henri Mondor.
- mise en place d'une artère fémorale droite et d'un désilet fémoral droit.
- Administration de culots, PVI, plaquettes.
- Restauration d'une hémodynamique sous Adrénaline.
- Une échographie abdominale effectuée en urgence retrouve un épanchement de faible abondance péri - hépatique.
- Pas d'épanchement pleural.
- Après avis conjoint de l'équipe de chirurgie digestive, un scanner thoraco-abdominal est réalisé qui montrera des contusions hépatiques diffuses, un épanchement intra-abdominal abondant, un hématome de la fosse iliaque gauche en regard de multiples fractures du bassin et surtout un saignement actif de l'artère iliaque gauche.
- Il existe aussi des fractures des apophyses transverses de D12 et de L3, une fracture du sacrum de l'aile iliaque gauche et du cadre obturateur droit.
- Au niveau cérébral, il existe une très discrète contusion frontale bilatérale, pas de collection intra-cérébrale et la structure médiane est en place.
- Très rapidement, la patiente est dirigée au bloc opératoire.
- Au bloc opératoire, un premier arrêt cardiaque sur table est noté dès l'incision.
- La patiente présentera un deuxième arrêt cardiaque sur table qui ne sera pas récupéré.
- polytraumatisme avec plaie de l'artère iliaque gauche sur fracas de bassin.
- Chirurgie de sauvetage en urgence.
- Décès de la patiente.
- chute d'environ 4 mètres après un malaise.
- Hypothyroïdie congénitale, traitement habituel 150 ? g lévothyrox (prise du traitement aléatoire) pas de suivi régulier.
- Tendinite bras droit.
- Prise de poids récente (environ 20 kg).
- Contexte social :
- ancienne SDF, bénéficie depuis peu de temps de l'aide médicale.
- Notion de malaise type lipothymie il y a un mois.
- Nouveau malaise le 20 / 02 avec probable perte de connaissance de quelques se-

condes, ayant entrainé une chute d'environ 4 mètres sur un sol en béton.
– Prise en charge SAMU 94 :
– GCS 15, amnésie de l'évènement, pas de signe de localisation, pupilles normales.
– Etats hémodynamique et respiratoire normaux, examen de l'abdomen normal.
– Douleur diffuse à la palpation du rachis cervical et thoracique, douleur au niveau de la rotule et des condyles fémoraux droits, petite dermabrasion face antérieure genou droit.
– Le bilan d'imagerie initial ne retrouve aucune lésion.
– Le séjour se caractérise par la stabilité de tous les paramètres.

A.2 Astronomy & Astrophysics (Astro)

– We report the detection of the extremely red object (ERO), HR10 (I-K 6.5, z=1.44), at 4.9 and 6.1mum (rest-frame) with ISOCAM, the mid-infrared (MIR) camera onboard the Infrared Space Observatory (ISO). HR10 is the first ERO spectroscopically identified to be associated with an ultra-luminous IR galaxy (ULIG) detected in the radio, MIR and sub-millimeter. The rest-frame spectral energy distribution (SED) of HR10 is amazingly similar to the one of Arp 220, scaled by a
– We investigate consequences of a continuously energy-injecting central engine of gamma-ray burst (GRB) afterglow emission, assuming that a highly magnetized pulsar is left beaming in the core of a GRB progenitor. Beaming and continuous energy-injection are natural consequences of the pulsar origin of GRB afterglows. Whereas previous studies have considered continuous energy-injection from a new-born pulsar to interpret the deviation of afterglow light curves of GRBs from
– We report on the discovery of the proper motion of the central star (CS) of the Planetary Nebula (PN)
objectSh2-68 (PN G030.6+06.2). Sh2-68 is a PN interacting with the interstellar medium (ISM), which impedes and finally stops the expansion of the nebula itself while the galactic motion of the CS leads to it moving out of the geometric center of the PN. Our observation is a direct confirmation of this process and combined with imaging data suggests that the ISM's
– A reduced network for CO production and removal, obtained via the use of objective techniques, is shown to produce reliable results over a wide range of physical conditions. Our analysis indicates that reactions involving species with long carbon chains are important in governing the CO abundance in some translucent regions.
– We present new observations of the D_2_CO emission towards the small cloud L1689N in the rho Ophiuchus complex. We surveyed five positions, three being a cut across a shock site and two probing the quiescent gas of the molecular cloud. We detected D_2_CO emission in the first three positions. The measured [D_2_CO]/[H_2_CO] is about 3%, whereas it is <=2% in the quiescent gas. We discuss the implications of these new observations, which suggest that the bulk of the D_2_CO
– When the eccentricity of a Maclaurin spheroid exceeds a critical value (e>0.83458318), circular orbits in the equatorial plane are unstable for a range of orbital radii outside the stellar surface - this is a purely Newtonian effect related to the oblateness of the

star. The orbital frequency in the marginally stable orbit, and all other orbits, around Maclaurin spheroids goes to zero in the limit e=1. The orbital and rotational frequencies in exact relativistic numerical

– We investigate the role of stellar axial rotation on the nitrogen nucleosynthesis at low metallicities Z. For this purpose, we have calculated models with initial masses between 2 and 60M_sun_ at Z=0.00001 from the zero age sequence to the phase of thermal pulses for models below or equal to 7M_sun_, and up to the end of central C-burning for the more massive stars. The models include all the main physical effects of rotation. We show that intermediate mass stars
– VLA observations of OH absorption towards the elliptical galaxy NGC 1052 are presented. Both OH lines, at 1665 and 1667MHz, were detected in absorption towards the center of NGC 1052. The hyperfine ratio of the two OH lines (tau_1667_/tau_1665_) is 2.6+/-0.8 as compared to 1.8 expected for the excitation under LTE conditions for an optically thin cloud. The column density of OH is estimated to be 2.73(+/-0.26)x10^14^cm^-2 assuming T_ex_ 10K. The centers of both
– The NS/CS ratio can be used to test if shocks or thermal evaporation remove grain ices during massive star formation. The two scenarios lead to differences in the subsequent chemistry : in particular, timescales are shorter if shocks are present. We have measured NS/CS ratios in six hot core sources through observation of high excitation NS, N^34^S, C^34^S and C^33^S with the JCMT. The NS/CS ratios we find are low, 0.02-0.05, and surprisingly consistent between sources.

B Ontologies

CETTE annexe présente l'ontologie de référence utilisée pour l'expérimentation de la section 5.2, ainsi que les ontologies produites à chacune des étapes importantes de la manipulation. Nous présentons ici uniquement la composante taxonomique de chaque ontologie. Et le format de représentation est simplement « (étiquette du concept parent, étiquette du concept fils) ».

B.1 Ontologie de référence pour le corpus Astro

- (property, length)
- (property, precision)
- (property, limit)
- (property, morphology)
- (property, dimension)
- (property, diagram)
- (property, altitude)
- (property, distance)
- (property, structure)
- (property, optical property)
- (property, anisotropy)
- (property, spectrum)
- (property, resolution)
- (property, viscosity)
- (property, rotation)
- (property, width)
- (property, radius)
- (property, intensity)
- (property, entropy)
- (property, isotropy)
- (property, luminosity)
- (property, parallax)
- (property, sensitivity)
- (property, composition)
- (property, illumination)
- (property, density)
- (property, mass)
- (property, contrast)

- (property, elasticity)
- (property, momentum)
- (property, temperature)
- (property, magnitude)
- (property, emission)
- (length, focal length)
- (length, jeans length)
- (limit, chandrasekhar limit)
- (limit, eddington limit)
- (dimension, diameter)
- (altitude, elevation)
- (distance, zenith distance)
- (structure, stellar structure)
- (structure, xray structure)
- (structure, large scale structure)
- (structure, thermal structure)
- (structure, fine structure)
- (optical property, albedo)
- (optical property, refractive index)
- (spectrum, radio spectrum)
- (spectrum, energy spectrum)
- (spectrum, xray spectrum)
- (spectrum, emission spectrum)
- (spectrum, continuous spectrum)
- (spectrum, line spectrum)
- (resolution, spectral resolution)
- (resolution, angular resolution)
- (rotation, differential rotation)
- (rotation, earth rotation)
- (rotation, synchronous rotation)
- (rotation, galactic rotation)
- (width, line width)
- (radius, tidal radius)
- (radius, schwarzschild radius)
- (intensity, residual intensity)
- (luminosity, xray luminosity)
- (parallax, trigonometric parallax)
- (composition, isotopic composition)
- (composition, chemical composition)
- (density, electron density)
- (density, gas density)
- (density, ion density)
- (density, flux density)
- (density, mass density)

- (momentum, angular momentum)
- (temperature, effective temperature)
- (temperature, brightness temperature)
- (temperature, high temperature)
- (magnitude, absolute magnitude)
- (magnitude, apparent magnitude)
- (magnitude, total magnitude)
- (magnitude, limiting magnitude)
- (magnitude, visual magnitude)
- (diameter, angular diameter)
- (optical property, surface brightness)
- (emission, xray emission)
- (line width, equivalent width)
- (density, column density)
- (density, surface density)
- (mass, gas mass)
- (mass, total mass)
- (temperature, dust temperature)
- (spectrum, power spectrum)
- (mass, high mass)
- (temperature, kinetic temperature)

B.2 Brouillon d'ontologie obtenue sur le corpus Astro par Dynamo

- (magnitude, absolute magnitude)
- (limit, chandrasekhar limit)
- (composition, chemical composition)
- (density, column density)
- (ConceptAgent :95, composition)
- (spectrum, continuous spectrum)
- (ConceptAgent :96, contrast)
- (ConceptAgent :94, density)
- (ConceptAgent :96, diagram)
- (#TOP, diameter)
- (rotation, differential rotation)
- (ConceptAgent :96, albedo)
- (#TOP, distance)
- (temperature, dust temperature)
- (rotation, earth rotation)
- (limit, eddington limit)
- (temperature, effective temperature)
- (#TOP, elasticity)
- (density, electron density)
- (ConceptAgent :92, elevation)

- (ConceptAgent :92, emission)
- (spectrum, emission spectrum)
- (ConceptAgent :92, altitude)
- (spectrum, energy spectrum)
- (ConceptAgent :96, entropy)
- (width, equivalent width)
- (structure, fine structure)
- (density, flux density)
- (length, focal length)
- (rotation, galactic rotation)
- (density, gas density)
- (mass, gas mass)
- (mass, high mass)
- (diameter, angular diameter)
- (temperature, high temperature)
- (ConceptAgent :95, illumination)
- (ConceptAgent :93, intensity)
- (density, ion density)
- (composition, isotopic composition)
- (#TOP, isotropy)
- (length, jeans length)
- (temperature, kinetic temperature)
- (structure, large scale structure)
- (#TOP, length)
- (momentum, angular momentum)
- (#TOP, limit)
- (magnitude, limiting magnitude)
- (spectrum, line spectrum)
- (width, line width)
- (ConceptAgent :94, luminosity)
- (#TOP, magnitude)
- (ConceptAgent :92, mass)
- (density, mass density)
- (ConceptAgent :94, momentum)
- (ConceptAgent :94, morphology)
- (resolution, angular resolution)
- (#TOP, parallax)
- (spectrum, power spectrum)
- (ConceptAgent :94, precision)
- (spectrum, radio spectrum)
- (#TOP, radius)
- (ConceptAgent :96, refractive index)
- (ConceptAgent :94, resolution)
- (intensity, residual intensity)

- (#TOP, rotation)
- (radius, schwarzschild radius)
- (#TOP, anisotropy)
- (ConceptAgent :92, sensitivity)
- (resolution, spectral resolution)
- (#TOP, spectrum)
- (structure, stellar structure)
- (ConceptAgent :93, structure)
- (ConceptAgent :96, surface brightness)
- (density, surface density)
- (rotation, synchronous rotation)
- (structure, thermal structure)
- (ConceptAgent :97, temperature)
- (magnitude, apparent magnitude)
- (radius, tidal radius)
- (magnitude, total magnitude)
- (mass, total mass)
- (parallax, trigonometric parallax)
- (ConceptAgent :93, viscosity)
- (magnitude, visual magnitude)
- (ConceptAgent :97, width)
- (emission, xray emission)
- (luminosity, xray luminosity)
- (spectrum, xray spectrum)
- (temperature, brightness temperature)
- (structure, xray structure)
- (distance, zenith distance)
- (#TOP, ConceptAgent :92)
- (ConceptAgent :92, ConceptAgent :93)
- (ConceptAgent :93, ConceptAgent :94)
- (ConceptAgent :93, ConceptAgent :95)
- (ConceptAgent :94, ConceptAgent :96)
- (ConceptAgent :96, ConceptAgent :97)

B.3 Ontologie obtenue sur Astro après la première modification

- (magnitude, absolute magnitude)
- (limit, chandrasekhar limit)
- (composition, chemical composition)
- (density, column density)
- (ConceptAgent :95, composition)
- (spectrum, continuous spectrum)
- (ConceptAgent :96, contrast)
- (ConceptAgent :99, density)

- (ConceptAgent :96, diagram)
- (ConceptAgent :99, diameter)
- (rotation, differential rotation)
- (ConceptAgent :96, albedo)
- (#TOP, distance)
- (temperature, dust temperature)
- (rotation, earth rotation)
- (limit, eddington limit)
- (temperature, effective temperature)
- (#TOP, elasticity)
- (density, electron density)
- (ConceptAgent :98, elevation)
- (ConceptAgent :98, emission)
- (spectrum, emission spectrum)
- (ConceptAgent :98, altitude)
- (spectrum, energy spectrum)
- (ConceptAgent :96, entropy)
- (width, equivalent width)
- (structure, fine structure)
- (density, flux density)
- (length, focal length)
- (rotation, galactic rotation)
- (density, gas density)
- (mass, gas mass)
- (mass, high mass)
- (diameter, angular diameter)
- (temperature, high temperature)
- (ConceptAgent :95, illumination)
- (ConceptAgent :93, intensity)
- (density, ion density)
- (composition, isotopic composition)
- (#TOP, isotropy)
- (length, jeans length)
- (temperature, kinetic temperature)
- (structure, large scale structure)
- (#TOP, length)
- (momentum, angular momentum)
- (#TOP, limit)
- (magnitude, limiting magnitude)
- (spectrum, line spectrum)
- (width, line width)
- (ConceptAgent :99, luminosity)
- (ConceptAgent :98, magnitude)
- (ConceptAgent :98, mass)

– (density, mass density)
– (#TOP, momentum)
– (#TOP, morphology)
– (resolution, angular resolution)
– (#TOP, parallax)
– (spectrum, power spectrum)
– (ConceptAgent :99, precision)
– (spectrum, radio spectrum)
– (#TOP, radius)
– (ConceptAgent :96, refractive index)
– (ConceptAgent :99, resolution)
– (intensity, residual intensity)
– (#TOP, rotation)
– (radius, schwarzschild radius)
– (#TOP, anisotropy)
– (ConceptAgent :98, sensitivity)
– (resolution, spectral resolution)
– (#TOP, spectrum)
– (structure, stellar structure)
– (ConceptAgent :93, structure)
– (ConceptAgent :96, surface brightness)
– (density, surface density)
– (rotation, synchronous rotation)
– (structure, thermal structure)
– (ConceptAgent :97, temperature)
– (magnitude, apparent magnitude)
– (radius, tidal radius)
– (magnitude, total magnitude)
– (mass, total mass)
– (parallax, trigonometric parallax)
– (ConceptAgent :93, viscosity)
– (magnitude, visual magnitude)
– (ConceptAgent :97, width)
– (emission, xray emission)
– (luminosity, xray luminosity)
– (spectrum, xray spectrum)
– (temperature, brightness temperature)
– (structure, xray structure)
– (distance, zenith distance)
– (#TOP, ConceptAgent :93)
– (ConceptAgent :93, ConceptAgent :95)
– (ConceptAgent :99, ConceptAgent :96)
– (ConceptAgent :96, ConceptAgent :97)
– (#TOP, ConceptAgent :98)

– (ConceptAgent :98, ConceptAgent :99)

B.4 Ontologie obtenue sur Astro après la deuxième modification

– (magnitude, absolute magnitude)
– (limit, chandrasekhar limit)
– (ConceptAgent :93, ConceptAgent :100)
– (composition, chemical composition)
– (density, column density)
– (ConceptAgent :95, composition)
– (spectrum, continuous spectrum)
– (ConceptAgent :96, contrast)
– (ConceptAgent :100, density)
– (ConceptAgent :93, diagram)
– (ConceptAgent :99, diameter)
– (rotation, differential rotation)
– (ConceptAgent :96, albedo)
– (#TOP, distance)
– (temperature, dust temperature)
– (rotation, earth rotation)
– (limit, eddington limit)
– (temperature, effective temperature)
– (ConceptAgent :93, elasticity)
– (density, electron density)
– (#TOP, elevation)
– (ConceptAgent :100, emission)
– (spectrum, emission spectrum)
– (#TOP, altitude)
– (spectrum, energy spectrum)
– (ConceptAgent :100, entropy)
– (width, equivalent width)
– (structure, fine structure)
– (density, flux density)
– (length, focal length)
– (rotation, galactic rotation)
– (density, gas density)
– (mass, gas mass)
– (mass, high mass)
– (diameter, angular diameter)
– (temperature, high temperature)
– (ConceptAgent :95, illumination)
– (ConceptAgent :100, intensity)
– (density, ion density)
– (composition, isotopic composition)

- (ConceptAgent :93, isotropy)
- (length, jeans length)
- (temperature, kinetic temperature)
- (structure, large scale structure)
- (#TOP, length)
- (momentum, angular momentum)
- (#TOP, limit)
- (magnitude, limiting magnitude)
- (spectrum, line spectrum)
- (width, line width)
- (ConceptAgent :99, luminosity)
- (#TOP, magnitude)
- (ConceptAgent :100, mass)
- (density, mass density)
- (ConceptAgent :100, momentum)
- (ConceptAgent :93, morphology)
- (resolution, angular resolution)
- (#TOP, parallax)
- (spectrum, power spectrum)
- (ConceptAgent :99, precision)
- (spectrum, radio spectrum)
- (#TOP, radius)
- (ConceptAgent :96, refractive index)
- (ConceptAgent :99, resolution)
- (intensity, residual intensity)
- (#TOP, rotation)
- (radius, schwarzschild radius)
- (ConceptAgent :93, anisotropy)
- (ConceptAgent :93, sensitivity)
- (resolution, spectral resolution)
- (ConceptAgent :93, spectrum)
- (structure, stellar structure)
- (ConceptAgent :100, structure)
- (ConceptAgent :96, surface brightness)
- (density, surface density)
- (rotation, synchronous rotation)
- (structure, thermal structure)
- (ConceptAgent :100, temperature)
- (magnitude, apparent magnitude)
- (radius, tidal radius)
- (magnitude, total magnitude)
- (mass, total mass)
- (parallax, trigonometric parallax)
- (ConceptAgent :93, viscosity)

- (magnitude, visual magnitude)
- (ConceptAgent :96, width)
- (emission, xray emission)
- (luminosity, xray luminosity)
- (spectrum, xray spectrum)
- (temperature, brightness temperature)
- (structure, xray structure)
- (distance, zenith distance)
- (#TOP, ConceptAgent :93)
- (ConceptAgent :93, ConceptAgent :95)
- (ConceptAgent :99, ConceptAgent :96)
- (#TOP, ConceptAgent :99)

B.5 Ontologie obtenue sur Astro après la troisième modification

- (magnitude, absolute magnitude)
- (limit, chandrasekhar limit)
- (ConceptAgent :93, ConceptAgent :100)
- (#TOP, ConceptAgent :104)
- (ConceptAgent :104, ConceptAgent :105)
- (composition, chemical composition)
- (density, column density)
- (ConceptAgent :95, composition)
- (spectrum, continuous spectrum)
- (ConceptAgent :104, contrast)
- (ConceptAgent :100, density)
- (ConceptAgent :93, diagram)
- (#TOP, diameter)
- (rotation, differential rotation)
- (ConceptAgent :105, albedo)
- (#TOP, distance)
- (temperature, dust temperature)
- (rotation, earth rotation)
- (limit, eddington limit)
- (temperature, effective temperature)
- (ConceptAgent :93, elasticity)
- (density, electron density)
- (#TOP, elevation)
- (ConceptAgent :100, emission)
- (spectrum, emission spectrum)
- (#TOP, altitude)
- (spectrum, energy spectrum)
- (ConceptAgent :100, entropy)
- (width, equivalent width)

- (structure, fine structure)
- (density, flux density)
- (length, focal length)
- (rotation, galactic rotation)
- (density, gas density)
- (mass, gas mass)
- (mass, high mass)
- (diameter, angular diameter)
- (temperature, high temperature)
- (ConceptAgent :95, illumination)
- (ConceptAgent :100, intensity)
- (density, ion density)
- (composition, isotopic composition)
- (ConceptAgent :93, isotropy)
- (length, jeans length)
- (temperature, kinetic temperature)
- (structure, large scale structure)
- (#TOP, length)
- (momentum, angular momentum)
- (#TOP, limit)
- (magnitude, limiting magnitude)
- (spectrum, line spectrum)
- (width, line width)
- (ConceptAgent :104, luminosity)
- (#TOP, magnitude)
- (ConceptAgent :100, mass)
- (density, mass density)
- (ConceptAgent :100, momentum)
- (ConceptAgent :93, morphology)
- (resolution, angular resolution)
- (#TOP, parallax)
- (spectrum, power spectrum)
- (ConceptAgent :105, precision)
- (spectrum, radio spectrum)
- (#TOP, radius)
- (ConceptAgent :104, refractive index)
- (ConceptAgent :104, resolution)
- (intensity, residual intensity)
- (#TOP, rotation)
- (radius, schwarzschild radius)
- (ConceptAgent :93, anisotropy)
- (ConceptAgent :93, sensitivity)
- (resolution, spectral resolution)
- (ConceptAgent :93, spectrum)

- (structure, stellar structure)
- (ConceptAgent :100, structure)
- (ConceptAgent :104, surface brightness)
- (density, surface density)
- (rotation, synchronous rotation)
- (structure, thermal structure)
- (ConceptAgent :100, temperature)
- (magnitude, apparent magnitude)
- (radius, tidal radius)
- (magnitude, total magnitude)
- (mass, total mass)
- (parallax, trigonometric parallax)
- (ConceptAgent :93, viscosity)
- (magnitude, visual magnitude)
- (ConceptAgent :104, width)
- (emission, xray emission)
- (luminosity, xray luminosity)
- (spectrum, xray spectrum)
- (temperature, brightness temperature)
- (structure, xray structure)
- (distance, zenith distance)
- (#TOP, ConceptAgent :93)
- (ConceptAgent :93, ConceptAgent :95)

B.6 Ontologie obtenue sur Astro après la quatrième modification

- (magnitude, absolute magnitude)
- (limit, chandrasekhar limit)
- (ConceptAgent :93, ConceptAgent :100)
- (ConceptAgent :110, ConceptAgent :105)
- (composition, chemical composition)
- (#TOP, ConceptAgent :110)
- (density, column density)
- (ConceptAgent :95, composition)
- (spectrum, continuous spectrum)
- (ConceptAgent :110, contrast)
- (ConceptAgent :100, density)
- (ConceptAgent :93, diagram)
- (#TOP, diameter)
- (rotation, differential rotation)
- (ConceptAgent :105, albedo)
- (#TOP, distance)
- (temperature, dust temperature)
- (rotation, earth rotation)

– (limit, eddington limit)
– (temperature, effective temperature)
– (ConceptAgent :93, elasticity)
– (density, electron density)
– (#TOP, elevation)
– (ConceptAgent :100, emission)
– (spectrum, emission spectrum)
– (#TOP, altitude)
– (spectrum, energy spectrum)
– (ConceptAgent :100, entropy)
– (width, equivalent width)
– (structure, fine structure)
– (density, flux density)
– (length, focal length)
– (rotation, galactic rotation)
– (density, gas density)
– (mass, gas mass)
– (mass, high mass)
– (diameter, angular diameter)
– (temperature, high temperature)
– (ConceptAgent :95, illumination)
– (ConceptAgent :100, intensity)
– (density, ion density)
– (composition, isotopic composition)
– (ConceptAgent :93, isotropy)
– (length, jeans length)
– (temperature, kinetic temperature)
– (structure, large scale structure)
– (#TOP, length)
– (momentum, angular momentum)
– (#TOP, limit)
– (magnitude, limiting magnitude)
– (spectrum, line spectrum)
– (width, line width)
– (ConceptAgent :110, luminosity)
– (#TOP, magnitude)
– (ConceptAgent :100, mass)
– (density, mass density)
– (ConceptAgent :100, momentum)
– (ConceptAgent :93, morphology)
– (resolution, angular resolution)
– (#TOP, parallax)
– (spectrum, power spectrum)
– (ConceptAgent :105, precision)

- (spectrum, radio spectrum)
- (#TOP, radius)
- (ConceptAgent :110, refractive index)
- (ConceptAgent :110, resolution)
- (intensity, residual intensity)
- (#TOP, rotation)
- (radius, schwarzschild radius)
- (ConceptAgent :93, anisotropy)
- (ConceptAgent :93, sensitivity)
- (resolution, spectral resolution)
- (ConceptAgent :93, spectrum)
- (structure, stellar structure)
- (ConceptAgent :100, structure)
- (ConceptAgent :110, surface brightness)
- (density, surface density)
- (rotation, synchronous rotation)
- (structure, thermal structure)
- (ConceptAgent :100, temperature)
- (magnitude, apparent magnitude)
- (radius, tidal radius)
- (magnitude, total magnitude)
- (mass, total mass)
- (parallax, trigonometric parallax)
- (ConceptAgent :93, viscosity)
- (magnitude, visual magnitude)
- (#TOP, width)
- (emission, xray emission)
- (luminosity, xray luminosity)
- (spectrum, xray spectrum)
- (temperature, brightness temperature)
- (structure, xray structure)
- (distance, zenith distance)
- (#TOP, ConceptAgent :93)
- (ConceptAgent :93, ConceptAgent :95)

C Mesure de similitude entre taxonomies

CETTE annexe présente la mesure utilisée pour comparer la similarité entre deux ontologies pour l'expérimentation de la section 5.2. Cette mesure est initialement présentée dans [Maedche et Staab, 2002]. Elle a pour but de comparer deux hiérarchies de concepts $\mathcal{H}_1$ et $\mathcal{H}_2$ constituant respectivement les ontologies $\mathcal{O}_1$ et $\mathcal{O}_2$.

C.1 Définitions

Nous présentons ici une définition simplifiée par rapport à [Maedche et Staab, 2002] d'une ontologie. Nous avons retenu ici uniquement les éléments nécessaires à la définition de la mesure.

Langage des concepts : $\mathcal{A}$ est l'ensemble des concepts C présents dans l'ontologie.

Lexique : $\mathcal{L}$ est l'ensemble des termes des concepts présents dans l'ontologie.

Fonction de référence : $\mathcal{F}$ est la fonction de $2^{\mathcal{L}}$ dans $2^{\mathcal{A}}$ associant l'ensemble d'entrées lexicales $\{L_i\} \subset \mathcal{L}$ à l'ensemble des concepts qu'il décrit. En général, à une entrée lexicale peut correspondre plusieurs concepts, et un concept peut être décrit par plusieurs entrées lexicales. Son inverse est notée $\mathcal{F}^{-1}$.

Hiérarchie des concepts : $\mathcal{H}$ est la relation associant un concept C à son père P. La relation $\mathcal{H}(C, P)$ est vraie si P est le père de C et fausse sinon.

Ontologie : $\mathcal{O}$ est le tuple $(\mathcal{A}, \mathcal{H}, \mathcal{L}, \mathcal{F})$.

C.2 Comparaison de taxonomies

Nous commençons par déterminer comment se comportent deux taxonomies du point de vue de deux concepts identifiés. Plus précisément, nous supposons que nous avons une étiquette $L \in \mathcal{L}_1^C \cap \mathcal{L}_2$ qui se réfère via $\mathcal{F}_1$ et $\mathcal{F}_2$ aux deux concepts C_1 et C_2 appartenant respectivement aux deux taxonomies distinctes $\mathcal{H}_1$ et $\mathcal{H}_2$. L'ensemble de tous les sur-concepts et les sous-concepts de C_1 (respectivement C_2) constitue sa *cotopie sémantique* (SC). Nous considérons la sémantique intensionnelle de C_1 (respectivement C_2) comme correspondant à cette *cotopie sémantique*.

$$SC(C_i, \mathcal{H}) = \{C_j \in \mathcal{A} | \mathcal{H}(C_i, C_j) \vee \mathcal{H}(C_j, C_i)\} \tag{C.1}$$

SC est surchargée pour s'appliquer aussi à des ensembles de concepts.

$$SC(\{C_1, ..., C_n\}, \mathcal{H}) = \bigcup_{i=1...n} SC(C_i, \mathcal{H}) \tag{C.2}$$

Le recouvrement taxonomique (TO) entre $\mathcal{H}_1$ et $\mathcal{H}_2$ du point de vue des concepts désignés par L peut alors être calculés en suivant $\mathcal{F}_1^{-1}$ et $\mathcal{F}_2^{-1}$ pour revenir au lexique commun.

$$TO'(L, \mathcal{O}_1, \mathcal{O}_2) = \frac{|\mathcal{F}_1^{-1}(SC(\mathcal{F}(\{L\}), \mathcal{H}_1) \cap \mathcal{F}_2^{-1}(SC(\mathcal{F}(\{L\}), \mathcal{H}_2))|}{|\mathcal{F}_1^{-1}(SC(\mathcal{F}(\{L\}), \mathcal{H}_1) \cup \mathcal{F}_2^{-1}(SC(\mathcal{F}(\{L\}), \mathcal{H}_2))|} \tag{C.3}$$

En moyennant sur toutes les entrées lexicales, nous pouvons alors calculer la similarité sémantique pour les deux hiérarchies données. De plus, nous devons considérer le cas où une entrée lexicale L est dans $\mathcal{L}_1$, mais pas dans $\mathcal{L}_2$. Alors, la supposition la plus simple est que L est simplement manquant de $\mathcal{L}_2$, et lorsque l'on compare les deux hiérarchies, l'approximation taxonomique la plus optimiste est celle qui recherche le recouvrement maximum pour une appartenance fictive de L à $\mathcal{L}_2$ par :

$$TO''(L, \mathcal{O}_1, \mathcal{O}_2) = \max_{C \in \mathcal{A}_2} \left\{ \frac{|\mathcal{F}_1^{-1}(SC(\mathcal{F}(\{L\}), \mathcal{H}_1) \cap \mathcal{F}_2^{-1}(SC(\mathcal{F}(\{L\}), \mathcal{H}_2))|}{|\mathcal{F}_1^{-1}(SC(\mathcal{F}(\{L\}), \mathcal{H}_1) \cup \mathcal{F}_2^{-1}(SC(\mathcal{F}(\{L\}), \mathcal{H}_2))|} \right\} \tag{C.4}$$

Grâce aux précédentes formules, nous pouvons définir la similarité moyenne $\overline{TO}$ entre les deux taxonomies $(\mathcal{H}_1, \mathcal{H}_2)$ des ontologies $(\mathcal{O}_1, \mathcal{O}_2)$ par les deux formules suivantes :

$$\overline{TO}(\mathcal{O}_1, \mathcal{O}_2) = \frac{1}{|\mathcal{L}_1|} \sum_{L \in \mathcal{L}_1} TO(L, \mathcal{O}_1, \mathcal{O}_2) \tag{C.5}$$

$$TO(L, \mathcal{O}_1, \mathcal{O}_2) = \begin{cases} TO'(L, \mathcal{O}_1, \mathcal{O}_2) & si \quad L \in \mathcal{L}_2 \\ TO''(L, \mathcal{O}_1, \mathcal{O}_2) & si \quad L \notin \mathcal{L}_2 \end{cases} \tag{C.6}$$

Bibliographie

François ARLABOSSE, Marie-Pierre GLEIZES et Michel OCCELLO : Méthodes de Conception. *Systèmes Multi-Agents*, 29:137–171, 2004.

Houssem ASSADI : *Construction d'ontologies à partir de textes techniques - Application aux systèmes documentaires*. Thèse de doctorat, Université Paris VI, 1998.

Sue ATKINS, Jeremy CLEAR et Nicholas OSTLER : Corpus design criteria. *Literary & Linguistic Computing*, 7, 1992.

Nathalie AUSSENAC-GILLES et Agnès BUSNEL : Une méthode de construction d'ontologies pour l'industrie de la fabrication de la fibre de verre. Rapport technique IRIT/2002-11-R., IRIT, Avril 2002.

Nathalie AUSSENAC-GILLES et Anne CONDAMINES : Rapport final de l'action spécifique ASSTICCOT. Rapport technique Rapport Interne IRIT/2003-23-R., Octobre 2003.

Nathalie AUSSENAC-GILLES et Anne CONDAMINES : Documents électroniques et constitution de ressources terminologiques ou ontologiques. *Revue I3*, 4(1):75–94, 2004. URL `http://archivesic.ccsd.cnrs.fr/sic_00001016.html`.

Bruno BACHIMONT : *Art et sciences du numérique : ingénierie des connaissances et critique de la raison computationnelle.* Mémoire d'habilitation à diriger des recherches, Université Technologique de Compiègne, Janvier 2004.

Jean-François BAGET, Étienne CANAUD, Jérôme EUZENAT et Mohand SAÏD-HACID : Les langages du web sémantique. *Revue I3, Hors série : Web Sémantique*, 2004.

Jie BAO et Vasant HONAVAR : Collaborative ontology building with wiki@nt. *Proceedings of the Workshop on Evaluation of Ontology-Based Tools (EON2004)*, 2004.

Françoise BAUDE, Denis CAROMEL, Fabrice HUET et Julien VAYSSIERE : Communicating Mobile Active Objects in Java. *Proceedings of HPCN Europe 2000*, 1823:633–643, 2000.

Mustapha BAZIZ : *Indexation conceptuelle guidée par ontologie pour la recherche d'information*. Thèse de doctorat, Université Paul Sabatier, 2005.

Federico BERGENTI, Marie-Pierre GLEIZES, Franco ZAMBONELLI et (EDS) : *Methodologies and Software Engineering for Agent Systems*. Kluwer Publishing, 2004.

Federico BERGENTI, Agostino POGGI, Giovanni RIMASSA et Paola TURCI : Comma : a multi-agent system for corporate memory management. *AAMAS '02 : Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1039–1040, 2002.

Alain BIDAULT, Christine FROIDEVAUX, Hélène GAGLIARDI, François GOASDOUÉ, Chantal REYNAUD, Marie-Christine ROUSSET et Brigitte SAFAR : Construction de médiateurs pour intégrer des sources d'information multiples et hétérogènes : le projet picsel. *Revue I3*, 2002.

Didier BOURIGAULT : *Un analyseur syntaxique opérationnel : SYNTEX*. CNRS & Université de Toulouse-Le Mirail, 2007. Habilitation à diriger des recherches.

Didier BOURIGAULT et Nathalie AUSSENAC-GILLES : Construction d'ontologies à partir de textes. *Actes de la 10ème conférence sur le Traitement Automatique des Langues Naturelles TALN2003*, pages 27–47, juin 2003.

Didier BOURIGAULT, Nathalie AUSSENAC-GILLES et Jean CHARLET : Construction de ressources terminologiques ou ontologiques à partir de textes : un cadre unificateur pour trois études de cas. *Revue d'Intelligence Artificielle (RIA), Numéro spécial sur les techniques informatiques de structuration de terminologies*, 18(1/2004), 2004.

Didier BOURIGAULT et Christian JACQUEMIN : Construction de ressources terminologiques. pages 215–234. Hermès Science, 2000.

Paul BUITELAAR, Philipp CIMIANO et Bernardo MAGNINI : *Ontology Learning From Text : Methods, Evaluation and Applications*. IOS Press, 2005.

Valérie CAMPS, Marie-Pierre GLEIZES et Pierre GLIZE : Une théorie des phénomènes globaux fondée sur des interactions locales. *Systèmes multi-agents – de l'interaction à la socialité – Actes des 6èmes JFIADSMA*, pages 207–220, 1998.

Davy CAPERA, Marie-Pierre GLEIZES et Pierre GLIZE : Mechanism Type Synthesis based on Self-Assembling Agents. *Journal on Applied Artificial Intelligence*, 18(9-10):921–936, 2004.

Cosmin CARABELEA, Olivier BOISSIER et Adina FLOREA : Autonomie dans les systèmes multi-agents : tentative de classification. *Actes des 11èmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, Novembre 2003.

Jean CHARLET : *Développements, résultats et perspectives pour la gestion des connaissances médicales*. Université Pierre et Marie Curie, Février 2003. Habilitation à diriger des recherches.

Djalel CHEFROUR et Françoise ANDRÉ : Développement d'applications en environnements mobiles à l'aide du modèle de composant adaptatif aceel. *Langages et Modèles à Objets (LMO). L'objet*, 9:77–90, 2003.

Anne CONDAMINES : *Sémantique et corpus spécialisés : Constitution de bases de connaissances terminologiques*. CNRS & Université de Toulouse-Le Mirail, 2003. Habilitation à diriger des recherches.

Nicolas de CONDORCET : *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Paris, 1785.

Stephen CRANEFIELD et Martin PURVIS : UML as an Ontology Modelling Language. *Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, July 31 - August 6 1999.

Béatrice DAILLE : *Approche mixte pour l'extraction de terminologie : statistique lexicale et filtres linguistiques*. Thèse de doctorat, Université Paris VII, 1994.

Pierre-Charles DAVID : *Développement de composants Fractal adaptatifs : un langage dédié à l'aspect d'adaptation*. Thèse de doctorat, École des Mines de Nantes, 2005.

Mike DEAN et Guus SCHREIBER : OWL web ontology language reference. W3C recommendation, W3C, February 2004.

John DOMINGUE, Martin DZBOR et Enrico MOTTA : *Handbook on Ontologies*, chapitre 27. Semantic Layering with Magpie. Springer, 2003.

Mark T. ELMORE, Thomas E. POTOK et Frederick T. SHELDON : Dynamic data fusion using an ontology-based software agent system. *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, 2003.

Chantal ENGUEHARD et Laurent PANTERA : Automatic natural acquisition of a terminology. *Journal of Quantitative Linguistics*, pages 27–32, 1995.

David FAURE : *Conception de méthode d'apprentissage symbolique et automatique pour l'acquisition de cadres de sous-catégorisation de verbes et de connaissances sémantiques à partir de textes : le système ASIUM*. Thèse de doctorat, Université Paris XI Orsay, 2000.

Mariano FERNANDEZ-LOPEZ, Asuncion GOMEZ-PEREZ et N. JURISTO : Methonlogy : From ontological art towards ontological engineering. *AAAI Symposium on Ontological Engineering*, 1997.

Fabien GANDON : *Ontology Engineering : a Survey and a Return on Experience*. INRIA, 2002.

Aldo GANGEMI, Nicolas GUARINO, Claudio MASOLO, Alessandro OLTRAMARI et Luc SCHNEIDER : Sweetening ontologies with dolce. *Processdings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, Octobre 2002.

Jean-Pierre GEORGÉ, Marie-Pierre GLEIZES et Pierre GLIZE : Conception de systèmes adaptatifs à fonctionnalité émergente : la théorie AMAS. *Revue d'intelligence artificielle*, 17 (4):591–626, 2003a.

Jean-Pierre GEORGÉ, Gauthier PICARD, Marie-Pierre GLEIZES et Pierre GLIZE : Living Design for Open Computational Systems. *12th IEEE International Workshops on Enabling Technologies, Infrastructure for Collaborative Enterprises*, pages 389–394, Juin 2003b.

Marie-Pierre GLEIZES et Pierre GLIZE : ABROSE : Multi Agent Systems for Adaptive Brokerage . *Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), Toronto, Ontario, Canada, 27/05/02-28/05/02*, mai 2002.

Marie-Pierre GLEIZES, Pierre GLIZE et Jo LINK-PEZET : FORSIC : A Self-Organizing Training System . *6th conference on Information Systems Analysis and Synthesis ISAS 2000 , ORLANDO (USA), 23/07/00-26/07/00*, juillet 2000a.

Marie-Pierre GLEIZES, Jo LINK-PEZET et Pierre GLIZE : An Adaptive Multi-Agent Tool for Electronic Commerce – IEEE Ninth International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 2000). *Second International Workshop on Knowledge Media Networking, Gettysburg, USA*, pages 59–66, 2000b.

Dominic GREENWOOD, Margaret LYELL, Ashok MALLYA et Hiroki SUGURI : The IEEE FIPA Approach to Integrating Software Agents and Web Services. *AAMAS'07*, May 2007.

Thomas R. GRUBER : The role of common ontology in achieving sharable, reusable knowledge bases. *KR'91 : Principles of Knowledge Representation and Reasoning*, 1991.

Nicolas GUARINO et Christopher A. WELTY : *Handbook on Ontologies*, chapitre 8. An Overview of OntoClean. Springer, 2003.

Zellig S. HARRIS : *Mathematical Structures of Language*. John Wiley and Sons, New-York, 1968.

Marti HEARST : Automatic acquisition of hyponyms froms large text corpora. *13th international Conference On Computational Linguistics (COLING)*, pages 539–545, 1992.

Jeff HEFLIN et James HENDLER : Dynamic ontologies on the web. *American Association for Artificial Intelligence Conference*, 2000.

Brian HENDERSON-SELLERS, Paolo GIORGINI et (EDS) : *Agent-Oriented Methodologies*. Idea Group Publishing, 2005.

Nathalie HERNANDEZ : *Ontologies de domaine pour la modélisation du contexte en recherche d'information*. Thèse de doctorat, Université Toulouse III, 2005.

Rinke HOEKSTRA, Joost BREUKER, Marcello Di BELLO et Alexander BOER : The lkif core ontology of basic legal concepts. *Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT)*, June 2007.

Ian HORROCKS : DAML+OIL : a description logic for the semantic web. *IEEE Data Engineering Bulletin*, 25(1):4–9, 2002.

Ian HORROCKS : WonderWeb, D29 Final Report. Rapport technique IST-2001-33052 D29, University of Manchester, Mars 2005. URL `http://wonderweb.semanticweb.org/deliverables/documents/D29.pdf`.

Ian HORROCKS, Dieter FENSEL, Jeen BROEKSTRA, Stefan DECKER, Michael ERDMANN, Carole GOBLE, Frank van HARMELEN, Michel KLEIN, Steffen STAAB, Rudi STUDER et Enrico MOTTA : OIL : The Ontology Inference Layer. Rapport technique IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, septembre 2000. URL `http://www.cs.vu.nl/~dieter/oil/Tr/oil.pdf`. See http ://www.ontoknowledge.org/oil/.

Ivar JACOBSON, Grady BOOCH et James RUMBAUGH : *Le processus unifié de développement logiciel*. Eyrolles, 2000.

Gaya Buddhinath JAYATILLEKE, Lin PADGHAM et Michael WINIKOFF : A model driven component-based development framework for agents. *International Journal of Computer Systems Science & Engineering*, July 2005.

Anneke KLEPPE, Jos WARMER et Wim BAST : *MDA Explained - The Model Driven Architecture : Practice and Promise*. Addison-Wesley, 2003.

Loredana LAERA, Ian BLACOE, Valentina TAMMA, Terry PAYNE, Jérôme EUZENAT et Trevor BENCH-CAPON : Argumentation over Ontology Correspondences in MAS. *AAMAS'07*, May 2007.

Sophie LE MOIGNO, Jean CHARLET, Didier BOURIGAULT et Marie-Christine JAULENT : Terminology extraction from text to build an ontology in surgical intensive care. *Proceedings of the AMIA 2002 annual symposium*, 2002.

Chantal LEMAY, Marie-Claude L'HOMME et Patrick DROUIN : Two methods for extracting "specific" single-word terms from specialized corpora : Experimentation and evaluation. *International Journal of Corpus Linguistics*, pages 227–255, 2005.

Kendall LISTER, Leon STERLING et Kuldar TAVETER : Reconciling Ontological Differences by Assistant Agents. *AAMAS'06*, May 2006.

Alexander MAEDCHE : *Ontology learning for the Semantic Web*. Kluwer Academic Publisher, 2002.

Alexander MAEDCHE et Steffen STAAB : Mining Ontologies from Text. *EKAW 2000*, pages 189–202, 2000a.

Alexander MAEDCHE et Steffen STAAB : Semi-automatic engineering of ontologies from text. *Proceedings of the Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE'2000)*, 2000b.

Alexander MAEDCHE et Steffen STAAB : Measuring Similarity between Ontologies. *EKAW 2002*, pages 251–263, 2002.

Christopher D. MANNING et Hinrich SCHÜTZE : *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

Diana MAYNARD, Milena YANKOVA, Alexandros KOURAKIS et Antonis KOKOSSIS : Ontology-based information extraction for market monitoring and technology watch. *ESWC Workshop : End User Aspects of the Semantic Web*, 2005.

Robert E. MCGRATH, Anand RANGANATHAN, Roy H. CAMPBELL et Denis MICKUNAS : Use of ontologies in pervasive computing environments. Rapport technique UIUCDCS-R-2003-2332 UILU-ENG-2003-1719, University of Illinois, Department of Computer Science, Avril 2003.

Ingrid MEYER et Kristen MACKINTOSH : *L'étirement du sens terminologique : aperçu du phénomène de la déterminologisation*. 2000.

Emmanuel MORIN : Acquisition de patrons lexico-syntaxiques caractéristiques d'une relation sémantique. *TAL 40/1*, pages 143–166, 1999.

James ODELL : Objects and Agents Compared. *Journal of Object Technology*, 1(1):41–53, 2002.

James ODELL, H. Van Dyke PARUNAK et Bernhard BAUER : Extending UML for Agents. *Proceedings of the Agent-Oriented Information Systems (AOIS) Workshop at the 17th National Conference on Artificial Intelligence (AAAI)*, pages 3–17, 2000.

OMG : Software Process Engineering Metamodel Specification. Rapport technique 1.1, formal/05-01-06, Object Management Group, 2005.

Kévin OTTENS : Analyse syntaxique par systèmes multi-agents adaptatifs : apport à la construction d'ontologies. Mémoire de D.E.A., Université Toulouse III, 2004.

Kévin OTTENS et Nathalie AUSSENAC-GILLES : Un algorithme multi-agent de classification pour la construction d'ontologies dynamiques. *EGC 2007*, 2007.

Kévin OTTENS, Nathalie AUSSENAC-GILLES, Marie-Pierre GLEIZES et Pierre GLIZE : Systèmes multi-agents pour l'extraction d'ontologies à partir de textes : revue de questions. *AGENTAL : Agents et Langue, Journée d'étude ATALA sur les Relations entre Systèmes Multi-Agents et Traitement Automatique des Langues*, mars 2004. URL `http://www.limsi.fr/Recherche/agental/Articles/agental\%20-\%20ottens.pdf`.

Kévin OTTENS, Marie-Pierre GLEIZES et Pierre GLIZE : A Multi-Agent System for Building Dynamic Ontologies. *AAMAS 2007*, May 2007.

Kévin OTTENS, Gauthier PICARD et Valérie CAMPS : Transformation de modèles d'agents dans la méthode ADELFE - Des stéréotypes de conception à l'implémentation. *L'Objet, Composants et Systèmes Multi-Agents*, 2006.

H. Van Dyke PARUNAK : "Go to the Ant" : Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69–101, 1997.

H. Van Dyke PARUNAK, Richard ROHWER, Theodore C. BELDING et Sven BRUECKNER : Dynamic decentralized any-time hierarchical clustering. *29th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*, August 2006.

Juan PAVÓN : *INGENIAS : Développement Dirigé par Modèles des Systèmes Multi-Agents*. Universidad Complutense Madrid, 2006. Habilitation à diriger des recherches.

Jennifer PEARSON : *Terms in Context, Studies in Corpus Linguistics*. John Benjamins Publishing Company, Amsterdam, 1998.

Anna PERINI et Angelo SUSI : Automating model transformations in agent-oriented modelling. *AOSE*, pages 151–168, Juillet 2005.

Gauthier PICARD : *Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente*. Thèse de doctorat, Université Toulouse III, 2004.

Gauthier PICARD, Davy CAPERA, Marie-Pierre GLEIZES et Pierre GLIZE : A Sample Application of ADELFE Focusing on Analysis and Design : The Mechanism Design Problem. *Fifth International Workshop on Engineering Societies in the Agents World (ESAW'04), 20-22 October 2004, Toulouse, France*, 3451:231–244, 2005.

Gauthier PICARD et Marie-Pierre GLEIZES : The ADELFE Methodology – Designing Adaptive Cooperative Multi-Agent Systems. *Methodologies and Software Engineering for Agent Systems (Chapter 8)*, pages 157–176, 2004.

Gauthier PICARD et Pierre GLIZE : Model and Analysis of Local Decision Based on Cooperative Self-Organization for Problem Solving . *Multiagent and Grid Systems*, 2(3):253–265, septembre 2006.

Christine RÉGIS, Thomas SONTHEIMER, Marie-Pierre GLEIZES et Pierre GLIZE : STAFF : un système multi-agent adaptatif en prévision de crues . *10ièmes Journées Francophones d'Intelligence Artificielle Distribuée et des Systèmes Multi-Agents, Lille, France, 28/10/02-30/10/02*, pages 87–98, octobre 2002.

Gilbert SAPORTA : *Probabilités Analyse des Données et Statistique*. Technip, 1990.

Ronny SIEBES et Frank van HARMELEN : Ranking agent statements for building evolving ontologies. *Workshop on Meaning Negotation, in conjunction with the Eighteenth National Conference on Artificial Intelligence*, July 2002.

John SINCLAIR : Preliminary recommendations on Corpus Typology. Rapport technique EAG–TCWG–CTYP/P, Expert Advisory Group on Language Engineering Standards (EAGLES), May 1996.

Monique SLODZIAN : *L'émergence d'une terminologie textuelle et le retour du sens*. 2000.

York SURE, Steffen STAAB et Rudi STUDER : *Handbook on Ontologies*, chapitre 6. On-To-Knowledge Methodology (OTKM). Springer, 2003.

Sylvie SZULMAN, Brigitte BIÉBOW et Nathalie AUSSENAC-GILLES : Structuration de terminologies à l'aide d'outils d'analyse de textes avec terminae. volume 43, pages 103–128. Hermès, Paris, 2002.

Patrick SÉGUÉLA : *Construction de modèles de connaissances par analyse linguistique de relations lexicales dans les documents techniques*. Thèse de doctorat, Université Paul-Sabatier - Toulouse III, mars 2001.

Arnaud THIEFAINE, Zahia GUESSOUM, Jean-François PERROT et Gilles BLAIN : Génération de systèmes multi-agents à partir de modèles. *Actes des Journées Francophones sur les Systèmes Multi-Agents*, pages 107–111, 2003.

Eugen WÜSTER : La théorie générale de la terminologie –un domaine interdisciplinaire impliquant la linguistique, la logique, l'ontologie, l'informatique et les sciences des objets. *Actes du colloque international de terminologie*, 1976.

Liste des figures

1.1 Extrait d'une ontologie du domaine de la fabrication de la fibre de verre [Aussenac-Gilles et Busnel, 2002] 9

2.1 Relation entre les systèmes fonctionnellement adéquats et les systèmes à milieu intérieur coopératif 28
2.2 Adaptation par auto-organisation 28
2.3 Les cinq premières définitions de travaux du processus d'ADELFE 31

3.1 Les différents modules d'un agent coopératif et leurs dépendances 39
3.2 Constituants d'un agent coopératif et visibilités associées (A pour attributs, M pour méthodes) 42
3.3 Une chaîne de type MDA pour l'aspect statique d'un agent 47
3.4 Exemple de classe d'agent coopératif obtenue en fin de conception 47
3.5 Etape 1 de la génération de la structure : ajout des classes d'attributs et de méthodes pour un stéréotype 47
3.6 Etape 2 de la génération de la structure : ajout de la classe d'espace de nom pour le stéréotype et de l'attribut self pour l'accès à ses méthodes 48
3.7 Etape 3 de la génération de la structure : ajout de la classe d'espace de nom pour chaque phase de l'agent (ici, action) et de l'attribut self correspondant . . 48
3.8 Etape 4 de la génération de la structure : ajout de la classe de l'agent 48
3.9 Entrées et sorties du compilateur d'agents 49

4.1 Architecture du système Dynamo 55
4.2 Extrait de réseau « Tête-Expansion » 56
4.3 Classification distribuée : étape 1 60
4.4 Classification distribuée : étape 2 61
4.5 Classification distribuée : étape 3 61
4.6 Arbre d'agents concept après stabilisation autonome du système 63
4.7 Arbre d'agents concept après intervention de l'ontologue 63

4.8 Arbre d'agents concept après stabilisation autonome du système sans la règle de couverture en tête . . . 65
4.9 Arbre d'agents concept après activation de la règle de couverture en tête . . . 65
4.10 Branche dégradée . . . 67
4.11 Branche simplifiée . . . 67
4.12 Classification, partitions constituées . . . 68

5.1 Résultats expérimentaux de l'algorithme distribué de classification . . . 75
5.2 Résultats expérimentaux avec les règles de « couverture en tête » . . . 76
5.3 Branche dégradée (bis) . . . 77
5.4 Branche simplifiée (bis) . . . 77
5.5 Résultats expérimentaux du passage d'un arbre binaire à une taxonomie . . . 78
5.6 Première taxonomie après exécution automatique sur le corpus Astro . . . 81
5.7 Stabilisation du système après la première modification (extrait centré sur ConceptAgent :93) . . . 82
5.8 Stabilisation du système après la seconde modification (extrait présentant la branche « mécanique et thermodynamique ») . . . 83
5.9 Stabilisation du système après la troisième modification (extrait présentant la branche « optique ») . . . 84
5.10 Stabilisation du système après la quatrième modification . . . 84

6.1 Architecture incluant l'approche par patrons . . . 88
6.2 Architecture incluant la pondération des relations du réseau tête-expansion . 90

Liste des exemples

1.1 Une phrase en français étiquetée par TreeTagger 16
1.2 Proposition de dépendances syntaxiques d'une phrase en français 17
1.3 Liste partielle des termes extraits par ANA sur un corpus d'acoustique 17
1.4 Format utilisé par Lexiclass pour la classification des individus [Assadi, 1998] 18
1.5 Un patron morpho-syntaxique 19