Graphes RDF et leur Manipulation pour la Gestion de Connaissances.

Mémoire d'Habilitation à Diriger les Recherches

soutenue le Mercredi 5 novembre 2008 par Fabien L. Gandon

devant un jury composé de

- Président : Pierre Bernhard
- Rapporteur : Nathalie Aussenac-Gilles
- Rapporteur : Marie-Laure Mugnier
- Rapporteur : Vincent Quint
- Examinateur : Bertrand Braunschweig
- Examinateur : Amedeo Napoli

INRIA Sophia Antipolis – Méditerranée

Université de Nice – Sophia Antipolis

à Rose,

« Ecoute plus souvent les choses que les êtres,
La voix du feu s'entend
Entends la voix de l'eau
Ecoute dans le vent le buisson en sanglot :
C'est le souffle des ancêtres.
Ceux qui sont morts ne sont jamais partis. »

Birago Diop, *Les contes d'Amadou Koumba*,
Editions Présence Africaine, Dakar, 1961.

Remerciements

à Rose Dieng-Kuntz, ma directrice de recherches depuis 1999.

à Olivier Corby et Alain Giboin mes collègues de recherche dans Edelweiss.

à tous les membres d'Acacia et Edelweiss que j'ai connus.

à Norman Sadeh et son laboratoire qui m'ont accueilli en post-doctorat.

à mes collègues de l'équipe Kewi avec qui je collabore.

à tous les partenaires de nos projets de recherche.

à chacun des membres du jury de cette habilitation.

à la communauté IC et GRACQ.

à l'INRIA et ses services.

à l'université de Nice – Sophia Antipolis.

Sommaire

1. Introduction 11
2. Gestion de connaissances dans des mémoires individuelles et collectives 15
2.1 Représenter des connaissances 16
2.2 Les ontologies informatiques 19
2.2.1 De l'Ontologie à l'ontologie 19
2.2.2 L'histoire d'une notion à la recherche d'un nom 20
2.2.3 Aux grands mots les grands remèdes 20
2.2.4 Que met-on dans une ontologie ? 21
2.2.5 Avec les meilleures intensions... 22
2.2.6 Ne pas confondre ontologie et taxinomie 25
2.2.7 Pourquoi avoir séparé ces connaissances des autres ? 26
2.2.8 Rendre l'implicite explicite : quelques applications 26
2.2.9 La vie rêvée des ontologies 27
2.2.10 Ontologie : une notion pleine d'avenir 30
2.2.11 Conclusion sur les ontologies 32
2.3 Mémoires individualisées 33
2.3.1 Les portefeuilles électroniques 33
2.3.2 Mémoire d'expériences 36
2.3.3 Mémoire de cours et enseignement assisté par ordinateur 37
2.3.4 Mémoire d'un dossier patient 39
2.3.5 De l'importance des mémoires individuelles 40
2.4 Mémoires de communautés 41
2.4.1 Mémoires organisationnelles 43
2.4.2 Mémoires de communautés ouvertes 48
2.5 Conclusion et discussion 56
2.6 Perspectives : ISICIL 57
3. Des graphes de représentation 59
3.1 Les graphes du web et la représentation de connaissances en ligne 60
3.2 Intra-web Sémantique et web sémantiques communautaires 65
3.3 Formalismes du web orientés graphes et leurs manipulations 67
3.3.1 Raisonneurs sur le web sémantique 67
3.3.2 Opérationnaliser RDF(S) comme des Graphes Conceptuels 71
3.3.3 Etendre les modèles RDF(S) en s'inspirant des graphes conceptuels 75
3.3.4 Factoriser les modèles de graphes et leurs manipulations 81

3.4 Conclusion 91
3.5 Perspectives 92
4. Graphes comme espaces métriques 97
4.1 Notions de proximité conceptuelle et distances sémantiques 98
4.1.1 Approches basées purement sur des structures ontologiques et notamment sur les liens hiérarchiques 100
4.1.2 Approches basées sur la hiérarchie augmentée par des informations extérieures 101
4.2 Distance et bases de connaissances distribuées 102
4.3 Distance et projection de graphes approchée 104
4.4 Distance et ultra-métrique de clustering 105
4.5 Les distances à l'état sauvage 114
4.6 Distance de cooccurrence et contexte en extension 116
4.7 Distance de signatures et contexte en intension 118
4.8 Comparaisons d'espaces métriques : LSA vs. ontologie 121
4.9 Conclusion 125
4.10 Perspectives 126
5. Structures de Graphes pour la gestion de la distribution 129
5.1 RDF et les graphes distribués 130
5.2 Architectures de distribution 132
5.3 Index d'arcs 134
5.3.1 Une mémoire distribuée 134
5.3.2 Une architecture logicielle distribuée : le paradigme agent 137
5.3.3 Protocoles d'archivages et de requêtes : interactions sociales artificielles 138
5.4 Index de motifs de graphes 142
5.4.1 Hubs : des services web pairs 143
5.5 Sources externes 151
5.6 Graphes représentant des services et leurs compositions 160
5.6.1 Services web sémantiques 162
5.6.2 Description et identification de services 164
5.6.3 Découverte et invocation de services 165
5.6.4 Composition de services 167
5.6.5 Composition entre les services et la mémoire 170
5.6.6 Discussion 172
5.7 Vers des écosystèmes de l'information 174
5.8 Conclusions 184
5.9 Perspectives 185
6. Graphes RDF et interactions 187

6.1 Formalisation et (Re)présentations 189
6.2 Informations personnelles et vie privée 195
6.2.1 Un exemple de scénario d'interrogation du *e-Wallet* 196
6.2.2 Architecture interne du *e-Wallet* 198
6.2.3 Implantation du *e-Wallet* 199
6.2.4 Synthèse et discussion 203
6.3 Localisation et contexte 205
6.3.1 Recommander des restaurants 207
6.3.2 Filtrer et router des messages 208
6.3.3 Gérer des posters virtuels 209
6.3.4 Cinéma et météorologie 211
6.3.5 Cartographie 212
6.3.6 Organiser une réunion 213
6.3.7 Projeter une présentation 214
6.3.8 Tests d'utilisation de la première version 215
6.3.9 Analyse des traces d'utilisation 216
6.3.10 Evaluation du service InfoBridge 219
6.3.11 Discussion 220
6.4 Substituts d'information 221
6.5 Wiki sémantique : SweetWiki 238
6.5.1 Wikis et Wikis sémantiques 240
6.5.2 Le principe de SweetWiki 242
6.5.3 Un wiki comme plate-forme d'application 244
6.5.4 Du web sémantique pour implémenter une folksonomie 245
6.5.5 Discussion 248
6.6 Web sémantique et dimension sociale 251
6.7 Vers des ontologies à l'état sauvage 256
6.8 Conclusion 260
6.9 Perspectives 261
7. Conclusion 263
8. Références 273
9. Table of figures 293

1. Introduction

Ce mémoire rassemble des extraits de publications de cinq années de recherches dans le laboratoire de m-Commerce de l'Université de Carnegie Mellon, puis dans les équipes Acacia et Edelweiss de l'INRIA de Sophia Antipolis. Mes travaux mobilisent plusieurs sous-domaines de l'informatique mais sont tous liés au domaine de l'ingénierie des connaissances par leurs approches et leurs applications. Ce mémoire rend compte des contributions auxquelles j'ai participé et les organise en cinq chapitres :

- Une introduction à la notion de mémoires telles que nous les étudions, qu'elles soient individualisées ou collectives ;
- Une visite guidée des représentations à base de graphes que nous utilisons et des caractéristiques que nous exploitons ;
- Un zoom sur un aspect particulier de nos représentations : leur utilisation comme des espaces métriques pour permettre de nouvelles formes d'inférences ;
- Un rappel de l'intérêt de ces représentations dans nos scénarios où les ressources sont naturellement distribuées ;
- Un tour d'horizon de nos travaux sur le problème ouvert de concilier des représentations conceptuelles de plus en plus complexes, riches et nombreuses avec le besoin de concevoir des interfaces et des visualisations intelligibles pour les utilisateurs.

Si l'on regarde le schéma résumant les axes de recherche de l'équipe Edelweiss, on peut positionner mon travail essentiellement sur deux directions de recherche : la représentation de connaissances et l'interaction avec les utilisateurs.

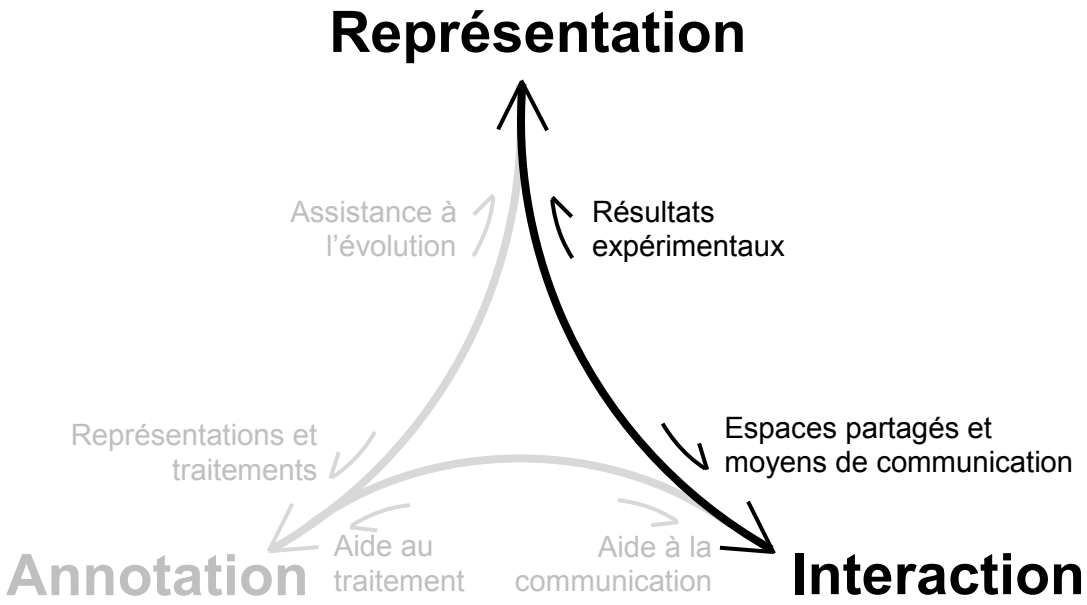


Figure 1. Positionnement sur les axes de recherche de l'équipe Edelweiss

Ce mémoire comporte cinq chapitres chacun se terminant systématiquement par un encart résumant mes contributions personnelles. Ces cinq chapitres abordent dans l'ordre : la notion de mémoire collective ; l'utilisation des graphes dans la représentation de connaissances ; l'utilisation des graphes de connaissances comme espaces métriques pour des inférences ; l'exploitation des structures de graphes pour la gestion de la distribution des ressources d'une mémoire ; l'utilisation des graphes dans la conception d'interactions avec les utilisateurs de nos systèmes.

Le premier chapitre montre que nos scénarios d'usages sont issus de problèmes de capitalisation, de gestion et de diffusion de connaissances. Nous étudions donc des mémoires individualisées (rattachées à un utilisateur donné), organisationnelles (rattachées à une organisation, notamment une entreprise) ou plus généralement communautaires (rattachées à une communauté d'intérêt ou de pratique). Nous donnons dans le premier chapitre des contextes d'application de nos travaux que nous utiliserons comme des scénarios motivants pour les chapitres suivants. Nous y définissons aussi notre positionnement en représentation des connaissances, notamment pour ce qui est de la représentation d'une mémoire à base d'ontologies.

Dans le deuxième chapitre, nous rappelons comment les formalismes à base de graphes peuvent être utilisés pour représenter des connaissances avec un degré variable de formalisation en fonction des besoins identifiés dans les scénarios d'application et des traitements à effectuer notamment pour la mise en place de webs sémantiques. Nous identifierons brièvement les caractéristiques de certains de ces formalismes qui sont

utilisés dans nos travaux et les opportunités d'extensions qu'ils offrent. Nous synthétiserons aussi une initiative en cours pour factoriser la définition des structures mathématiques partagées par ces formalismes et réutiliser l'algorithmique des traitements communs à ces structures.

Dans le troisième chapitre nous expliquons que l'ontologie offre un support à d'autres types de raisonnement que la dérivation logique. Par exemple, la hiérarchie de notions contenue dans une ontologie peut être vue comme un espace métrique permettant de définir des distances pour comparer la proximité sémantique de deux notions. Nous avons mis en œuvre cette idée dans plusieurs scénarios comme l'allocation distribuée d'annotations, la recherche approchée ou le *clustering*. Nous résumons dans ce troisième chapitre diverses utilisations que nous avons faites des distances sémantiques et discutons notre position sur ce domaine. Nous donnons les scénarios d'utilisation et les distances utilisées dans un échantillon représentatif de projets que nous avons menés. Pour nous, cette première série d'expériences a permis de démontrer l'intérêt et le potentiel des distances, et aussi de souligner l'importance du travail restant à faire pour identifier et caractériser les familles de distances existantes et leur adéquation respective aux tâches pour lesquelles les utilisateurs souhaitent être assistés.

Dans le quatrième chapitre, nous rappelons qu'un web sémantique, tel que nous l'utilisons dans nos scénarios, qu'il soit public ou sur l'intranet d'une entreprise, repose généralement sur plusieurs serveurs web qui proposent chacun différentes ontologies et différentes bases d'annotations utilisant ces ontologies pour décrire des ressources. Les scénarios d'usage amènent souvent un utilisateur à formuler des requêtes dont les réponses combinent des éléments d'annotation distribués entre plusieurs de ces serveurs.

Ceci demande alors d'être capable :

(1) d'identifier les serveurs susceptibles d'avoir des éléments de réponse ;
(2) d'interroger des serveurs distants sur les éléments qu'ils connaissent sans surcharger le réseau ;
(3) de décomposer la requête et router les sous-requêtes vers les serveurs idoines ;
(4) de recomposer les résultats à partir des réponses partielles.

Nous avons, avec le web sémantique, les briques de base d'une architecture distribuée. Le quatrième chapitre résume un certain nombre d'approches que nous avons proposées pour tenir compte de la distribution et gérer des ressources distribuées dans les webs sémantiques que nous concevons.

Les ontologies et les représentations de connaissances sont souvent dans le cœur technique de nos architectures, notamment lorsqu'elles utilisent des représentations formelles. Pour interagir avec le web sémantique et ses applications, le cinquième chapitre rappelle que nous avons besoin d'interfaces qui les rendent intelligibles pour les utilisateurs finaux. Dans nos systèmes d'inférences, des éléments de connaissances sont manipulés et combinés, et même si les éléments de départ étaient intelligibles, l'intelligibilité des résultats, elle, n'est pas préservée par ces transformations.

Actuellement, et dans le meilleur des cas, les concepteurs d'interfaces mettent en œuvre des transformations *ad hoc* des structures de données internes en représentations d'interface en oubliant souvent les capacités de raisonnement que pourraient fournir ces représentations pour construire de telles interfaces. Dans le pire des cas, et encore trop souvent, les structures de représentation normalement internes sont directement mises à nu dans des *widgets* sans que cela soit justifié et, au lieu d'assister l'interaction, ces représentations alourdissent les interfaces.

Puisqu'elles reçoivent les contributions d'un monde ouvert, les interfaces du web sémantique devront être, au moins en partie, générées dynamiquement et rendues pour chaque structure devant rentrer en contact avec les utilisateurs. Le cinquième et dernier chapitre souligne cette opportunité croissante d'utiliser des systèmes à base d'ontologies dans l'assistance aux interactions avec les utilisateurs.

Fabien L. Gandon
17 aout 2008

2. Gestion de connaissances dans des mémoires individuelles et collectives

Dans l'équipe Acacia de 1992 à 2004 puis dans l'équipe Edelweiss, les scénarios d'usages sont issus de problèmes de capitalisation, de gestion et de diffusion de connaissances. A travers ces scénarios, j'ai été amené à concevoir des représentations et des raisonnements pour des mémoires individualisées (rattachées à un utilisateur donné), organisationnelles (rattachées à une organisation, notamment une entreprise) ou plus généralement communautaires (rattachée à une communauté d'intérêt ou de pratique).

Nous verrons dans cette section certains des contextes d'application de mes travaux et nous les utiliserons comme des scénarios motivants pour les chapitres suivants. Nous définirons aussi notre positionnement en représentation des connaissances, notamment pour ce qui est de la représentation d'une mémoire à base d'ontologies.

2.1 Représenter des connaissances

« même un petit peu de sémantique
peut déjà vous emmener très loin »

Née en 1992, l'équipe de recherche Acacia visait à mettre au point des outils qui aideraient la phase d'acquisition des connaissances lors de la construction d'un système expert, et a évolué vers le développement d'aides méthodologiques et logicielles (*i.e.* modèles, méthodes, outils) pour l'acquisition, la modélisation et la capitalisation des connaissances ; en particulier pour la construction, la gestion et la diffusion de mémoires d'entreprise.

L'équipe a étudié plusieurs types de mémoires comme les mémoires techniques ou mémoires métier, les mémoires de projets ou mémoires d'expériences, les mémoires de veille, etc. [1] L'approche systématique développée au cours des projets est celle d'une mémoire hybride connaissances-documents où la construction de la mémoire d'entreprise repose sur l'exploitation des connaissances sous-jacentes à des documents et leur contexte organisationnel, et sur la gestion des liens entre documents et bases de connaissances.

Rapidement, avec le développement du web, l'équipe en est venue à considérer les problèmes de diffusion des connaissances à travers un serveur de connaissances via un réseau Intranet ou Internet et à considérer le web comme un moyen privilégié pour l'aide à la gestion des connaissances distribuées intra-entreprise ou inter-entreprises. Le modèle d'architecture générique de nos applications est devenu celui d'un serveur web permettant la recherche d'informations dans une mémoire d'entreprise hétérogène, cette recherche étant intelligemment guidée par des modèles de connaissances et des ontologies.

Dès lors nous maintenons deux types de représentations dans nos systèmes : des documents (représentations informelles de connaissances) et des modèles (représentations formelles de connaissances). Pour la représentation des modèles de connaissances ou des ontologies, nous nous appuyons sur des méthodes d'acquisition et de modélisation [1], sur le formalisme des graphes conceptuels de Sowa [2] pour la sémantique et l'opérationnalisation de nos représentations et sur les technologies et standard du web[1] pour les formats d'échange et les interfaces d'accès.

Notre approche allie donc des méthodes de l'intelligence artificielle symbolique pour la représentation et le raisonnement sur des connaissances dans des formalismes graphiques et logiques, et des technologies du web afin proposer un paradigme de mémoires hybrides où les documents et leurs métadonnées sont organisés dans des modèles de connaissances formels.

[1] Voir W3C, http://www.w3.org

L'équipe Edelweiss poursuit ce travail en élargissant les mémoires d'entreprises à des mémoires de communautés virtuelles en ligne de façon générale. L'objectif étant toujours d'assister les échanges d'information et la capitalisation des connaissances mais en considérant des groupes d'utilisateurs se formant spontanément, sans nécessairement des structures organisationnelles prédéfinies et de façon parfois éphémère. Nos objectifs sont de concevoir et maintenir une plateforme de gestion de connaissances dans une communauté, offrant des moyens d'interaction ergonomiques, et exploitant des représentations à base de graphes et d'ontologies.

L'équipe étudie depuis longtemps le formalisme des graphes conceptuels pour la représentation de connaissances sous forme de graphes étiquetés orientés et la conception d'algorithmes d'inférences dans ces graphes. Après avoir pris le virage XML pour la matérialisation des mémoires [3], Olivier Corby en 1999 détecte les similitudes existant entre les formalismes RDF(S) du web sémantique [4] et les graphes conceptuels et commence à développer Corese [5], un prototype de moteur de recherche et une API Java permettant des inférences sur des annotations RDF en traduisant les triplets RDF en des arcs de graphes conceptuels et vice versa. Ce rapprochement entre les graphes conceptuels et RDF(S) permet de transférer des résultats et de proposer si nécessaire, des extensions en reposant sur le cadre théorique des graphes conceptuels largement étudiés, implantés et éprouvés durant ces trente dernières années.

Dans ce cadre d'un web sémantique pour une mémoire collective, les problèmes de recherche qui se posent sont :

- Quelles méthodes et quels outils d'aide à la modélisation pour la conception d'une ontologie et des annotations ?
- Comment assister la mise en place et le maintien du consensus ontologique?
- Comment gérer l'évolution des ontologies et les conséquences sur les autres objets les utilisant (cohérence) ?
- Comment prendre en compte différents points de vue et différents profils utilisateurs ?
- Comment améliorer les formalismes de représentation en équilibrant expressivité et efficacité?
- Comment améliorer le raisonnement pour s'approcher du mode de pensée des utilisateurs ou de tâches assistant ce mode de pensée ?

Ces questions suivent les axes de recherche d'Edelweiss et l'idée de reposer sur une plate-forme de graphes générique pour implanter les différents niveaux de représentation et de manipulation des connaissances.

Parmi les représentations de connaissances, l'ingénierie des connaissances a donné naissance, dans le début des années 90, à un nouvel objet de l'intelligence artificielle symbolique et à un nouvel outil conceptuel pour la modélisation des connaissances : les ontologies.

Cet objet est une théorie logique qui rend compte partiellement mais explicitement : (1) des notions mobilisées par la description d'une réalité (2) des règles contraignant la structure de ces descriptions. Par exemple : les livres sont des documents, les documents ont un titre et un auteur, les auteurs sont des personnes ou des groupes de personnes, les personnes ont un nom, etc. Une ontologie se compose donc d'un système symbolique, d'opérations permises sur ce système (ex. règles de réécriture, règles d'inférence) et d'une interprétation.

Dans le cadre des mémoires collectives, une ontologie sert à capturer les notions qui interviennent dans les scénarios de recherche et de gestion de la connaissance. Elle fournit alors des primitives pour annoter des ressources et décrire des acteurs (groupes et individus). Les modèles à base d'ontologies permettent la conception d'inférences améliorant les mécanismes de gestion et d'exploitation de la mémoire. Des exemples d'inférences classiques sont :

- La généralisation/spécialisation des notions (ex. un livre est un document, donc si je cherche un document, un livre est une réponse valide), permettant d'améliorer le rappel et la flexibilité des méthodes de recherche.
- La définition formelle d'une notion (ex. un directeur est une personne qui dirige un groupe de personnes) permet de détecter cette notion même si elle n'a pas été explicitement capturée (ex. si "M. Santau dirige la société DC4Plus qui est un 'Groupe de personnes'" le système en déduit automatiquement que "M. Santau est un directeur").

Nous allons donc nous arrêter sur cet objet le temps d'une section, puisqu'il va nous suivre tout au long de ce mémoire.

2.2 Les ontologies informatiques

Cette section reprend un article de vulgarisation de la notion d'ontologie [6] que j'avais écrit pour la revue Interstice. Nous y verrons la nature, l'histoire et l'intérêt de cet objet informatique dans les systèmes d'informations.

2.2.1 De l'Ontologie à l'ontologie

Le terme « ontologie », construit à partir des racines grecques *ontos* (ce qui existe, l'existant) et *logos* (le discours, l'étude), est un mot que l'informatique a emprunté à la philosophie au début des années 80 : on le trouve dans l'article de McCarthy [7] et dans le livre de John Sowa [2] avant qu'il ne devienne célèbre avec l'article de Thomas Gruber [8].

En philosophie, l'Ontologie est une branche fondamentale de la Métaphysique, qui s'intéresse à la notion d'existence, aux catégories fondamentales de l'existant et étudie les propriétés les plus générales de l'être. Sa première apparition date de 1606 [9]. Si vous ouvrez un dictionnaire tel que le Petit Larousse Illustré, la définition n'éclairera probablement pas beaucoup votre lanterne quant à l'intérêt d'importer cette notion en informatique : « (1) Etude de l'être en tant qu'être, de l'être en soi (2) Etude de l'existence en général, dans l'existentialisme ».

Pourtant, à y regarder de plus près, nous pourrions, à l'extrême inverse, penser que beaucoup d'ingénieurs en informatique sont des « messieurs Jourdain » de l'ontologie. Par exemple, lorsque pour implanter une application, les ingénieurs en informatique conçoivent un schéma de classes, ils s'interrogent sur les objets que cette application va manipuler, les classes qui les regroupent, les caractéristiques communes à tous les objets de chaque classe, les relations qui peuvent exister entre ces objets, etc. En d'autres termes, ces ingénieurs s'interrogent sur ce qui définit ces classes d'objets, ce qui permet d'identifier qu'un objet appartient à une classe, ce que cette appartenance signifie en termes de contenu ou de manipulations possibles. Bref, ils s'interrogent sur la définition existentielle des classes d'objets mobilisés dans les scénarios de l'application qu'ils développent.

Vu sous cet angle, l'ingénieur qui conçoit ses représentations logicielles n'est-il pas plus proche de l'ontologue qui interroge nos conceptualisations du monde qu'il ne le semblait initialement ? Si la programmation orientée objets présente cette ressemblance avec la notion d'ontologie informatique, c'est qu'elles ont un ancêtre commun : les systèmes de l'intelligence artificielle symbolique. Les débuts de cette branche de l'intelligence artificielle se confondent avec les débuts de l'informatique, car dès ses prémices, l'informatique a perpétué le rêve des concepteurs d'automates de simuler voire dépasser l'intelligence humaine avec des systèmes artificiels.

2.2.2 L'histoire d'une notion à la recherche d'un nom

La branche de l'intelligence artificielle à laquelle nous nous intéressons ici est qualifiée de symbolique parce qu'elle repose sur des représentations formelles des connaissances, sous la forme de symboles que le système peut stocker et manipuler (par exemple, langages et opérations logiques, structures et opérations de graphes). Contrairement à d'autres approches, ces représentations sont à la fois compréhensibles par les humains et manipulables par les systèmes, en appliquant des règles de manipulation définies sur les symboles de ces représentations et dont l'interprétation simule, par exemple, un raisonnement.

Ainsi, dès les années 70, la notion d'ontologie existait, sans être nommée et de façon transversale, dans les différents systèmes de représentation de connaissances : c'est la TBox des logiques de descriptions [10], où l'on décrit les types de termes qui existent dans notre représentation et leurs caractéristiques ; c'est le support des graphes conceptuels [2], où l'on décrit des hiérarchies de multi-héritage entre des types de concepts ou des types de relations ; ce sont enfin les schémas des « Frames » et les classes des langages de représentation par objets.

Il aura pourtant fallu attendre les années 90 pour que le mot « ontologie » soit adopté par toute la communauté et sa définition fait encore couler de l'encre (électronique).

2.2.3 Aux grands mots les grands remèdes

Comble de l'histoire donc, cette notion d'ontologie, qui s'attache tant à la définition précise des concepts que nous manipulons, a longtemps cherché une définition et un nom. Ce retard est probablement largement dû à la nature même de la notion d'ontologie, qui est abstraite.

Pour tenter de définir la notion d'ontologie informatique, il est utile de rappeler que l'Ontologie désigne l'étude des propriétés générales de ce qui existe. En important cette notion en informatique, nous sommes passés de la science (l'Ontologie) à un objet (une ontologie).

Une ontologie informatique est une représentation de propriétés générales de ce qui existe dans un formalisme permettant un traitement rationnel. C'est le résultat d'une formulation exhaustive et rigoureuse de la conceptualisation d'un domaine. Cette conceptualisation est souvent qualifiée de partielle car il est illusoire de croire pouvoir capturer dans un formalisme toute la complexité d'un domaine. Notons aussi que le degré de formalisation d'une ontologie varie avec l'usage qui en est envisagé.

C'est pour cet aspect de description de l'existant et de ses catégories que les ontologies informatiques ont emprunté leur nom à l'Ontologie philosophique. De ce rapprochement vient aussi la possibilité d'adapter des méthodes de la philosophie pour proposer des méthodes d'ingénierie d'ontologies.

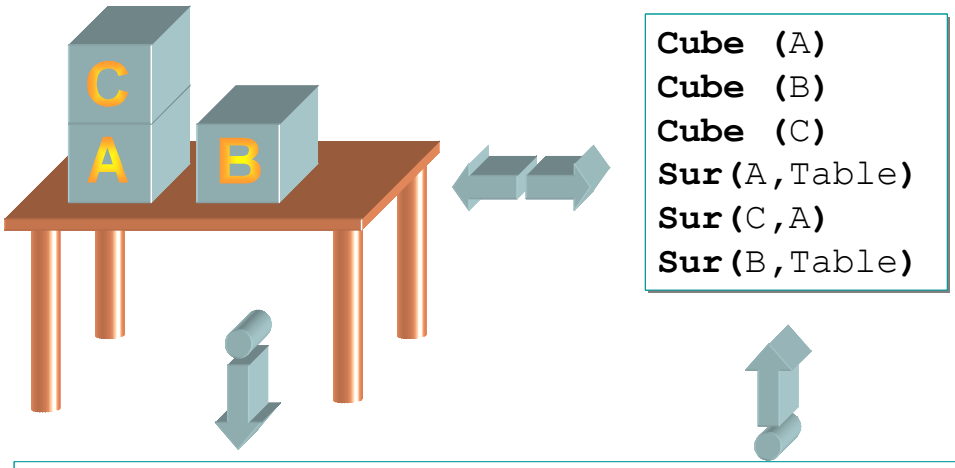


Figure 2. Un exemple schématique d'ontologie : l'incontournable exemple des cubes.

Pour faire simple, considérons l'exemple de la figure ci-dessus. On y voit une certaine « scène du monde ». La description de cette scène demande deux choses : (1) un vocabulaire non ambigu, aussi appelé vocabulaire conceptuel ou ontologie ; (2) une énonciation des faits de la scène, reposant sur l'utilisation du vocabulaire de l'ontologie.

D'un point de vue pratique, une ontologie informatique permet, en particulier grâce aux travaux de l'intelligence artificielle symbolique sur les systèmes à base de connaissances et les moteurs d'inférence, d'implanter des mécanismes de raisonnement déductif, de classification automatique, de recherche d'information, et d'assurer l'interopérabilité entre plusieurs systèmes de ce type.

2.2.4 Que met-on dans une ontologie ?

Une ontologie définit des concepts (principes, idées, catégories d'objets, notions potentiellement abstraites) et des relations. Elle inclut généralement une organisation hiérarchique des concepts pertinents et des relations qui existent entre ces concepts, ainsi que des règles et axiomes qui les contraignent.

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

- une intension : c'est un ensemble de propriétés qualitatives ou fonctionnelles communes aux individus auxquels le concept s'applique, et permettant de définir le concept, par exemple : « `C'est une sous-catégorie de véhicules de transports automobiles, conçus et aménagés pour le transport d'un petit nombre de personnes (7 ou moins) ainsi que d'objets de faible encombrement, et dotés d'au minimum trois roues` » ;

- une extension : un ensemble d'entités qui entrent dans cette catégorie, par exemple : `{ la Twingo de Rose, le Kangoo d'Olivier, la 306 de Fabien, la Clio d'Alain… }`.

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

Figure 3. Trois types de signes en sémiotique.

Dans le cas précédent, nous pouvons donner comme exemples de représentations linguistiques, les termes de « `voiture` », « `automobile` », « `auto` » ou « `véhicule automobile` », ou encore « `tacot` », « `bagnole` », « `tire` » ou « `caisse` ». Nous dissocions donc les concepts et leurs manifestations linguistiques. Un terme n'est pas un concept, et vice-versa. Un terme peut être ambigu, alors qu'un concept n'a qu'un seul sens, une seule définition dans une ontologie donnée. Il faut alors gérer les problèmes de synonymie (un concept dénoté par plusieurs termes) et d'homonymie (un terme dénotant plusieurs concepts).

De la même façon que pour les concepts, l'ontologie définit des relations pouvant exister entre les instances de ces concepts. Prenons une relation volontairement anonyme notée R, nous pouvons aussi lui associer :

- une intension, par exemple : « `R est une relation entre une personne ou un groupe qui a créé un document, et son contenu intellectuel, son arrangement ou sa forme` » ;
- une extension, par exemple : `{ (Hugo, Notre Dame de Paris), (Jean Markale, Le cycle du Graal)… }` ;
- des représentations linguistiques : « `a écrit` », « `auteur de` »...

Les relations possèdent en plus une « signature », une liste spécifiant les types d'instances qu'elles relient, soit pour notre exemple : `(Personne ou groupe, Document)`.

2.2.5 Avec les meilleures intensions...

Dans une ontologie, les intensions sont organisées, structurées et contraintes pour représenter notre conception du monde et de ses contraintes (par exemple, une voiture est forcément un véhicule). L'ontologie capture les intensions et les lois qui les régissent, afin de rendre compte des aspects de la réalité choisis pour leur pertinence dans les scénarios d'application considérés. La représentation des intensions et de l'ontologie peut faire appel à des langages plus ou moins formels (graphes, logiques, langue restreinte), selon l'utilisation envisagée pour l'ontologie. La construction

formelle de l'intension donne une représentation précise et non ambiguë de la manière dont on peut concevoir son sens, ce qui permet sa manipulation logicielle et son utilisation comme une primitive de représentation de connaissances pour décrire et structurer, par exemple, des données, des logiciels, des utilisateurs, des communautés, etc.

Dans l'ontologie, les intensions sont habituellement organisées en taxinomie ou hiérarchie de types. On appelle « subsomption » le fait de placer une catégorie sous une autre ; c'est aussi le lien qui en résulte entre la sous-catégorie subsumée et la catégorie mère. L'importance de l'organisation taxinomique se justifie par le fait que la classification ou identification (le fait de déterminer si quelque chose appartient à une classe) et la catégorisation (le fait d'identifier les catégories existantes) sont des inférences très courantes que nous faisons à longueur de journée. Prenons l'exemple simple d'une conversation entre deux personnes :

« - Tu me conseilles un restaurant ?

- Il y a la pizzeria Torino.

- Merci. »

Dans une conversation aussi banale, la première personne a généralisé sa requête au concept de restaurant, qui représente la catégorie la plus abstraite recouvrant toutes les formes de réponses acceptables. La deuxième a, probablement sans même y prêter attention, utilisé sa taxinomie de concepts pour en déduire qu'une pizzeria est un restaurant, et que par conséquent sa réponse est pertinente. Le fait que cette connaissance taxinomique soit partagée est implicite, puisque la deuxième personne suppose que sa réponse sera comprise sans préciser qu'une pizzeria est un restaurant, et que c'est effectivement le cas. Le recours à des conceptualisations partagées et aux inférences qu'elles permettent est donc au cœur d'activités aussi simples que cet échange d'information. Le fait de rendre explicites les connaissances ontologiques et de s'assurer de leur nature consensuelle est un des problèmes majeurs de l'ingénierie ontologique.

Ainsi, dans un système d'information, le simple ajout de cette connaissance peut permettre d'améliorer considérablement les capacités des machines. Prenons l'exemple, très simplifié, où l'on recherche des livres écrits par un certain « Hugo ». Si votre système d'information se contente de travailler au niveau textuel, avec les mots clefs « Hugo » et « Livre », vous verrez apparaître plusieurs problèmes :

- le bruit : le système ne saura pas faire la différence entre le nom de famille « Hugo », le prénom « Hugo » ou le nom de rue « Hugo » ;
- le silence : le système, s'il rencontre le terme « R-o-m-a-n », ne saura pas qu'il est pertinent pour votre requête, car il cherche le mot « L-i-v-r-e ».

Si maintenant, vous expliquez au système quelques aspects de notre réalité sur les humains (Homme et Femme sont des sous-types d'Humain, qui est lui-même un sous-type d'Être Vivant), les documents (Roman et Nouvelle sont des sous-types de Livre, qui est lui-même un sous-type de Document) et les relations entre les deux, avec leurs signatures (par exemple, il existe une relation Auteur, qui peut s'établir entre un Document et un Humain.)...

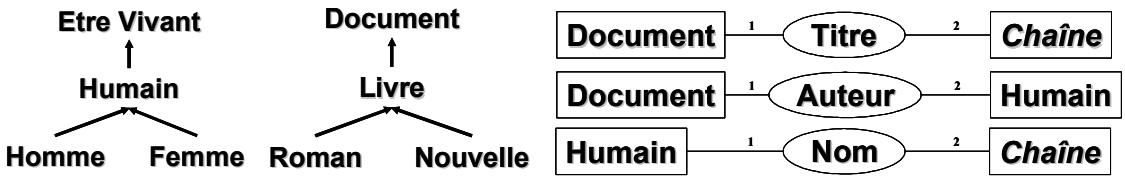


Figure 4. Une micro-ontologie pour notre scénario de recherche de livres.

puis que vous utilisez ce vocabulaire pour décrire la réalité, ici, qu'un homme dont le nom est « Hugo » a écrit un roman intitulé « Notre Dame de Paris »...

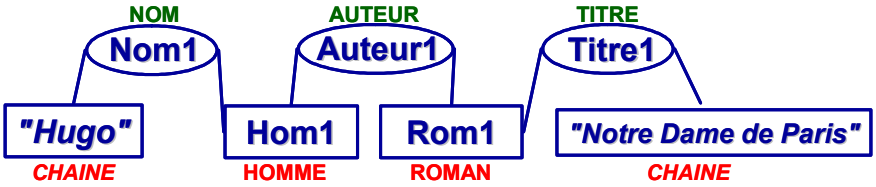


Figure 5. Une annotation utilisant le vocabulaire conceptuel de notre ontologie pour décrire un livre en donnant son titre et son auteur.

alors vous pouvez formuler une requête non ambiguë avec ce même vocabulaire pour rechercher les documents écrits par un certain « Hugo ».

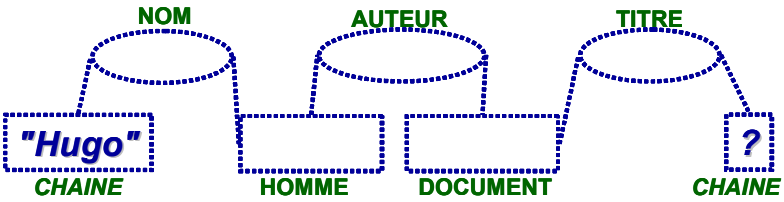


Figure 6. Une requête utilisant le vocabulaire conceptuel de notre ontologie pour rechercher des livres en donnant leur auteur.

Et en utilisant la logique de votre langage, le système peut inférer qu'un roman est un livre, un livre est un document, donc un roman est un document, et que la réponse « Hugo a écrit le roman Notre Dame de Paris » est valide.

2.2.6 Ne pas confondre ontologie et taxinomie

En voyant l'exemple précédent, il ne faut cependant pas confondre ontologie et taxinomie. Les connaissances ontologiques dépassent largement les connaissances taxinomiques. Ainsi, on peut trouver dans une ontologie :

- des connaissances de composition, par exemple : en chimie (catégories d'éléments), en production (catégories de pièces), en médecine (catégories anatomiques), etc. ;
- des définitions complètes, par exemple : une personne est un directeur si et seulement si il existe une organisation dirigée par cette personne ;
- des contraintes d'intégrité, par exemple : un livre édité a un et un seul ISBN, un parent ne peut pas être plus jeune que ses enfants ;
- des fonctions de calcul, par exemple : le rythme cardiaque conseillé pour une personne lors d'un effort cardio-vasculaire est égal à (220 - âge) x 0.65 ;
- des propriétés algébriques, par exemple : la relation « est marié avec » est symétrique, cela signifie que si Thomas est marié avec Stéphanie, alors le système peut aussi déduire que Stéphanie est mariée avec Thomas, et vice-versa ;
- des connaissances par défaut, par exemple : par défaut une voiture a quatre roues ;
- des relations inverses, par exemple : « faire partie de » est l'inverse de « inclure », c'est-à-dire que si une portière fait partie d'une voiture, alors la voiture inclut la portière, et vice-versa ;
- des règles spécifiques au domaine considéré, par exemple : en biologie, pour chaque récepteur qui active une fonction moléculaire, si cette fonction joue un rôle dans le fonctionnement de l'organisme, alors le récepteur joue le même rôle.

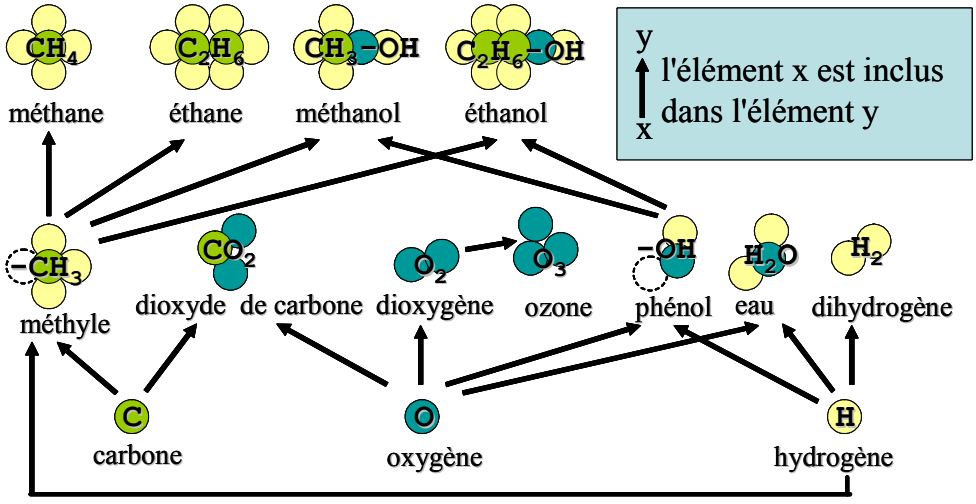


Figure 7. Exemple d'ontologie en chimie : composition de molécules ; une partonomie / méronymie[2] en chimie organique est une connaissance ontologique.

[2] La partonomie ou méronymie est une relation partitive hiérarchisée : une relation de partie à tout souvent nommée « partie de ».

Le contenu d'une ontologie varie aussi avec le type d'ontologie considéré. Une ontologie de domaine contiendra des connaissances propres à un domaine de connaissances (par exemple, l'aviation). Une ontologie de tâche contiendra des connaissances propres à une activité (par exemple, le diagnostic). Une ontologie de haut niveau contiendra des connaissances abstraites très générales, destinées à rassembler d'autres ontologies (par exemple, des notions d'entité, d'événement, de rôle, etc.). Le contenu dépendra aussi du degré de formalisation (langue naturelle, langage restreint, formalisme simple, logiques complexes). Notons en particulier que l'on fait communément la différence entre :

- des ontologies légères : ontologies qui ne comportent typiquement pas ou peu de définitions formelles et qui se focalisent souvent sur la représentation de hiérarchies de types ne nécessitant pas des langages très expressifs (ex : RDFS) ;
- des ontologies lourdes : ontologies qui donnent des définitions formelles précises pour les primitives qu'elles définissent, en utilisant des langages de représentation beaucoup plus expressifs que ceux des ontologies légères (ex : OWL DL).

2.2.7 Pourquoi avoir séparé ces connaissances des autres ?

Les raisons en sont multiples. Tout d'abord, une ontologie permet de factoriser des connaissances. Dans un modèle, les connaissances ontologiques sont des connaissances toujours vraies, quels que soient l'état du système et les descriptions faites. L'ontologie permet de les factoriser et ainsi de ne pas avoir à les répéter pour chaque occurrence. Par exemple, on dira dans l'ontologie qu'une voiture est un véhicule (car c'est toujours vrai), mais on ne lui donnera pas une couleur, car cela change d'une voiture à une autre.

Un autre avantage est de pouvoir réutiliser et échanger des connaissances. Les connaissances ontologiques étant séparées, elles peuvent être réutilisées dans plusieurs applications, et ces réutilisations (totales ou partielles) peuvent constituer la base d'une interopérabilité entre différents systèmes.

Enfin, il est possible de compiler les connaissances et d'optimiser les inférences. Les connaissances ontologiques peuvent faire l'objet de traitements particuliers pour leur donner des structures efficaces, certifier leur cohérence et optimiser les inférences qui les exploitent. Par exemple, une ontologie permet le calcul d'une fermeture transitive comme « si un coupé est une voiture, et une voiture est un véhicule, alors un coupé est un véhicule ».

2.2.8 Rendre l'implicite explicite : quelques applications

Beaucoup d'entités sociales ont à gérer et maintenir des connaissances, que ce soit leur raison d'être (réseaux d'intérêt, équipes de recherche, écoles, etc.) ou un résultat de leur fonctionnement (entreprises, administrations, associations...). De l'agilité de ces entités à détecter, mémoriser, se remémorer et activer leurs connaissances dépend leur agilité à répondre au monde extérieur (innovation des recherches, temps de réponse au marché, qualité des formations, etc.). Dans ce contexte, la connaissance est un capital et un

système d'information performant est un atout primordial. Au royaume de l'information, les connaisseurs sont rois.

Mais les systèmes d'information collectifs sont des applications singulièrement contraintes. Les différences d'expérience, les différentes formations, les différentes cultures, les différents besoins, les différents points de vue, les différentes langues ou jargons, les différents médias et formats, les différents contextes d'utilisation, les différents droits d'accès, etc., posent une diversité de contraintes qui peuvent entraîner un système dans un cercle vertueux ou dans un cercle vicieux, selon leur degré d'importance pour les usages envisagés et la façon dont elles sont respectées par les solutions choisies.

L'introduction d'une ontologie dans un système d'information vise à réduire, voire éliminer, la confusion conceptuelle et terminologique et à tendre vers une compréhension partagée pour améliorer la communication, le partage, l'interopérabilité et le degré de réutilisation possible. Une ontologie informatique offre un cadre unificateur et fournit des « primitives », des éléments de base pour améliorer la communication entre les personnes, entre les personnes et les systèmes, et entre les systèmes.

Intégrer une ontologie à un système d'information permet donc de déclarer formellement un certain nombre de connaissances utilisées pour caractériser les informations gérées par le système et de se baser sur ces caractérisations et la formalisation de leur signification pour automatiser des tâches de traitement de l'information.

Dans un moteur de recherche c'est, par exemple, pouvoir améliorer la recherche d'information dans sa précision, en évitant des ambiguïtés au niveau terminologique (provenant de l'homonymie) ou dans son taux de rappel, en intégrant des notions plus précises ou équivalentes (en utilisant la synonymie, l'hyponymie) ou en déduisant des connaissances implicites (par exemple, des règles d'inférence) ou en relâchant des contraintes trop strictes en cas d'échec de la requête (par généralisation) ou en regroupant des résultats trop nombreux selon leur similarité pour les présenter de façon plus conviviale (regroupement ou *clustering* conceptuel). Le moteur de recherche Corese applique cette approche dans de nombreux domaines comme nous le verrons en sections 2.3 et 2.4.

2.2.9 La vie rêvée des ontologies

Les ontologies sont des objets vivants, et chaque étape de leur cycle de vie pose des problèmes de recherche. Ce cycle de vie rassemble sept activités : détection des besoins, conception, gestion et planification, évolution, diffusion, utilisation, évaluation.

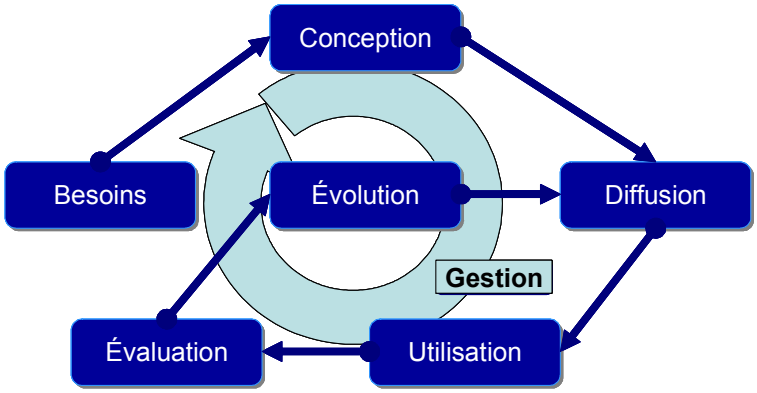


Figure 8. Le cycle de vie d'une ontologie.

Besoins et évaluation : L'activité de détection des besoins, préalable à la conception, et l'activité d'évaluation, lorsqu'une ontologie est utilisée, posent des problèmes méthodologiques de recueil (analyse d'entretiens, questionnaires et sondages, étude de l'ergonomie et des usages) et d'identification (par exemple, modélisation par scénarios). En complément, la phase de détection des besoins demande un état des lieux initial approfondi, car elle ne peut reposer sur des études précédentes ou des retours d'utilisation, comme c'est le cas pour l'évaluation.

Conception et évolution : La phase de conception initiale et la phase d'évolution ont elles aussi en commun un certain nombre de problèmes :

- spécification des solutions (conception participative, maquettage, prototypage) ;
- acquisition des connaissances nécessaires (analyse de textes, traitement automatique de la langue naturelle, plates-formes collaboratives) ;
- conceptualisation et modélisation (design pattern ontologiques, méta-ontologies, entretien avec les experts) ;
- formalisation (méthodes et outils de l'Ontologie formelle, logiques de descriptions et algorithmes de tableaux, analyse formelle de concepts, graphes conceptuels, formalismes du web sémantique RDF/S et OWL) ;
- intégration de ressources existantes (alignement automatique d'ontologies, traduction) ;
- implantation (graphes conceptuels, logiques de description, formalismes objets).

Un autre problème de conception et d'évolution est l'obtention et le maintien d'un consensus sur les choix de représentation et de conceptualisation faits dans l'ontologie. Suivant les usages, ce problème appelle des « collecticiels » et des outils de gestion des points de vue, des terminologies, des langues et des jargons différents.

Notons aussi que l'évolution pose le problème de la maintenance de ce qui repose déjà sur l'ontologie. En effet, une ontologie est à la fois un objet vivant intéressant en soi et un ensemble de « primitives » pour décrire des faits du monde et des algorithmes sur ces faits. Lorsque l'ontologie change, ses changements ont un impact sur tout ce qui a été construit au-dessus. Le maintien de la cohérence dans une ontologie et au-dessus d'une ontologie, l'historique et la gestion des versions, la ré-ingénierie et la propagation des changements après modification, sont des questions de recherche encore largement

ouvertes. La maintenance de l'ontologie soulève donc des problèmes d'intégration technique et des problèmes d'intégration aux usages.

Diffusion : La phase de diffusion s'intéresse au déploiement et à la mise en place de l'ontologie. Les problèmes de cette phase sont fortement contraints par l'architecture des solutions. Dans un contexte d'application web, on reposera sur des technologies idoines. Pour le partage de fichiers, des architectures pair-à-pair ou autres architectures distribuées peuvent être utilisées. Pour l'intégration d'applications, des architectures de services web peuvent être une solution. Dans toutes ces architectures (serveurs web, services web, pair à pair, agents, etc.) la distribution des ressources (données, modèles, applications et utilisateurs) et leur hétérogénéité (syntaxes, sémantiques, protocoles, contextes, etc.) posent des problèmes de recherche sur l'interopérabilité (alignement et médiation) et le passage à l'échelle (larges bases, optimisation d'inférences, propagation de requêtes, syndication de données, composition de services, etc.).

Utilisation : La phase d'utilisation regroupe toutes les activités reposant plus ou moins directement sur la disponibilité de l'ontologie, par exemple, l'annotation de ressources (traitement de la langue, rétro-ingénierie de base de données, etc.), la résolution de requête (algorithme de projection de graphes avec contraintes, recherche approchée en utilisant des distances sémantiques définies sur l'ontologie pour quantifier les approximations faites), la déduction de connaissances et l'aide à la décision (moteurs d'inférence à base de règles), la navigation assistée et les services contextuels (analyse de contexte, identification et composition de services), l'analyse de gros volumes de connaissances (*clustering*, recherche de motifs récurrents, veille).

Toutes ces activités ont en commun de poser le problème de la conception des interactions avec l'utilisateur et de leur ergonomie (interfaces dynamiques, lien sémiotique-sémantique, profils et contextes d'utilisation). Sur ce point, l'ontologie apporte à la fois de nouvelles solutions (par exemple, les inférences exploitent les ontologies pour la génération dynamique d'éléments d'interfaces) et de nouveaux problèmes (par exemple, la complexification des modèles de données engendre des problèmes pour leur représentation et l'interaction avec ces représentations).

Gestion : L'activité permanente de gestion et planification souligne qu'il est important d'avoir un travail de suivi et une politique globale pour détecter ou déclencher, préparer et évaluer les itérations du cycle et s'assurer que l'on reste dans le cercle vertueux des systèmes d'information (où se succèdent contribution, utilisation, création).

2.2.10 Ontologie : une notion pleine d'avenir

La notion d'ontologie, qui précède largement l'utilisation du mot, ne semble pas prête à disparaître. Au contraire, le spectre d'applications et de domaines s'intéressant aux ontologies ne cesse de s'élargir.

Anciennement réservée aux systèmes experts simulant des raisonnements humains dans des domaines spécifiques, l'ontologie se retrouve maintenant dans une large famille de systèmes d'information. Elle est utilisée pour décrire et traiter des ressources multimédia ; asseoir l'interopérabilité d'applications en réseaux ; piloter des traitements automatiques de la langue naturelle ; construire des solutions multilingues et interculturelles ; permettre l'intégration de sources hétérogènes d'information ; décrire des protocoles d'interactions complexes ; vérifier la cohérence de modèles ; permettre les raisonnements temporel et spatial ; faire des approximations logiques ; etc.

Ces utilisations des ontologies se retrouvent dans de nombreux domaines d'application : intégration d'informations géographiques, gestion de ressources humaines, aide à l'analyse en biologie, commerce électronique, enseignement assisté par ordinateur, bibliothèques numériques, échanges commerciaux entre partenaires industriels, suivi médical informatisé, etc.

Un courant particulièrement prometteur pour l'expansion des systèmes à base d'ontologies est celui du web sémantique. Il s'agit d'une extension du web actuel, dans laquelle l'information se voit associée à un sens bien défini, améliorant la capacité des logiciels à traiter l'information disponible sur le web. L'annotation de ces ressources d'information du web repose sur des ontologies elles aussi disponibles et échangées sur le web. Grâce au web sémantique, l'ontologie a trouvé un formalisme standard à l'échelle mondiale et s'intègre dans de plus en plus d'applications web, sans même que les utilisateurs ne le sachent. Cela se fait au profit des logiciels qui, à travers les ontologies et les descriptions qu'elles permettent, peuvent proposer de nouvelles fonctionnalités.

De ce fait, de plus en plus d'ontologies de domaines sont disponibles : ontologie de la génétique, ontologie de la géométrie, ontologie pour les musées, ontologie médicale, ontologie pour l'enseignement, ontologie pour le bâtiment, ontologie de systèmes documentaires, ontologie pour la gestion, ontologie dans le secteur automobile, etc.

À l'heure où l'ontologie se dote d'une ingénierie, cette expansion est loin d'être finie. Parmi ses dernières évolutions, l'ontologie qui s'appliquait essentiellement à des données (documents, images, vidéos) est maintenant utilisée pour décrire des logiciels (par exemple, des services web), leurs caractéristiques fonctionnelles (types d'entrées, types de sorties), et non fonctionnelles (coût, qualité). Elle pourrait ainsi permettre l'identification, l'invocation et la composition dynamique d'applications à l'échelle du web en utilisant des services distribués, ou au sein même d'une application à base de composants.

De même, l'ontologie commençait déjà à être utilisée pour décrire les utilisateurs et s'étend maintenant à la description du contexte d'interaction, pour doter les applications de ce que l'on appelle une conscience du contexte. Cela concerne les préférences de l'utilisateur (langue, goûts, droits, etc.), les caractéristiques du terminal (mobile, vocal, etc.), la situation géographique (à l'étranger, dans une salle avec imprimante, etc.), l'activité en cours (au volant, en présentation, etc.), l'historique d'utilisation.

Enfin, si l'ontologie est actuellement utilisée pour faciliter l'accès à des informations et des applications, on pressent aussi son utilisation dans la description et l'application de règles de sécurité et de confidentialité décrites à de hauts niveaux d'abstraction, permettant de restreindre les accès avec une grande flexibilité. Ainsi, dans un système d'information à base d'ontologies, la confidentialité et ses règles reposent aussi sur la sémantique des ontologies et les inférences qu'elle permet pour contrôler l'accès à l'information et la précision de l'information diffusée.

Nos systèmes d'information sont de plus en plus complexes. Cette complexité, même si elle est artificielle, puisqu'il s'agit de technologie, pose des défis scientifiques ardus qu'il nous faudra relever pour voir l'expansion technologique continuer, par exemple, dans l'assistance des sciences de la vie.

La possibilité de concevoir des systèmes qui se reconfigurent, s'adaptent au contexte, détectent leurs fautes, et même se corrigent dans une certaine mesure, apparaît comme un facteur de passage à l'échelle pour la croissance technologique. Rendre explicites les conceptualisations du monde sur lesquelles se basent les architectures logicielles, les structures de données, les choix de conception, c'est aussi participer à cette évolution des applications et de leur programmation.

Le défi actuel des ontologies est de passer dans les pratiques d'ingénierie logicielle, pour que les conceptualisations actuellement sous-jacentes, en filigrane dans le code, dans ses commentaires ou dans sa documentation dans le meilleur des cas, soient le plus souvent possible rendues explicites et capturées dans des formalismes. Ainsi exposées, elles offriraient des possibilités d'inférence aux systèmes informatiques, de réflexivité sur leurs connaissances et leurs traitements, d'introspection, d'alignement dynamique pour permettre l'interopérabilité, d'évolution dynamique et de description de l'« affordance[3] » des composants logiciels, c'est-à-dire leur capacité à indiquer comment s'interfacer avec eux et les utiliser, afin de rendre leurs interactions plus dynamiques et leur gestion plus automatique, et d'aller vers une informatique plus autonome.

Alors que les applications web s'infiltrent dans tous nos systèmes d'informations, le web est définitivement passé du statut de base documentaire à celui d'une machine virtuelle universelle combinant des ressources de tous horizons, mises à dispositions par les serveurs et les services du web. On peut imaginer un nouveau paradigme de programmation, où les structures de données seraient des représentations basées sur des

[3] L'affordance est la capacité d'un objet à suggérer sa propre utilisation.

ontologies partagées, et où les applications seraient obtenues par composition de services (logiciels personnels, services en ligne, appels à des grilles, etc.).

Après la programmation orientée objet, la « programmation orientée ontologie » ?

2.2.11 Conclusion sur les ontologies

Nous venons donc de voir que l'ontologie est une notion de l'intelligence artificielle symbolique. Dans la suite de ce chapitre, nous allons dresser un panorama de différents types de mémoires que nous avons conçues et utilisées avec des applications dans plusieurs domaines.

Cependant, dans tous les exemples que nous allons voir, il y a une constante : les mémoires utilisent une représentation de connaissances à base d'ontologies *i.e.* un système symbolique dans lequel on distingue notamment :

- Des ontologies : des représentations symboliques constituant la formulation exhaustive et rigoureuse d'une conceptualisation partielle et fournissant un vocabulaire conceptuel.
- Des bases d'annotations : des représentations symboliques constituant des descriptions de faits de la réalité que l'on souhaite mémoriser et que l'on formule avec le vocabulaire conceptuel de l'ontologie.
- Des requêtes : des représentations symboliques constituant des descriptions de faits que l'on cherche à retrouver dans la mémoire et que l'on formule avec le vocabulaire conceptuel de l'ontologie.
- Des règles : des représentations symboliques constituant des manipulations que l'on automatise à chaque fois que des conditions sont vérifiées et que l'on formule avec le vocabulaire conceptuel de l'ontologie.

Ces composants forment la base commune des mémoires que nous construisons.

2.3 Mémoires individualisées

Dans cette section, nous donnerons quatre exemples de mémoires que l'on peut qualifier d'individualisées car elles sont conçues et instanciées pour être associées à un individu même si, dans leur exploitation, plusieurs acteurs peuvent intervenir.

Avec la multiplication des rôles et des moyens d'interaction, les utilisateurs multiplient aussi les traces numériques et les données personnelles ; ils créent de plus en plus de mémoires individualisées qu'il est important de gérer efficacement (notamment pour éviter les pertes et les répétitions) et dans le respect des personnes physiques et morales.

2.3.1 Les portefeuilles électroniques

Au cœur de la gestion des connaissances, les problématiques les plus étudiées sont le captage, la mémorisation et la diffusion des connaissances. De même, les technologies telles que le web sémantique mettent l'accent sur des solutions techniques aux problèmes d'interopérabilité. Parallèlement, le développement des services web, la multiplication des appareils en réseaux et leur déploiement dans une informatique mobile [11], ambiante et ubiquitaire accroissent considérablement les connaissances et les services disponibles en ligne.

Dans les articles [12], [13], [14], [15], nous montrons comment une approche ontologique, peut améliorer la gestion des connaissances privées et de la confidentialité. Comme nous le résumons ici, la modélisation orientée ontologie a permis la conception d'inférences pour contrôler l'accès et ajuster la précision des réponses, voire même mentir lorsque cela s'avère moins dangereux que de refuser de répondre.

Le projet *my*Campus est celui d'un environnement ouvert conciliant les technologies des systèmes multi-agents et du web sémantique pour implanter une plate-forme d'accès mobiles à des services en ligne [13]. Ces services peuvent utiliser les informations contextuelles et personnelles des utilisateurs et l'architecture assure le respect de leur vie privée. La première version a été validée début 2003 sur le campus de Carnegie Mellon où l'environnement était accessible grâce à des agendas électroniques connectés et localisés via le réseau sans-fil de l'université. Une expérience de 3 jours a impliqué 11 utilisateurs choisis pour couvrir un large spectre de profils et libres de se connecter depuis n'importe où sur le campus et d'utiliser les deux premiers services développés : le Concierge (recommandation de restaurants) et le Messager (filtrage et routage des messages). Les utilisateurs devaient en plus maintenir leur agenda à jour de façon à assurer une bonne connaissance du contexte tout au long de l'expérience. L'évaluation a montré une acceptation globale positive et la connaissance du contexte a montré une amélioration systématique des résultats par rapport à des profils d'utilisateur statiques. Par exemple, pour l'agent de filtrage des messages, dans 70% des cas, la meilleure décision ne peut être prise sans connaissance du contexte [16].

Comme le montre la Figure 9, la plate-forme *my*Campus est un environnement multi-agents où les utilisateurs souscrivent à différents types d'*agents-services*. Ces agents

répondent à une attente et aident à différentes activités (ex : établir une réunion, filtrer des messages, préparer un voyage). Pour opérer, chaque agent va avoir besoin de différentes informations sur son propriétaire et éventuellement sur d'autres utilisateurs.

L'accès à une information sur un utilisateur est contrôlé par son *e-Wallet* et les règles de confidentialité qu'il contient. Les agents ne sont pas limités aux ressources personnelles des utilisateurs et accèdent également à des services web publics, à des documents web, des ontologies et des descriptions du web sémantique, *etc.* Dans *my*Campus, ceci inclut l'accès à une variété de ressources et services tels que des services web de localisation, de météorologie, *etc.*

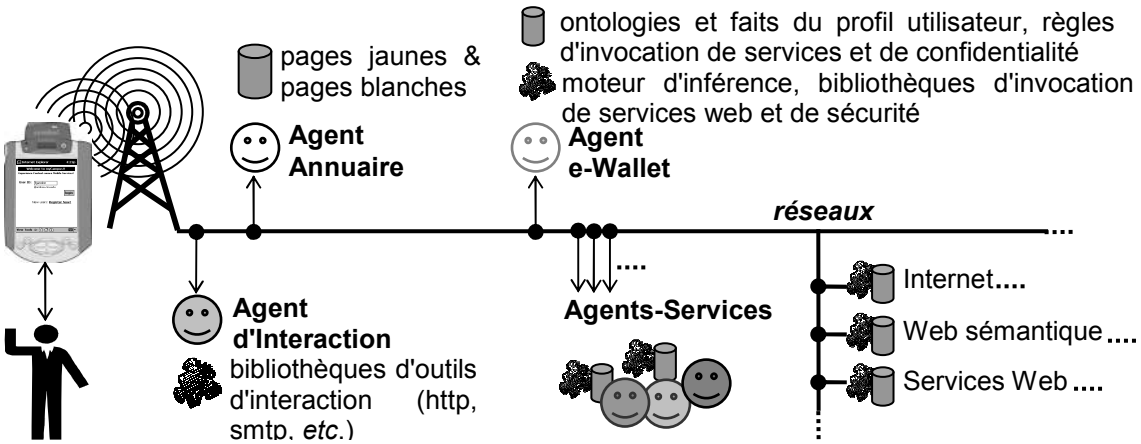


Figure 9. Un environnement ouvert pour l'accès mobile aux services en ligne.

Le *e-Wallet* (copies d'écran en Figure 10) archive les connaissances statiques au sujet de l'utilisateur (ex : nom, téléphone, *etc.*) ; c'est une mémoire individualisée. Il sait aussi obtenir plus de connaissances en faisant appel à une variété de ressources (ex : agenda électronique, localisation sur le réseau sans-fil, *etc.*), chacune représentée par un service web. Cette connaissance est stockée sous forme de règles qui lient différents types de connaissances contextuelles à une ou plusieurs invocations possibles de services. Ces règles permettent au *e-Wallet* d'identifier et d'activer les ressources les plus appropriées pour répondre à une question sur son propriétaire (ex : accéder au calendrier pour découvrir sa disponibilité, utiliser le réseau sans-fil pour le localiser). Des règles de confidentialité, enregistrées dans le *e-Wallet*, sont personnalisées par l'utilisateur et assurent que chaque type d'information est uniquement révélé aux interlocuteurs autorisés à y accéder dans le contexte courant. Elles ajustent aussi l'exactitude ou l'inexactitude des informations fournies selon les préférences de révision spécifiées par l'utilisateur (ex : ne pas révéler la salle où je suis, mais simplement le bâtiment). Les aspects formels du e-Wallet sont donnés en section 6.2.

Le e-Wallet est un premier exemple très représentatif d'une mémoire individualisée : une mémoire attachée à un utilisateur pour la gestion de ses informations privées.

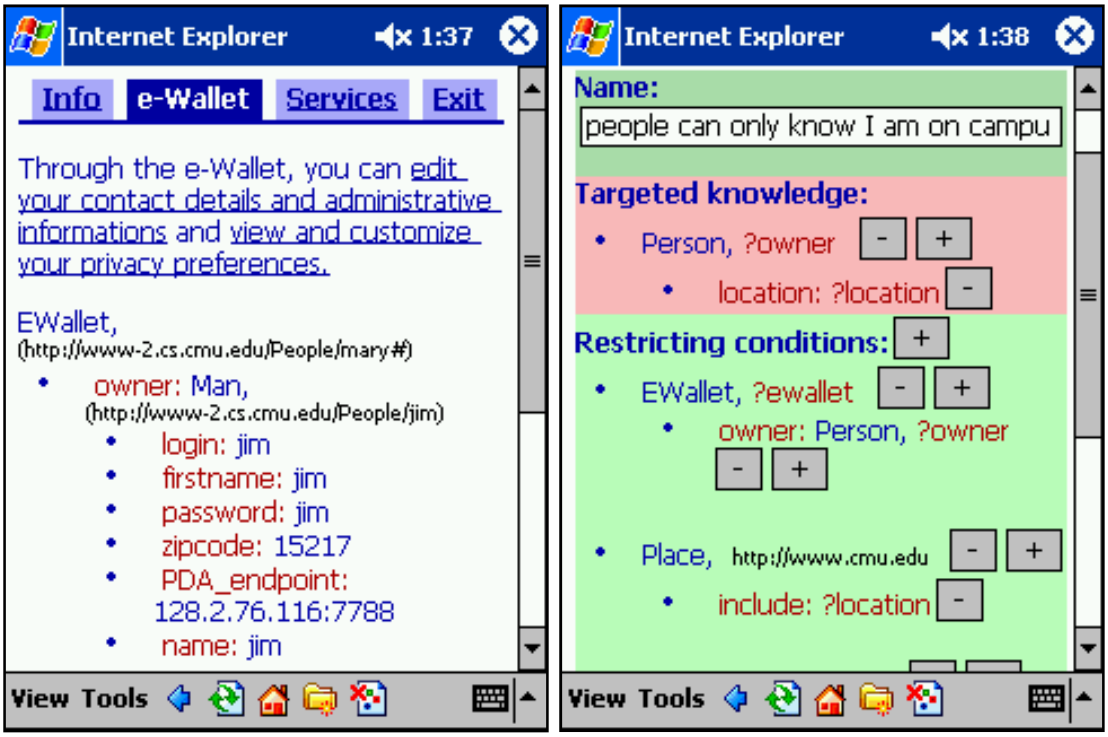


Figure 10. Interfaces d'accès aux connaissances et règles d'accès du *e-Wallet*

Dans *my*Campus l'accès au système se fait à partir d'agendas électroniques reliés au réseau sans-fil de l'université de Carnegie Mellon. Cependant, cette architecture fonctionne tout aussi bien pour des scénarios avec des postes fixes et plus généralement dans des environnements où les utilisateurs se connectent par différents canaux et dispositifs d'accès. Les informations sur le dispositif et le canal peuvent être traitées comme des attributs du contexte et rendues disponibles via le *e-Wallet*.

Nous avons suivi une architecture FIPA [17] incluant en particulier des *Agents annuaires* permettant à un agent/utilisateur à la recherche d'un service d'obtenir une liste d'agents potentiellement fournisseurs de ce service. Des *Agents d'Interaction* sont responsables des échanges avec l'utilisateur. Cela inclut la gestion des sessions de connexion et des interactions avec les autres agents. Même si nous nous concentrerons sur des scénarios impliquant des utilisateurs individuels, cette architecture s'étend à des scénarios où les utilisateurs sont des organismes entiers ayant chacun un ou plusieurs *e-Wallet* ; on retombe alors dans les scénarios de mémoires organisationnelles tels que nous les verrons dans les sections suivantes.

La deuxième version du système *my*Campus a été terminée et démontrée fin 2003 sur des scénarios illustrant l'intérêt du *e-Wallet*. A titre d'exemple, un service de cartographie donne à l'utilisateur une carte des environs en utilisant les réponses de son *e-Wallet* : dans le cas où l'utilisateur est disposé à révéler son code postal, la carte est plus précise que lorsqu'il est seulement disposé à révéler la ville la plus proche. Plusieurs autres services ont été développés et testés [16] comme décrit en section 6.3.

2.3.2 Mémoire d'expériences

Dans le projet MeatAnnot, qui constitue le sujet et les résultats de la thèse de Khaled Khelif [18], nous avions à concevoir un autre type de mémoire individualisée : la mémoire des expériences d'un biologiste, la bibliographie associée à ces expériences et les annotations de cette bibliographie.

Dans une communauté comme celle de la biologie cellulaire, la quantité de publications scientifiques est telle que l'on conçoit aisément l'enjeu que représente l'intégration des connaissances pour améliorer la recherche d'information, des recoupements des résultats, des travaux similaires, des expériences liées, etc. Les communautés de la biologie et de la médecine sont d'ailleurs parmi les pionnières du développement d'ontologies. Cependant, les grands corpus existant font qu'une méthode d'annotation purement manuelle ne passerait pas à l'échelle.

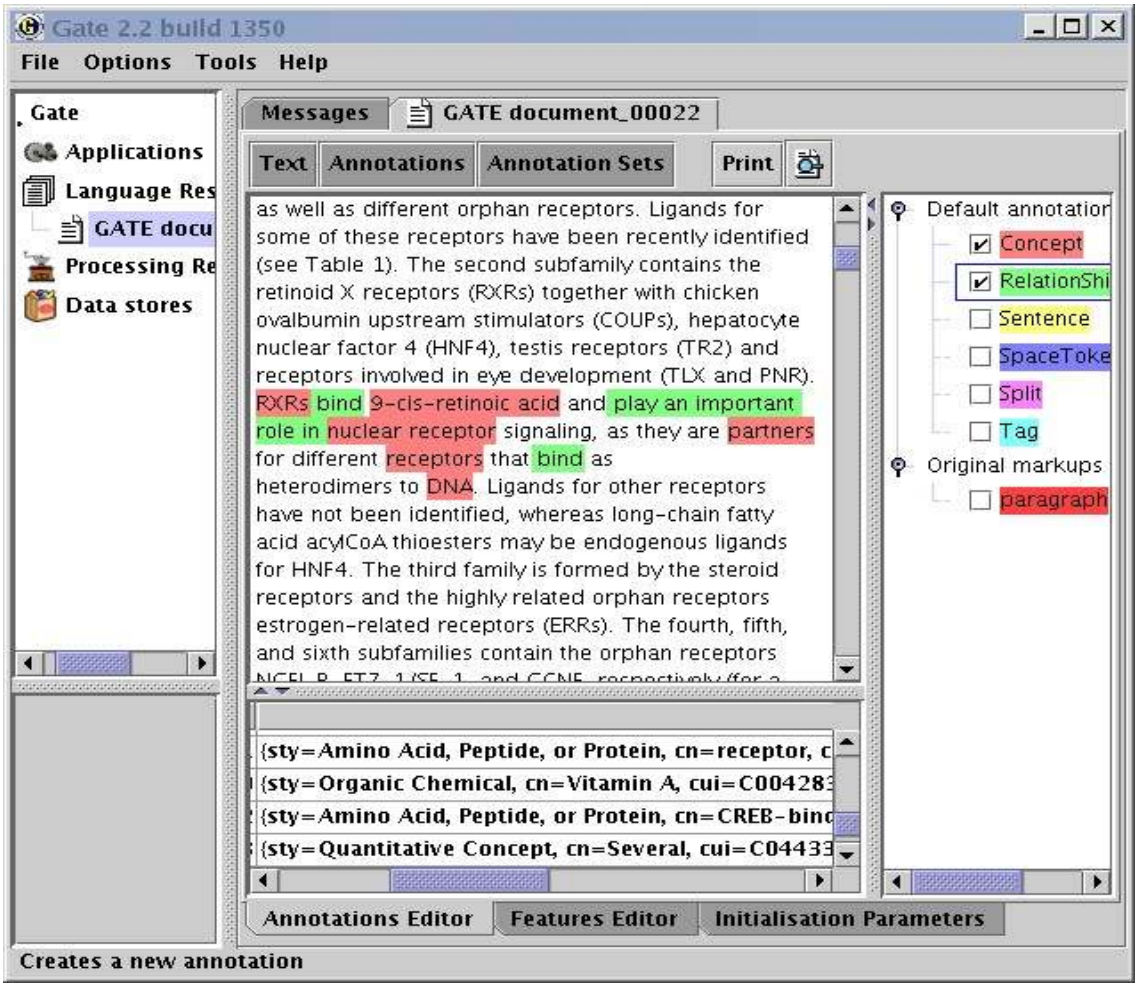


Figure 11. MeatAnnot utilise la plate-forme de traitement de la langue naturelle GATE et des ontologies pour permettre l'annotation automatique d'articles en biologie. [18]

En couplant les ontologies avec des outils d'analyse de la langue naturelle, l'outil MeatAnnot permet d'annoter formellement des textes avec des connaissances qu'ils décrivent. Par exemple, nous pouvons extraire d'un article une relation de causalité entre un gène et une maladie et annoter cet article avec cette connaissance, afin de l'utiliser d'une part dans des raisonnements d'analyses d'expériences et pour la confrontation de résultats, et d'autre part de garder une trace de la source de chaque connaissance.

Dans un premier temps, l'ontologie pilote donc l'analyse de texte en fournissant les termes à chercher (représentations linguistiques associées aux concepts et aux relations), le sens qui peut leur être associé (les intensions), et les structures qui peuvent être extraites (les signatures des relations) ; ces contraintes dirigent et focalisent l'analyse. Dans un deuxième temps, l'ontologie est utilisée, comme dans les autres exemples de ce chapitre, dans des inférences de recherche d'information.

2.3.3 Mémoire de cours et enseignement assisté par ordinateur

Dans le projet QBLS qui faisait le sujet de la thèse de Sylvain Dehors [19], chaque enseignant maintenait une mémoire individuelle où ses cours étaient annotés et réorganisés pour offrir des parcours individualisés aux étudiants.

Chaque étudiant assistant à un cours l'assimile selon un parcours qui lui est propre. Les acquis et les impasses sont différents d'un étudiant à un autre. C'est pourquoi, dans le système QBLS, la structure d'un support de cours est enrichie à l'aide d'une ontologie des éléments pédagogiques (définition, théorème, exemple, question, etc.), afin de générer dynamiquement une interface de travaux dirigés ou de travaux pratiques assurant une grande souplesse de navigation et suivant la progression de l'étudiant dans l'ensemble des notions à acquérir.

Dans cette application, l'ontologie définit et organise essentiellement les rôles des notions pédagogiques. Par exemple, elle définit les notions : « illustration », « définition », « exemple », etc. La sémantique de ces notions est utilisée dans la génération des supports pour distinguer les aspects fondamentaux (à présenter en priorité) des aspects qui ne le sont pas.

La sémantique fixée permet aussi, d'un point de vue ergonomique, de définir des conventions pour les interfaces (par exemple, un code de couleurs) ou de définir formellement des structures récurrentes pour créer des vues abstraites (par exemple, rendre plus synthétique le graphe des parcours des étudiants sur un support pédagogique).

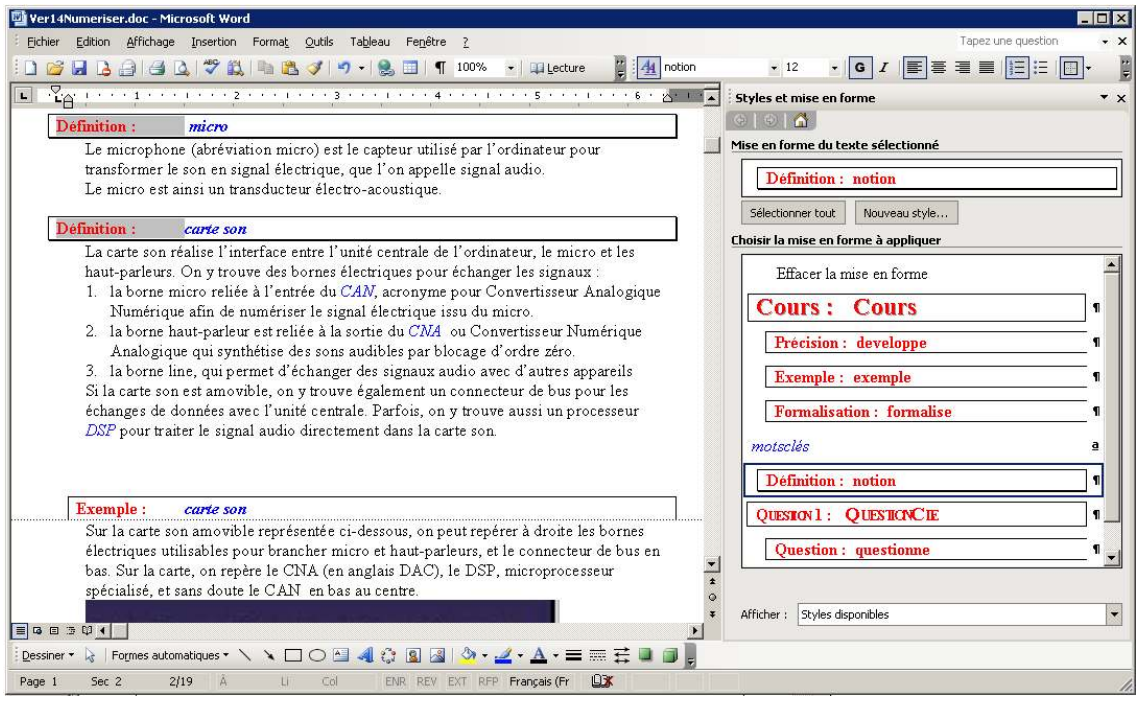


Figure 12. Cours entré sous Microsoft Word (en haut) en utilisant des styles prédéfinis et (en bas) le même cours automatiquement réorganisé pour des séances de TP ou TD

2.3.4 Mémoire d'un dossier patient

Une décision thérapeutique peut impliquer plusieurs experts médicaux et peut aussi nécessiter une connaissance détaillée des antécédents du patient. Le dossier d'un patient est une mémoire individualisée.

Le système Ligne de Vie [20] repose sur une ontologie pour décrire les symptômes, les diagnostics, les options et les choix thérapeutiques, pour intégrer différentes contributions au dossier d'un patient et pour permettre une collaboration non ambiguë des personnes et des systèmes.

L'ontologie est utilisée ici comme référentiel commun dans une activité collaborative.

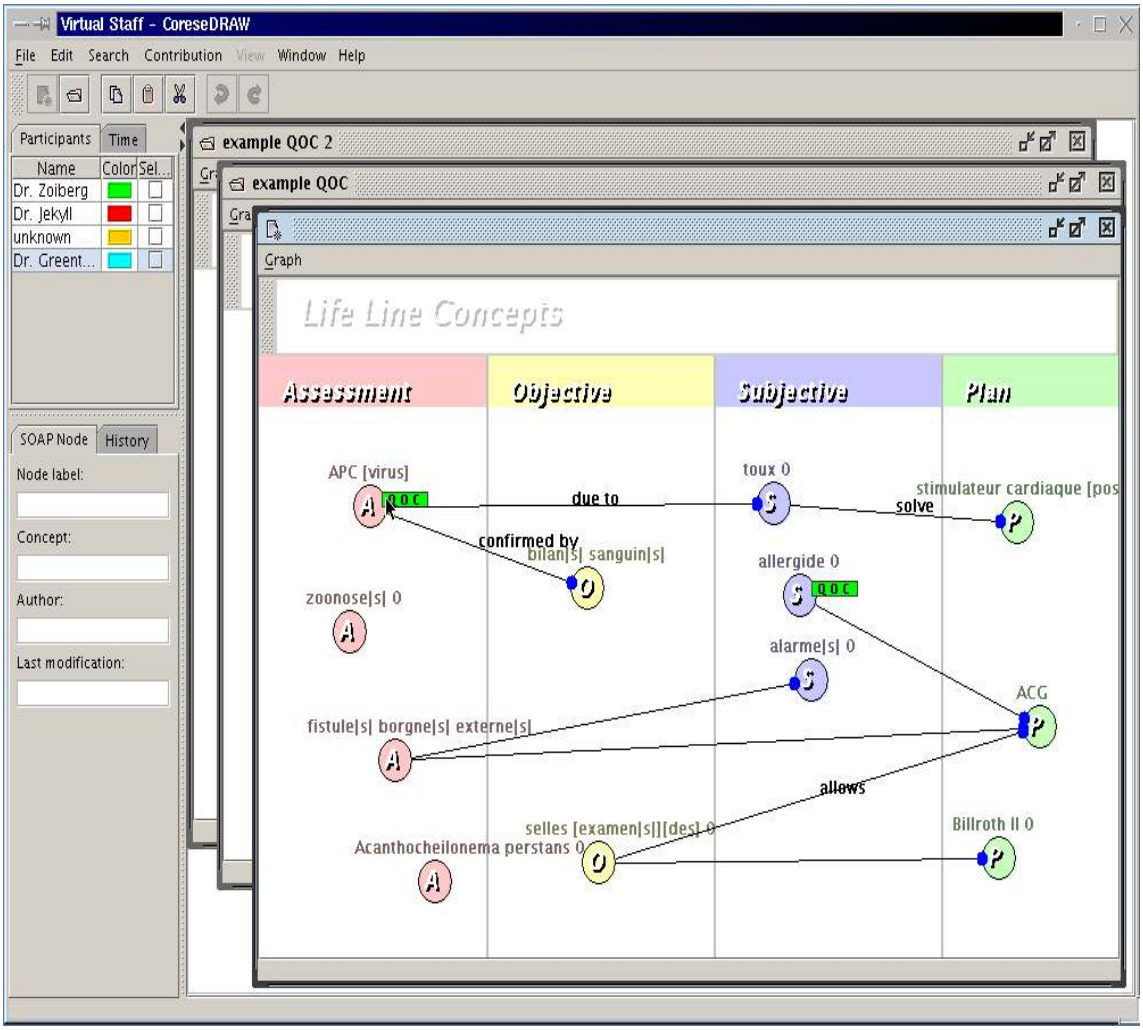


Figure 13. Application au suivi médical et à la coopération autour du dossier d'un patient : Structurer le dossier patient pour aider la coopération médicale entre acteurs d'un réseau de soins

2.3.5 De l'importance des mémoires individuelles

Nous générons et laissons de plus en plus de traces numériques. Les applications génèrent pour nous des multitudes de profils et d'archives ; parfois redondantes et l'on souhaiterait plus d'interopérabilité ; parfois trop facilement rapprochables et l'on souhaiterait plus de contrôle sur leurs interactions.

Représenter et gérer les mémoires individualisées n'a pas seulement un intérêt technique mais aussi un rôle éthique. En réifiant ces mémoires, nous soulignons l'existence de ces connaissances et la nécessité de les gérer, non seulement efficacement et intelligemment mais aussi, dans le respect des individus, de leur vie privée et de la séparation des différents rôles qu'ils jouent dans notre société.

2.4 Mémoires de communautés

La globalisation des échanges d'informations de cette dernière décennie a provoqué une modification des règles des marchés économiques et forcé les entreprises et autres organismes à adapter leur structure et à améliorer leur temps de réaction. Les frontières organisationnelles sont de moins en moins tangibles, et les systèmes d'information soutiennent leur perpétuelle restructuration en devenant de véritables réseaux nerveux assistant la gestion orientée projet, les équipes virtuelles, la veille pour détecter des mouvements aussitôt que possible, et surtout essayer de les prévoir à partir de signaux faibles pour être novateur.

Les organisations et leurs acteurs ont ressenti la nécessité d'une politique de gestion de l'information pour assurer l'acquisition, la diffusion, et l'évolution des connaissances vitales à leur activité. L'intérêt pour les méthodologies et les outils permettant l'acquisition, la capitalisation et la gestion des connaissances organisationnelles s'est donc considérablement accru ces dernières années, comme en témoigne la notion maintenant très répandue de "*knowledge management*" [1].

Ces problématiques, nous les étudions au sein d'Acacia et Edelweiss sous l'appellation de mémoire organisationnelle *i.e.* une représentation persistante, explicite, désincarnée des connaissances et des informations dans une organisation, afin de faciliter leur accès, leur partage et leur réutilisation par les membres adéquats de l'organisation, dans le cadre de leurs tâches individuelles et collectives. L'enjeu de la construction d'un système de gestion de mémoire organisationnelle est de réussir l'intégration cohérente de la connaissance dispersée dans l'organisation afin d'en promouvoir la croissance, la communication et la préservation.

Plus généralement, une mémoire collective peut exister dès lors que des personnes conjuguent leurs efforts au sein d'une communauté pour agir collectivement. La création d'Edelweiss a réorienté nos scénarios vers l'assistance aux communautés de pratique ou d'intérêt en ligne. Ceci demande : un accès aux informations communes pertinentes, un support à la communication et l'interaction, une aide à l'intégration de nouveaux membres, une assistance de ses cycles et processus de fonctionnement, etc. Dans tous ces scénarios la mémoire communautaire cherche à éviter qu'une communauté n'oublie une connaissance utile et à l'activer et la présenter aux membres de la communauté dès qu'elle leur est pertinente.

Une mémoire collective peut prendre plusieurs formes depuis une base documentaire faiblement structurée jusqu'à une base de connaissances modélisées et formalisées. Entre ces deux extrêmes existe un continuum dans lequel se situent les mémoires documentaires à base de connaissances : les documents y forment une masse d'informations qu'il n'est pas nécessaire de formaliser complètement, mais leur indexation repose sur des modèles de connaissances permettant d'améliorer les capacités de navigation, de recherche et d'exploitation de cette mémoire.

Les problèmes de recherche se déclinent alors selon deux axes :

- Matérialisation de la mémoire : pour formaliser les connaissances utiles à la recherche d'informations pertinentes, il faut choisir des formalismes de représentations *i.e.* des systèmes symboliques avec une interprétation unique permettant de capturer le sens des structures d'indexation de la mémoire. Quels formalismes faut-il choisir ou développer pour une mémoire donnée ? Quels modèles développer pour assister la structuration de la mémoire ?
- Processus mémoriels : la gestion de la mémoire requiert des inférences capables d'utiliser et maintenir les structures sémantiques de son indexation pour assister les tâches collectives et individuelles. Quelles inférences aident ces tâches ? Comment simuler ces inférences par des opérations sur les systèmes symboliques utilisés ? Quelles architectures utiliser pour implanter ces solutions ? Que se passe-t-il dans un contexte distribué ?

Pour répondre à ces questions, nous avons fait appel à trois domaines :

- L'ingénierie des connaissances qui étudie et propose des méthodes et des outils pour acquérir, modéliser, formaliser et manipuler des connaissances à travers des systèmes symboliques.
- L'intelligence artificielle distribuée qui propose des paradigmes et des outils permettant de concevoir des sociétés artificielles de composants logiciels faiblement couplés et autonomes, pouvant être déployés sur des réseaux ou dans des environnements virtuels afin d'y accomplir des tâches collectives.
- Le web qui met à notre disposition des normes pour l'intégration et l'échange de connaissances et d'applications.

Dans cette section, nous allons donner quelques exemples de scénarios de mémoires organisationnelles, notamment des mémoires d'entreprises, et quelques scénarios de mémoires pour des communautés en ligne.

2.4.1 Mémoires organisationnelles

Application à la gestion de documents dans le domaine du bâtiment : Le logiciel Aprobatiom permet de guider la recherche de documents lors de la conception d'un nouveau bâtiment, afin d'identifier et de réutiliser des documents traitant de la réalisation de bâtiments ressemblant au nouveau projet. Ce logiciel faisait l'objet d'un projet COLOR avec le CSTB (Centre Scientifique et Technique du Bâtiment).

La mémoire identifiée ici est une mémoire de projet *i.e.* l'organisation souhaite se rappeler des projets de construction précédents pour accélérer le montage d'un nouveau projet.

Dans cette mémoire, l'ontologie est utilisée pour guider l'utilisateur dans l'expression d'une requête (par exemple, en montrant les contraintes des signatures des relations, c'est-à-dire les instances qui peuvent être reliées). L'ontologie sert aussi à caractériser et comparer les différents documents gérés (par exemple, pour vérifier les types en prenant en compte la subsomption, c'est-à-dire les liens hiérarchiques entre catégories).

Figure 14. Premières interfaces de requêtes pour Aprobatiom en 2001

Application à la distribution des connaissances : Dans une entreprise, les connaissances, leurs sources et leurs utilisations sont distribuées. Le système CoMMA a été conçu pour gérer l'annotation distribuée de documents. Il faisait notamment le sujet de ma thèse [21].

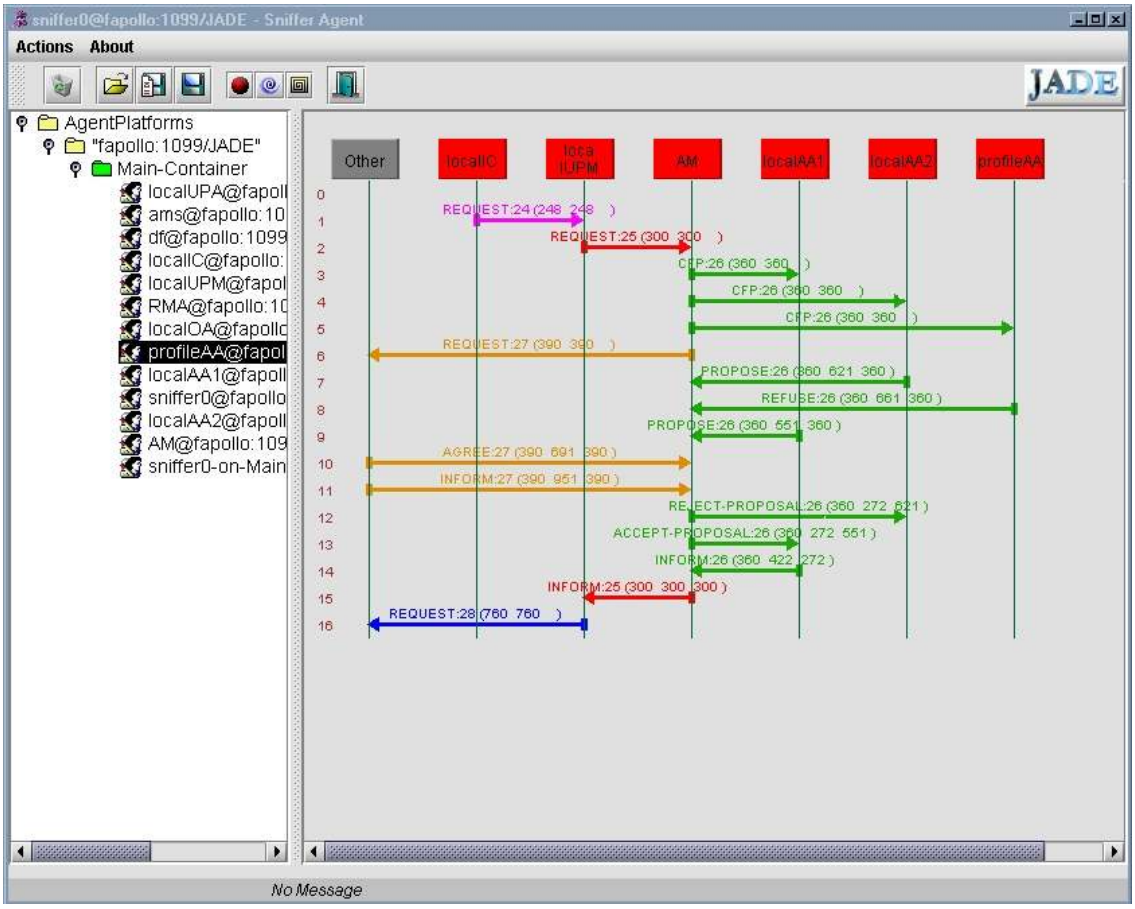


Figure 15. Dans le système CoMMA, négociation entre agents logiciels gérant la distribution des connaissances (ici, des enchères pour décider de la meilleure base d'archivage pour une nouvelle annotation).

Prenons par exemple deux scénarios où l'ontologie est utilisée dans le système CoMMA.

Le premier scénario met en scène un employé dans le cadre d'une activité de veille technologique (mémoire d'entreprise pour la veille), qui identifie et annote un document intéressant pour son entreprise. L'ontologie est ici utilisée pour caractériser ce document, archiver cette annotation avec des annotations similaires pour maintenir la spécialisation des archives, identifier les profils de personnes potentiellement intéressées par ce document et leur envoyer un message de notification. Pour définir formellement la notion de similarité, le graphe de subsomption de l'ontologie est utilisé comme espace métrique fournissant une distance pour évaluer, par exemple, la proximité sémantique de deux annotations (voir chapitre 4).

Le deuxième scénario concerne un nouvel employé qui recherche des informations sur les activités et structures de son entreprise. Sa requête met en jeu différentes sources distribuées et l'ontologie est utilisée pour identifier les sources pertinentes et découper et distribuer la requête (par exemple, comparaison logique entre les contraintes de typage de la requête et les statistiques sur le contenu des bases en prenant en compte la hiérarchie des types et les signatures des relations).

Application à la veille dans le domaine du bâtiment : Aucune organisation n'est isolée, elle vit dans une culture, un pays, une société, un marché, etc. et beaucoup d'informations intéressantes (car se rapportant à l'environnement de l'organisation, aux activités dans son domaine, etc.) sont disponibles sur le web ouvert. Dans le cadre du DEA [22] puis de la thèse de Tuan Dung Cao [23], nous avons considéré le cas d'une mémoire de veilleurs qui doivent surveiller et intégrer dans la mémoire de leur organisation les signaux faibles et mouvements de leur domaine.

La tâche d'annotation peut rapidement devenir fastidieuse si un grand ensemble de documents pertinents est découvert et l'hypothèse de l'annotation manuelle devient alors peu réaliste. Certains sites web ayant une structure plutôt statique qui, même si elle est implicite, fournit des indices structurels, ce travail a exploité cette structure pour automatiser des règles d'extraction. Ceci permet à l'utilisateur de produire automatiquement des annotations à partir de la teneur des ressources et de les intégrer instantanément au système d'information de l'entreprise.

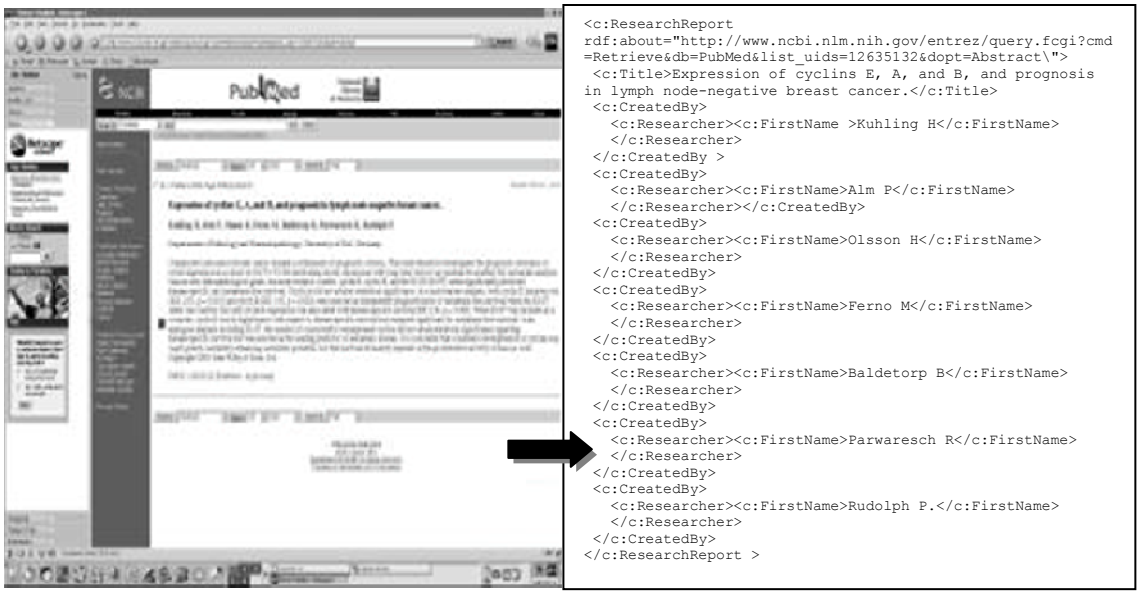


Figure 16. Un exemple d'extraction d'annotation de PubMed [22].

Application à la gestion du cycle de vie d'un produit : Le projet européen SevenPro [24] [25] [26] vise à développer des outils et des technologies afin d'aider un ingénieur à concevoir de nouveaux objets en lui fournissant des rendus 3D de l'objet à concevoir ou d'objets similaires conçus précédemment.

L'ingénieur utilise ces vues pour accéder aux informations sur ces objets quelque soit la provenance de ces informations. Le scénario inclut notamment l'extraction d'annotations à partir de documents textuels et de fichiers CADs (conception et dessin industriel assistés par ordinateur).

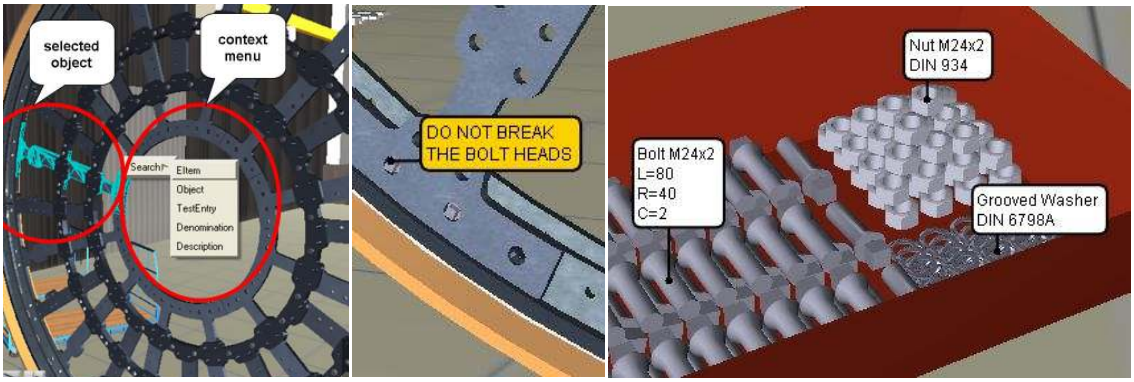


Figure 17. Visualisation 3D enrichie par des requêtes SPARQL dans SevenPro pour annoter des scènes explicatives sur le montage d'un produit

Le cas d'usage de SevenPro visualisé en Figure 17 est celui d'une entreprise de fabrication de moulins ayant plusieurs sources d'information sur chaque produit : bases de CAO, documents textuels techniques, catalogues numériques, base de normes publiques, etc. L'un des objectifs de SevenPro est d'obtenir des annotations sur ces documents à partir de la fouille de la structure des documents de CAO, de la fouille de textes et de techniques d'apprentissage.

Comme nous le verrons dans en section 5.4, le système gère la distribution des requêtes entre les différentes bases archivant ces annotations et les réponses peuvent être intégrées à des visualisations 3D pour l'aide à la conception, les tests, l'aide au montage, la formation, etc.

Conclusion sur les mémoires organisationnelles : la gestion des connaissances (*Knowledge Management*) et les systèmes d'information internes en entreprise ont maintenant une longue histoire [1]. Les technologies internet dans un premier temps, et le web dans un deuxième temps ont complètement réorganisé ces systèmes en intranets puis en intrawebs. Avec l'arrivée des formalismes du web sémantique, se pose maintenant la question de l'intégration de cette nouvelle évolution pour aller vers des intraweb sémantiques.

Le web ouvert et les webs internes ont souvent fonctionné par fertilisation croisée. Les dernières évolutions telles que les web services et web services sémantiques, le web sémantique et les approches appelées 'web 2.0' montrent des avantages que les organisations aimeraient intégrer et concilier avec leurs processus métiers.

A titre d'exemple la flexibilité et la dynamique des applications web 2.0 est vue par certains de nos partenaires industriels comme une approche pouvant améliorer leurs méthodes et outils de veille. Il y a donc un champ d'application énorme et d'importants défis d'intégration à relever pour que ces mémoires organisationnelles passent elles aussi à leur version suivante. Certains parlent déjà de l'entreprise 2.0.

2.4.2 Mémoires de communautés ouvertes

Application à la gestion des compétences : Cet exemple s'applique à la communauté de la Telecom Valley de Sophia Antipolis. Dans cette Telecom Valley, qui compte environ 70 membres, représentant plus de 10 000 emplois, la variété des compétences et des échanges possibles est non seulement élevée mais aussi dynamique et hautement spécialisée.

Dans ce contexte, comprendre le paysage industriel pour les institutionnels régionaux ou trouver un partenaire pour les acteurs locaux sont de véritables problèmes d'intercompréhension et d'intégration d'informations.

Le portail KmP [27] permet la visualisation et l'analyse des échanges entre les membres de la Telecom Valley. Ce portail public, reposant sur une ontologie des compétences, permet à chaque acteur de décrire ses atouts et ses besoins afin d'améliorer la visibilité du parc et de faciliter de nouveaux partenariats.

Là aussi, l'ontologie est utilisée pour piloter la génération d'interfaces, par exemple en fournissant des contraintes de typage pour la génération des choix multiples dans un formulaire. À partir des interfaces générées, les échanges peuvent être analysés, et le système peut trouver des partenaires possibles (inférences de types dans la recherche d'information), même si le partenaire parfait n'existe pas (utilisation de l'ontologie comme espace métrique pour la recherche approchée).

Enfin, l'état du parc, ses points forts et ses niches, peuvent être visualisés en temps réel (utilisation de l'ontologie dans la définition des similarités dans un algorithme de *clustering*).

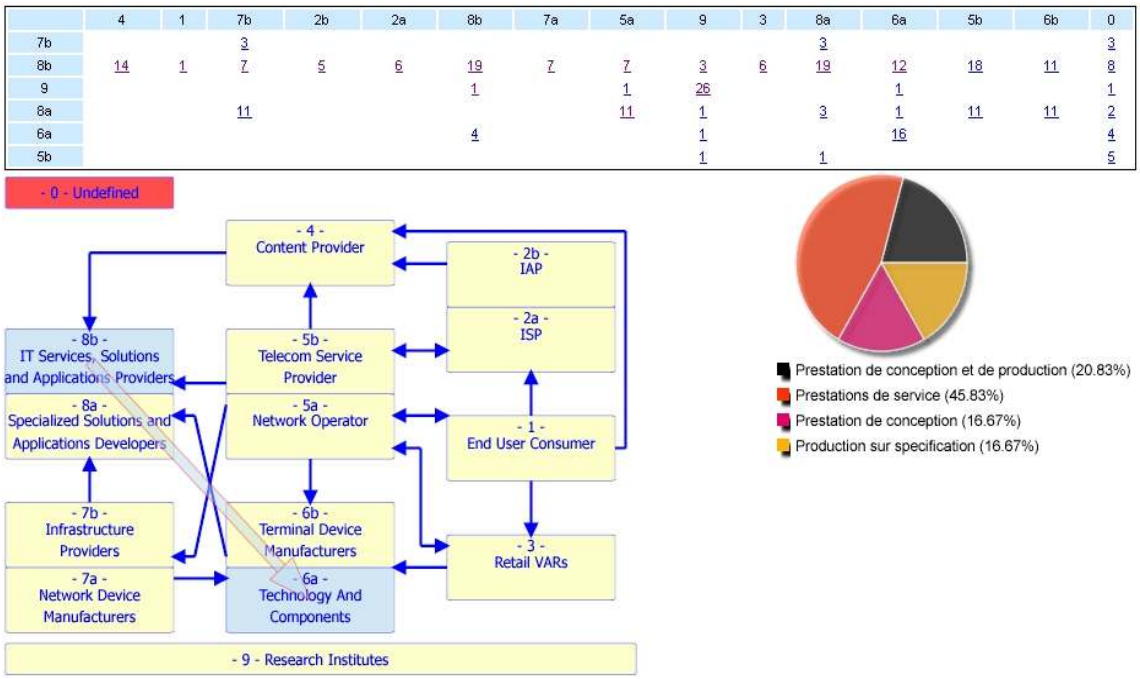


Figure 18. Dans le portail KmP, visualisation de l'analyse des échanges entre les membres de la Telecom Valley.

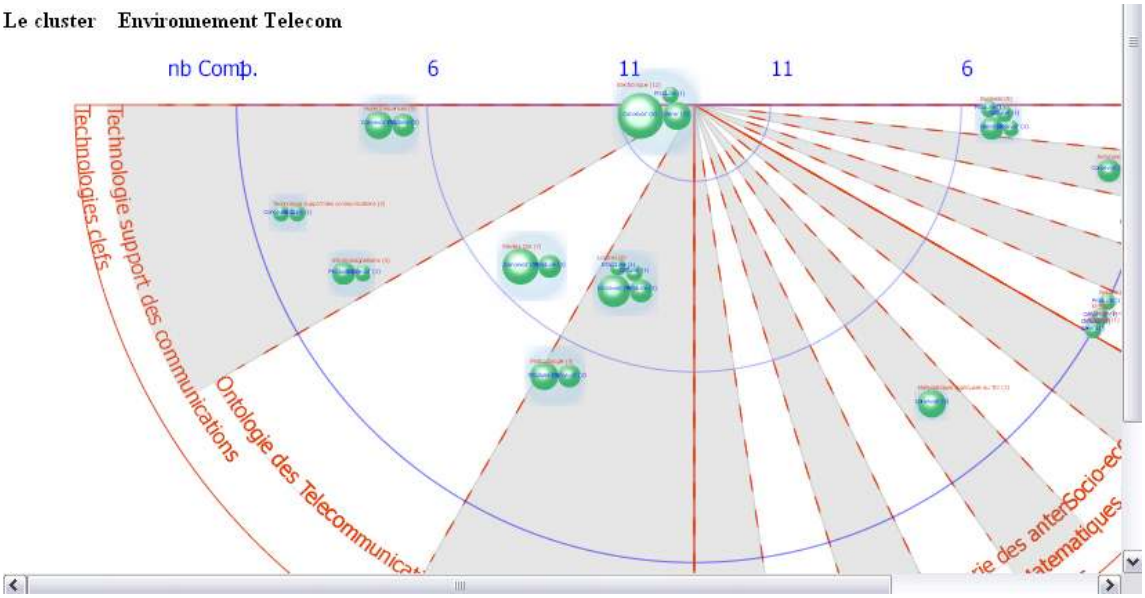


Figure 19. Visualisation des groupes de compétences sur la Telecom Valley.

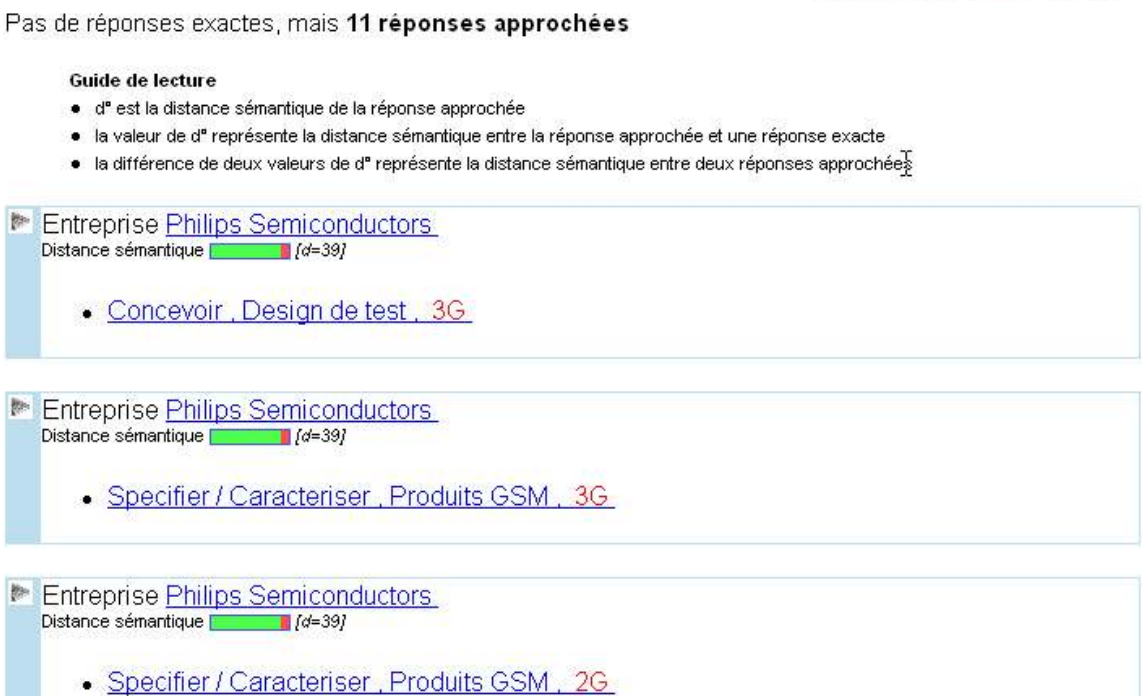


Figure 20. Résultats de recherche approximée pour une requête n'ayant pas de résultat exact dans la base : « qui conçoit des produits wifi sur Sophia ? »

Le portail KmP est donc une mémoire communautaire partagée entre des institutionnels régionaux et des entreprises et organismes de la Telecom Valley. Cette mémoire n'est liée ni à un individu ni à une organisation particulière mais bien à une communauté émergente de Sophia Antipolis.

Instituts collaborant pour des projets sur la capture du CO_2 : le projet e-Wok [28] a pour objectif de tirer bénéfice des travaux entrepris sur le web sémantique pour développer des systèmes opérationnels autorisant la coopération sur internet entre différentes organisations (entreprises, instituts, ...) impliquées dans un workflow d'ingénierie. Et ce, en mettant en place un ensemble de portails communicants (e-Wok hubs) proposant à la fois des applications web accessibles à des utilisateurs finaux et des services web accessibles aux applications métiers.

Ces portails sont la brique de base d'une architecture dédiée à l'exploitation des techniques de traitement des données et des connaissances. Le scénario applicatif, dans le domaine des géosciences, vise à gérer la mémoire de plusieurs projets sur la capture et le stockage de CO2, tout en exploitant les résultats de la veille technologique sur le domaine.

Outre l'architecture distribuée de ce système de gestion des connaissances, e-Wok s'intéresse aussi à l'ingénierie ontologique participative en particulier avec un essai de mise en œuvre dans l'éditeur collaboratif d'ontologies ECCO.

ECCO est l'acronyme d'Editeur Collaboratif et Contextuel d'Ontologies. Il a pour fonction d'aider les personnes participant à la construction d'une ontologie à partir de textes 1) à collaborer entre elles (Figure 21) à recueillir le contexte dans lequel apparaissent, dans des textes sources, les termes dénotant les concepts et les relations de l'ontologie. [29]

ECCO permet de créer une ontologie en suivant une série d'étapes (workflow) qui couvrent l'ensemble du cycle de conception de l'ontologie à partir de textes : 1) acquisition de textes sources ; 2) extraction de termes candidats et de leurs contextes dans les textes sources ; 3) élaboration collaborative d'un vocabulaire à partir des termes candidats, en distinguant notamment les concepts des relations ; 4) mise en place de la hiérarchie des termes du vocabulaire ; 5) édition formelle de l'ontologie ; 6) ajout/édition de règles au format de CORESE ; 7) tests de l'ontologie à l'aide de requêtes SPARQL exécutées par CORESE.

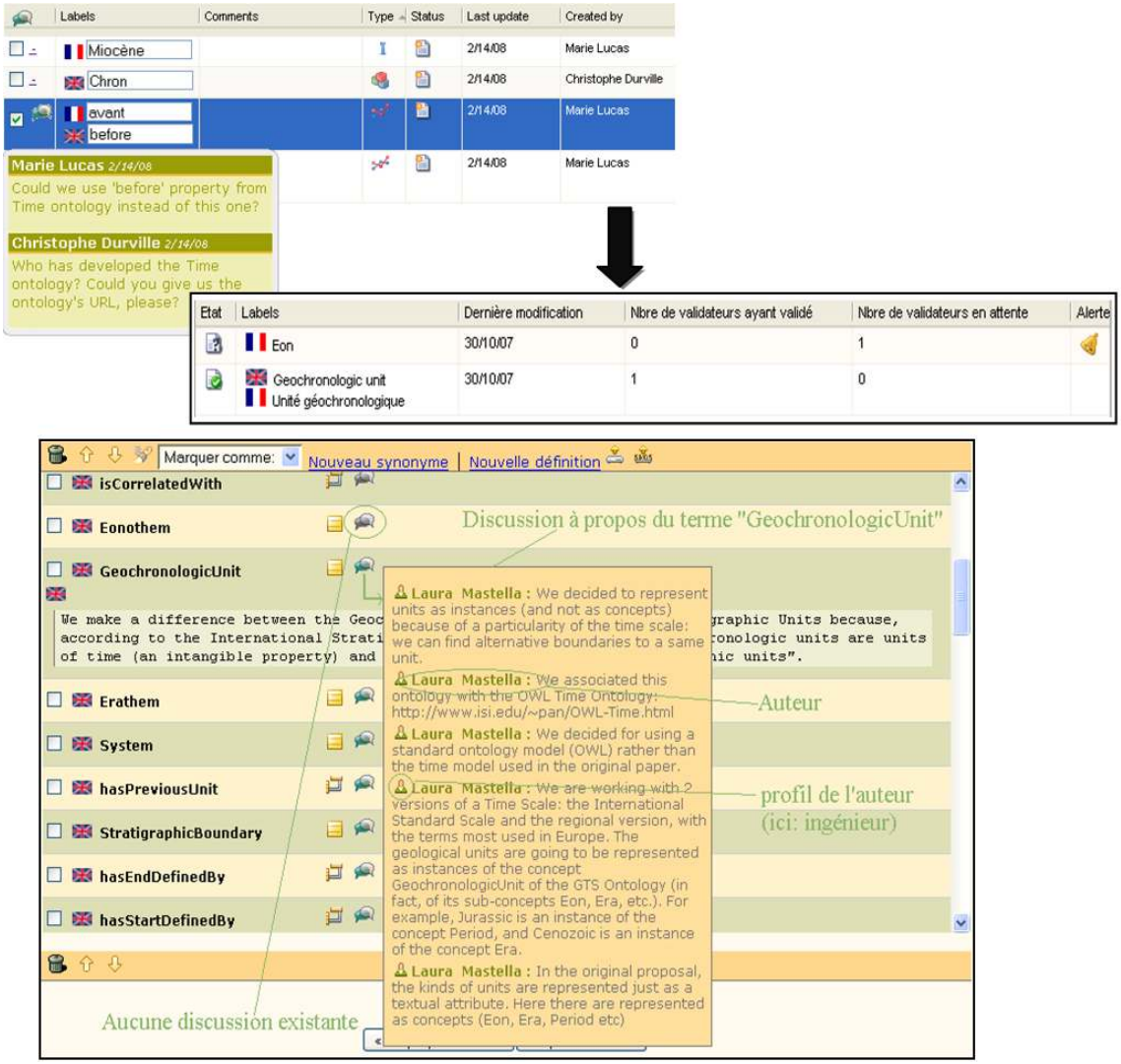


Figure 21. Quelques interfaces d'ECCO illustrant les aspects collaboratifs de cet éditeur.

Assister des communautés de pratiques pédagogiques : Le projet Palette [30] [31] intègre des informaticiens, des enseignants et scientifiques et des membres de communautés de pratique dont l'objectif est de faciliter et d'améliorer l'apprentissage individuel et collectif par le développement de services technologiques intégrés assistant l'apprentissage en ligne et l'organisation de la communauté en ligne.

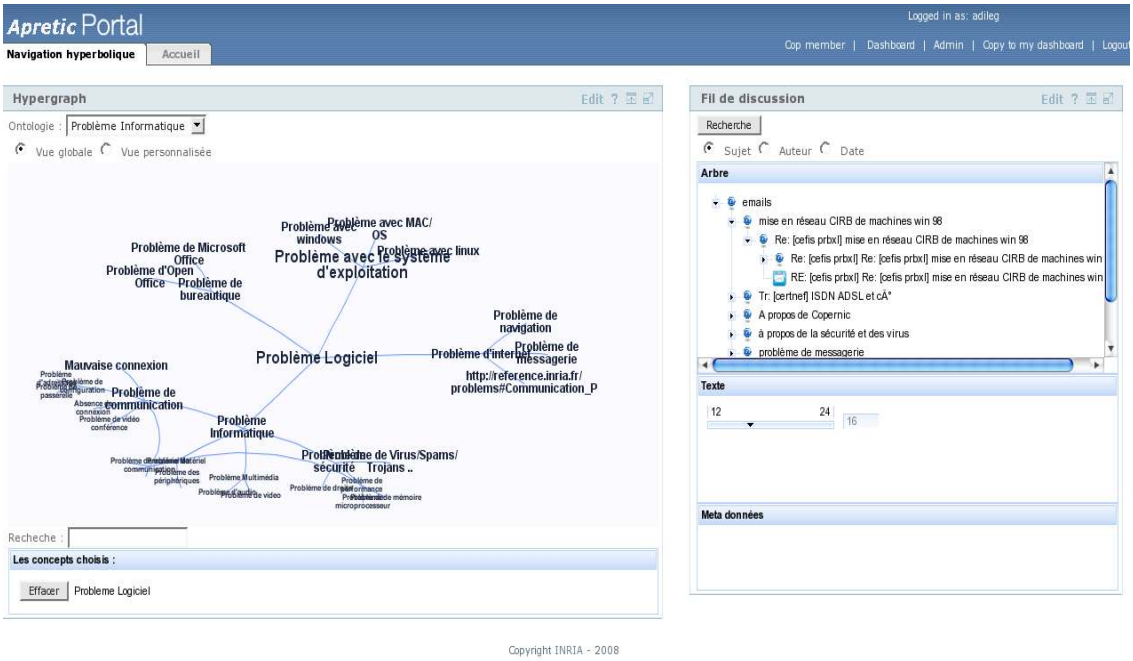


Figure 22. Traitement d'un corpus de mails pour en extraire des annotations, les organiser et naviguer comme dans une FAQ [32]

L'objectif principal du projet Palette est la facilitation et la mise en valeur des apprentissages individuel et collectif à travers des communautés de pratique, des groupes de personnes ayant des interactions fréquentes, qui partagent une préoccupation, une série de problèmes, ou une passion sur un sujet, et qui souhaitent approfondir leurs connaissances et leur expertise dans ce domaine.

Assistés d'une utilisation appropriée des applications en réseau et des technologies du web, ces communautés ont le potentiel de devenir un ferment fondamental pour le déploiement d'environnements d'apprentissage qui, à l'avenir, soutiendraient les professionnels, les organisations et les individus dans le domaine de l'éducation.

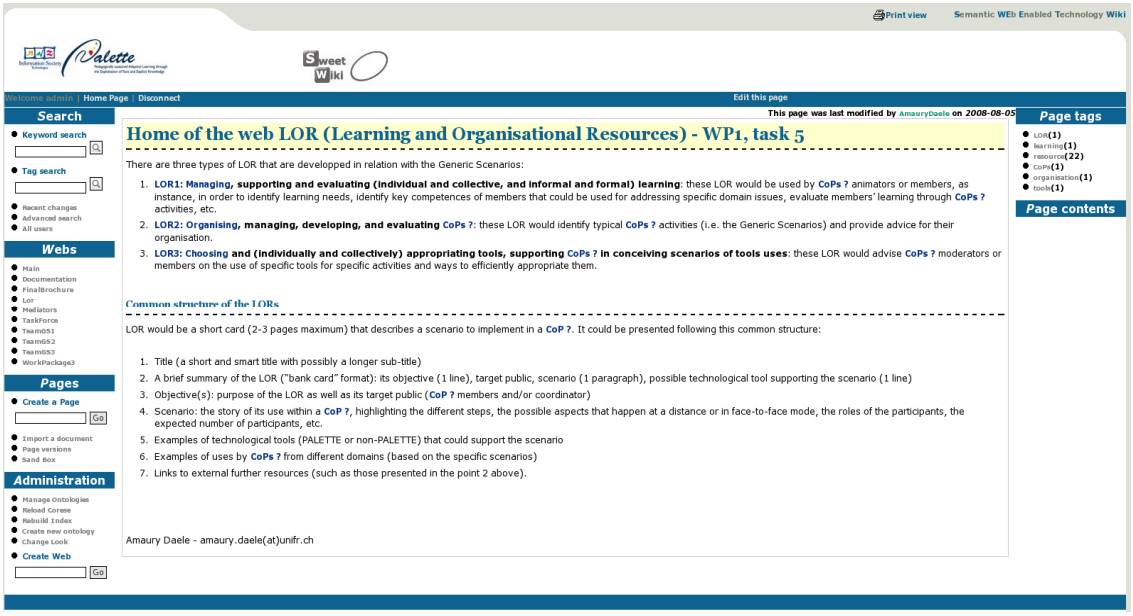


Figure 23. Utilisation d'un wiki sémantique dans une communauté de pédagogues [31] [30]; voir aussi la section 6.5 sur SweetWiki [33].

En intégrant dix communautés qui participent au projet et évaluent ses choix, ses objectifs et ses contraintes, Palette permet d'analyser leur situation actuelle et, d'imaginer et de rendre effectives des améliorations possibles grâce à l'expérimentation de nouvelles activités et outils. Cela comprend en particulier : la gestion et l'utilisation des documents d'une communauté, la création de liens sociaux grâce à une meilleure sensibilisation aux réseaux sociaux et aux interactions sociales, la représentation, l'organisation et l'extraction efficace des connaissances créées, et le suivi des processus de décision.

Assister des communautés de pratiques en sciences de la vie : Sealife [34] est un projet européen en cours depuis 2006 avec La Biotec de Dresden, City University, Manchester University, Edinbourg University, Scionics et l'INRIA.

L'objectif de ce projet est la conception et la réalisation d'un navigateur sémantique dédié aux sciences de la vie. Ce navigateur fera le lien entre le web existant et l'infrastructure eScience qui est en train d'émerger. Il proposera à l'utilisateur des services web eScience tout en prenant en considération le contenu sémantique des pages qu'il est en train de visiter. Ce navigateur sera testé avec trois scénarios portant sur :

(i) la médecine basée sur des faits (EBM),
(ii) la fouille des brevets et de la littérature et
(iii) la biologie moléculaire.

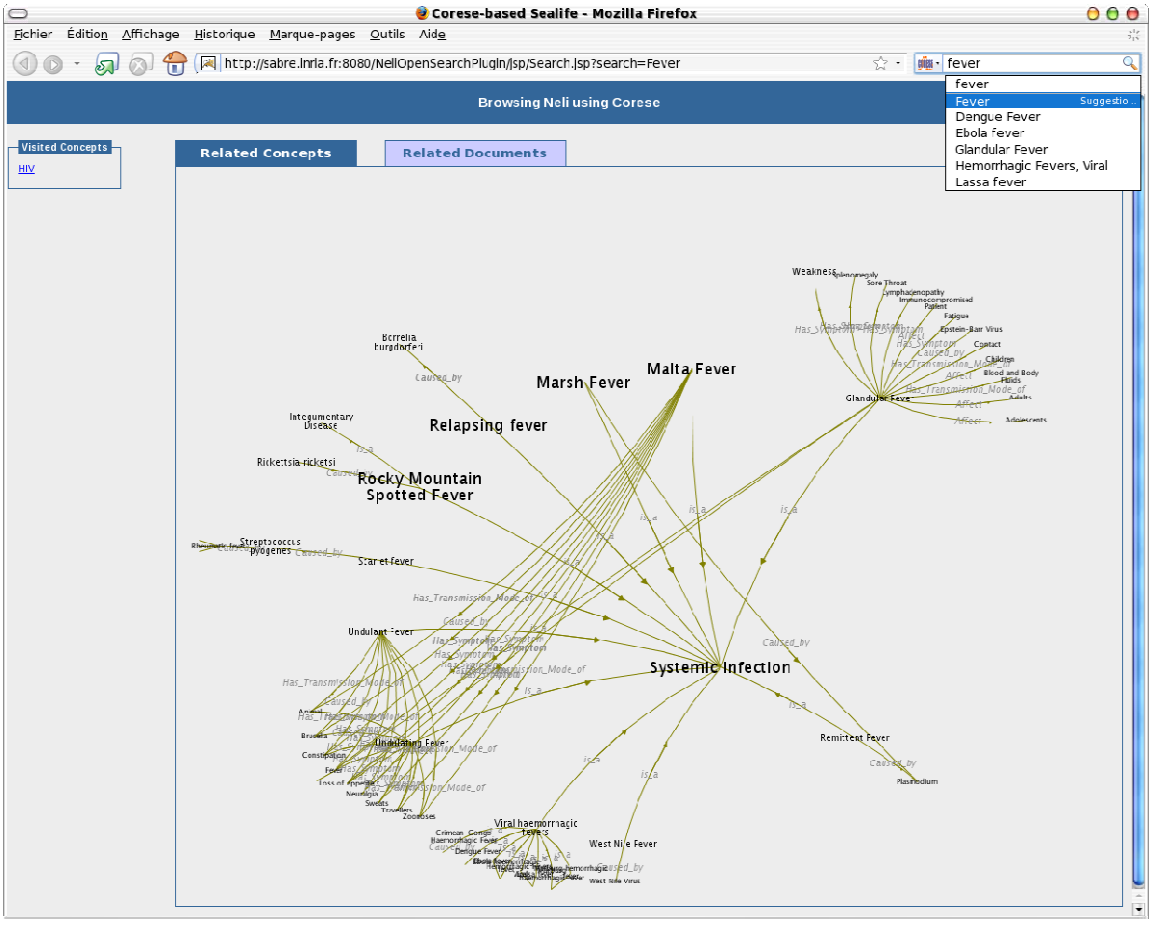


Figure 24. Navigation hyperbolique dans les annotations et l'ontologie de Sealife avec suggestion de mots-clefs par interrogation de Corese durant la frappe et gestion de l'historique. Ici l'utilisateur navigue autour du concept « fever »

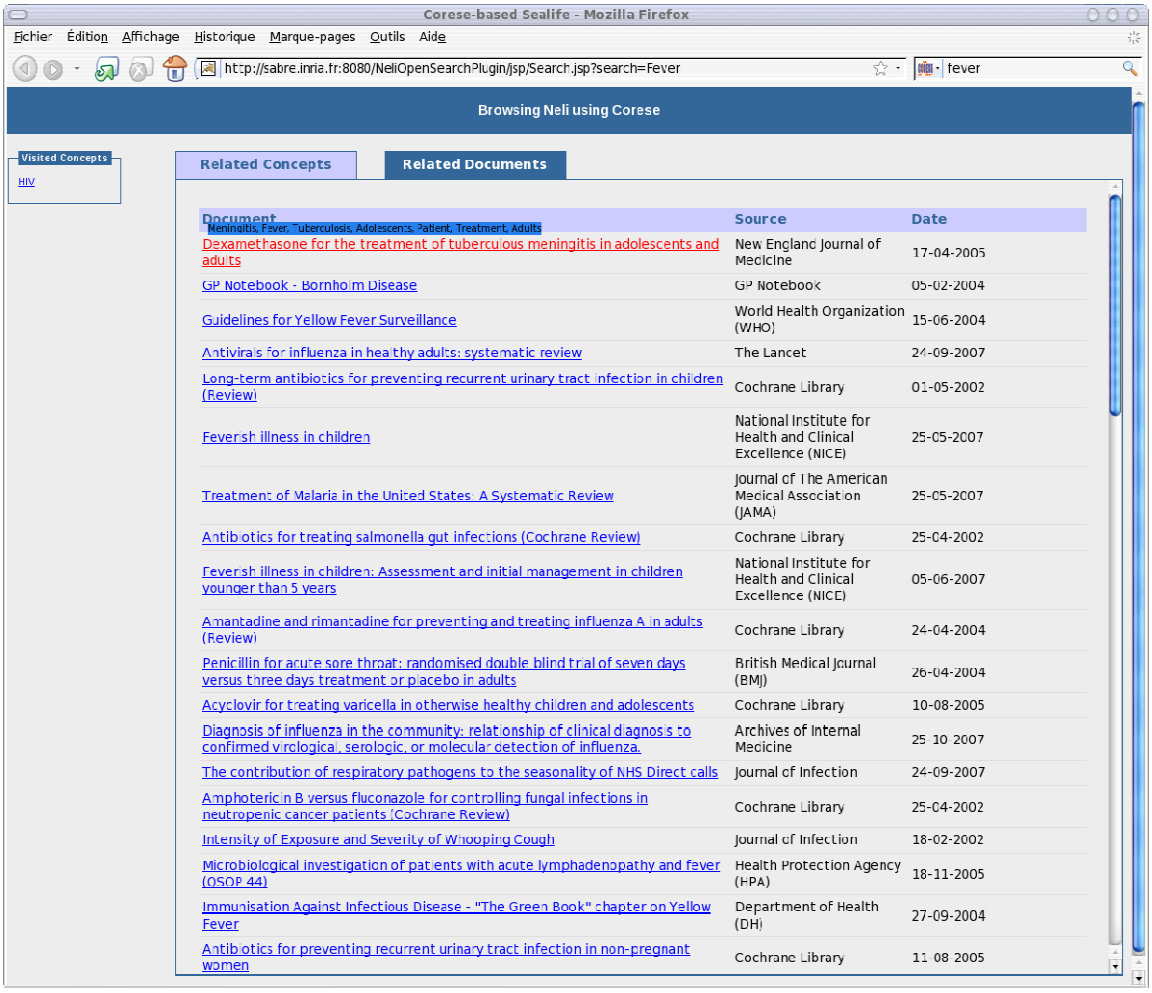


Figure 25. Une fois un nœud sélectionné dans la navigation hyperbolique, les documents annotés par le concept sélectionné sont listés, triés et documentés par leurs métadonnées.

Conclusion des mémoires communautaires : en créant la notion de présence et d'action sur le web, les applications web ont permis le développement de communautés en ligne et les interfaces dynamiques et les pratiques du web 2.0 ont considérablement accéléré cette tendance.

Ces communautés d'intérêts ou de pratiques, d'individus ou d'organisations traversent les structures socio-économiques classiques et multiplient les rôles que nous jouons. Elles génèrent des connaissances et d'autres traces numériques de nos activités qu'il nous gérer de façon efficace et en respect des différents acteurs individuels et collectifs.

Ces nouvelles mémoires ouvrent donc un nouveau champ d'application avec des caractéristiques très particulières : une émergence spontanée des communautés ; des structures et des fonctionnements agiles, fluides ; une nature parfois éphémère ; des règles de fonctionnement très spontanées et un besoin d'animation spécifique.

2.5 Conclusion et discussion

Dans cette section nous avons identifié et exemplifié plusieurs types de mémoires :

1. Des mémoires individualisées : le e-Wallet qui est une mémoire attachée à un utilisateur pour la gestion de ses informations privées ; la mémoire des expériences d'un biologiste, la bibliographie associée à cette expérience et les annotations de cette bibliographie ; la mémoire pédagogique d'un enseignant contenant ses cours annotés et réorganisés pour offrir des parcours individualisés aux étudiants ; la représentation sémantique d'un dossier-patient mémorisant son historique médical et les logiques de décision suivies, pour aider la coopération médicale entre acteurs d'un réseau de soins.
2. Des mémoires organisationnelles : pour la gestion de documents lors de la conception de bâtiments ou de produits industriels ; pour organiser et catalyser la veille technologique dans une entreprise ; pour accélérer l'intégration d'un nouvel employé dans une équipe.
3. Mémoires de communautés ouvertes : pour identifier les compétences au sein d'une technopole et favoriser les partenariats ; pour assister les échanges de données et d'applications entre des organismes partenaires d'un projet ; pour partager des pratiques et des ressources pédagogiques dans un réseau d'enseignants d'une même discipline ; pour améliorer les connaissances et services offerts en ligne des biologistes.

Chacune de ces mémoires utilise la représentation des connaissances à base de graphes (voir chapitre 3 suivant) et le raisonnement à base d'ontologies. Le spectre des applications et des domaines s'intéressant aux ontologies ne cesse de s'élargir. Outre l'argument d'un engouement patent pour cette nouvelle génération de systèmes d'information, ce panorama préfigure aussi le problème de la gestion des multiples mémoires auxquelles un utilisateur accède et des interactions entre ces mémoires. Gérer les différents rôles des utilisateurs et permettre tout en les contrôlant les échanges et les applications entre leurs différentes mémoires est un problème ouvert.

Nos systèmes d'information sont de plus en plus complexes. Cette complexité, même si elle est artificielle, puisqu'il s'agit de technologie, pose des défis scientifiques ardus qu'il nous faudra relever pour voir l'expansion technologique continuer. Injecter des connaissances dans nos systèmes ne sert pas uniquement à assister de nouvelles fonctionnalités applicatives ; en rendant nos conceptualisations explicites, les représentations à base d'ontologies permettent aussi de documenter, justifier et expliquer les comportements de nos systèmes. Au-delà d'une capacité explicative, et de nouvelles capacités d'interaction, ces approches ouvrent aussi des opportunités pour plus de réflexivité dans nos systèmes ce qui pourrait être un atout important d'une programmation orienté ontologie si elle venait à se réaliser

2.6 Perspectives : ISICIL

Pour conclure ce chapitre introductif à nos scénarios d'application par une perspective, nous mentionnerons le nouveau projet ANR nommé « ISICIL » visant à assister les communautés d'intérêt pour améliorer les capacités de veille d'une organisation. En étant à la frontière entre les mémoires d'entreprise et les communautés d'intérêt, ce projet est un archétype du passage des axes de l'équipe Acacia aux axes de l'équipe Edelweiss.

Récemment, les communautés d'intérêt en ligne ont vu le jour et ont commencé à construire des répertoires de références dans leurs domaines d'intérêt à une vitesse et une réactivité impressionnantes. Comme exemples de communautés d'intérêt en ligne, nous pouvons citer le célèbre Wikipedia, les amateurs de musique sur mp3.com, le système d'exploitation open source Debian ou la communauté des investisseurs *business angels* fool.com. L'une des forces des outils assistant ces communautés est leur capacité à transformer des utilisateurs normalement passifs en des participants actifs et producteurs. La diversité et la masse des utilisateurs sont utilisées pour faire face à la diversité et la masse des sources d'information.

La veille scientifique, économique et technologique est un élément essentiel de la capacité des organisations d'aujourd'hui, et pourtant la diversité croissante des sources d'information à surveiller dans chaque domaine d'intérêt demeure un défi de taille pour toute organisation quelque soit sa taille. Il y a donc une demande croissante pour importer les outils et les pratiques qui ont fait le succès de ces communautés en ligne dans les systèmes d'information d'entreprise. Des blogs et wikis sont mis en place dans de plus en plus d'intranets.

Cependant, d'un côté, les outils web 2.0 montrent des limites quand il s'agit d'automatiser certaines tâches ou de contrôler certains processus, comme habituellement requis dans un environnement d'entreprise. D'un autre côté, les systèmes d'information plus structurés souffrent souvent d'un manque de convivialité et d'une faible assistance à la capture des connaissances. En outre, dans le cadre de la veille, les structures organisationnelles peuvent également fournir une assistance aux différentes étapes de ces processus, pour s'assurer que les normes de qualité des entreprises, les processus sont suivis et les règles sont respectées.

Le défi du projet ISICIL qui commencera en 2009 est de concilier les nouvelles applications virales du web avec des représentations formelles et des processus d'entreprise pour les intégrer dans les pratiques de veille en entreprise. Plus précisément, ISICIL propose de concevoir, d'étudier et d'expérimenter l'utilisation de nouveaux outils d'assistance aux tâches d'intelligence en entreprise. Ces outils s'appuieront sur les interfaces avancées des applications du web 2.0 (blog, wiki, *social bookmarking*) pour les interactions et sur les technologies du web sémantique pour l'interopérabilité et le traitement de l'information.

Chapitre 2 : récapitulatif de mes contributions personnelles.

Dans ce chapitre j'ai rappelé la notion de mémoires telles que j'ai été amené à en construire dans différents projets.

Parmi les projets listés dans cette section :

J'ai été responsable et architecte du projet myCampus à Carnegie Mellon pendant un an et j'ai conçu, implanté et validé les représentations et les raisonnements du e-Wallet [14] [13] ainsi que les services et l'accès mobile et localisé sur le campus [16].

J'ai été l'architecte principal du système multi-agents de CoMMA, de ces représentations, protocoles et inférences [21] et j'ai encadré les extensions [22] faites pendant le DEA et la thèse de Tuan-Dung Cao.

J'ai été rédacteur et porteur du projet Aprobatiom sur les mémoires dans la conception de bâtiments durant lequel j'ai encadré deux étudiants de master pour la réalisation du prototype.

J'ai participé au montage et à la soutenance du projet européen SevenPro. Je contribue actuellement dans trois tâches de ce projet : l'utilisation d'une architecture distribuée pour l'intégration des sources de la mémoire [24] ; la définition de métriques sémantiques pour la recherche de cas de conception passés ; l'utilisation des représentations et raisonnements à base de connaissances dans des environnements de réalité virtuelle pour la conception industrielle.

J'ai conçu et formalisé les représentations et inférences du projet KmP pour la capture, l'organisation, la recherche et la navigation dans les compétences de la Telecom Valley de Sophia Antipolis [27].

J'ai participé au montage du projet e-Wok Hub et j'interviens sur les questions d'éditeur d'ontologie collaboratif [29] et sur la mise en place d'une architecture de services web sémantiques.

Je suis co-concepteur du logiciel SweetWiki [33] utilisé pour le projet Palette. J'ai co-encadré quatre stagiaires de master et une thèse en cours, et j'implémente la partie RDFa GRDDL qui fait l'objet d'une diffusion dans le cadre d'un standard au W3C [35] [36] [37] [38].

J'interviens dans le projet Sealife pour la conception de distances sémantiques utilisées pour la désambiguïsation d'annotations extraites par analyse linguistique [39].

J'ai aussi présenté un travail de vulgarisation [6] de la notion d'ontologie informatique qui, depuis quinze ans maintenant, percole dans un nombre croissant de domaines fondamentaux et applicatifs de l'informatique.

Ce travail m'a aussi amené à rédiger plusieurs chapitres dans l'ouvrage collectif de notre équipe [1].

Enfin, je suis l'instigateur et le porteur du nouveau projet ANR ISICIL.

3. Des graphes de représentation

Dans ce chapitre, nous rappelons comment les formalismes à base de graphes peuvent être utilisés pour représenter des connaissances avec un degré variable de formalisation en fonction des besoins identifiés dans les scénarios d'application et des traitements à effectuer.

Nous nous attacherons à deux aspects importants dans ces représentations :

- leur appartenance aux standards du web qui nous assure une interopérabilité et une réutilisation de résultats de recherche et développement à une échelle jamais connue en représentation des connaissances.
- l'omniprésence des structures de graphes dans ces représentations et les caractéristiques de celles-ci qui seront notamment utilisées dans tous les chapitres suivants de ce mémoire.

Nous identifierons et rappellerons brièvement les caractéristiques de certains de ces formalismes qui sont utilisés dans nos travaux et les opportunités d'extensions et de recherches qu'ils offrent.

Nous synthétiserons aussi une initiative en cours pour factoriser la définition des structures mathématiques partagées par ces formalismes et réutiliser l'algorithmique des traitements communs à ces structures.

3.1 Les graphes du web et la représentation de connaissances en ligne

On parle beaucoup du graphe du web mais il y a maintenant une multitude de graphes différents inclus et mis à disposition par le web. Au-delà de l'éternel graphe des liens hypertextes et des graphes d'accointance issus des traces d'interactions entre utilisateurs ou entre ressources, il existe maintenant un certain nombre de formalismes qui permettent de mettre en ligne des données ou métadonnées structurées en graphes.

Le langage le plus important est sans aucun doute RDF [40] : un modèle de graphes orientés étiquetés qui, avec les arbres XML [41], est une structure de base pour toutes les activités de standardisation au W3C (Figure 26).

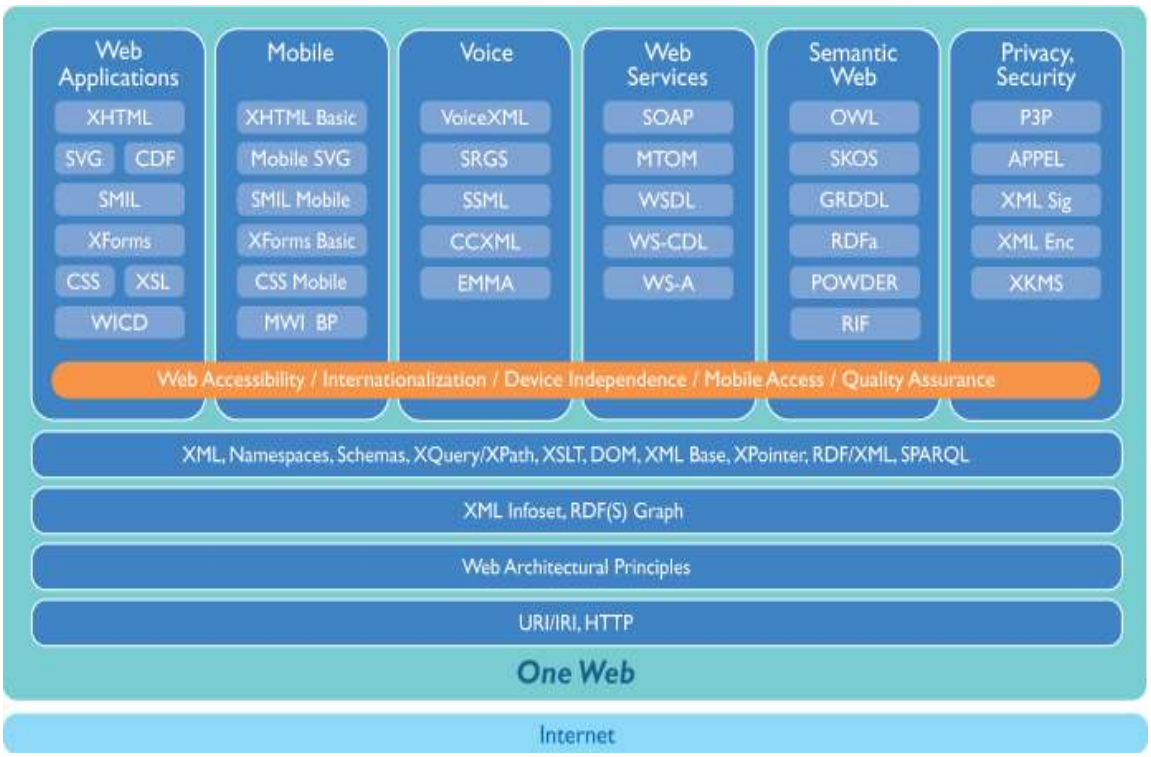


Figure 26. Piles des activités de standardisation au W3C reposant sur les structures d'arbre de XML et de graphe de RDF (source : www.w3.org).

RDF est un modèle de représentation de données et métadonnées décomposées en triplets pour décrire et connecter des ressources *i.e.* des objets anonymes ou identifiés par un URI. Toute connaissance représentée en RDF se décompose donc sous la forme `(sujet, prédicat, objet)`. Le triplet est l'atome de connaissance en RDF, sa plus petite division comme l'illustre la Figure 27 en reprenant la célèbre illustration de Douglas R. Hofstadter [42].

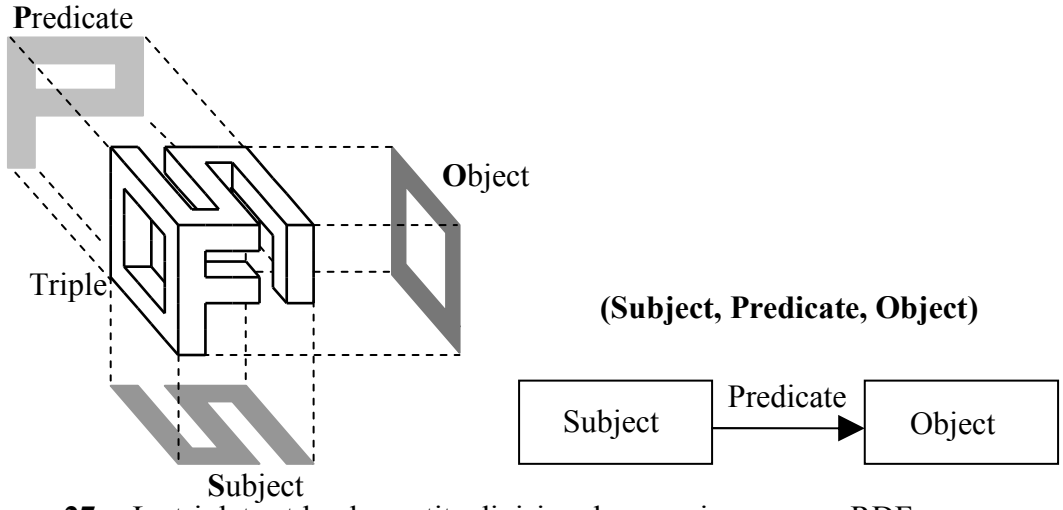


Figure 27. Le triplet est la plus petite division de connaissances en RDF

Par exemple l'assertion « doc.html a pour auteur Fabien et parle du web » peut se décomposer en deux triplets RDF `(doc.html, author, #me)` et `(doc.html, subject, "web")`. Les triplets peuvent être aussi vus comme les arcs d'un graphe orienté étiqueté qui seraient distribués sur le web. Ce modèle est doté d'une syntaxe XML [43], entre autres syntaxes, permettant de représenter, enregistrer et échanger des graphes RDF (Figure 28).

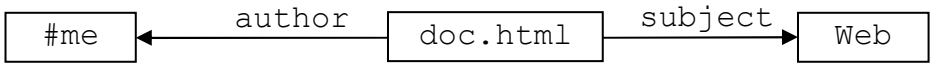


Figure 28. Triplets RDF comme les arcs d'un graphe orienté étiqueté distribué sur le web.

Il existe pour RDF une syntaxe XML (Figure 29) mais il existe aussi une autre syntaxe permettant de l'inclure dans du HTML, notamment à travers les attributs du langage HTML d'où le nom de cette syntaxe : RDFa [37] pour « *RDF in attributes* ». Les triplets RDF sont combinés avec les structures HTML, ce qui fait qu'en un seul fichier et une seule inscription, des données dans un document deviennent compréhensibles pour l'humain et accessibles aux applications (Figure 30).

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:inria="http://inria.fr/schema#" >
  <rdf:Description rdf:about="http://inria.fr/rr/doc.html">
    <inria:author rdf:resource="http://inria.fr/~fabien#me" />
    <inria:subject>web</inria:subject>
  </rdf:Description>
</rdf:RDF>
```

Figure 29. Exemple de syntaxe XML pour RDF.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:inria="http://inria.fr/schema#">
<head><title>A page about my doc</title></head>
<body>
 <p>I wrote a <a rev="inria:author"
                 about="http://inria.fr/~fabien#me"
                 href="http://inria.fr/rr/doc.html">
  document about the
    <span about="http://inria.fr/rr/doc.html"
          property="inria:subject">web</span>
 </a>.</p>
</body>
</html>
```

Ce code s'affiche ainsi : I wrote a document about the web.

et contient les mêmes triplets RDF que le code XML en Figure 29

Figure 30. Exemple de syntaxe RDFa pour du RDF dans une page web.

Le web classique (HTML) et le web structuré (XML) ont cependant déjà mis beaucoup d'informations et de données en ligne. Des métadonnées ou des données utiles aux applications du web sémantique sont parfois déjà incluses plus ou moins explicitement dans ces ressources. RDFa fournit une syntaxe permettant d'intégrer des métadonnées RDF dans le code d'une page web en HTML.

GRDDL [38] est une solution duale à la définition de laquelle nous avons participé [36] et qui fournit un mécanisme pour déclarer qu'un document XML contient des données qui peuvent être représentées en graphes RDF et pour lier à ce document une transformation (en particulier une feuille de style XSLT) afin d'extraire ces données. Des pages web contenant des métadonnées en RDFa ou microformats[4], ou des documents XML peuvent ainsi être transformés en données au format RDF et utilisés par des applications du web sémantique. Nous participons aussi à la mise en place d'un profile GRDDL pour RDFa [35] permettant automatiquement d'extraire en RDF/XML le RDF inclus dans une page web en RDFa.

SPARQL fournit un langage de requête [44] sur les graphes RDF, un langage de résultats [45] pour représenter les réponses à une requête et un protocole [46] pour soumettre une requête à un serveur distant et recevoir les réponses.

RDFS [47] est un langage de déclarations et de descriptions légères des types de ressources (appelés Classes) et de leurs relations (appelées Propriétés). RDFS permet de nommer et définir un vocabulaire utilisé dans des graphes d'annotations RDF : nommer les classes de ressources existantes ; nommer les relations qui existent entre ces classes et donner leur signature *i.e.* le type des ressources qu'elles relient. RDFS permet aussi des inférences élémentaires notamment en exploitant les liens hiérarchiques entre classes, les liens hiérarchiques entre propriétés ou les signatures des propriétés. En

[4] Voir http://microformats.org/

donnant un URI aux concepts qui sont importants pour une application, RDFS permet de déclarer le squelette taxonomique d'une ontologie dans un langage universel.

OWL [48] est une recommandation donnant trois couches d'extension de l'expressivité logique de RDFS : OWL Lite, OWL DL et OWL Full. Les deux premières couches d'extension sont basées sur les logiques de descriptions qui permettent des inférences supplémentaires telles que la vérification de la cohérence d'un schéma, la classification automatique des types pour générer automatiquement des hiérarchies, ou l'identification automatique du type d'une ressource en fonction de ses propriétés. OWL permet la définition de classes par l'énumération de leur contenu ou par l'union, l'intersection, le complément et la disjonction d'autres classes. OWL permet aussi la caractérisation des propriétés (restriction de leur valeur ou de leur cardinalité) et de leurs propriétés algébriques (symétrique, transitive, fonctionnelle, inversement fonctionnelle, propriété inverse). Enfin, OWL permet la gestion des équivalences entre schémas, des versions et de l'annotation du schéma lui-même. OWL 2.0 [49], une seconde version, est en préparation ajoutant de nouvelles extensions de l'expressivité.

Légèrement en marge du web sémantique, la recommandation en cours de POWDER [50] fournira un mécanisme pour associer une description à un groupe de ressources en fonction de leurs URI et un protocole pour obtenir des métadonnées sur des ressources avant d'y accéder. Un exemple d'assertion possible en POWDER serait : toute ressource dont l'URI commence par `http://www.inria.fr/sophia/fr/` concerne le centre de recherche de l'INRIA géographiquement situé à Sophia Antipolis et est en Français. La représentation de ces associations serait en RDF et OWL.

SKOS signifie « système simple d'organisation des connaissances » [51]. SKOS propose un modèle de représentation moins orienté vers la formalisation logique et destiné à représenter des ressources terminologiques, des classifications thématiques, des glossaires, des thésaurus, des folksonomies[5], ou tout autre type de vocabulaire contrôlé et structuré. Impliquant des experts en science bibliothécaire, SKOS est un modèle RDF, donc à base de graphes. Elle vise à faciliter la publication de vocabulaires structurés pour leur utilisation dans le cadre du web sémantique. SKOS suit l'approche « centrée-concept » de RDF dans le sens où ses objets primitifs ne sont pas des termes, mais des concepts abstraits identifiés par des URI et dont les termes sont des propriétés. SKOS offre plusieurs primitives pour la représentation des thésaurus. A chaque concept, peuvent être attachés : au maximum un label ou terme préférentiel par langue naturelle ; des labels ou termes synonymes dans chaque langue ; des définitions, des notes de travail, des exemples, et autres commentaires ; des relations vers d'autres concepts notamment les relations « plus générique » (*broader*) et « plus spécifique » (*narrower*). Le modèle graphique SKOS offre à la fois une solution intermédiaire au développement d'ontologies lourdes en OWL, un formalisme de représentation intermédiaire pour une

[5] Une folksonomie est un système de classification collaborative décentralisée spontanée formé par l'union des tags choisis par les utilisateurs pour classifier une ressource.

ontologie en construction et un modèle compatible et combinable avec des descriptions en RDFS et OWL pour des modèles hybrides.

Les *topics maps* sont un formalisme graphique datant du début des années 90 et doté d'un standard ISO en l'an 2000. Ils permettent de représenter les sujets dont parlent des ressources documentaires et les relations entre ces sujets. Les sujets (*topics*) sont des représentations informatiques de sujets du monde réel. Ils sont nommés et ont des occurrences, c'est-à-dire des ressources documentaires pertinentes pour ce sujet. Ces sujets participent à des associations (relations) dans lesquelles ils ont un rôle. Afin de publier et d'échanger des représentations *topic maps* sur le web, le formalisme s'est doté d'une syntaxe XML appelée XTM qui signifie *XML Topic Maps*. [52]

Notons aussi l'existence de CWL pour *Common Web Language* [53] visant à décrire le contenu et les métadonnées des pages web destinés à être converti en une ou plusieurs langues naturelles pour être présentées à un utilisateur. CWL compte RDF parmi ses représentations et produit donc aussi des graphes.

Enfin le groupe incubateur RDB2RDF au W3C examine les liens entre les bases de données relationnelles d'un côté et le modèle RDF et les schémas RDFS et OWL de l'autre. Deux livrables sont en particulier prévus : un rapport sur les approches pour transformer des données de tables relationnelles en triplets RDF ; un rapport sur l'association de requêtes SQL à des classes OWL. [54]

Tous les exemples de cette section montrent qu'un grand nombre de formalismes et de procédés sont disponibles pour mettre en ligne sur le web des graphes de données et métadonnées, et notamment des graphes RDF.

3.2 Intra-web Sémantique et web sémantiques communautaires

Un intraweb est un web formé par les serveurs HTTP d'un intranet. Le couplage d'un intranet à l'Internet et d'un intraweb au web ouvert fournit un moyen d'accès unifié aux informations internes et externes disponibles en ligne. C'est maintenant l'infrastructure privilégiée pour la mise en place d'une mémoire d'entreprise. En se développant, ces intrawebs peuvent, comme le web, souffrir « d'infobésité » avec des quantités de données et d'informations disponibles mais enterrées et dormantes dans leur masse.

Pour qu'elle soit utile et utilisée, une mémoire collective doit être adaptée aux besoins de sa communauté, et les mécanismes de son exploitation doivent être efficaces. Elle doit faciliter l'accès aux documents, offrir des moyens de recherche précis et complets, s'adapter aux différents profils d'utilisateurs et aux différents contextes d'utilisation, s'activer d'elle-même lorsqu'elle détecte la pertinence d'un document pour un profil et un contexte, ou lorsqu'un processus de la communauté passe à une nouvelle étape (ex : une évaluation vient d'être soumise sur un document).

Pour mettre en place ces mécanismes, l'indexation et la recherche sont très vite limitées si elles restent au niveau des chaînes de caractères (ex : réponses manquées dues à une synonymie ou un hyponymie ignorée). En franchissant la barrière des symboles un système d'indexation et de recherche peut manipuler les aspects sémantiques utiles pour les recherches qui lui sont soumises.

XML est devenu un standard pour l'échange et le stockage pérennes d'informations. Il est donc un excellent candidat pour matérialiser une structuration plus forte de la mémoire. Cependant, les documents d'une communauté ne sont pas forcément disponibles au format XML. De plus, la seule exploitation de la structure n'est pas suffisante pour une recherche intelligente ; plus qu'un balisage structurel, il nous faut capturer le sens des structures ou du moins les caractéristiques de ce sens qui permettent les traitements dont nous avons besoin. Pour cela, l'approche du web sémantique consiste à annoter sémantiquement les ressources du web par les aspects pertinents pour guider leur exploitation. Ces annotations sont représentées en RDF que nous venons de présenter car RDF permet de décrire les relations existant entre ces ressources dans un format qu'une machine peut traiter.

En utilisant RDF pour l'annotation d'une mémoire organisationnelle, on peut mettre en place un intraweb sémantique où l'on décrit les ressources de la mémoire par des annotations sémantiques internes ou externes aux ressources, utilisées ensuite pour fouiller plus efficacement la masse d'informations. En étendant la problématique d'une organisation à une communauté d'intérêt ou de pratique, on peut reposer sur le modèle RDF pour organiser les ressources manipulées par cette communauté et assister les interactions entre ses membres.

Les problèmes de recherche posés sont alors les suivants :

- L'identification des caractéristiques des formalismes utilisés et l'exploitation de ces spécificités dans des inférences.
- L'adéquation et les limites des approches du web sémantique pour des problèmes de la gestion des connaissances.
- Les passerelles entre les méthodes et outils de l'ingénierie des connaissances et ceux du web Sémantique pour proposer, par exemple, l'extension des formalismes et des opérateurs disponibles ou suggérer de nouvelles techniques de recherche d'information.

Les sections suivantes se focalisent sur le premier point en défendant notamment l'intérêt des formalismes à base de graphes tels que nous les utilisons, et en montrant l'exploitation de certaines de leurs spécificités dans des représentations puis dans des inférences (voir chapitre 4).

3.3 Formalismes du web orientés graphes et leurs manipulations

Les applications du web et, de façon générale, l'automatisation des tâches de traitement de l'information ne cessent de générer des scénarios d'application des recherches sur la représentation des connaissances et le raisonnement. Nous venons de le voir, de plus en plus de standards et techniques sont déployés pour offrir sur le web un espace et des outils formels d'échanges et de manipulation de connaissances (RDF, RDFS, SKOS, OWL, GRDDL, RDFa, µFormats, etc.). Certaines couches de ces formalismes (ex : la couche OWL DL) ont des opérationnalisations immédiates découlant de leur parenté avec des formalismes existants (ex : logiques de description). Nous nous intéressons ici aux couches qui s'appuient directement sur la structure de graphe des connaissances manipulées.

3.3.1 Raisonneurs sur le web sémantique

Sesame [55] est une architecture générique pour le stockage persistant de RDF(S) dans une base de données et l'interrogation de ce RDF(S) avec le langage RQL. RQL [56] est un langage de requête RDF défini par le moyen d'un ensemble de requêtes fondamentales, un ensemble de filtres de base et un moyen de construire de nouvelles requêtes par une composition fonctionnelle et des itérateurs. Lors de l'analyse d'une requête RQL, Sesame construit une requête optimisée évaluée au moyen d'une série d'appels à la couche de stockage et d'inférence de Sesame.

Par comparaison RDQL [56] permet l'interrogation de données RDF au niveau de la structure, tandis que Sesame permet l'interrogation au niveau sémantique. En ce sens notre objectif est beaucoup plus proche de Sesame. Toutefois Sesame ne considère pas la sémantique des XSD datatypes, ni ne permet d'avoir des règles d'inférence exploitant les ontologies.

Jena [57] est l'un des moteurs actuels les plus complets et propose une persistance en mémoire ou en base de données. Il implante RDF, RDFS et OWL ainsi que les requêtes SPARQL et propose un moteur en chaînage avant (RETE), arrière (programmation logique) et hybride. Ce moteur est utilisé pour implanter la sémantique de RDFS et OWL. Le modèle de Jena repose sur une structure prédéfinie de bases de données.

Triple [58] est un langage de requêtes pour divers modèles de données du web sémantique. Le noyau Triple est un langage de requête RDF fondé sur la logique de Horn étendue par des fonctionnalités syntaxiques pour intégrer des primitives RDF comme les espaces de nommage, les ressources et les réifications. Ce langage peut être compilé en logique de Horn et exécuté par des moteurs Prolog. Triple n'est pas limité à l'interrogation de données RDF. Il dispose d'une architecture en couche lui permettant d'interroger d'autres modèles de données avec différents types de sémantique (ex : RDFS et DAML + OIL) et en ce sens il rejoint notre certitude qu'il faut abstraire les structures et algorithmes des différents langages que nous manipulons. Le noyau de Triple est étendu par des règles pour l'axiomatisation de la sémantique de RDFS ; elles

peuvent être utilisées avec un moteur d'inférence pour dériver des connaissances supplémentaires à partir d'un schéma RDFS.

DAMLJessKB [59] et son successeur OWLJessKB ainsi que le cœur du eWallet [13] sont des outils de raisonnement pour DAML [60] ou OWL Lite. Ils ont tous les trois intégré JESS [61] un moteur de règles de production qu'ils appliquent à la sémantique de RDF, RDFS, DAML, XSD (*XML Schema Datatypes* [62]) et OWL Lite. Ils implantent une traduction des triplets RDF vers des faits du langage CLIPS de Jess et utilisent des règles de production pour décrire la sémantique des langages traduits. A l'aide de Jess, ces systèmes peuvent effectuer des raisonnements sur les classes et les instances.

OntoBroker [63] et son successeur On2broker sont des pionniers des systèmes à base d'ontologies basés sur les Frame Logics. OntoBroker gère des métadonnées intégrées dans des documents HTML avec des balises tout comme On2Brocker gère des annotations RDF. Dans les deux systèmes, les ontologies et les requêtes sont exprimées en langage *frame logics* qui permet la représentation d'une hiérarchie de types de concepts, d'une hiérarchie de types de relations et de règles. Le moteur de requête traduit ces représentations en Logique de Horn pour répondre à une requête.

OntoBroker, On2Broker, DAMLJessKB, OWLJessKB et partagent avec notre initiative le même principe général et les expressivités de leurs langages de représentation à base d'ontologie sont comparables. Toutefois, ces systèmes sont dédiés à des capacités générales de raisonnement sans exploiter les structures de graphes par exemple pour optimiser des opérations spécifiques à la recherche d'informations. Leurs langages de requêtes sont proches des logiques sur lesquelles ils se basent et restent dans leurs limites.

WebKB [64] [65] est un pionnier des serveurs web d'ontologies et des robots web basés sur les graphes conceptuels. Sur la base de sa simplicité et de l'exhaustivité de sa représentation, ses auteurs rejettent l'utilisation directe des langages basés sur XML. WebKB interprète et traduit automatiquement en graphes conceptuels des déclarations exprimées dans une notation linéaire des graphes et intégrées à des documents web. WebKB fournit des commandes de requête pour interroger les propriétés lexicales ou structurelles des documents HTML ou afficher des spécialisations ou généralisations d'un concept, d'une relation ou d'un graphe.

OntoSeek [66] est un autre système conçu pour la recherche d'information basée sur le contenu de pages jaunes et de catalogues de produits en ligne. Il combine les graphes conceptuels et des ontologies du domaine correspondant. Il met l'accent sur les contraintes lexicale et sémantique dans le codage des ressources en graphes conceptuels et la construction de requêtes. Requêtes et annotations des ressources sont des graphes conceptuels lexicaux entre lesquels on cherche des homomorphismes.

WebKB, OntoSeek et la plate-forme que nous spécifierons en section 3.3.4 partagent le même type de représentation à base de graphes, par conséquent, le même principe d'homomorphisme correspondant à une requête sur les graphes d'annotation en prenant

en compte la subsomption des types de concepts et des types de relations. Mais ni WebKB ni OntoSeek n'intègre RDF(S) ou les règles dans leur représentation à base d'ontologie.

Avec la standardisation de OWL DL, les moteurs à base de logiques de descriptions ont pris une importance particulière. Citons : Fact et son successeur Fact ++ [67], KAON2 [68] (une branche de KAON qui lui est resté focalisé sur RDFS), Racer [69], Pellet [70]. Ces moteurs offrent des opérations classiques en logiques de descriptions : identification, classification, validation. L'interrogation se limite en général à des requêtes conjonctives et les mécanismes d'interrogation ne sont pas prévus pour exploiter la structure des graphes ni accepter des graphes de grande taille.

En conclusion, aucune des contributions citées ci-dessus ne cherche à mettre en place un modèle pivot et une plate-forme ouverte et open-source pour implanter efficacement les aspects des représentations relevant des graphes. Elles sont toutes liées à un langage particulier (le plus souvent une logique) et/ou à une application particulière.

En ce qui concerne les outils permettant de raisonner avec des graphes conceptuels, les deux principaux outils sont : CoGITaNt [71] développé conjointement par le LIRMM et le LERIA, plate-forme dédiée à la mise en œuvre de mécanismes de raisonnement sur les graphes conceptuels spécifiquement ; et Amine (développé par l'INSEA au Maroc), plate-forme dédiée au développement de systèmes intelligents, qui repose sur une combinaison de Prolog et de graphes conceptuels. Citons également Prolog+CG basé sur un précurseur d'Amine, à usage éducatif.

Corese [72] [73] [5] [74] est un moteur de recherche sémantique basé sur les graphes conceptuels. Il implante RDF/S et SPARQL et il est développé dans l'équipe Acacia puis Edelweiss. Plus précisément, il implante la sémantique de RDF, RDFS, ainsi que les *datatypes*, les propriétés transitives, symétriques et inverses de OWL Lite.

Comme le montre la Figure 31, CORESE combine les avantages d'utiliser le langage RDF(S)/XML pour exprimer et échanger des méta-données, et les mécanismes de requête et d'inférence disponibles pour le formalisme des Graphes Conceptuels.

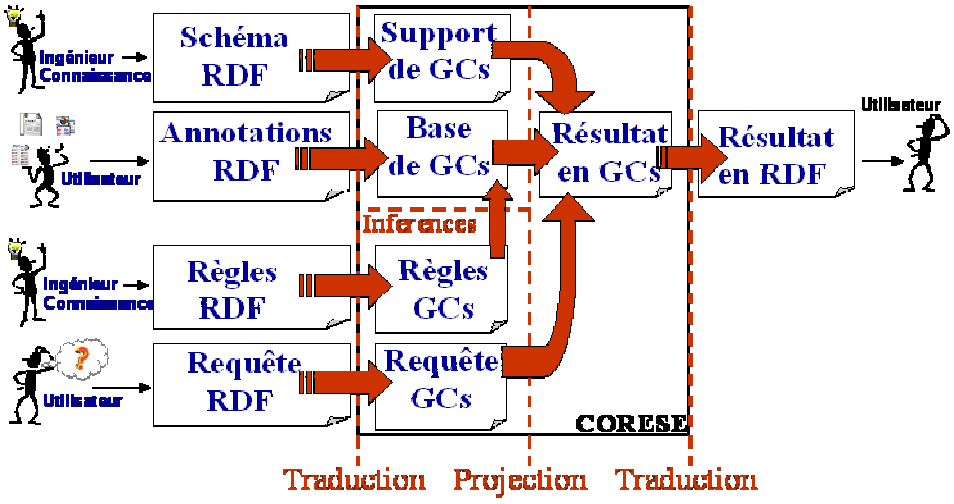


Figure 31. Principe général du moteur de recherche sémantique Corese

Le mécanisme de projection tient compte des liens de spécialisation décrits dans les hiérarchies traduites à partir du schéma. Ainsi le moteur de recherche exploite les connaissances ontologiques pour améliorer le taux de rappel de sa recherche, par exemple : si un utilisateur demande les documents concernant l'intelligence artificielle et que le système connaît un document concernant les systèmes multi-agents, il sera capable d'utiliser l'ontologie pour inférer que les systèmes multi-agents sont une branche de l'intelligence artificielle et donc que ce document est une réponse pertinente.

Corese possède également un langage de règles et un moteur d'inférences. Il s'agit d'un moteur de règles en chaînage avant, utilisé en particulier pour vérifier des restrictions de propriétés en OWL DL. Ces inférences, et les règles de production éventuellement ajoutées par l'application utilisant Corese, sont appliquées jusqu'à saturation, avant la résolution des requêtes.

Corese permet de faire des recherches approchées [74] par relation de type ainsi que des recherches de chemin dans les graphes. Corese peut être intégré à un serveur web sémantique grâce à la librairie Sewese/SemTag [75]. Corese a été et est utilisé dans plus d'une vingtaine d'applications[6].

[6] http://www-sop.inria.fr/teams/edelweiss/wiki/wakka.php?wiki=CoreseApplications

3.3.2 Opérationnaliser RDF(S) comme des Graphes Conceptuels

Les modèles du web et de l'Internet sont, autant que faire se peut, organisés en couches d'extension et le web sémantique n'y déroge pas (Figure 32). Le web sémantique compte actuellement cinq niveaux d'extensions bien établis :

- RDF : un modèle de triplets pour l'annotation des ressources du web, composant des graphes de description sérialisables dans une syntaxe XML ;
- RDF/S et OWL (Lite, DL, Full) : quatre langages de représentation d'ontologies offrant des expressivités approximativement imbriquées et croissantes.

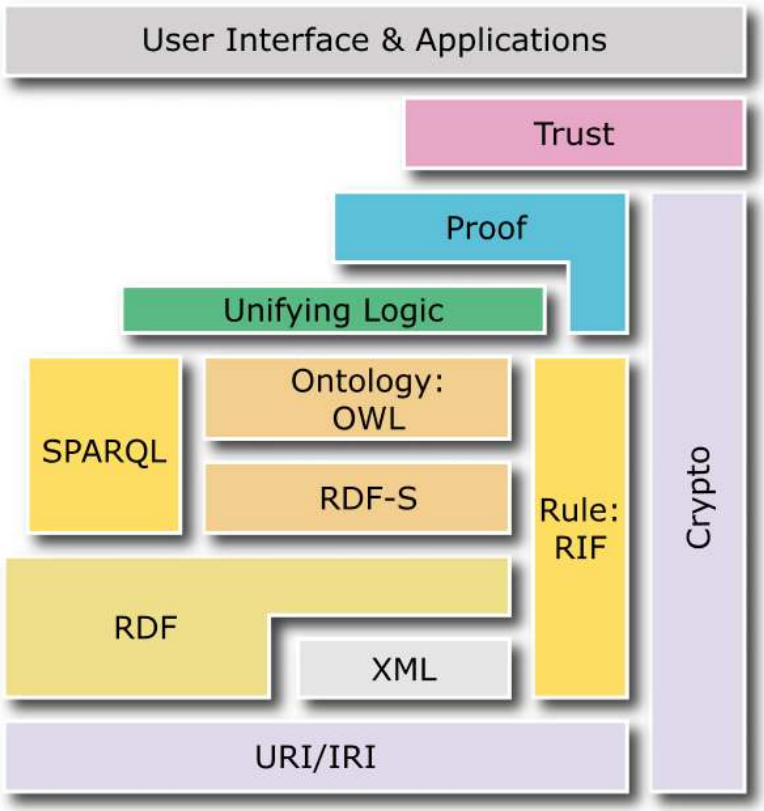


Figure 32. Les couches du web sémantique (www.w3.org 2007)

RDF, la base de la pile du web sémantique, permet de décrire des ressources avec trois aspects importants :

(1) un modèle de triplets doté d'une syntaxe XML permettant la sérialisation d'un ensemble d'assertions et fournissant un format d'échange et de stockage de ces assertions ;
(2) un modèle de prédicats binaires en logique du premier ordre, de la forme `Propriété(Ressource,Valeur)` dotant RDF d'une sémantique formelle permettant la définition de mécanismes de raisonnement ;
(3) un modèle de graphe orienté étiqueté dont les sommets sont les ressources et les valeurs, et dont les arêtes sont les assertions étiquetées par les prédicats. Ce modèle permet une interprétation intuitive des représentations et une manipulation basée

sur les opérations de graphes. C'est cet aspect des formalismes de représentation du web sémantique qui nous intéressera ici.

Le besoin de représentation de connaissances ontologiques pour assurer la cohérence sémantique des assertions a conduit au développement de RDFS (extension de RDF) et des langages OWL Lite, OWL DL, OWL Full. RDFS sert essentiellement à modéliser des connaissances factuelles, alors que OWL Lite et OWL DL sont issus des logiques de descriptions (LD) [10] et permettent la mise en œuvre des raisonnements classificatoires.

En introduisant la famille OWL DL, les standards du web sémantique ont donné une place importante aux logiques de descriptions dans les modèles de représentation du web sémantique. Les logiques de descriptions proposent des familles de langages à l'expressivité modulée au gré des constructeurs admis et de la complexité associée. Cependant, dans leur ensemble, ces langages et les algorithmes de décision associés (ex : algorithmes de tableaux) sont tournés vers la classification de concepts et l'identification d'instances [10]. Ce type de raisonnement permet de répondre à certains problèmes du web sémantique (ex : validation et complétion d'une ontologie), mais au prix de la perte de deux des trois points de vues initiaux : un format à base de triplets sans restriction, une représentation graphique intuitive de ces triplets. Par ailleurs, les LD ne permettent pas la mise en œuvre de langages de requêtes conjonctives et ne permettent pas l'expression de règles qui sont pourtant deux besoins essentiels du web sémantique en cours de standardisation (cf. SPARQL, RIF, etc.). A l'inverse, les langages utilisant différentes sortes d'appariements de graphes pour les raisonnements permettent très bien la mise en œuvre de ces opérations ainsi qu'un certain nombre d'extensions [76] [77].

Il existe une communauté de chercheurs travaillant autour de la représentation de connaissances et des raisonnements à partir de graphes. Ces chercheurs conduisent des recherches de toute nature (théorique, méthodologique, technologique, applicative) sur des formalismes de graphes issus de la famille des Graphes conceptuels (GC) [2]. Ces formalismes sont entièrement graphiques puisque toutes les connaissances sont représentées par des graphes et tous les mécanismes de raisonnement sont réalisés par des opérations de graphes (basées essentiellement sur une notion d'homomorphisme appelée projection). Ceci permet une visualisation des raisonnements et l'utilisation des propriétés structurelles des graphes pour optimiser les algorithmes et/ou définir des sous problèmes de plus faible complexité [78] [79] [71] [80], une démarche similaire étant appliquée avec succès dans le domaine très proche des réseaux de contraintes.

Par ailleurs le formalisme est logiquement fondé puisqu'à chaque type de connaissances est associée une transformation dans la logique des prédicats assurant adéquation et complétude sur le fragment équivalent de cette logique.

Les GC sont donc un formalisme qui, comme les LD, propose une famille de langages de représentations d'expressivité variable. Cette famille repose sur un autre paradigme (modèle de graphes avec une fondation logique), d'autres constructeurs (ex : lambda

expressions, graphes imbriqués, règles de graphes) et d'autres opérations (ex : projection, jointure). Ces opérations peuvent être utilisées dans d'autres parties du spectre des tâches identifiées dans les scénarios du web sémantique.

Les principes de base de CORESE sont expliqués dans [74]. Le principe est celui de l'implication logique entre une requête et une description [81] : soit un modèle de descriptions et de requêtes alors une description *D* répond à une requête *R* si la description *D* implique logiquement la requête *R* noté $D \rightarrow R$. Il s'agit alors de trouver un algorithme d'appariement de ces descriptions logiques (descriptions de documents et descriptions de requêtes) le plus efficace possible. Il changera d'un système à un autre en fonction des formalismes utilisés, de leur expressivité, et des caractéristiques des réponses que l'on veut assurer.

Le principe d'une représentation et d'une recherche à base d'ontologies est que l'implication logique dans la fonction de recherche prend en compte les connaissances ontologiques. Soit un modèle de descriptions et de requêtes alors une description *D* répond à une requête *R* en considérant une ontologie O si la description *D* et l'ontologie *O* impliquent logiquement la requête *R* noté $D \wedge O \rightarrow R$.

Dans CORESE, les modèles et représentations reposent sur le formalisme des graphes conceptuels et la fonction de recherche repose sur l'opérateur de projection de graphes. L'implication logique est rendue équivalente à un homomorphisme de graphes.

La similarité entre le langage RDF et les Graphes conceptuels a été démontrée de plusieurs façons [5] [82] [83] [78]. A titre d'exemple on peut noter que :

- dans les deux modèles il y a une distinction entre la connaissance ontologique (support des graphes conceptuels, schémas RDFS et OWL) et la connaissance assertionnelle ;
- dans les deux modèles, la connaissance assertionnelle est positive, conjonctive et existentielle et est représentée par des graphes orientés étiquetés ;
- la hiérarchie des classes (resp. propriétés) d'un schéma RDF est comparable à la hiérarchie de types de concept (resp. relation) d'un support GC ;
- les relations en RDF comme en GC sont des citoyens de premier ordre comme les classes *i.e.* elles sont déclarées en dehors des classes (resp. types de concept).
- les deux mécanismes de déduction sont équivalents : subsomption RDFS et projection de graphes conceptuels (et plus généralement homomorphisme de graphes étiquetés).

Il existe aussi un certain nombre de différences entre RDF/S et les GC. Par exemple : RDF permet la multi-instanciation qui n'a pas d'équivalent direct dans les GC et une déclaration de propriété en RDF peut indiquer plusieurs *domains* et/ou *ranges* alors que ce n'est pas le cas dans les GC simples et demande une nouvelle famille comme les GC à types conjonctifs. Inversement, les GC permettent des relations d'arité supérieure à deux alors que les graphes RDF sont binaires et les différentes familles de Graphes conceptuels proposent des extensions qui vont bien au-delà de l'expressivité de RDF/S.

CORESE propose des solutions à ces différences notamment pour la traduction de RDF en graphes conceptuels. Une multi-instanciation de *n* classes $C_1, ..., C_n$ est traduite par la création à la volée d'un type conjonctif étant, par multi-héritage, la spécialisation la plus générale des *n* classes. Un procédé identique est utilisé pour les propriétés ayant plusieurs *domains* et/ou *ranges*.

L'étude des rapports entre ces deux modèles pose donc un certain nombre de problèmes de recherche mais elle est aussi motivée par la valeur ajoutée que l'on peut trouver dans un tel rapprochement. Ainsi, on peut remarquer que l'opération de projection définie dans les graphes conceptuels peut être utilisée dans le cadre du web sémantique pour fournir un algorithme exploitant les connaissances ontologiques dans la recherche d'information afin d'améliorer la précision (ex : utilisation des structures de typage) et le rappel (ex : utilisation des implications logiques de la subsomption de types) [84]. La projection est le couteau suisse des raisonnements dans les graphes conceptuels. Elle remplace l'implication logique de recherche d'information par une opération de spécialisation de graphes :

$$D \wedge O \rightarrow R \Leftrightarrow G_1 \leq G_2$$

En d'autres termes, un graphe conceptuel G_1 (et par extension un graphe RDF) implique logiquement un graphe conceptuel G_2 ssi G_1 est une spécialisation de G_2 noté $G_1 \leq G_2$. De plus on dit qu'un graphe G_1 est une spécialisation de G_2 ssi il existe une projection de G_2 dans G_1 telle que tous les nœuds concepts et relation de G_2 sont projetés sur des nœuds de G_1 dont le type est le même ou une spécialisation selon l'ontologie spécifiée.

Le principe formel simplifié est alors le suivant pour un ensemble de triplets et un schéma RDFS donné :

- un triplet RDF est traduit par une relation binaire de graphes conceptuels :
  (s, p, o) ⇒ [s]→(p)→[o]
  (avec *s* et o respectivement premier et deuxième argument de *p*)
- une relation *sub-class of* ou *sub-property of* est traduite par une spécialisation de labels :
  $(C_1, subClassOf, C_2) \Rightarrow C_1 \leq C_2$
  $(P_1, subPropertyOf, P_2) \Rightarrow P_1 \leq P_2$

Soit alors un graphe conceptuel orienté, étiqueté et bipartite $G = (C,R,E,l)$ obtenu ainsi où *C* et *R* sont respectivement ses ensembles de nœuds concepts et nœuds relations, *E* est l'ensemble des arcs et *l* est une fonction d'étiquetage des nœuds concepts et nœuds relations respectivement par un couple *(type, marqueur)* pour les concepts et par un type de relation pour les relations.

Les types sont issus de hiérarchies construites par les relations de spécialisation. Le marqueur est soit un identifiant unique soit le marqueur générique * tel que pour tout couple d'identifiants uniques i_1, i_2 on a $i_1 \leq i_2$ ssi $i_1 = i_2$; pour tout identifiant *i* on a $i \leq *$ et pour tout couple *(t, m)* on a $(t_1, m_1) \leq (t_2, m_2)$ ssi $t_1 \leq t_2$ et $m_1 \leq m_2$.

Dans ce cadre simplifié, la projection d'un graphe RDF *G* sur un graphe RDF *H* peut alors se définir comme la projection des deux graphes traduits en graphes conceptuels $G = (C_G,R_G,E_G,l_G)$ et $H = (C_H,R_H,E_H,l_H)$ *i.e.* une fonction *M* de C_G dans C_H et de R_G dans R_H telle que :

- l'adjacence et l'ordre des arcs soient préservés : $\forall(r,c)\in E_G$ on a $(M(r),M(c)) \in E_H$ et si *c* est le i^{ème} voisin de *r* dans *G* alors *M(c)* est le i^{ème} voisin de *M(r)* dans *H*
- la compatibilité des étiquettes soit préservée : $\forall x\in C_G\cup R_G \;\; l_H(M(x)) \leq l_G(x)$

Cette opération de projection fournit non seulement une opérationnalisation d'un moteur de recherche en RDF mais elle peut aussi être utilisée dans un moteur de règles pour opérationnaliser la recherche de prémisses vérifiées. [74]

3.3.3 Etendre les modèles RDF(S) en s'inspirant des graphes conceptuels

En considérant la structure de graphe de RDF, nous pouvons proposer un certain nombre d'extensions qui ne sont pas nécessairement évidentes lorsque l'on considère uniquement l'interprétation logique des triplets RDF et qui, pourtant, sont très utiles dans plusieurs de nos scénarios. A titre d'exemple, on peut citer : la notion de graphe nommé [85] qui peut être utilisée pour représenter des contextes [73] [72], la recherche approchée [74], le parcours de chemins [72], la génération d'index [86] [24], etc.

Nous détaillerons ici une proposition que nous avons faite pour la déclaration de la source (provenance) de triplets RDF sérialisés en RDF/XML [87] [88]

Lorsque l'on effectue des requêtes ou des raisonnements sur les données et métadonnées du web sémantique, les sources de ces métadonnées peuvent être d'une grande importance.

Dans une requête SPARQL sur une collection de graphes, le mot-clé GRAPH est utilisé pour restreindre la recherche à des graphes nommés particuliers. Toutefois, le modèle de données RDF met l'accent sur l'expression de triplets avec un sujet, un prédicat et un objet mais ni ce modèle ni sa syntaxe RDF / XML ne fournissent un mécanisme permettant de préciser la source de chaque triplet. Un moyen typique serait une syntaxe XML pour associer aux triplets encodés en RDF/XML une IRI (International Resource Identifier, une version internationalisée des URI) précisant leur origine.

C'est ce que nous proposons dans [87] [88] : une extension minimale de la syntaxe (un seul attribut) permettant de préciser pour les triplets représentés en RDF / XML la source qui doit leur être associée.

Dans cette section, nous utiliserons les espaces de nommages et préfixes suivants :

```
rdf     http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs    http://www.w3.org/2000/01/rdf-schema
cos     http://www.inria.fr/acacia/corese#
dc      http://purl.org/dc/elements/1.1/
foaf    http://xmlns.com/foaf/0.1/
```

En SPARQL, l'utilisation du mot-clef GRAPH peut fournir une IRI pour sélectionner un graphe ou utiliser une variable qui sera, dans la requête, successivement liée aux IRI de tous les graphes nommés des données RDF. La requête en Figure 33 recherche deux

motifs de graphes dans les graphes nommés de la base et renvoie des solutions où les variables `?srcname` et `?srctitle` sont liées aux IRI des graphes correspondants.

Malheureusement, la syntaxe de SPARQL pour les sources n'a pas d'équivalent dans la syntaxe RDF/XML. Nous proposons donc un mécanisme possible pour normaliser la déclaration des sources dans un graphe RDF sérialisé en RDF / XML.

En utilisant le moteur SPARQL Corese, nous avons mis en place et testé une extension de la syntaxe RDF / XML : un attribut `cos:graph` peut être inséré dans un document RDF / XML pour préciser l'IRI d'une source. La valeur de cet attribut est interprétée comme une référence IRI.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?srcname ?name ?srctitle ?title
WHERE
  {
    GRAPH ?srctitle
    {
      ?doc dc:title ?title .
      ?doc dc:creator ?author
    }
    GRAPH ?srcname  {  ?author foaf:name ?name  }
  }
```

Figure 33. Requête SPARQL utilisant la source des triplets ou graphes nommés.

L'IRI[7] de la source d'un triplet est :

1. la source spécifiée par l'IRI d'un attribut `cos:graph` sur l'élément XML encodant ce triplet, si celui-ci est spécifié, sinon
2. la source spécifiée par l'IRI de l'élément parent de celui encodant ce triplet ; source obtenue à l'issue de l'application récursive des mêmes règles au parent, sinon
3. l'IRI de la base du document.

L'IRI de la base d'un document est déterminé par la RFC 2396 [89], à savoir que la base est l'IRI utilisé pour récupérer le document entité ou entité externe. En d'autres termes, si aucune source n'est spécifiée, l'URL du document RDF / XML sérialisant le graphe RDF est utilisée comme la source par défaut.

La portée de la déclaration d'une source s'étend du début de la balise ouvrante dans laquelle il apparaît jusqu'à la fin de la balise fermante correspondante, à l'exclusion des portées des déclarations de sources contenues à l'intérieur de cette balise. Une telle déclaration s'applique à tous les éléments et attributs dans sa portée. Dans le cas d'une balise vide, le champ d'application est la balise elle-même.

[7] IRI : International Resource Identifier, une version internationalisée des URI

Une seule source peut être déclarée comme attribut sur un élément.

Ainsi, l'attribut `cos:graph` peut être utilisé sur n'importe quel élément nœud ou élément propriété du graphe pour indiquer que le contenu est associé à une IRI source ; la source la plus spécifique de la portée actuelle étant systématiquement appliquée.

Nous autorisons expressément les sources à `null` : la forme `cos:graph=""` indique l'absence de source, de sorte que la source associée est explicitement nulle et même l'IRI de base du document ne sera pas considérée.

Considérons le graphe RDF Figure 34 qui indique qu'une ressource a un titre « RDF Source » et un créateur et que ce créateur est de type personne et a un nom « Fabien Gandon » et une boîte aux lettres « mailto:fgandon@inria.fr ». La sérialisation de ce graphe est donnée par la Figure 35 et les triplets correspondants sont donnés par la Figure 36.

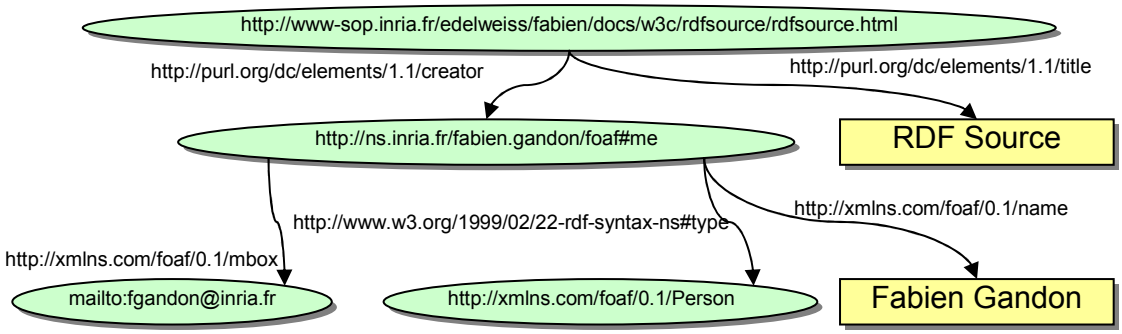


Figure 34. Un exemple de graphe RDF classique

```
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
 <rdf:Description                                    rdf:about="http://www-
sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html">
  <dc:title>RDF Source</dc:title>
  <dc:creator>
    <foaf:Person rdf:about="http://ns.inria.fr/fabien.gandon/foaf#me">
     <foaf:name>Fabien Gandon</foaf:name>
     <foaf:mbox rdf:resource="mailto:fgandon@inria.fr"/>
   </foaf:Person>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

Figure 35. Un exemple de sérialisation du graphe RDF en Figure 34

```
<http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html>
          dc:title   "RDF Source"
<http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html>
          dc:creator <http://ns.inria.fr/fabien.gandon/foaf#me>
<http://ns.inria.fr/fabien.gandon/foaf#me>      rdf:type   foaf:Person
<http://ns.inria.fr/fabien.gandon/foaf#me>      foaf:name  "Fabien Gandon"
<http://ns.inria.fr/fabien.gandon/foaf#me>      foaf:mbox
     <mailto:fgandon@inria.fr>f:mbox rdf:resource="mailto:fgandon@inria.fr"/>
```

Figure 36. Triplets correspondants au graphe RDF en Figure 34

L'extension de la syntaxe RDF / XML proposée ici transforme les triplets en quadruplets avec un quatrième terme étant l'IRI de la source du triplet. En termes de graphes, on peut considérer être dans un modèle d'hypergraphe où les arcs deviennent des hyperarcs avec un sommet supplémentaire correspondant à l'URI de la source. Examinons l'exemple précédent augmenté de deux occurrences de l'attribut `cos:graph` comme montré en Figure 37. Il en résulte que tous les triplets à propos de la personne sont associés à la source `http://www.inria.fr` y compris la déclaration de type `foaf:Person`.

```
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cos="http://www.inria.fr/acacia/corese#"
  cos:graph="http://www.w3.org">
 <rdf:Description rdf:about="http://www-
sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html">
  <dc:title>RDF Source</dc:title>
  <dc:creator>
    <foaf:Person rdf:about="http://ns.inria.fr/fabien.gandon/foaf#me"
      cos:graph="http://www.inria.fr">
     <foaf:name>Fabien Gandon</foaf:name>
     <foaf:mbox rdf:resource="mailto:fgandon@inria.fr"/>
   </foaf:Person>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

Figure 37. Un exemple de sérialisation avec des déclarations de source

```
<http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html>
   dc:title "RDF Source" <- http://www.w3.org
<http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html>
   dc:creator <http://ns.inria.fr/fabien.gandon/foaf#me> <- http://www.w3.org
<http://ns.inria.fr/fabien.gandon/foaf#me> rdf:type foaf:Person
    <- http://www.inria.fr
<http://ns.inria.fr/fabien.gandon/foaf#me> foaf:name "Fabien Gandon"
    <- http://www.inria.fr
<http://ns.inria.fr/fabien.gandon/foaf#me> foaf:mbox <mailto:fgandon@inria.fr>
    <- http://www.inria.fr
```

Figure 38. Les quadruplets obtenus à partir de la sérialisation en Figure 37

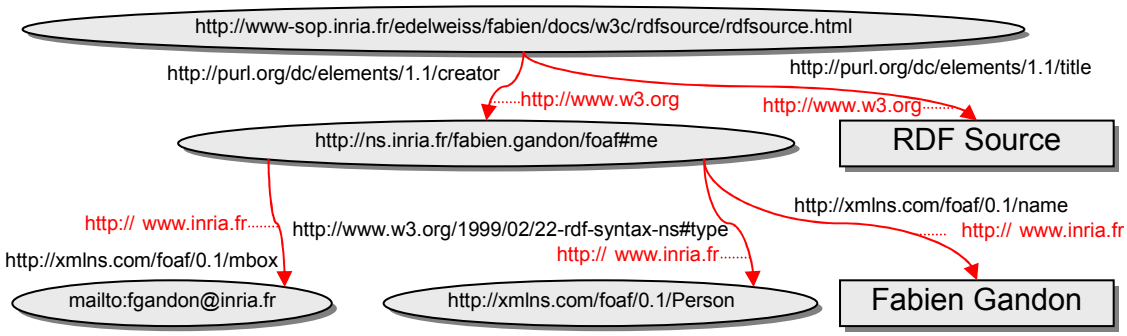


Figure 39. Graphe obtenu à partir de la sérialisation en Figure 37

Notons cependant qu'il est dangereux de changer de sources autour des nœuds anonymes (*blank nodes* ou quantification existentielle) : un nœud anonyme ne peut appartenir qu'à une seule source ; le changement de source sur les propriétés d'un nœud anonyme engendre la division du nœud anonyme en plusieurs nœuds anonymes, un pour chaque source comme dans l'exemple en Figure 40 où la balise `<foaf:Person>` ne contient aucun attribut d'identification (*about* ou *id*) et est donc un nœud anonyme.

```
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cos="http://www.inria.fr/acacia/corese#"
  cos:graph="http://www.w3.org">
 <rdf:Description rdf:about="http://www-
sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html">
  <dc:title>RDF Source</dc:title>
  <dc:creator>
    <foaf:Person>
     <foaf:name>Fabien Gandon</foaf:name>
     <foaf:mbox rdf:resource="mailto:fgandon@inria.fr"
                        cos:graph="http://www.inria.fr"/>
   </foaf:Person>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

Figure 40. Un exemple déclaration de source sur la propriété d'un nœud anonyme

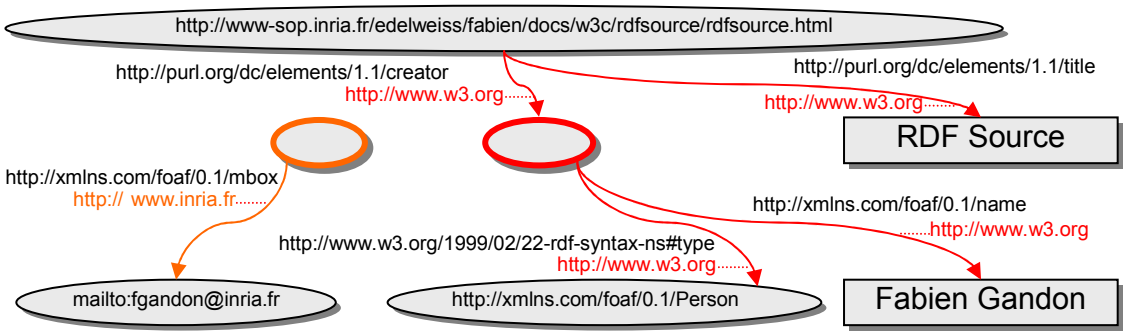


Figure 41. Graphe RDF sérialisé en Figure 40

```
<http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html> dc:title "RDF Source"
       <- http://www.w3.org
<http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html> dc:creator _:a
       <- http://www.w3.org
_:a rdf:type foaf:Person <- http://www.w3.org
_:a foaf:name "Fabien Gandon" <- http://www.w3.org
_:b foaf:mbox <mailto:fgandon@inria.fr> <- http://www.inria.fr
```

Figure 42. Quadruplets générés par le RDF sérialisé en Figure 40

L'ensemble des cas problématiques, et notamment les Containers, est étudié ailleurs [87] [88]. Dans ces références, nous passons aussi en revue des cas d'utilisation spéciaux comme l'utilisation dans des déclarations de schéma et leur documentation

dont des exemples sont donnés en Figure 43 (la définition d'une propriété vient d'une autre source) et Figure 44 (un commentaire a été ajouté par une autre source pour attester d'une validation).

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:cos="http://www.inria.fr/acacia/corese#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns="http://www.w3.org/2000/01/rdf-schema#"
         cos:graph="http://ns.inria.fr/2006/05/research_onto.rdfs"
         xml:base="http://ns.inria.fr/2006/05/research_onto.rdfs">

  <Class rdf:ID="Man">
    <subClassOf rdf:resource="#Person"/>
    <subClassOf rdf:resource="#Male"/>
    <label xml:lang="en">man</label>
    <comment xml:lang="en">an adult male person</comment>
  </Class>

  <rdf:Property rdf:about="http://xmlns.com/foaf/0.1/name"
    cos:graph="http://xmlns.com/foaf/0.1/"
    rdfs:label="name" rdfs:comment="A name for some thing.">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
    <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/>
    <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2000/01/rdf-
schema#label"/>
  </rdf:Property>

</rdf:RDF>
```

Figure 43. Spécification d'une source extérieure pour la déclaration d'une propriété

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:cos="http://www.inria.fr/acacia/corese#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns="http://www.w3.org/2000/01/rdf-schema#"
         cos:graph="http://ns.inria.fr/2006/05/research_onto.rdfs"
         xml:base="http://ns.inria.fr/2006/05/research_onto.rdfs">

<owl:Ontology rdf:about="http://ns.inria.fr/2006/05/research_onto.rdfs">
   <rdfs:label>research ontology</rdfs:label>
   <rdfs:comment>defines primitives to describe research
activities</rdfs:comment>
   <rdfs:comment cos:graph="http://www.w3.org">RDFS_VALID</rdfs:comment>
   <owl:versionInfo>1.3</owl:versionInfo>
</owl:Ontology>

</rdf:RDF>
```

Figure 44. Annotation du graphe RDF de l'ontologie en identifiant sa source

Cette spécification a été construite à partir de cas d'utilisation de plusieurs de nos projets. Toutefois, il est un cas que nous n'avons considéré : le cas où l'on veut associer plusieurs sources à un triplet. Nous n'avons pas trouvé une bonne syntaxe pour ce cas et nous ne savons pas ce que cela impliquerait pour les *blank nodes* en termes de requêtes SPARQL.

3.3.4 Factoriser les modèles de graphes et leurs manipulations

Outre l'opportunité de considérer de nouvelles fonctionnalités, le rapprochement graphes conceptuels et RDF/S est aussi motivé par un point commun aux deux langages : une représentation des connaissances reposant sur un modèle de graphes. Cette structure de graphe offre des opportunités d'optimisation de l'indexation et de l'organisation des connaissances ainsi qu'un espace permettant de définir des inférences dépassant le raisonnement logique ; on peut par exemple exploiter les hiérarchies de types des ontologies pour définir des espaces métriques dont les distances sont ensuite utilisées dans des inférences de relaxation de contraintes dans des requêtes de recherche d'information approchée ; voir [74] et le chapitre 4.

Les applications actuelles citées en première section et les travaux de nos équipes sœurs de par le monde recouvrent de multiples domaines (annotation, recherche et réorganisation de ressources audiovisuelles, assistance à la conception en mécanique, aide à la décision en agroalimentaire, aide à l'échange de résultats entre biologistes, intégration de services et informations, gestion de ressources culturelles, etc.).

Dans toutes ces applications, les connaissances peuvent être représentées par des graphes et les raisonnements peuvent être mis en œuvre par des opérations de graphes. C'est pourquoi nous avons identifié un nouvel objectif pour cette communauté qui serait la réalisation d'une plate-forme logicielle permettant l'abstraction aussi bien des différents types de graphes utilisés (dépendant du langage formel utilisé par l'application) que des différents raisonnements nécessaires, tout en assurant de nombreux services génériques : algorithmes efficaces, persistance et distribution, visualisation et édition, etc. Le but serait ainsi de pouvoir développer à faible coût ces différentes applications, tout en bénéficiant de la forte valeur ajoutée de la plate-forme. La généricité nous assurerait également que les développements spécifiques à une application puissent être réutilisés.

Depuis un an et demi, le projet Griwes[8] [90] vise à la spécification d'une plate-forme générique de représentation de connaissances et de raisonnements à base de graphes, une plate-forme commune générique reposant sur une abstraction de plusieurs modèles de graphes (RDF, Graphes conceptuels) et diverses extensions de ces modèles motivées par des scénarios applicatifs rencontrés par chaque équipe participante.

Les équipes de ce projet s'intéressent à des langages de représentation de connaissances et de raisonnement à base de graphes ou représentables par des graphes comme les graphes conceptuels, les *topic maps*, les formalismes du web sémantique, le langage de modélisation par objets UML, les bases de données relationnelles, les réseaux de contraintes, etc. Ces langages sont étudiés d'un point de vue théorique (complexité, algorithmes, relations entre langages) et sont implémentés dans des outils qui sont utilisés dans de nombreux domaines applicatifs.

[8] Projet de l'appel COLOR, Griwes, http://www.inria.fr/sophia/acacia/project/griwes/

Utilisant initialement les graphes conceptuels de Sowa [26] comme langage de base, les outils de ces équipes s'adaptent aujourd'hui à de nouveaux langages. Cependant, rien n'assure que les différents modules d'extension conçus sur ces outils puissent interagir, que ce soit au sein des équipes concernées ou entre ces équipes. De plus un certain nombre de traitements et d'optimisations de ces différents modules pourraient être factorisés permettant ainsi de toujours bénéficier des dernières avancées.

Pour répondre à ce frein au développement, les équipes souhaitent maintenant "penser la généricité" en la mettant au cœur des versions futures de leurs outils ce qui les conduit naturellement à envisager le développement d'une plate-forme commune unique mais modulaire : la mise en commun de nos expériences et de nos compétences est un premier atout, et la diversité de nos domaines applicatifs ne peut que bénéficier à notre objectif de généricité. La mise en œuvre reste un problème ouvert.

Les réunions de travail mensuelles ont amené les participants à raffiner une architecture multi niveaux (distinction d'un noyau « graphes et opérations de graphes », d'un niveau « représentation de connaissances », d'un niveau « langages et stratégies » et d'un niveau « interfaces utilisateurs et interfaces programmatiques »). Une telle plate-forme serait un outil fédérateur pour notre communauté de recherche sur les représentations des connaissances à base de graphes. Ce projet permettrait d'atteindre une masse critique nécessaire dans un outil ouvert factorisant des développements jusque là indépendants.

Nous reprenons dans la suite des résultats de l'article et du rapport de recherche publiés dans le cadre de Griwes [90] et résumons les résultats obtenus par les deux équipes (Edelweiss, INRIA et RCR, LIRMM) qui, dans ce cadre, collaborent au rapprochement de leurs modèles et outils.

Le travail de rapprochement que nous effectuons dans Griwes est essentiellement focalisé sur les deux premiers niveaux d'abstraction d'un modèle commun dont la spécification est la plus avancée :

Structures mathématiques pour les Graphes : il s'agit du noyau de l'architecture qui spécifie les graphes et objets mathématiques fondamentaux (ex : les préordres) utilisés dans la caractérisation de primitives de représentation de connaissances (ex : hiérarchie de types). Ce niveau devra assurer la persistance, la distribution et les traitements de bas niveaux des graphes avec une efficacité algorithmique maximale. Le modèle que nous avons choisi repose sur des hypergraphes étiquetés et une famille de *mappings* associés.

Structures de représentation dans une base de connaissances : ce niveau peut être vu comme un langage de représentation de connaissances abstrait, en dehors de toute syntaxe particulière. On s'intéresse ici à des constructions à partir de graphes qui auront des statuts différents dans une base de connaissances, par exemple : des faits, des règles, des contraintes, des requêtes, des hiérarchies, etc. Ce niveau servira de base à la définition des algorithmes plus évolués : application d'une règle, test de déduction, test de généralisation, recherche d'une ou de toutes les réponses à une requête, etc.

Bien que stables, ces niveaux demanderont encore à être revus et validés notamment par des experts extérieurs au projet et, dans leur forme actuelle, laissent un certain nombre de questions ouvertes quant à la distribution des graphes, aux typages et comparateurs élémentaires, aux techniques d'indexation performantes, aux algorithmes et structures de données efficaces correspondant à ces opérateurs et objets mathématiques, etc.

Le niveau encore peu étudié est celui des **langages et stratégies :** c'est à ce niveau que se définissent les différents langages de représentation de connaissances (ex : RDFS, graphes conceptuels, etc.) et les stratégies d'inférences qui leur sont appliquées (ex : faire de la validation, faire de la dérivation etc.).

Les aspects encore non étudiés sont les suivants :

Les interfaces programmatiques : ce niveau gère la communication de la plate-forme avec d'autres outils, que ce soit des outils de développement ou des applications spécifiques, à travers des interfaces programmatiques spécifiées (API) et des protocoles standards (ex : SOAP).

Les interfaces utilisateurs : on peut associer une ou plusieurs méthodes de visualisation à des expressions de chaque langage. Mais l'interface utilisateur ne se limite pas à des connaissances représentées par des graphes : pour certains types de connaissances, il peut être plus intuitif de les lire ou de les éditer sous forme de tables ou de formulaires. Ce besoin sera pris en compte.

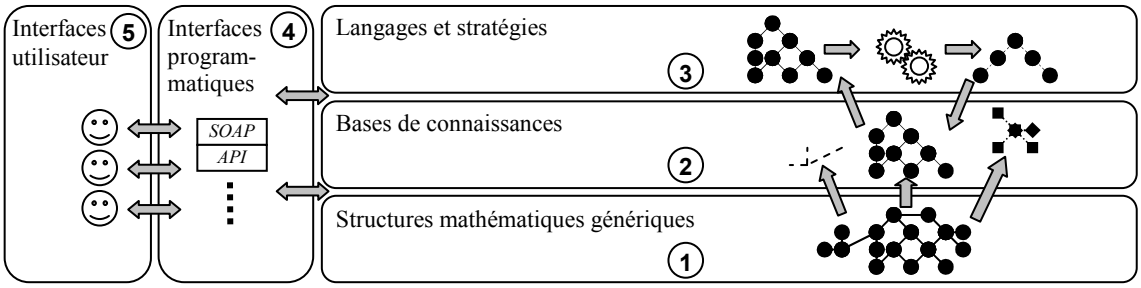


Figure 45. Architecture de Griwes

Par la suite, nous allons donner quelques détails sur les couches basses 1 et 2 et renvoyons le lecteur au rapport de recherche disponible sur le site de Griwes pour plus de détails. La couche 1, dite couche de structure est la couche centrale de l'architecture de Griwes. Nous mentionnerons ici quelques unes des définitions données dans cette couche et destinées à caractériser des primitives génériques de représentation de connaissances.

La primitive centrale est l'ERGraph pour Graphe Entité-Relation. Il sert à décrire un ensemble d'entités et les relations entre ces entités. Une entité est tout ce qui peut faire l'objet d'une représentation conceptuelle. Une relation peut représenter une propriété d'une entité ou peut porter sur deux entités ou plus.

Les relations peuvent avoir n'importe quel nombre d'arguments (relations d'arité quelconque), y compris zéro, et ces arguments sont totalement ordonnés. En termes de théorie des graphes, un ERGraph est un hypergraphe orienté, où les sommets représentent les entités et les hyperarcs représentent les relations de ces entités.

Toutefois, un hypergraphe a une représentation graphique naturelle associée : un graphe bipartite, où les deux types de nœuds représentent respectivement les entités et les relations, et les arêtes reliant un nœud relation à un nœud entité représentent les arguments de la relation, les arcs incidents à un nœud relation sont totalement ordonnés selon l'ordre des arguments de la relation.

Les nœuds (Entités) et hyperarcs (relations) d'un ERGraph ont des étiquettes. Au niveau de la structure, ce ne sont que des éléments d'un ensemble L, qui peut être défini en intension ou en extension. Les étiquettes obtiennent un sens au niveau 2, celui des primitives de représentation de connaissances.

Définition d'un ERGraph : Un ERGraph défini par rapport à un ensemble d'étiquettes L est un quadruplet $G=(E_G, R_G, n_G, l_G)$ où

- E_G et R_G sont deux ensembles finis disjoints respectivement de nœuds entités et de nœuds relations ou hyperarcs.
- $n_G : R_G \rightarrow E_G^*$ associe à chaque relation un tuple fini d'entités appelées les arguments de la relation. Si $n_G(r)=(e_1,...,e_k)$ nous notons $n_G^i(r)=e_i$ le $i^{ème}$ argument de r.
- $l_G : E_G \cup R_G \rightarrow L$ est une fonction d'étiquetage des nœuds entités et des nœuds relations.

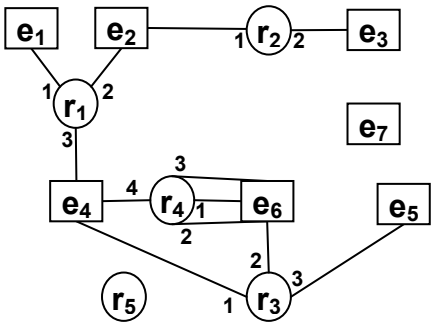


Figure 46. Exemple de graphe bipartite représentant un ERGraph

Pour certaines primitives de représentation des connaissances et certains algorithmes, il est utile de distinguer certaines entités d'un graphe. A cette fin, nous définissons une deuxième primitive de base, appelée λ–ERGraph.

Définition d'un λ-ERGraph : Un λ-ERGraph λ_G est le couple formé d'un ERGraph G et d'un tuple d'entités de G : $\lambda_G = ((e_1,...e_k), G), e_i \in E_G$. Nous disons alors que k est la taille de λ_G et que les $e_1,...e_k$ sont les entités distinguées de G.

Définition d'un sous-ERGraph induit : Soit $G=(E_G, R_G, n_G, l_G)$ un ERGraph. Soit $E_{G'}$ un sous ensemble de E_G. Le sous-ERGraph de G induit par $E_{G'}$ est l'ERGraph $G'=(E_{G'}, R_{G'}, n_{G'}, l_{G'})$ défini par :

(1) $R_{G'}= \{ r \in R_G \mid \forall 1 \leq i \leq card(n_G(r)) , n_G^i(r) \in E_{G'} \}$

(2) $n_{G'}$ est la restriction de n_G à $R_{G'}$

(3) $l_{G'}$ est la restriction de l_G à $E_{G'} \cup R_{G'}$

Définition d'une fusion : Soit $G=((g_1,...g_k), G')$ et $H=((h_1,...h_k), H')$ deux λ-ERGraphs de même taille, la fusion de H dans G modifie G' en ajoutant une copie $C(H')$ de H' à G' en fusionnant pour tout $1 \leq i \leq k$ les entités $C(h_i)$ et g_i. Il est à noter que les étiquettes des deux entités fusionnées sont obtenues en appliquant une méthode définie aux couches supérieures.

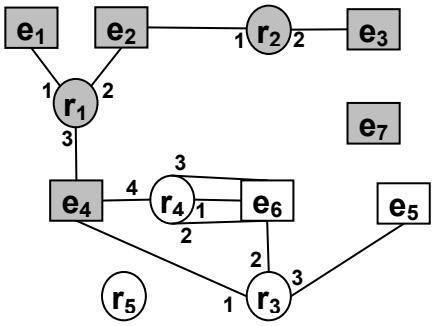


Figure 47. Sous-ERGraph induit par $E_{G'}=\{e_1, e_2, e_3, e_4, e_7\}$

Intuitivement, un *mapping* associe (met en correspondance) des entités d'un ERGraph requête à des entités d'un ERGraph dans une base de connaissances d'ERGraphs. Le *mapping* des entités d'un graphe est une opération fondamentale pour la comparaison et le raisonnement avec des ERGraphs ; c'est l'opération de base utilisée par les requêtes et les règles.

Définition d'un EMapping : Soient G et H deux ERGraphs, un EMapping de H dans G est une fonction partielle M de E_H dans E_G *i.e.* une relation binaire (ou fonction) qui associe chaque élément de E_H avec au plus un élément E_G ; tous les éléments de E_H ne sont pas nécessairement associés à des éléments de E_G.

Le *mapping* est une opération de base utilisée dans de nombreuses opérations plus complexes, comme les règles. Notons que par défaut un *EMapping* est partiel. Cela nous permet de manipuler et de raisonner sur des *EMappings* au cours du processus de *mapping* des graphes alors même qu'il n'est pas terminé. Le cas échéant, une fois ce processus terminé, le *EMapping* est dit total si toutes les entités du graphe requête H sont *mappées*. En général nous utilisons des *mappings* qui permettent de préserver certaines caractéristiques choisies des graphes (ex : la compatibilité des étiquettes, des informations structurelles, etc.); la Figure 48 montre la hiérarchie des *mappings* définis dans Griwes. Nous ne donnerons pas ici toutes les définitions disponibles dans le rapport du projet.

En particulier, un *ERMapping* contraint la structure des graphes qui sont mis en correspondance et un $EMapping_{<X>}$ contraint les étiquettes des entités mises en correspondances dans les graphes *mappés i.e.* les images du graphe requête dans les graphes cibles. Un *ERMapping* est un *EMapping* qui assure aussi que chaque relation de H est mise en correspondance avec une relation de G ayant la même arité. Un $EMapping_{<X>}$ est un *EMapping* qui satisfait une relation de compatibilité X sur les étiquettes des entités *mappées*. Un $ERMapping_{<X>}$ est à la fois un *ERMapping* et un $EMapping_{<X>}$. Un homomorphisme est un *ERMapping* total. D'autres caractéristiques

intéressantes à considérer sont : les *mappings* injectifs, les *mappings* surjectifs, les *mapping* fidèles (préserver l'absence d'hyperarcs), etc.

Définition d'un ERMapping : Soient *G* et *H* deux ERGraphs, un ERMapping de *H* dans *G* est un EMapping *M* de *H* dans *G* tel que : Soit *H'* un sous-ERGraph de *H* induit par $M^{-1}(E_G)$, $\forall r' \in R_{H'}\ \exists r \in R_G$ tel que $card(n_{H'}(r')) = card(n_G(r))$ et $\forall\ 1 \leq i \leq card(n_G(r))$, $M(n_{H'}{}^{i}(r')) = n_G{}^{i}(r)$. Nous appelons *r* le support de *r'* dans *M* et notons $r \in M(r')$

Pour information, la projection en graphes conceptuels correspond alors à un $Homomophism_{<X>}$ où *X* est un préordre sur *L* l'ensemble des étiquettes.

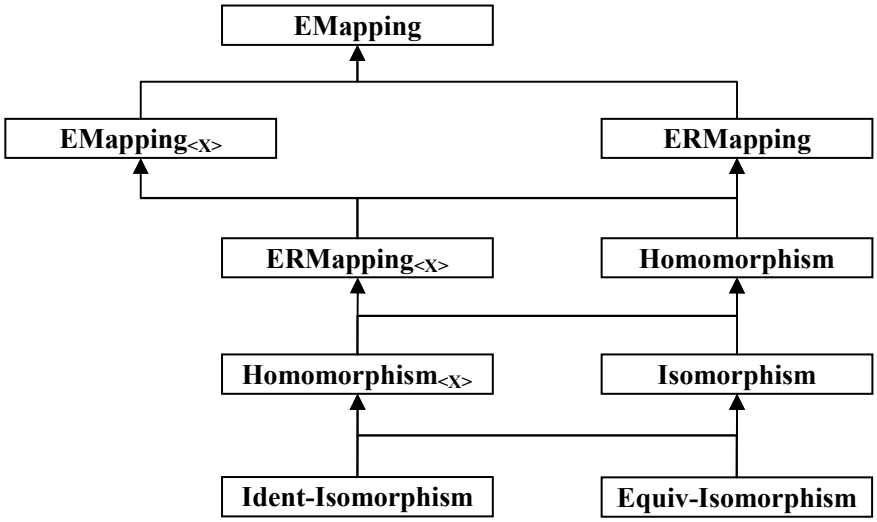
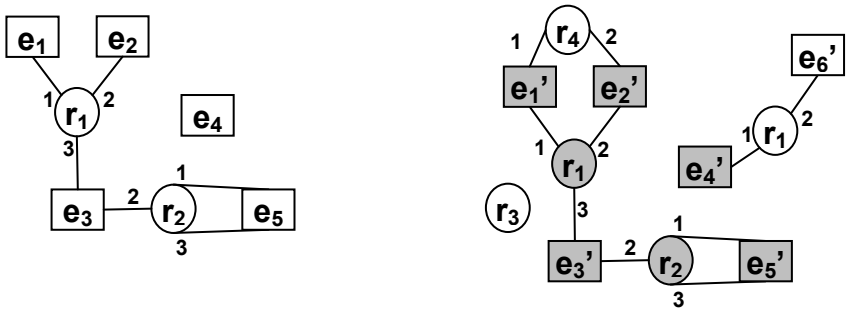


Figure 48. Hiérarchie des *mappings* définis dans Griwes



ERGraph *H* ERGraph *G*

Figure 49. Un homomorphisme de *H* dans *G*

Nous définissons la preuve d'un *mapping* comme une forme de "réification" du *mapping*. Une preuve fournit une vue statique du fonctionnement dynamique du *mapping*, une photo, permettant ainsi d'accéder à l'information relative à l'état du *mapping*. La preuve formelle d'un *mapping* est un ensemble d'associations détaillant exactement l'association de chaque entité et chaque relation du graphe requête *H* à des entités et relations du graphe cible *G*.

Nous suivons la hiérarchie des *mappings* décrite précédemment et associons à chaque type de *EMapping* une notion de preuve : *EProof*, *ERProof* et $ERProof_{<X>}$. Par exemple la preuve d'un homomorphisme correspond à la preuve d'un total $ERMapping_{<X>}$ où X est un préordre sur l'ensemble des étiquettes *L* et défini comme suit :

Définition d'une EProof : Soient *G* et *H* deux *ERGraphs*, et *M* un *EMapping* de *H* dans *G*. La EProof de *M* est l'ensemble $M_E = \{ (e_H,e_G) \in E_H \times E_G \mid e_G=M(e_H) \}$.

Définition d'une ERProof : Soient *G* et *H* deux *ERGraphs*, et *M* un *ERMapping* de *H* dans *G*. Soit *H'* un sous-ERGraph de *H* induit par $M^{-1}(E_G)$. Une *ERProof* de *M* est un couple $P=(M_E,M_R)$ où M_E est la EProof de *M* et $M_R= \{(r_1,r'_1),\dots (r_k,r'_k)\}$ avec $\{r_1,\dots,r_k\}=R_{H'}$ et $\forall 1 \le i \le k\ r'_i \in M(r_i)$.

Définition d'une EProof$_{<X>}$: Soient *G* et *H* deux *ERGraphs*, et *M* un *EMapping*$_{<X>}$ de *H* dans *G*. Une *EProof*$_{<X>}$ de *M* est un ensemble $M_{EX}= \{(e_1,e'_1,p_1)\dots (e_k,e'_k,p_K)\}$ où $\{(e_1,e'_1)\dots (e_k,e'_k)\}$ est l'*EProof* de *M* et $\forall 1 \le i \le k\ p_i$ est une preuve de $(l_G(M(e)), l_H(e)) \in X$.

Notons qu'à ce stade, aucune supposition n'est faite sur la structure des preuves p_i.

Définition d'une ERProof$_{<X>}$: Soient *G* et *H* deux *ERGraphs*, et *M* un *EMapping* de *H* dans *G*. Une *ERProof*$_{<X>}$ de *M* est un couple $P=(M_{EX},M_{RX})$ où M_{EX} est l'*EProof*$_{<X>}$ de *M* et $M_{RX}= \{(r_1,r'_1,p_1)\dots (r_k,r'_k,p_K)\}$ où $\{(r_1,r'_1)\dots (r_k,r'_k)\}$ est le deuxième élément de l'*ERProof* de *M* et $\forall 1 \le i \le k\ p_i$ est une preuve de $(l_G(M(r)), l_H(r)) \in X$.

Un système de contraintes pour un *EMapping* est une fonction $\mathcal{C}$ qui fixe des conditions supplémentaires qu'un *EMapping* doit satisfaire pour être correcte. Ce système prend la forme d'une expression évaluable qui doit être évaluée à vrai pour que le *EMapping* considéré satisfasse la contrainte.

Définition d'un système de contraintes de EMapping : Un système de contraintes pour un *EMapping M* de *H* dans *G* est une fonction $\mathcal{C}(E)$ où *E* est le triplet *(H,P,V)* appelé environnement, avec *P* la preuve de *M* et *V* une relation binaire associant à des variables v_i une entité ou une relation unique de *H*. Cette fonction peut prendre les valeurs *{true, false, unknown, error}*.

Un *EMapping M* satisfait (resp. viole) un système de $\mathcal{C}$ si $\mathcal{C}(M)=true$ (resp. si $\mathcal{C}(M)=false$).

Cette partie des spécifications a été motivée par des scénarios utilisant des langages de requête expressifs comme SPARQL. Par exemple, considérons la requête SPARQL suivante et en particulier sa clause FILTER en ligne 7:

```
1. PREFIX inria: <http://www.inria.fr#>
2. SELECT ?student ?name
3. WHERE {
4.  ?student rdf:type inria:Student
5.  ?student inria:name ?name .
6.  ?student inria:age ?age .
7.  FILTER (xsd:integer(?age) > 22 && regex(?name, "^A")) }
```

Figure 50. Requête SPARQL avec une clause FILTER à traduire en système de contraintes

Les triplets de la requête peuvent être considérés comme un graphe requête demandant les étudiants (ligne 4), avec leur nom (ligne 5) et leur âge (ligne 6):

```
[Student]-
   (name)->[?name]
   (age)->[?age].
```

La ligne 7 toutefois est une contrainte supplémentaire qui doit être satisfaite pour que la correspondance soit valide ; elle précise que la valeur entière de l'âge doit être supérieure à 22 et que le nom doit commencer par un «A».

Ces types de contraintes supplémentaires ont motivé la définition des systèmes de contraintes dans nos spécifications, mais les systèmes de contraintes sont également envisagés pour fournir un moyen d'accès efficace pour indexer les graphes, par exemple pour récupérer tous les arcs d'un graphe satisfaisant une contrainte donnée.

Après ces quelques exemples de structures mathématiques de base, nous allons donner les définitions centrales de la deuxième couche de l'architecture : la couche de représentation des connaissances.

Dans notre architecture, une base de connaissances B est définie par un vocabulaire, une ou plusieurs bases de données, éventuellement une base de règles et une base de requêtes. *B= (Vocabulary, Fact Base* $^{+}$*, Rule Base*, Query Base*)*

Un vocabulaire est un ensemble d'ensembles nommés, non nécessairement disjoints, appelés sous-ensembles du vocabulaire et un ensemble de préordres définis sur l'union de ces ensembles nommés :

Définition d'un vocabulaire : Un vocabulaire V est un tuple $V = \left(U = \bigcup_{1 \le i \le k} V_i, \left(\le_1, \ldots, \le_q\right)\right)$ où les V_i sont k ensembles d'éléments et les $\le_i$ sont q préordres sur U.

Définition d'un Fait : Un Fait est un ERGraph.

Définition d'une Base de Faits : Une Base de Faits est un ensemble de Faits.

Notons que tous les *ERGraph G* d'une base de faits respectent $l_G : E_G \cup R_G \rightarrow L$ où L est construit à partir de U l'ensemble des éléments du vocabulaire V de la base de connaissances.

Définition d'une Requête : Une requête est le couple $Q=(q, \mathcal{C})$ d'un λ-ERGraph $q=((e_1,\ldots e_k), G)$ et d'un système de contraintes $\mathcal{C}$.

Les réponses à une requête dépendent du type de *EMapping* utilisé pour interroger la base. Dans les définitions suivantes, la lettre *X* représente un type du *EMapping*.

X-Réponse à une requête : Soit $Q=(((e_1,\ldots e_k), G), \mathcal{C})$ une requête et F un fait. $A=(a_1,\ldots a_k)$ est une X-Réponse à Q dans F ssi il existe un EMapping M de type X de G dans F satisfaisant $\mathcal{C}$ tel que $M(e_i)=a_i$.

Notons que la preuve d'une X-réponse est la preuve du *EMapping* associé à cette X-réponse.

Définition d'une base de requêtes : Une base de requêtes est un ensemble de requêtes.

Définition d'une règle : Une règle est le couple *R=(H,C)* d'une requête *H=(G, 𝒞)* et d'un λ-ERGraph *C* de même taille que *G*. *H* est appelé l'hypothèse de la règle et *C* sa conclusion.

Règle *X*-applicable : Une règle *R=(H,C)* est *X*-applicable à un fait *F* ssi il existe une X-Réponse à *H* dans *F*.

X-application d'une Règle : Soit *R=(H,C)* une règle X-applicable à un fait *F*, et *A* une X-Réponse à *H* dans *F*. La X-Application de *R* sur *F* pour la réponse *A* fusionne *C* avec le λ-ERGraph *(A,F)*.

Définition d'une ERFunction : Une *ERFunction F* est une fonction associant à une *ERProof P* une étiquette ou une erreur.

Définition d'un ERGraph fonctionnel : un *ERGraph* fonctionnel est un *ERGraph* où certaines entités ou relations sont étiquetées par des *ERFunctions*.

Evaluation d'un ERGraph fonctionnel : L'évaluation d'un *ERGraph* fonctionnel *G* par rapport à une *EProof P* et un environnement *E* est une copie *G'* de *G* où chaque étiquette fonctionnelle est remplacée par l'évaluation de sa fonction sur *P*. Si l'une des évaluations retourne une erreur alors *G'=∅*.

Définition d'une règle fonctionnelle : Une règle fonctionnelle est une règle *R = (H, C)* où C est un λ-ERGraph fonctionnel.

X-application d'une règle fonctionnelle : Soit *R = (H, C)* une règle fonctionnelle X-applicable à un fait *F*, avec *A* la X-Réponse à *H* dans *F* et *P* une preuve de cette X-réponse. La X-Application de la règle fonctionnelle *R* sur *F* par rapport à *P* fusionne l'évaluation de *C* sur *P* et le λ-ERGraph *(A, F)*.

Définition de la coréférence : Une relation de coréférence *R* est une relation d'équivalence sur l'ensemble des entités de *G*.

Définition d'une forme normale : Soit G un ERGraph avec une relation de coréférence *R* et une fonction de $fusion(E_1,E_2,\ldots, E_n)$ qui retourne une nouvelle entité à partir d'un ensemble d'entités, la forme normale de *G* est le graphe *NF(G)* obtenu par la fusion de toutes les entités d'une même classe d'équivalence définie par *R* comme une nouvelle entité calculée en appelant la fonction de fusion sur les entités de cette classe.

Coréférence et *fusion* sont des fonctions abstraites qui doivent être précisées dans la couche de définition des langages.

Une fois définies les primitives de ce deuxième niveau de représentation de connaissances, nous avons cherché à valider ce modèle en essayant de définir des langages connus notamment RDF. Nous reproduisons ci-après le tableau de [90] qui correspond à ces définitions.

Définition du préordre sur les étiquettes RDF : soit $\leq_{RDF}$ un préordre sur *V* tel que

- $x \leq_{RDF} y$ si $y \in$ *Blanks* l'ensemble des nœuds anonymes
- $x \leq_{RDF} y$ si $x, y \in Literals^2$ et *value(x)=value(y)*
- $x \leq_{RDF} y$ si $x=y$

Primitives RDF	Traduction dans le modèle de Griwes
Blank	Membre d'un sous-ensemble spécifique du vocabulaire et défini en intension.
Literal	Membre d'un sous-ensemble spécifique du vocabulaire et défini en intension.
Literal ^^datatype	Membre d'un sous-ensemble spécifique du vocabulaire et défini en intension.
Literal @lang	Membre d'un sous-ensemble spécifique du vocabulaire et défini en intension.
URI ref	Membre d'un sous-ensemble spécifique du vocabulaire et défini en intension.
Triplet : sujet, prédicat, objet (x p y)	Une relation dans ERGraph ; elle devrait intuitivement être binaire mais des informations de codage supplémentaires peuvent utiliser des relations n-aires. Ex : des relations quaternaires spécifiant la source du triplet et réifiant le type de la propriété. Le ERGraph *G* inclut la relation R_p telle que $n_G(R_p)=(e_x,e_p,e_y)$
graphe RDF *G* (i.e. un ensemble de triplets sur un vocabulaire donné)	Un ERGraph *E* tel que pour tout terme distinct *t* apparaissant dans un triplet de *G*, *E* contienne une entité distincte *e(t)* et pour tout triplet *(s,p,o)* de *G*, *E* contienne une relation *r* telle que $n_E(r)=(e(s),e(p),e(o))$. Remarque, dans un ERGraph RDF bien formé : - il n'y a pas d'entité isolée ; - le premier argument des relations n'est jamais un littéral ; - le nom d'une relation est une URI ref.
Noeud RDF	Entités apparaissant en position 1 et 3 d'une relation.
Vocabulaire RDF (rdf:Property, rdf:type)	Un sous-ensemble spécifique du vocabulaire et défini en extension pour RDF.
Implication simple en RDF	*H* implique *G* ssi il existe un Homomorphism$_{\leq RDF}$ de *G* dans la forme normale NF(H) définie par $coref_{RDF}$ et $fusion_{RDF}$.
Axiomes RDF	le ERGraph representant les triplets axiomatiques def RDF et ajoutés à chaque base de faits.
x rdf:type t	Représenté comme tout autre triplet. (NB : *t* peut être intégré à l'étiquette de l'entité représentant *x*)
Règle 1 de RDF **IF** *x p y* in RDF graph *G* **THEN** *p rdf:type rdf:Property*	*R=(H,C)* où *H=((e(y)),H')* avec *H'* le graphe associé à *{(x,y,z)}* où *x*, *y* et *z* sont des nœuds anonymes et *C=((e(u)),C')* avec *C'* le graphe associé à *{(u, rdf:type, rdf:Property)}* où *u* est un nœud anonyme et *rdf:type* et *rdf:Property* sont des URI refs du vocabulaire RDF.
Règle 2 de RDF **IF** *x p y^^d* in RDF graph *G* and *y^^d* well-typed **THEN** *y^^d rdf:type d*	R=(Q,D) une règle fonctionnelle, où *Q=(H,C)* avec *H=((e(z)),H')* avec *H'* le graphe associé à *{(x,y,z)}* où *x*, *y* et *z* sont des nœuds anonymes et *C* satisfait ssi *e(z)* est étiqueté par un littéral datatypé bien typé. *D=((e(a)),D')* est un λ-ERGraph fonctionnel associé à *{(a, rdf:type, fun:getType(im(e(z)))), (x, fun:id(im(r(y)), fun:getNormalForm(im(e(z)))) , (fun:getNormalForm(im(e(z))), rdf:type, fun:getType(im(e(z)))) }* avec *a* un nœud anonyme et *rdf:type* une URI ref du vocabulaire RDF et *fun:getType()* une fonction extrayant le type d'un littéral.

Figure 51. Tableau de définition de RDF dans le modèle de Griwes

Les résultats de l'initiative Griwes devraient fournir les bases d'une spécification de la plate-forme générique visée. Il nous faut maintenant passer à l'étape suivante qui demande une structure de projet plus importante et qui vise à :

(1) la conception et l'implantation des structures de données et algorithmes associés aux structures mathématiques identifiées dans les spécifications, avec un intérêt particulier pour leur optimisation et leur paramétrage ;
(2) la conception de l'architecture logicielle de la plate-forme permettant l'extension et l'évolution de celle-ci en accord avec son statut de logiciel libre et open-source ;
(3) la mise en place et l'animation d'une communauté d'utilisateurs et de développeurs et des outils de gestion du cycle de vie de la plate-forme.

3.4 Conclusion

Comme nous le disions en introduction de plus en plus de standards et techniques sont déployés pour offrir sur le web un espace et des outils formels d'échanges et de manipulation de connaissances (RDF, RDFS, SKOS, OWL, GRDDL, RDFa, µFormats, etc.). Certaines couches de ces formalismes (ex : la couche OWL DL) ont des opérationnalisations immédiates découlant de leur parenté avec des formalismes existants (ex : logiques de description) mais nous avons montré dans ce chapitre que de nombreux formalismes offrent des structures de graphes pour lesquelles la communauté des graphes conceptuels pourrait contribuer à bien des titres.

Nous avons rappelé les approches existantes et expliqué comment les opérations de graphes peuvent être utilisées pour opérationnaliser des opérations sur des représentations de connaissances dont l'interprétation et les caractéristiques algorithmiques sont connues et prouvées.

Nous avons montré comment les formalismes à base de graphes peuvent être utilisés pour représenter des connaissances avec un degré variable de formalisation en fonction des besoins identifiés dans les scénarios d'application et des traitements à effectuer.

Nous avons donné les caractéristiques de certains de ces formalismes qui sont utilisés dans nos travaux et les opportunités d'extensions qu'ils offrent avec, en exemple, le cas du nommage des graphes pour permettre d'utiliser la provenance en RDF.

Nous avons ensuite donné des extraits de résultats d'une initiative en cours pour factoriser la définition des structures mathématiques partagées par les formalismes à base de graphes et réutiliser l'algorithmique des traitements communs à ces structures.

3.5 Perspectives

Pour saisir l'ampleur des contributions possibles, il nous faut aussi souligner les opportunités futures *i.e.* les couches de standardisation en cours ou à venir. En effet, en parallèle au déploiement de ces premiers résultats, de nouveaux problèmes sont identifiés et de nouvelles initiatives voient le jour qui, elles aussi, offrent de nouvelles perspectives à la représentation et au raisonnement à base de graphes ; nous en citerons deux : signatures et graphes imbriqués ; règles de graphes

Signatures et graphes imbriqués : en perdant la notion de graphe, les représentations logiques ne perdent pas uniquement une structure de visualisation ou d'optimisation, elles perdent aussi une structure d'identification et donc un moyen de modularisation des représentations, de nommage des ces modules, de traçabilité et d'identification des provenances, d'authentification, etc. Les problématiques de graphes nommés, de sérialisation canonique d'un graphe et de signature reviennent actuellement sur le devant de la scène. Là encore les graphes conceptuels ont déjà des résultats sur le domaine, notamment pour les graphes imbriqués [91].

Règles de graphes : après la représentation et l'échange de connaissances, le W3C a lancé une initiative sur l'échange de règles d'inférence RIF pour *Rules Interchange Format* [92]. Chacune de ces règles représente un raisonnement, une déduction, élémentaire. Par exemple « **si** un employé travaille sur un sujet donné **alors** son groupe travaille aussi sur ce sujet ». RIF a pour objectifs : de fournir un format d'échange de ces règles sur le web [93]; de permettre l'interopérabilité des systèmes à base de règles ; et de fournir un langage de règles pour le web sémantique [94]. Des initiatives telles que RIF montrent l'intérêt de standardiser l'échange de règles (règles de production, règles de décision etc.) sur le web et posent la question d'un langage de règles s'appliquant aux représentations de connaissances du web sémantique. Les Graphes conceptuels ont eux-mêmes proposé et évalué des langages de règles de graphes qui pourraient être adaptés et servir de base à de telles initiatives [95] [96] [97]

Enfin pour bien comprendre que l'expressivité recherchée est toujours un problème ouvert mentionnons encore deux initiatives :

- OWL 2.0 [49]: trois ans après la première version de OWL, il y a un certain nombre de demandes pour faire évoluer les langages proposés (ex : restriction qualifiée de la cardinalité, extension du support des datatypes, union disjointe) et aussi une volonté d'identifier de nouveaux fragments de OWL avec des propriétés (ex : expressivité et complexité) bien identifiées. Cette demande a donné lieu à la création d'un nouveau groupe de travail.
- SKOS [51] est l'exemple même au sein du W3C que d'autres langages de représentation ont vu le jour et utilisent des modèles à base de graphes. SKOS, par exemple, est proposé pour représenter des thésaurus, des index thématiques, des glossaires, des vocabulaires contrôlés, etc.

Etre capable d'accepter et d'intégrer différents langages ou différents fragments de langages et de factoriser les structures de données et algorithmes efficaces pouvant leur être appliqués est réellement un enjeu majeur. D'autres communautés comme les logiques de description [10] ou les *Topic Maps* (rapport du SWBP au W3C [98]) l'ont compris et ont su mettre en place des modèles pivots et des plates-formes associées.

En 2007, le projet COLOR Griwes a montré comment un modèle abstrait commun était envisageable et quelles pouvaient en être les spécifications. Il nous paraît maintenant stratégique de matérialiser ce travail en abordant à la fois les problèmes de structures de données efficaces et d'algorithmes optimisés pour ce modèle pivot et la constitution d'une communauté autour d'une plate-forme open-source syndiquant les contributions.

Enfin, une perspective actuelle dans le cadre du développement des métadonnées et de leurs représentations en ligne, ces dernières années, est liée au "tagging social" qui s'est imposé au sein du web 2.0 comme le principal moyen de classification de données en très grand nombre. Le résultat de cette activité d'étiquetage est la génération d'ensembles de tags non contrôlés, appelés folksonomies. Cette génération libre pose plusieurs problèmes : le problème de l'ambiguïté (un tag dénotant plusieurs concepts), le problème de variations d'écritures et de synonymie (plusieurs tags dénotant un même concept), le statut de ces tags par rapport au document ou aux données tagués et le manque de représentations explicites qui permettraient une plus grande exploitation automatique de ces structures (notamment pour la recherche ou l'échange d'informations). La thèse de Freddy Limpens que je co-encadre et commencée en septembre 2007, proposera une évolution de ces systèmes à base de folksonomies, en tenant compte, notamment, des travaux existant dans le domaine de la représentation des connaissances, en particulier ceux à base d'ontologies.

Les approches identifiées actuellement [99] montrent que la nature sociale des échanges de connaissances n'est pas en contradiction avec les possibilités offertes par les systèmes à base d'ontologies formelles. Ce point est symptomatique de la tension entre l'utilité de représentations formelles dans un système à base de connaissances et l'utilité de représentations informelles dans les interactions quotidiennes entre individus.

Parce que les tags sont simples et naturels d'emploi, parce que les folksonomies se constituent d'elles mêmes comme un effet secondaire de l'usage des tags, elles sont considérées, à juste titre, comme extrêmement performantes puisqu'elles utilisent la masse et la diversité des utilisateurs pour organiser la masse et la diversité des ressources d'information.

Parce qu'elles ont des méthodologies parfois lourdes, parce qu'elles reposent sur des formalismes complexes, les ontologies sont considérées comme des objets coûteux et difficilement réconciliables avec des interactions naturelles.

Cependant, comme le montrent les exemples abordés dans [99], il est possible d'utiliser une ontologie pour décrire une folksonomie, l'utilisation des tags ou tout autre activité liée aux usages du web communautaire. Opposer ontologies et folksonomies n'a donc pas de raison d'être dans l'absolu puisque, pour un scénario donné, elles peuvent avoir des rôles complémentaires à jouer dans une même solution applicative.

Enfin, lorsque l'on regarde les cycles de vie d'une ontologie et d'une folksonomie, les approches présentées en [99] dévoilent une opportunité de faire cohabiter ces deux objets :

(1) les scénarios d'utilisation d'une folksonomie finissent souvent par identifier des cas d'usages où l'on souhaiterait pouvoir exploiter plus avant des structures dans ces ensembles de tags ;
(2) la création et la maintenance d'une ontologie doivent toujours trouver des méthodes pour éviter le goulot d'étranglement de l'acquisition des connaissances, par exemple en se nourrissant de folksonomies. Permettre de fusionner certaines structures d'une folksonomie et d'une ontologie, représenter explicitement et maintenir ces points de jonction ; gérer les frictions entre les deux cycles de vie ; assister et automatiser le moins intrusivement possible les échanges entre ces deux objets en arrière plan des usages ; etc. Tels sont à notre avis des enjeux intéressants et fertiles pour les systèmes d'information communautaires.

D'un point de vue représentation des connaissances tout l'enjeu sera de proposer des formalismes qui non seulement accommodent les particularités de ces différents objets (folksonomies, thésaurus, ontologies, etc.) mais permettent aussi d'opérationnaliser leurs cycles de vie et les échanges entre eux.

Chapitre 3 : Récapitulatif de mes contributions personnelles.

Dans ce chapitre, je rappelais comment les formalismes à base de graphes peuvent être utilisés pour représenter des connaissances.

A travers ma participation à plusieurs groupes de travail du W3C, j'ai étudié et contribué à plusieurs initiatives de ce consortium. Je fais partie du SWBP *working group* sur les méthodes d'utilisation des formalismes du web sémantique, je suis l'un des éditeurs des documents de standardisation de GRDDL et l'éditeur principal des *use cases* [36], je suis aussi le concepteur du profil de RDFa [35].

Je suis l'auteur principal d'une soumission officielle pour l'extension du standard RDF afin de prendre en compte la nécessité de nommage des graphes RDF pour le suivi de la provenance [87] [88].

Je participe aux réflexions sur Corese qui implante RDF en utilisant les graphes conceptuels et j'ai notamment spécifié et étudié ses mécanismes de distances pour une projection approchée [74].

Je suis l'instigateur, l'éditeur et le porteur du projet Griwes qui s'intéresse à factoriser les structures mathématiques et l'algorithmique communes aux formalismes de graphes les plus courants [90].

4. Graphes comme espaces métriques

Une ontologie offre un support à d'autres types de raisonnement que la dérivation logique. Par exemple, la hiérarchie de notions contenue dans une ontologie peut être vue comme un espace permettant de définir des métriques pour comparer la proximité sémantique de deux notions. Nous avons mis en œuvre cette idée dans plusieurs scénarios comme l'allocation distribuée d'annotations, la recherche approchée ou le *clustering*.

Nous résumons ici diverses utilisations que nous avons faites des distances sémantiques et discutons notre position sur ce domaine. Nous donnons les scénarios d'utilisation et les distances utilisées dans un échantillon représentatif de projets que nous avons menés.

Pour nous, cette première série d'expériences a permis de démontrer l'intérêt et le potentiel des distances, et aussi de souligner l'importance du travail restant à faire pour identifier et caractériser les familles de distances existantes et leur adéquation respective aux tâches pour lesquelles les utilisateurs souhaitent être assistés.

4.1 Notions de proximité conceptuelle et distances sémantiques

Intuitivement, nous sommes tous portés à dire que le concept de *berline* est plus proche du concept de *monospace* que de celui d'*avion* ; cependant, nous pensons aussi que le concept de *berline* est plus proche du concept d'*avion* que du concept de *livre*. Ces distances intuitives peuvent être simulées par exemple pour améliorer les moteurs de recherche du web dans leurs algorithmes de filtrage et de tri des réponses.

En informatique, une ontologie est une théorie logique partielle rendant explicite une conception de la réalité [8] [100]. Les définitions en intension d'une ontologie sont donc naturellement traduites en des représentations logiques exploitées notamment dans des inférences de dérivation logique, par exemple pour améliorer le rappel en recherche d'information. Cependant, ces mêmes définitions et leurs relations peuvent être vues comme des espaces, notamment des graphes ou réseaux sémantiques, qui peuvent être dotés de métriques servant de base à toute une autre gamme d'inférences.

L'idée d'évaluer les appariements conceptuels en se basant sur des réseaux sémantiques date des premiers travaux sur la simulation de la mémoire sémantique chez l'humain.

En 1968, Quillian [101] simule la définition et la remémoration d'un concept comme un processus de traçage ou d'activation propagée à travers un réseau de liens étiquetés reliant ces concepts. Cette activation était utilisée pour identifier des intersections dans ce réseau définitionnel et ainsi fournir un système capable de comparer deux concepts. Parmi les cinq types de liens utilisés par Quillian, on trouve déjà le lien de subordonné ou subsomption ("isa").

En 1975, Collins et Loftus [102] étendent le travail de Quillian avec treize hypothèses parmi lesquelles :

- Hypothèse #1 : l'activation décroît au fur et à mesure de sa propagation et de façon inversement proportionnelle à la "force" des liens qu'elle parcourt ;
- Hypothèse #5 : le réseau est organisé selon des critères de similarité sémantique *i.e.* soient deux concepts, plus ils ont de propriétés en commun et plus ils sont proches ;
- Hypothèse #9 : le lien de subordonné est le premier lien à considérer lors de l'appariement de deux concepts.

La proximité conceptuelle de deux concepts peut prendre bien des formes par exemple, la complémentarité fonctionnelle (ex : le marteau et le clou), ou la similarité fonctionnelle (ex : le marteau et le tournevis). Ce dernier exemple correspond à la famille des similarités sémantiques où l'appariement des concepts est basé sur les caractéristiques définitionnelles qu'ils partagent (ex : le tournevis et le marteau sont tous les deux des outils à main). La structure de données naturelle pour un tel raisonnement

est la taxonomie[9], où les types/catégories sont regroupés par des liens "est sous type de / est sous catégorie de" en fonction des caractéristiques qu'ils partagent (ex : marteau, tournevis, pinces, scie, rabot, etc. sont des sous-types d'outil à main).

En 1989, Rada *et al.* [103] expliquent qu'en appliquant l'algorithme d'activation propagée à un réseau sémantique contenant uniquement des liens de subsomption (est sous type de) on obtient une forme de distance sémantique définie pour tout couple de concept du réseau. Rada *et al.* défendent l'hypothèse selon laquelle la proximité sémantique est une métrique fonction du nombre d'arcs minimum séparant deux concept dans la hiérarchie de subsomption. Ils utilisent la distance $Dist(C_1,C_2)= Min (NombreArcsEntre(C_1,C_2))$ et montrent que c'est une métrique. Rada *et al.* appliquent cette distance à la recherche documentaire à base de requêtes booléennes. Cependant la comparaison entre une requête booléenne et l'indexation booléenne des documents, reposait sur une moyenne des distances qui avait un comportement contre intuitif autour de zéro.

A partir de ces premiers travaux, nous pouvons identifier deux principales tendances dans la définition d'une distance sémantique sur une hiérarchie de concepts :

1. Les approches qui reposent uniquement sur la structure de la hiérarchie de concepts comme espace métrique [103] [104];
2. Les approches qui reposent sur des informations extérieures supplémentaires (ex : statistiques sur l'utilisation des concepts) pour construire leur distance. [105] [106]

Une structure supportant naturellement le raisonnement sur les similarités sémantiques est la hiérarchie des types telle que l'on peut la trouver dans un support en graphes conceptuels, dans la TBox des logiques de description, dans un schéma RDFS, etc. En effet, dans ce squelette taxonomique de l'ontologie, les liens de subsomption groupent les types suivant les caractéristiques définitionnelles qu'ils partagent. Lorsqu'elle est appliquée au graphe d'une hiérarchie, une proximité calculée par propagation donne une distance sémantique, la première et la plus simple étant celle qui compte les arcs séparant deux sommets [103].

Dans le domaine des graphes conceptuels, cette première approche est utilisée, en particulier pour proposer une projection ne donnant plus uniquement des valeurs booléennes *i.e.* une similarité $S:C^2\rightarrow[0,1]$ où 1 correspond à la valeur vraie de la projection classique et toute autre valeur donne une idée de la similarité entre le graphe projeté et le graphe source. L'utilisation initiale faite par Sowa visait à permettre des déplacements latéraux dans le treillis des types. Ralescu et Fadlalla [107] l'ont utilisée pour relaxer les contraintes de l'opérateur de jointure. Plus récemment, Zhong *et al.* [108] ont utilisé une distance atténuée par la profondeur des types dans l'ontologie pour construire une mesure de similarité entre graphes conceptuels.

[9] nous appellerons taxinomie ou taxonomie une classification en arbre ou en treillis de catégories, classes ou types.

Dans la suite, nous rappelons un certain nombre des distances connues dans chacun des deux types d'approche, puis nous donnons un aperçu des applications et des travaux que nous menons autour de cette notion de proximité ou distance sémantique à travers une opérationnalisation du web sémantique basée sur les graphes conceptuels.

4.1.1 Approches basées purement sur des structures ontologiques et notamment sur les liens hiérarchiques

Les approches purement basées sur la hiérarchie diffèrent essentiellement dans leur façon de combiner les profondeurs et les chemins entre les concepts comparés.

L'algorithme de Rada et al. [103] présenté en introduction est le plus simple : il compte le nombre minimum d'arcs à parcourir pour aller d'un concept à l'autre dans la hiérarchie des concepts.

$Dist_{Rada}(C_1,C_2)= Min\ (NombreArcsEntre(C_1,C_2))$

En 1994, Wu et Palmer [104] proposent une amélioration basée sur le ratio entre la profondeur du plus petit concept ancêtre commun et la profondeur des deux concepts comparés.

$Dist_{WuPalmer}(C_1,C_2)= 2*N_3 / (N_1 + N_2 + 2*N_3)$

avec C_3 le plus petit ancêtre commun de C_1 et C_2 et, N_1 le nombre de nœuds sur le chemin de C_1 à C_3, N_2 le nombre de nœuds sur le chemin de C_2 à C_3, et N_3 le nombre de nœuds sur le chemin de C_3 à la racine de la hiérarchie.

Leacock et Chodorow [109], utilisent aussi la longueur minimale du chemin entre deux concepts :

$Sim_{Leacok}(C_1,C_2)= -log\ [min_{p \in \{<C1,C2>\}}(length\ (p)) / (2*Max)\ ]$

où $p \in \{<C_1,C_2>\}$ est un chemin de C_1 à C_2 dans la hiérarchie, *length(p)* est le nombre d'arcs de du chemin *p*, et *Max* est la profondeur maximale de la hiérarchie.

En 2004, Zargayouna et Salotti [110] intègrent au calcul la distance entre les concepts et le bas de l'ontologie.

$Sim_{Zargayouna}(C_1,C_2)= 2*Depth(C_3) /(\ Depth_C(C_1) + Depth_C(C_2) + spec\ (C_1,C_2))$

$spec(C_1,\ C_2) = depth_b(C)\ *\ distance(C,C_1)\ *\ distance(C,C_2)$

avec $depth_b(C)$ le nombre maximum d'arcs qui séparent *C* de *bottom* et *distance(C, Ci)* le nombre d'arcs entre *C* et C_i, *C* le plus petit ancêtre commun de C_1 et C_2 en nombre d'arcs, *depth(C)* est le nombre d'arcs qui séparent *C* de la racine et $depth_c(C_i)$ le nombre d'arcs qui séparent C_i de la racine en passant par *C*.

Un autre type d'approche s'intéresse à parcourir tous les types de relations du modèle (pas uniquement les liens de subsomption) comme par exemple Hirst et St-Onge [111]

$Weigth_{HirstStOnge}(<C_1,C_2>)= C- length\ (<C_1,C_2>) - k*\ nb\_changes(<C_1,C_2>)$

avec $length\ (<C_1,C_2>)$ la longueur du chemin de C_1 à C_2 et $nb\_changes(<C_1,C_2>)$ le nombre de changements de direction sur le chemin de C_1 à C_2. Une similarité peut être

définie en prenant le poids minimal de tous les chemins existants entre C_1 et C_2 et 0 s'il n'y a pas de chemin.

Enfin il existe un autre courant qui s'intéresse aux propriétés associées à un concept, notamment dans sa définition, pour calculer une similarité. Ainsi, Tversky [112] propose de se baser sur les différences entre les ensembles de caractéristiques des concepts C_A et C_B notés A et B :

$sim_{Tversky}(A, B) = \alpha\, g(A \cap B) - \beta\, g(A - B) - \gamma\, g(B - A)$

où α, β, et γ sont des constantes de calibration.

4.1.2 Approches basées sur la hiérarchie augmentée par des informations extérieures

Resnik [113] [114] [105] utilise la théorie de l'information et attache à chaque concept une évaluation du caractère informatif (contenu d'information) de ce concept C, défini comme une fonction de la probabilité de rencontrer ce concept C: $IC(C)= -log\ p(C)$. Le calcul de cette valeur repose donc sur la capacité à évaluer statistiquement la probabilité de rencontrer un concept donné dans un contexte fixé. A partir de cette définition, la similarité entre deux concepts est définie comme la quantité d'information qu'ils ont en commun *i.e.* le contenu d'information le plus élevé parmi leurs ancêtres communs dans la hiérarchie.

$Sim_{Resnik}(C_1,C_2)= max_{\ C \in \{Ancêtres\ communs\ (C1,C2)\}}\ [IC(C)]$

où $p(C)$ est approximé par la fréquence de C dans un corpus. Resnik remarquera que l'approche de Leacock et Chodorow [109] donne de meilleurs résultats que la sienne.

Resnik a essayé d'appliquer cette technique en travaillant directement sur les mots mais a rencontré des comportements contre intuitifs dus à la polysémie. L'approximation de la probabilité de rencontrer un concept est faite par un calcul statistique sur le corpus utilisé pour tester la recherche d'information ; elle dépend donc du scénario et du corpus considérés.

En 1997, Jiang et Conrath [106] améliorent l'approche de Resnik en intégrant au calcul de la distance l'évaluation du caractère informatif des deux concepts comparés :

$Dist_{JiangConrath}(C_1,C_2) = IC(C_1) + IC(C_2) - (2*IC(C_3))$

avec C_3 le plus petit ancêtre commun de C_1 et C_2

En 1998, Lin [115] réutilise les mêmes éléments mais sous la forme d'un quotient

$Dist_{Lin}(C_1,C_2) = (2*IC(C_3)) / (IC(C_1) + IC(C_2))$

avec C_3 le plus petit ancêtre commun de C_1 et C_2.

4.2 Distance et bases de connaissances distribuées

Dans un web sémantique d'entreprise, les scénarios amènent souvent la contrainte de bases d'annotations distribuées (assertions RDF à propos de ressources intranet). Pour gérer cette distribution, nous avons proposé une architecture et des protocoles permettant en particulier de maintenir la spécialisation des bases d'annotations quant aux sujets abordés dans leurs assertions [21].

Chaque archive de notre architecture maintient une structure appelée ABIS (*Annotation Base Instances Statistics*) décrivant des statistiques sur les types de triplets (relations binaires imposées par le modèle RDF) présents dans leur base d'annotations. Par exemple si, dans l'ontologie, il existe une propriété `Auteur` avec la signature :

`[Document]` → `(Auteur)` → `[Personne]`

L'ABIS pourra contenir des statistiques sur l'existence des instances de triplets ayant les signatures suivantes :

`[Article]` → `(Auteur)` → `[Etudiant]`

`[Essai]` → `(Auteur)` → `[Philosophe]`

...

L'ABIS est construit lors de la transformation des annotations RDF en graphes conceptuels et capture la contribution d'une archive à la mémoire globale en termes de types de connaissances contenues dans cette archive. L'ABIS fournit un moyen de comparer le contenu de deux bases et nous l'utilisons pour maintenir la spécialisation des bases d'annotations grâce à une distance sémantique définie entre un ABIS et une nouvelle annotation.

Pour comparer deux types primitifs, nous utilisons la distance de [103] comptant le nombre d'arcs sur le chemin le plus court qui relie ces deux types à travers la hiérarchie ; voir formule (1). En utilisant cette distance, on peut définir une distance entre deux triplets RDF (ou deux instances d'une relation binaire), comme étant la somme des distances entre : les types des deux relations, les types des deux concepts en premier argument (*domain*) et les types des deux concepts en deuxième argument (range); voir formule (2). La distance entre un triplet et un ABIS est alors définie comme la distance minimale entre ce triplet et les triplets recensés par l'ABIS ; voir formule (3). Et finalement, la distance entre une annotation et un ABIS est la somme des distances entre chaque triplet de l'annotation figurant dans l'ABIS ; voir formule (4). Les caractéristiques exactes de ces distances sont étudiées dans [21].

$dist(t_1,t_2) = length(t_1,lcst(t_1,t_2)) + length(t_2, lcst(t_1,t_2))$ (1)

où *lcst(t1,t2))* est le plus proche supertype commun de t_1 et t_2.

dist(triple1,triple2)=

dist(domain(triple1), domain(triple2)) +

dist(predicate(triple1), predicate(triple2)) +

dist(range(triple1), range(triple2)) (2)

$$dist(triple, ABIS) = \min_{triple_i \in ABIS} (dist(triple, triple_i)) \quad (3)$$

$$dist(An, ABIS) = \sum_{triple_j \in An} dist(triple_j, ABIS) \quad (4)$$

Cette distance donne une fonction d'évaluation / fonction de coût utilisée comme critère dans un protocole de mise aux enchères des nouvelles annotations à archiver : chaque nouvelle annotation est mise aux enchères entre les archives existantes ; chaque archive fait une offre qui correspond à la distance entre son ABIS et l'annotation ; l'archive avec l'offre la plus petite gagne l'annotation. Ce protocole permet de maintenir la spécialisation des bases et ainsi de faciliter l'optimisation de la résolution de requêtes distribuées en utilisant les ABIS pour la décomposition et le routage des projections.

Dans ce premier exemple, la définition d'une distance conceptuelle sur la hiérarchie des types permet de construire un consensus calculatoire (distance) au dessus du consensus ontologique (hiérarchie), et de l'utiliser dans un consensus protocolaire (enchères).

Initialement utilisée pour un protocole de mémoire distribuée, la section suivante explique comment cette distance a ensuite été intégrée au moteur de recherche de chaque base pour proposer une nouvelle fonctionnalité : la recherche approchée de connaissances par relaxation des contraintes de typage.

4.3 Distance et projection de graphes approchée

La plate-forme CORESE [74] intègre une fonctionnalité de recherche approchée qui démontre une autre application des inférences simulant la proximité conceptuelle. CORESE utilise une extension de la distance atténuée par la profondeur [108] des types dans le treillis de l'ontologie ; voir formules (5) et (6).

$$\forall (t_1,t_2) \in {H_c}^2; t_1 \le t_2 \text{ on a } l_{H_c}(t_1,t_2) = \sum_{\{t \in \langle t_1,t_2 \rangle, t \neq t_1\}} \left[\frac{1}{2^{depth(t)}} \right] \quad (5)$$

et $\forall t \in H_c$; on a $l_{H_c}(t,t) = 0$

avec H_c la hiérarchie des types de concepts, $<t_1,t_2>$ le chemin le plus court entre deux types de concepts t_1 et t_2, et *depth(t)* la profondeur de t dans l'ontologie *i.e.* le nombre d'arcs sur le chemin le plus court entre t et la racine ⊤

$$\forall (t_1,t_2) \in {H_c}^2 \text{ on a } dist_{COR}(t_1,t_2) = \min_{\{t \ge t_1, t \ge t_2\}} \left(l_{H_c}(t_1,t) + l_{H_c}(t_2,t) \right) \quad (6)$$

D'un point de vue mathématique $dist_{COR}$ est une dissimilarité ou semi-distance *i.e.* elle respecte toutes les caractéristiques d'une distance sauf l'inégalité triangulaire :

$$\forall t \in H_c \text{ on a } dist_{COR}(t,t) = \min_{\{t' \ge t, t' \ge t\}} \left(l_{H_c}(t,t') + l_{H_c}(t,t') \right) = 2 * l_{H_c}(t,t) = 0$$

puisque $t \le t$

$$\forall (t_1,t_2) \in {H_c}^2 \text{ on a } dist_{COR}(t_1,t_2) = 0 \Rightarrow l_{H_c}(t_1,t) = l_{H_c}(t_2,t) = 0 \Rightarrow t_1 = t_2$$

$$\forall (t_1,t_2) \in {H_c}^2 \text{ on a } dist_{COR}(t_1,t_2) = \min_{\{t \ge t_1, t \ge t_2\}} \left(l_{H_c}(t_1,t) + l_{H_c}(t_2,t) \right)$$
$$= \min_{\{t \ge t_2, t \ge t_1\}} \left(l_{H_c}(t_2,t) + l_{H_c}(t_1,t) \right) = dist_{COR}(t_2,t_1)$$

L'inégalité triangulaire ne tient pas de façon générale pour tout type pris dans la hiérarchie à cause du multi-héritage. Par contre elle peut être vérifiée pour tout type pris parmi les ancêtres des deux types considérés ; cette version affaiblie du principe de parcimonie nous suffit ici.

En utilisant cette distance, on peut relaxer la contrainte d'égalité ou de spécialisation des types lors de la projection en la remplaçant par une contrainte de proximité utilisant une distance conceptuelle comme celle définie en (6). On obtient alors une projection approchée pour l'appariement de graphes requêtes (ex : un motif recherché par un utilisateur) et de graphes faits (ex : une annotation RDF) ; une telle projection préserve l'adjacence et l'ordre des arcs mais permet de relaxer les contraintes de typage.

Dans ce deuxième exemple, la distance est utilisée pour remplacer une contrainte logique ($t_1(x) \Rightarrow t_2(x)$) par une contrainte numérique ($d(t_1,t_2) < seuil$). Cette transformation permet de relaxer une requête donnée par un utilisateur lorsqu'elle ne donne pas (suffisamment) de résultats. Pour accéder aux connaissances d'une base, une alternative aux requêtes est la navigation dans les annotations. La section suivante montre comment, là encore, les distances peuvent être utilisées.

4.4 Distance et ultra-métrique de clustering

Dans le cadre du projet KmP, nous nous sommes intéressés à la construction d'un algorithme de regroupement (*clustering*) des compétences présentes sur la Télécom Valley de Sophia Antipolis et annotées en RDF [27]. Le regroupement normalement effectué manuellement par les experts en gestion s'est révélé être un algorithme de regroupement monothétique (*monothetic clustering*). La représentation recherchée demandait de pouvoir fournir des moyens de contrôler simplement le niveau de détail et de granularité choisis pour générer le regroupement. En analyse de données [116], une structure classique supportant le choix des niveaux de détail est le dendrogramme, un arbre qui, à chaque niveau de coupure, donne une solution de regroupement plus ou moins fin.

Un dendrogramme (ou arbre indicé ou hiérarchie indicée) repose sur une ultra-métrique, c'est-à-dire une distance avec une inégalité triangulaire sur-contrainte :

$$\forall t',\ t_1,\ t_2 \in H_c^3 \quad dist(t_1,t_2) \leq max(dist(t_1,t'),\ dist(t_2,t'))$$

Nous avons donc cherché à construire cette ultra-métrique à partir de la distance sémantique de CORESE. Dans KMP, nous avons fait l'hypothèse que nos types étaient disjoints. Par conséquent le treillis des types est un arbre. En effet si un type pouvait avoir deux parents cela signifierait que les deux classes parentes ont une intersection non vide et par conséquent qu'elles ne sont pas disjointes. Nous n'avons considéré pour cela que des hiérarchies en arbres simples.

Le calcul de cette distance est donc équivalent à trouver le plus petit commun sur-type (appelé LCST) entre deux types (t_1 et t_2) et la longueur du chemin de subsumption entre un type et son sur-type (SubPath) on a alors comme distance :

$$dist_{TP}(t_1,t_2) = SubPath(t_1, LCST(t_1,t_2)) + SubPath(t_2, LCST(t_1,t_2))$$

avec $LCST(t_1,t_2)$ le plus petit commun sur-type de t_1 et t_2 et $SubPath(t,t')$ défini par :

- pour $t=t'$ par définition $SubPath(t, t') = 0$
- pour $t \neq t'$ et $t \not\leq t'$ $SubPath(t, t') = \infty$
- pour $t \neq t'$ et $t \leq t'$ $SubPath(t, t') = Length(t,ST(t)) + SubPath(ST(t), t')$

où $Length(t, ST(t))$ est la longueur d'un lien de subsomption entre un type t et son sur-type direct $ST(t)$ avec $Length(t, ST(t)) = \left[\frac{1}{2}\right]^{depth(ST(t))}$

où $depth(t)$ est la profondeur du type t dans l'arbre telle que :

$$\begin{cases} depth(\mathsf{T}) = 0 & \text{avec } \mathsf{T} \text{ racine de l'arbre} \\ depth(t) = 1 + depth(ST(t)) & \text{où } ST(t) \text{ est le sur-type direct de } t \end{cases}$$

Comme nous travaillons sur un arbre, la distance *SubPath* a été simplifiée par rapport à la section précédente ; le chemin est maintenant unique et, lorsque qu'il est défini, la distance peut être calculée directement :

$$SubPath(t,t') = \sum_{n=depth(t')}^{depth(t)-1}\left[\frac{1}{2}\right]^n = \frac{1}{2^{depth(t')}} \times \sum_{n=0}^{depth(t)-depth(t')-1}\left[\frac{1}{2}\right]^n$$

$$= \frac{1}{2^{depth(t')}} \times \frac{1-\frac{1}{2^{depth(t)-depth(t')}}}{\frac{1}{2}}$$

$$= \frac{1}{2^{depth(t')-1}} \times \left(1-\frac{1}{2^{depth(t)-depth(t')}}\right)$$

$$= \frac{1}{2^{depth(t')-1}} - \frac{1}{2^{depth(t)-1}}$$

Ce qui nous donne une distance exact ayant pour formule (7).

$$dist_{TP}(t_1,t_2) = \frac{1}{2^{depth(LCST(t_1,t_2))-1}} - \frac{1}{2^{depth(t_1)-1}} + \frac{1}{2^{depth(LCST(t_1,t_2))-1}} - \frac{1}{2^{depth(t_2)-1}}$$

$$dist_{TP}(t_1,t_2) = \frac{1}{2^{depth(lcst(t_1,t_2))-2}} - \frac{1}{2^{depth(t_1)-1}} - \frac{1}{2^{depth(t_2)-1}} \quad (7)$$

Montrons que nous avons bien à faire à une distance au sens mathématique :

(I) montrons que $dist_{TP}(t_1,t_1)=0$

$$dist_{TP}(t_1,t_1) = \frac{1}{2^{depth(LCST(t_1,t_1))-2}} - \frac{1}{2^{depth(t_1)-1}} - \frac{1}{2^{depth(t_1)-1}}$$

$$= \frac{1}{2^{depth(LCST(t_1,t_1))-2}} - \frac{1}{2^{depth(t_1)-2}}$$

Or comme la subsumption est reflexive (*i.e.* $t_1 \leq t_1$) le plus petit commun sur-type d'un même type est lui-même (*i.e.* $LCST(t_1,t_1)=t_1$) donc

$$dist_{TP}(t_1,t_1) = \frac{1}{2^{depth(t_1)-2}} - \frac{1}{2^{depth(t_1)-2}} = 0$$

(II) Montrons que $dist_{TP}(t_1,t_2) = dist_{TP}(t_2,t_1)$ pour cela remarquons que le plus petit commun sur-type pour deux types est par définition commutatif (*i.e.* $LCST(t_1,t_2)= LCST(t_2,t_1)$) donc

$$dist_{TP}(t_1,t_2) = \frac{1}{2^{depth(lcst(t_1,t_2))-2}} - \frac{1}{2^{depth(t_1)-1}} - \frac{1}{2^{depth(t_2)-1}}$$

$$= \frac{1}{2^{depth(LCST(t_2,t_1))-2}} - \frac{1}{2^{depth(t_2)-1}} - \frac{1}{2^{depth(t_1)-1}}$$

$$= dist_{TP}(t_2,t_1)$$

(III) Montrons que $dist_{TP}(t_1,t_2) = 0 \Rightarrow t_1=t_2$

Repartons des définitions initiales :

$$dist_{TP}(t_1,t_2) = SubPath(t_1,LCST(t_1,t_2)) + SubPath\ (t_2,LCST(t_1,t_2))$$

et

$$SubPath\ (t,t') = \sum_{n=depth(t')}^{depth(t)-1} \left[\frac{1}{2}\right]^n$$

Nous avons alors très clairement une somme de deux sigmas de termes positifs :

$$dist_{TP}(t_1,t_2) = \sum_{n=depth(LCST(t_1,t_2))}^{depth(t_1)-1} \left[\frac{1}{2}\right]^n + \sum_{n=depth(LCST(t_1,t_2))}^{depth(t_2)-1} \left[\frac{1}{2}\right]^n$$

La seule façon pour que cette somme soit nulle est que les deux sigmas soient nuls et pour cela il ne faut aucune itération de ces sigmas *i.e.*:

$depth(t_1)= depth(LCST(t_1,t_2))$ qui n'est possible que si $LCST(t_1,t_2)=t_1$

et

$depth(t_2)= depth(LCST(t_1,t_2))$ qui n'est possible que si $LCST(t_1,t_2)=t_2$

donc

$dist_{TP}(t_1,t_2) = 0 \Rightarrow LCST(t_1,t_2)=t_1 \wedge LCST(t_1,t_2)=t_2 \Rightarrow t_1=t_2$

(4) Montrons que $dist_{TP}(t_1,t_2) \leq dist_{TP}(t_1,t') + dist_{TP}(t',t_2)$

Par définition le LCST est le plus petit commun sur-type et, comme nous sommes dans un arbre, il n'y en a qu'un seul. Dans un arbre, s'il existait un autre chemin plus court de t_1 à t_2 alors il devrait exister un autre LCST ce qui est absurde.

Sinon on peut aussi remarquer que

$$dist_{TP}(t_1,t') + dist_{TP}(t',t_2) = \frac{1}{2^{depth(LCST(t_1,t'))-2}} + \frac{1}{2^{depth(LCST(t_2,t'))-2}} - \frac{1}{2^{depth(t_1)-1}} - \frac{1}{2^{depth(t_2)-1}} - \frac{1}{2^{depth(t')-2}}$$

et en considérant la formule (7) nous construisons la différence D suivante :

$$D = dist_{TP}(t_1,t') + dist_{TP}(t',t_2) - dist_{TP}(t_1,t_2) = \frac{1}{2^{depth(LCST(t_1,t'))-2}} + \frac{1}{2^{depth(LCST(t_2,t'))-2}} - \frac{1}{2^{depth(LCST(t_1,t_2))-2}} - \frac{1}{2^{depth(t')-2}}$$

Nous cherchons alors à prouver que $\forall t'\ D \geq 0$

(1) Si $t' > LCST(t_1,t_2)$ alors

$depth(LCST(t_1, t') < depth(t') < depth\ LCST(t_1,t_2)$

$depth(LCST(t_2, t') < depth(t') < depth\ LCST(t_1,t_2)$

donc D est positive

(2) Si $t' < LCST(t_1,t_2)$ alors considérons t_1,

si $t'<t_1$ alors $LCST(t_1, t') = t_1$ et $LCST(t_2, t') = LCST(t_1,t2)$ donc

$D = \frac{1}{2^{depth(t_1)-2}} - \frac{1}{2^{depth(t')-2}}$ et comme $t'<t_1$ on a $depth(t')> depth(t_1)$ donc $D>0$

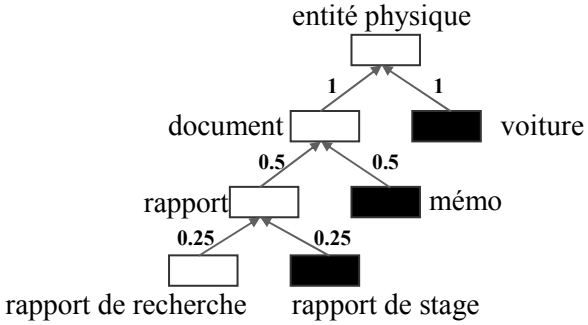
si $t_1<t'$ alors $LCST(t_1,t')=t'$ et comme $t_1<t'<LCST(t_1,t_2)$ on a $LCST(t_2,t')=LCST(t_1,t_2)$ donc $D=0$

idem en considérant t_2

(3) Si $t'=t_1$ ou $t'=t_2$ ou $t'=LCST(t_1,t_2)$ D est instantanément nulle

(4) Sinon t' et $LCST(t_1,t_2)$ sont dans deux branches distinctes de l'arbre donc :
$LCST(t_1,t')> LCST(t_1,t_2)$ et $LCST(t_2,t')> LCST(t_1,t_2)$
donc $depth(LCST(t_1,t'))<depth(LCST(t_1,t_2))$ et
$depth(LCST(t_2,t'))>depth(LCST(t_1,t_2))$ et $D>0$

Cependant pour construire le dendrogramme voulu dans KmP, il nous fallait une ultra-métrique, ce qui n'est pas le cas de $dist_{TP}$. En effet, considérons le contre-exemple suivant :



Soit t_1*="rapport de stage"* et t_2*="voiture"* alors $dist_{TP}(t_1,t_2)=2,75$

si nous prenons t'*="document"* $dist_{TP}(t_1,t')=1,75$ et $dist_{TP}(t',t_2)=2$ donc

donc $dist_{TP}(t_1,t_2) \not\leq max\ (dist_{TP}(t_1,t'), dist_{TP}(t',t_2))$

On ne peut donc pas construire un arbre indicé directement avec cette distance. Comme pressenti, l'arbre ontologique avec sa distance classique ne correspond pas directement à une hiérarchie indicée.

Remarquons d'abord que la différence entre notre taxonomie de types et les hiérarchies indicées des ultra-métriques est que dans le cas de la taxonomie ontologique, la distance instance-racine n'est pas unique et maximale comme dans les hiérarchies indicées où la distance singleton-racine est toujours égale à la hauteur de la hiérarchie indicée.

Nous avons donc regardé les classes de distances entre types, au lieu de chercher une correspondance directe classe-type. Une première distance candidate avait été identifiée en utilisant l'opérateur *max(...)* :

$$dist_{MH}(t_1,t_2) = \max_{\forall t<LCST(t_1,t_2)} (Length(t,ST(t)) = \left[\frac{1}{2}\right]^{depth(LCST(t_1,t_2))}$$

avec *ST(t)* le super-type de *t*

Cependant, comme le montre la Figure 52, cette transformation recrée des niveaux équivalents aux niveaux de l'ontologie, ce qui n'apporte pas une grande valeur ajoutée d'un point de vue usage.

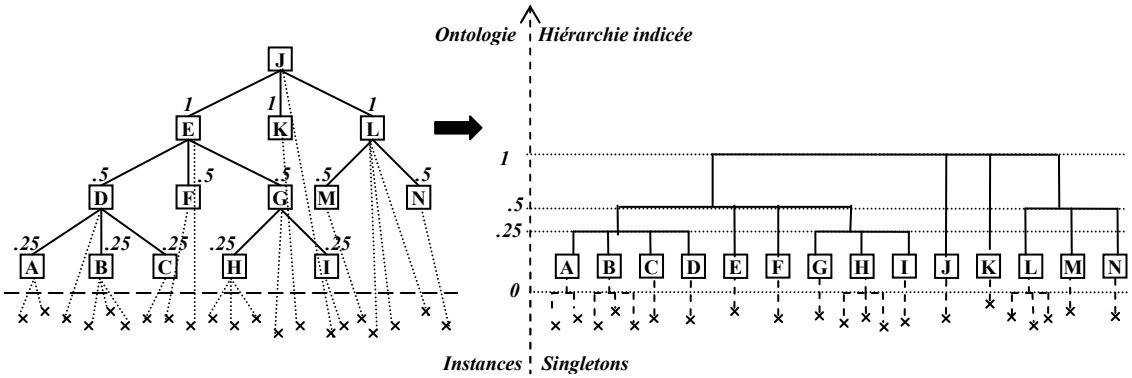


Figure 52. Transformation effectuée par $dist_{MH}$

En voulant éviter que les classes soient regroupées par niveau d'ontologie, nous avons cherché un autre critère. Le critère qui correspond à l'application de KmP revient, à profondeur égale, à regrouper les classes les moins détaillées en premier. D'où une deuxième distance candidate :

$$dist_{CH}(t_1,t_2) = \max_{\forall st\leq LCST(t_1,t_2)} \left(SubPath\left(st, LCST(t_1,t_2)\right)\right) \quad \text{quand } t_1 \neq t_2$$

$$dist_{CH}(t_1,t_2) = 0 \text{ quand } t_1 = t_2$$

$dist_{CH}$ permet lors des regroupements de différencier les classes regroupant déjà des niveaux de détails des classes plus superficielles. A titre d'exemple, la Figure 53 montre comment $dist_{CH}$ crée un niveau supplémentaire par rapport à $dist_{MH}$ en Figure 52.

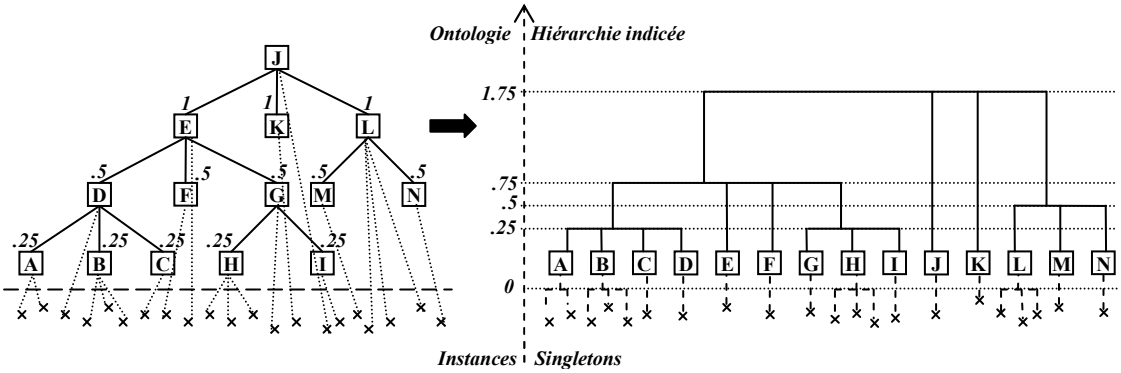


Figure 53. Transformation effectuée par $dist_{CH}$

Pour être précis, $dist_{MH}$ prend ses valeurs dans $E_{MH} = \left\{ \frac{1}{2^n}; 0 \le n < D \right\} \cup \{0\}$

alors que $dist_{CH}$ prend ses valeurs dans $E_{CH} = \left\{ \sum_{i=m}^{n} \frac{1}{2^i}; 0 \le m \le n < D \right\} \cup \{0\}$

où D est la profondeur maximale de l'ontologie.

Donc, pour $dist_{MH}$ le nombre maximum d'échelons possibles dans la hiérarchie indicée est $Card(E_{MH})=D+1$

Pour $dist_{CH}$, le nombre maximum d'échelons possibles dans la hiérarchie indicée est calculé ainsi : pour une profondeur d, le nombre d'échelons possibles $NL(d)$ est récursivement défini par

$NL(d) = NL(d-1) + s$

car une profondeur supplémentaire rajoute un arc possible à tous les chemins partant de la racine donc comme $NL(0) = 1$ (un seul échelon pour une profondeur nulle) d'où :

$$Card(E_{CH}) = NL(D) = \left(\sum_{n=1}^{D} n \right) + 1 = \frac{D(D+1)}{2} + 1 = \frac{D^2}{2} + \frac{D}{2} + 1$$

Par conséquent pour une même profondeur D, on a une différence de précision possible entre les deux distances candidates, donnée par :

$$Card(E_{CH}) - Card(E_{MH}) = \frac{D^2}{2} - \frac{D}{2} = \frac{D}{2}(D-1) > 0 \text{ avec D} > 1$$

Donc la distance $dist_{CH}$ donnera potentiellement plus d'échelons et dans un ordre de grandeur quadratique par rapport à la profondeur.

Nous avons donc retenu cette distance et avons dû prouver alors qu'il s'agissait bien d'une ultra-métrique :

(I) $dist_{CH}(t,t)=0$ par définition.

(II) Montrons que $dist_{CH}(t_1,t_2) = dist_{CH}(t_2,t_1)$

$$dist_{CH}(t_1,t_2) = \max_{\forall st \le LCST(t_1,t_2)} \left(dist\left(st, LCST(t_1,t_2)\right)\right)$$

$$= \max_{\forall st \le LCST(t_2,t_1)} \left(dist\left(st, LCST(t_2,t_1)\right)\right) = dist_{CH}(t_2,t_1)$$

puisque LCST(t_1,t_2)= LCST(t_2,t_1) *i.e.* le plus petit commun sur-type à deux types est indépendant de l'ordre dans lequel on considère les deux types.

(III) Montrons que $dist_{CH}(t_1,t_2) = 0 \Rightarrow t_1=t_2$

$$dist_{CH}(t_1,t_2) = \max_{\forall st \le LCST(t_1,t_2)} \left(dist\left(st, LCST(t_1,t_2)\right)\right)$$

$$\ge dist\left(t_1, LCST(t_1,t_2)\right) + dist\left(t_2, LCST(t_1,t_2)\right) > 0$$

donc le seul moyen que $dist_H(t_1,t_2) = 0$ c'est que $t_1=t_2$

(IV) montrons que $\forall t'\ dist_{CH}(t_1,t_2) \le max(dist_{CH}(t_1,t'), dist_{CH}(t_2,t'))$

Si $t_1=t_2$ alors $dist_{CH}(t_1,t_2) =0$ et l'inégalité est vérifiée.

S $t' \le t_1$ and $t_1 \not\le t_2$ et $t' \not\le t_2$ alors

$$dist_{CH}(t',t_2) = \max_{\forall st \le LCST(t',t_2)} \left(dist\left(st, LCST(t',t_2)\right)\right) = \max_{\forall st \le LCST(t_1,t_2)} \left(dist\left(st, LCST(t_1,t_2)\right)\right)$$

puisque qi $t' \le t_1$ alors $LCST(t',t_2)=LCST(t_1,t_2)$ ou plus généralement le plus petit sur-type commun de t_1 et t_2 est le même que pour les sous-types de t_1 et t_2 puisque nous sommes dans un arbre.

Si $t' \le t_2$ et $t_2 \not\le t_1$ et $t' \not\le t_1$ le raisonnement est le même *mutatis mutandis*.

Si $t_1 \le t_2$ et $t' \le t_2$ alors $LCST(t',t_2)=t_2$ et $LCST(t_1,t_2) =t_2$ donc $dist_{CH}(t_2,t')= dist_{CH}(t_1,t_2)$. Si $t_2 \le t_1$ et $t' \le t_1$ le raisonnement est le même, *mutatis mutandis*.

If $t' \not\le t_2$ and $t' \not\le t_1$ alors $LCST(t_1,t_2) \le LCST(t',t_1)$ ou $LCST(t_1,t_2) \le LCST(t',t_2)$ si non nous aurions $LCST(t_1,t_2)>LCST(t',t_1)$ et $LCST(t_1,t_2)>LCST(t',t_2)$ et comme $t_1 \ne t_2$, $t' \not\le t_2$ and $t' \not\le t_1$, il faudrait que t' ou un de ses ancêtres ait deux frère ce qui est impossible dans un arbre. Donc $LCST(t_1,t_2) \le LCST(t',t_1)$ alors $dist_{CH}(t_1,t_2) \le dist_{CH}(t_1,t')$ puisque $\{st\ ;\ st \le LCST(t_1,t_2)\} \subset \{st\ ;\ st \le LCST(t',t_1)\}$. De même si $LCST(t_1,t_2) \le LCST(t',t_2)$ alors $dist_{CH}(t_1,t_2) \le dist_{CH}(t_2,t')$ puisque $\{st\ ;\ st \le LCST(t_1,t_2)\} \subset \{st\ ;\ st \le LCST(t',t_2)\}$. Ainsi donc dans les deux cas, l'inégalité est vérifiée.

Donc $dist_{CH}$ est une ultra-métrique qui peut être utilisée pour définir des niveaux de détails par exemple dans une interface.

Avant de quitter les aspects formels de $dist_{CH}$, prouvons que les distances entre classes plus profondes sont plus petites mais qu'à profondeur égale les classes ayant le moins de sous classes ont des distances plus petites. Soit la configuration suivante (Figure 54) où A_D est de profondeur maximale et B_d est de profondeur d mais sans enfant.

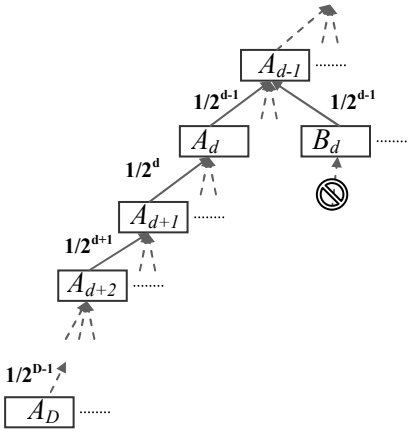


Figure 54. Exemple pour la vérification de l'ordre de regroupement

La distance maximale entre deux sous-classes de profondeur supérieure à d dans une ontologie de profondeur D est :

$$distsub_{Max}(d,D)=\sum_{i=d}^{D-1}\left[\frac{1}{2}\right]^{i}=\frac{1}{2^{d}}\sum_{i=0}^{D-d-1}\frac{1}{2^{i}}=\frac{1}{2^{d}}\times\frac{1-\frac{1}{2^{D-d}}}{\frac{1}{2}}=\frac{1}{2^{d-1}}\left(1-\frac{1}{2^{D-d}}\right)=\frac{1}{2^{d-1}}-\frac{1}{2^{D-1}}$$

avec $\lim_{D\to\infty} distsub_{Max}(d,D)=\frac{1}{2^{d-1}}$

La distance minimale entre classes de profondeur d est $dist_{Min}(d)= 1/2^{d-1}$

Donc comme l'ontologie est de profondeur finie : $dist_{Min}(d)>distsub_{Max}(d,D)$

Donc les classes sont regroupées par ordre croissant de profondeur et décroissant de détails *i.e.* pour un même niveau de profondeur, les classes ayant peu de descendants seront regroupées en premier mais après que les descendants de chaque classe de ce niveau aient été regroupés.

Enfin puisque le regroupement suit la hiérarchie de l'ontologie on peut donner un nom à tous les regroupements *R* à n'importe quelle étape du regroupement

Name(Cl) = Name (LCST({t ; type t ∈ Cl}))

Un exemple de regroupement des compétences sur la Télécom Valley est donné en Figure 55.

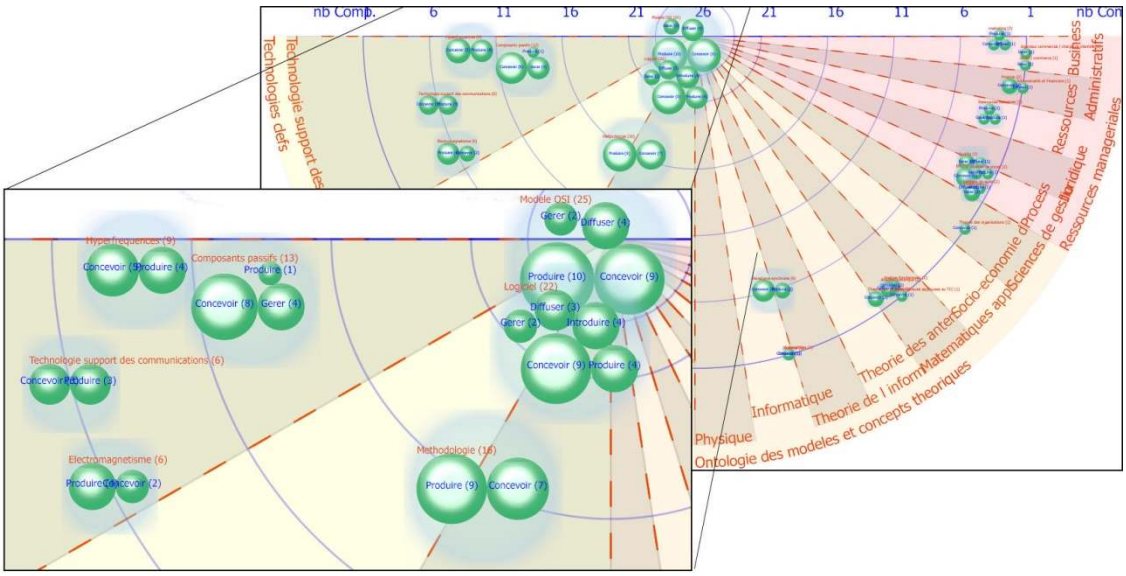


Figure 55. Vue en Radar sur 180° des regroupements de compétences.

4.5 Les distances à l'état sauvage

En parallèle avec ces premières explorations des caractéristiques, interprétations et applications des distances conceptuelles, nous avons commencé à questionner la valeur de ces distances et leur fidélité par rapport aux proximités naturellement ressenties par les humains.

Pour cela, nous avons commencé une étude empirique et statistique. L'une des hypothèses testées fut : *"Est-il juste de considérer que les frères sont à égale distance du père et à égale distance les uns des autres ou est-ce un effet secondaire du fait que l'on repose sur la structure des chemins de subsomption ?"*.

Afin d'étudier ces distances dans leur « milieu naturel » et de les comparer avec leurs simulations informatiques, nous avons conçu une plate-forme permettant de réaliser, gérer et d'analyser des expériences où les participants organisent et regroupent spatialement des concepts selon leur proximité intuitive [117] [118]; voir Figure 56.

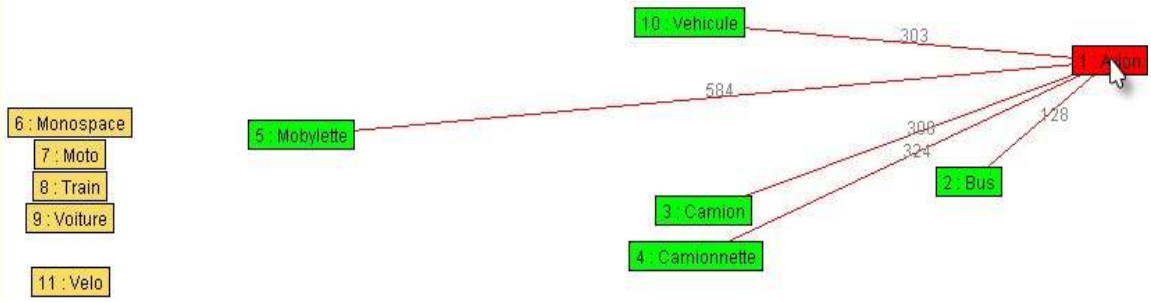


Figure 56. Applet de l'exercice de placement

A partir de ces exercices, des analyses statistiques sont faites. Prenons l'exercice de la Figure 56 effectué par 30 participants de 13 à 50 ans. Les distances capturées ont été normalisées avant d'en calculer la moyenne, l'écart type et la variance. La Figure 57 montre les distances entre le concept *camion* et la liste des autres concepts à placer.

On peut lire sur ce graphique que le concept *camionnette* est en moyenne très proche de *camion* et, étant donné que la variance est très faible, qu'il s'agit d'un consensus. Le concept *véhicule* est particulier dans cette liste puisque dans une ontologie n'utilisant que ces concepts il serait naturellement placé comme père des autres concepts et que ces autres concepts seraient entre eux des frères. On voit notamment que la distance entre *camion* et ses frères est parfois plus petite (4 cas) et parfois plus grande (5 cas) que la distance à son père *véhicule* ce qui serait impossible à faire dans un hiérarchie de types classique comme celles utilisées dans la majorité des distances puisque l'on passe forcément par le père pour aller à un frère ; notons que l'on peut introduire des classes abstraites pour résoudre ce problème et faire de véhicule un ancêtre plus éloigné. Il reste beaucoup à faire dans cette étude mais si ces résultats se confirmaient, ils montreraient qu'une structure de subsomption seule ne permet pas de simuler de tels comportements.

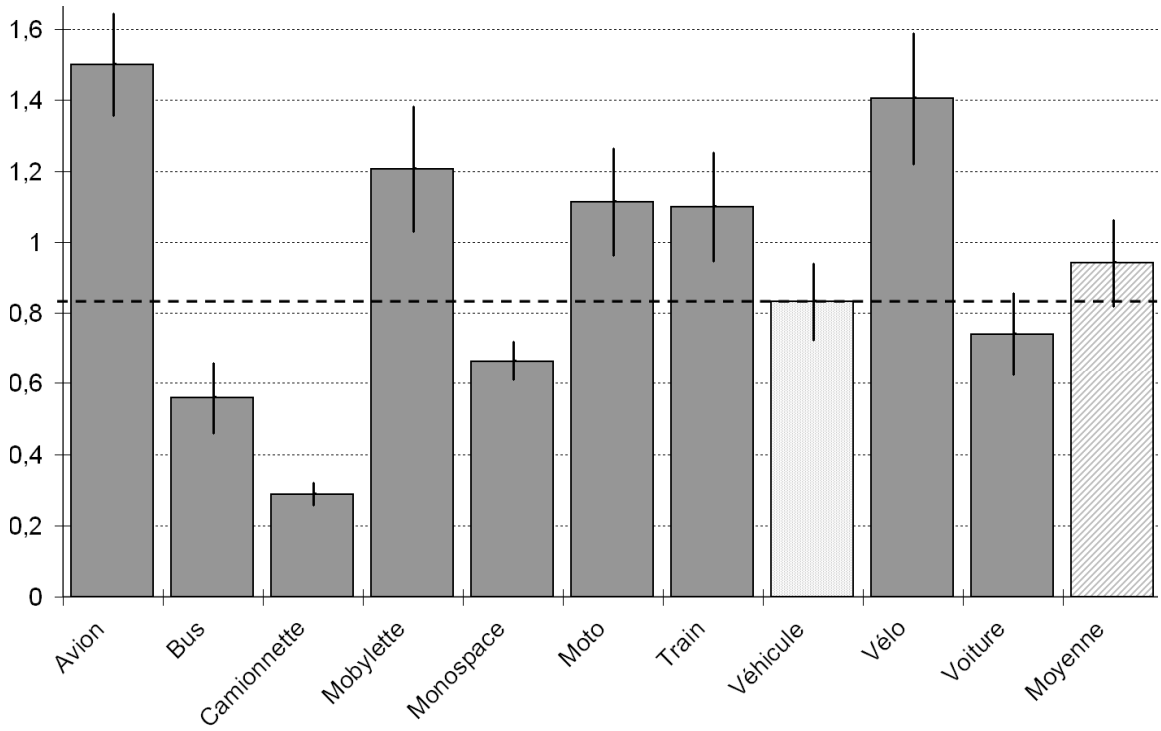


Figure 57. Distances moyenne entre Camion et d'autres véhicules sur un échantillon de trente personnes.

4.6 Distance de cooccurrence et contexte en extension

Nous présentons dans cette section, notre premier essai pour considérer la proximité d'usage de deux types *i.e.* la fréquence avec laquelle ces deux types sont employés ensemble dans des descriptions. Soient deux types de concepts $t_x, t_y \in H_c^2$, on définit le comptage de cooccurrences comme le nombre de triplets impliquant ces deux types :

$$count(t_x,t_y) = || \{t \text{ triplets RDF} \mid t = (x,p,y) \wedge (x,rdf:type,T_x) \wedge (y,rdf:type,T_y)\} ||$$

On définit alors une dissimilarité en prenant l'inverse :

$$disim_{count}(t_x,t_y) = \frac{1}{1+count(t_x,t_y)} \in [0,1]$$

Nous ne parlons pas de distance ici car cette fonction $disim_{count}$ ne remplit pas les conditions de l'inégalité triangulaire : il nous suffit de trouver t_z tel que des ressources typées par t_x et t_z soient très fréquents et idem pour t_y et t_z alors on peut avoir :

$$disim_{count}(t_x,t_y) = \frac{1}{1+count(t_x,t_y)} \geq \frac{1}{1+count(t_x,t_z)} + \frac{1}{1+count(t_z,t_y)} = disim_{count}(t_x,t_z) + disim_{count}(t_z,t_y)$$

Il en va de même pour les autres propriétés : $disim_{count}(t_1,t_1) \neq 0$ et $disim_{count}(t_1,t_2) = 0 \not\Rightarrow t_1 = t_2$ puisque par construction 0 est la valeur limite à l'infini et un type n'est pas forcément souvent utilisé avec lui-même dans un même triplet, $disim_{count}(t_1,t_2) \neq disim_{count}(t_2,t_1)$ puique par défaut un prédicat n'est pas symmétrique,

Nous n'avons donc absolument pas une distance au sens mathématique mais cette dissimilarité capture une proximité d'usage des types. En effet, comme nous le disions en section 4.2, une même signature de relation peut engendrer bien des familles de triplets par spécialisation des types spécifiés. De plus, détail important, la signature en RDFS est utilisée pour de l'inférence de type (ajout supplémentaire de types) et non de la validation de type. Par conséquent, la variété effective des types sur les instances desquels une propriété est utilisée peut être bien plus grande que la liste des types obtenue par la fermeture transitive de la subsumption de ses domaine et co-domaine.

Reprenons l'exemple de la section 4.2 où dans l'ontologie, il existe une propriété Auteur avec la signature :

`[Document] → (Auteur) → [Personne]`

Cette dissimilarité pourra tenir compte de l'existence des instances de triplets suivantes dans son calcul :

`[Plan] → (Auteur) → [Ingénieur]`

`[Guide] → (Auteur) → [Commercial]`

Une utilisation immédiate de cette distance est d'assister les utilisateurs dans des interfaces de navigation (ex: *tag cloud* Figure 58), de requête ou d'annotation en suggérant des types co-occurrents au dernier type sélectionné (Figure 59).

Figure 58. Nuages de termes obtenus avec la distance en fonction d'un concept sélectionné par l'utilisateur

Figure 59. Suggestion dans l'interface en utilisant la distance pour ordonner les options alors qu'un utilisateur construit une requête simple.

4.7 Distance de signatures et contexte en intension

L'ontologie capture des caractérisations en intension des concepts et des relations du domaine, et en particulier des relations en intension entre ces concepts. Nous l'avons vu, l'espace le plus utilisé par les distances à base d'ontologies est le graphe de la hiérarchie des types de concepts. Nous nous intéressons ici à augmenter ce graphe avec les chemins de la hiérarchie des types de relations et les chemins des signatures des relations. Rappelons que la signature d'une relation ou propriété est la spécification des types de concepts qu'elle relie. En RDF cette signature concerne les deux arguments de la propriété dont les types sont spécifiés par les valeurs des propriétés *range* et *domain*. Formellement, on repose donc sur une première extension du méta-modèle RDFS introduisant une propriété symétrique parente directe du domaine (*domain*) et co-domaine (*range*) et appelée signature (définition 1).

Définition 1 : la propriété `cos:signature` est telle que

`rdfs:domain`$(T_p, T_x) \Rightarrow$ `cos:signature`(T_p, T_x) // généralisation du domaine

`rdfs:range`$(T_p, T_x) \Rightarrow$ `cos:signature`(T_p, T_x) // généralisation du co-domaine

`cos:signature`$(T_x,T_y) \Leftrightarrow$ `cos:signature`(T_x,T_y) // symétrie

Définition 2 : un chemin de signatures $C_S(T_x,T_y)$ entre les types $T_x,T_y \in H_C^2$ est un chemin de T_x à T_y composé exclusivement d'arcs de type `cos:signature` inférés à partir des signatures de H_R : $C_S(T_x,T_y):=< T_x,$ *signature,* $T_1,$ *signature,* $T_2,$ *signature,...signature,* $T_n,$ *signature* $T_y>$

Définition 3 : une distance de signature $d_S(T_x,T_y)$ entre les types $T_x,T_y \in H_C^2$ est définie par $d_S(T_x,T_x):=0$ et $d_S(T_x,T_y):= min_{\{Ci \in \{ Cs(Tx,Ty) \}\}} long(C_i)$ avec

$long(<T_x,$ *signature,* $T_1,$ *signature,* $T_2,$ *signature, ...* *signature,* $T_n,$ *signature* $T_y >) := n$

Intuitivement, avec cette distance, deux types sont proches s'il y a (en intension) une possibilité de les impliquer dans un graphe d'annotation concis. Par exemple *document* et *pays* peuvent être proches dans une ontologie s'il existe les déclarations suivantes (et les implications par subsomption et symétrie) :

rdfs:domain(auteur, document) $\Rightarrow$ *cos:signature(document, auteur)*

rdfs:range(auteur, personne) $\Rightarrow$ *cos:signature(auteur, personne)*

rdfs:domain(nationalité, personne) $\Rightarrow$ *cos:signature(personne, nationalité)*

rdfs:range(nationalité, pays) $\Rightarrow$ *cos:signature(nationalité, pays)*

soit $d_S(document, pays)=3$

Cette première version permet de comprendre le principe mais ne rend pas compte de la subsomption des types de relations et des types de leur signature ; en d'autres termes, si une relation (ex : titre) spécifie un type (ex : document) dans sa signature, elle inclut les sous-types de ce type (ex : livre, roman, etc.) et cette spécification se propage au sous-type de la relation (ex : sous-titre, titre court, etc). Pour rendre compte de ce point, nous avons défini et nous expérimentons la distance suivante (définition 4).

Définition 4 : la propriété `cos:sous-type-et-signature` est telle que

$\texttt{cos:signature}(T_p, T_x) \Rightarrow \texttt{cos:sous-type-et-signature}(T_p, T_x, w_{sig})$

$\texttt{rdfs:subClassOf}(T_p, T_x) \Rightarrow \texttt{cos:sous-type-et-signature}(T_p, T_x, w_{subclass})$

$\texttt{rdfs:subPropertyOf}(T_p, T_x) \Rightarrow \texttt{cos:sous-type-et-signature}(T_p, T_x, w_{subprop})$

$\texttt{cos:sous-type-et-signature}(T_x, T_y, w) \Leftrightarrow \texttt{cos:sous-type-et-signature}(T_x, T_y, w)$

Les poids introduits permettent de contrôler le bruit que l'on génère en mélangeant deux types de relations. Les définitions du chemin et de la distance sont des adaptations des définitions 2 et 3 *mutatis mutandis* avec l'exception que la longueur d'un chemin est maintenant pondérée : *long(<sous-type-et-signature* (C_x, X_1, w_0)*, sous-type-et-signature* (X_1, X_2, w_1)*, …, sous-type-et-signature*(X_n, C_y, w_n) *>) :=*Σw_n

Une utilisation testée actuellement pour cette distance est le pilotage de l'analyse de la langue naturelle utilisée pour lever l'ambigüité dans la génération automatique des annotations à partir de textes [39]. Il s'agit de prédire les patrons d'annotation les plus probables pour l'extraction de connaissances d'un texte donné en fonction d'une ontologie donnée. Très tôt les similarités et distances sémantiques ont été utilisées pour désambigüiser les termes par exemple avec Renik [105] dont on a déjà vu la distance. Plus récemment Baziz [119] utilise aussi une similarité sémantique entre concepts pour désambiguïser des termes via le réseau sémantique d'une ontologie. Il utilise l'ontologie dans un premier temps pour récupérer les différents sens possibles, puis pour calculer les similarités entre les différents sens de ces concepts en utilisant des mesures de proximité sémantique connues dans la littérature.

La Figure 60 montre une expérience dans laquelle, en jouant sur les poids de la définition 4, on passe en neuf étapes d'une distance n'utilisant que les chemins de subsomption à une distance n'utilisant que des chemins de signatures. Pour chaque jeu de poids, la courbe donne la précision de la désambiguïsation pour 10 termes apparaissant chacun dans 100 documents. Le point intéressant ici est que le meilleur résultat est obtenu pour un espace métrique mixte.

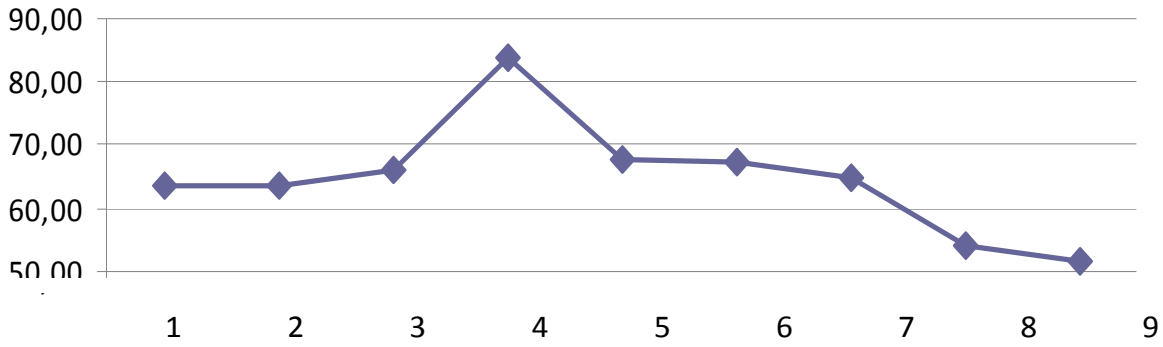


Figure 60. Précision de la désambiguïsation entre subsomption et signature

4.8 Comparaisons d'espaces métriques : LSA vs. ontologie

Dans le projet Edccaeteras (Expériences sur les Distances Conceptuelles, Campagnes et Analyses pour ETayer l'Elaboration de Représentations et Algorithmes de Simulation) nous nous sommes intéressés à comparer deux espaces métriques obtenus très différemment [120]: celui de la distance utilisée dans CORESE donc basée sur une ontologie et celui d'un espace construit par la méthode LSA.

L'analyse Sémantique Latente ou LSA [121] construit un espace vectoriel à partir d'un corpus de textes.

LSA utilise une matrice *X* qui décrit les occurrences des termes d'un corpus dans les documents de ce corpus. C'est une matrice creuse dont les lignes correspondent aux « termes » et dont les colonnes correspondent aux « documents ». Formellement, l'élément $x_{i,j}$ de *X* donne la fréquence du terme *i* dans le document *j* ; un vecteur ligne t_i de *X* correspond à un terme, et ses composantes donnent l'importance de ce terme dans chaque document ; un vecteur colonne d_j de *X* correspond à un document, et ses composantes donnent l'importance de chaque terme dans le texte de ce document.

$$X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{pmatrix} = \begin{pmatrix} t_1^T \\ \vdots \\ t_m^T \end{pmatrix} = (d_1 \quad \dots \quad d_n)$$

$$t_i^T = (x_{1,1} \quad \dots \quad x_{1,n})$$

$$d_j = \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{m,1} \end{pmatrix}$$

Les termes issus du corpus peuvent être lemmatisés et corrigés. Les fréquences sont obtenues en général par la méthode du TF*IDF [122]. On effectue alors une décomposition de *X* en valeurs propres, qui donne deux matrices orthonormales *U* et *V* et une matrice diagonale *Σ* contenant les valeurs propres σ_i de *X*

$$X = U\Sigma V^T = (u_1 \quad \dots \quad u_l) \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_l \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_l \end{pmatrix} = \begin{pmatrix} \hat{t}_1^T \\ \vdots \\ \hat{t}_l^T \end{pmatrix} \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_l \end{pmatrix} (\hat{d}_1 \quad \dots \quad \hat{d}_l)$$

Cette décomposition permet ensuite de réduire le rang des matrices en ne considérant que les *k* plus grandes valeurs propres, ainsi que les vecteurs propres correspondants dans *U* et *V*, on obtient une approximation de rang *k* de la matrice des occurrences.

LSA considère que cette approximation, traduit les vecteurs « termes » et « documents » dans l'espace des « concepts » ; cet espace ayant réduit de taille chaque dimension correspond maintenant à des termes qui ont été regroupés par corrélation.

Le vecteur $\hat{t}_i$ possède alors k composantes, qui chacune donne l'importance du terme *i* dans chacun des *k* différents « concepts ». De même, le vecteur $\hat{d}_j$ donne l'intensité des relations entre le document *j* et chaque concept.

On peut alors utiliser les vecteurs pour comparer les documents et les termes entre eux. Pour les termes, on peut ainsi utiliser la similarité par cosinus pour comparer les termes x et p en comparant leurs vecteurs $\hat{t}_x$ et $\hat{t}_y$ ainsi :

$$sim_{LSA}(x,y) = \cos(\widehat{\hat{t}_x, \hat{t}_y}) = \frac{\hat{t}_x \cdot \hat{t}_y}{\|\hat{t}_x\| \cdot \|\hat{t}_y\|}$$

Ainsi LSA transforme la matrice des occurrences en une « relation » entre les termes et des « concepts », et une relation entre ces concepts et les documents. Cette relation peut être vue comme une similarité. Cette organisation entre termes et concepts est employée pour :

- la comparaison de documents dans l'espace des concepts (regroupement, classement et partitionnement de documents) ;
- la recherche de relations entre les termes (résolution de synonymie et de polysémie) ;
- étant donnée une requête, traduire les termes de la requête dans l'espace des concepts, pour retrouver des documents liés sémantiquement (recherche d'information).

La première expérience effectuée dans *Edccaeteras* s'est intéressée à regarder la corrélation entre des distances obtenues par analyse LSA d'un corpus et des distances obtenues par la distance de CORESE pour deux ontologies plus ou moins structurées. Le corpus utilisé rassemble plusieurs années du journal « Le Monde ». La première ontologie est volontairement extrêmement plate, Figure 61 :

Figure 61. Première ontologie pour la comparaison LSA vs. distance ontologique

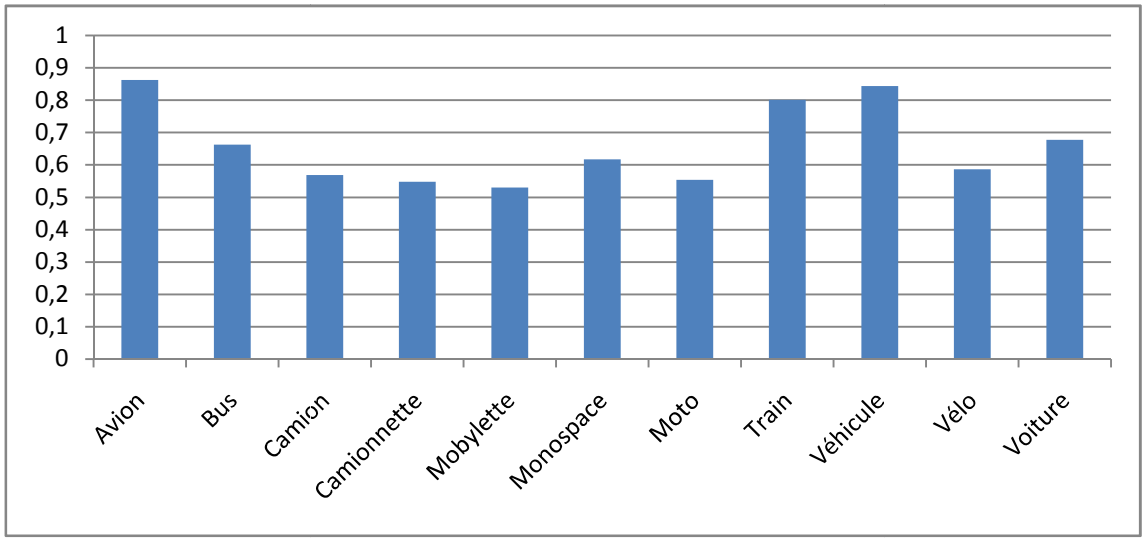


Figure 62. Corrélations LSA distance - ontologique pour la première ontologie

Avec une moyenne de 0,659 et un écart de moyen de 0,100 ces premières corrélations ne sont pas excellentes, sachant qu'une corrélation intéressante doit se rapprocher de 1.

Si maintenant nous augmentons la structuration de l'ontologie, Figure 63 :

Figure 63. Deuxième ontologie pour la comparaison LSA vs. distance ontologique

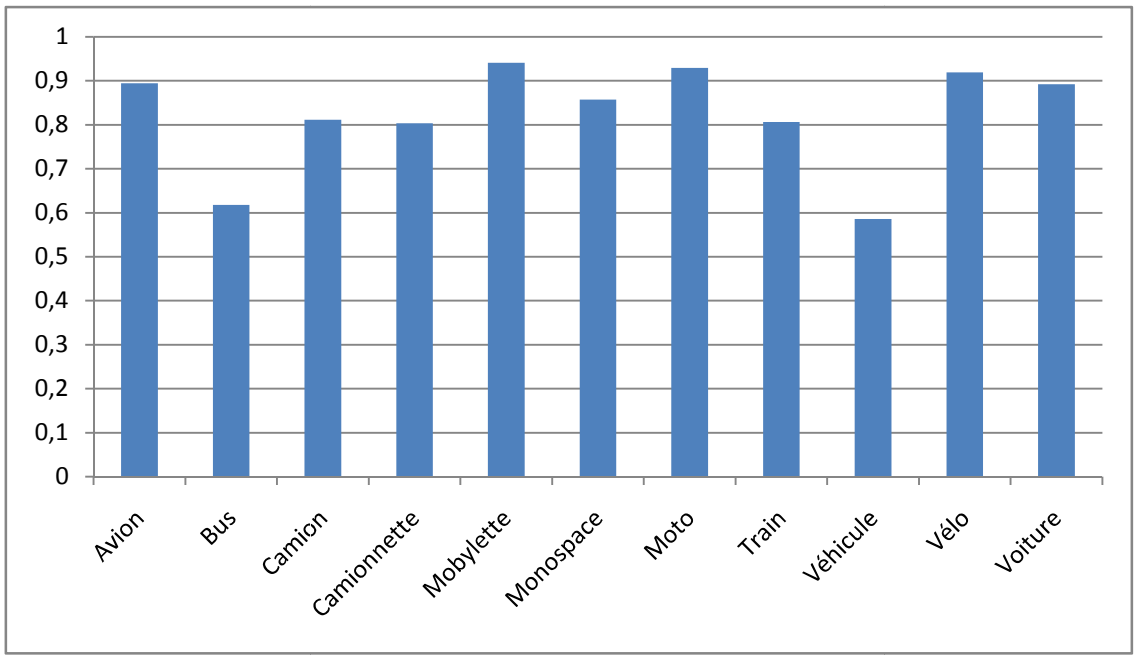


Figure 64. Corrélations LSA distance - ontologique pour la deuxième ontologie

Avec une moyenne de 0,823 et un écart de 0,089 ces deuxièmes corrélations sont bien meilleures. Ceci permet de considérer deux perspectives intéressantes :

1. Les distances à base d'ontologies offrent une bonne approximation de ce que l'on obtient par des statistiques.
2. Les distances à base de corpus peuvent aussi aider à valider des modélisations par exemple en signalant les endroits où se trouve une mauvaise corrélation entre le modèle ontologique et une analyse LSA.

D'autres expériences menées dans *Edccaeteras* ont donné des résultats préliminaires qui restent encore à analyser et confirmer [120] en particulier :

- La lemmatisation du corpus n'améliore pas systématiquement les performances de LSA ; il semble en effet que dans certains cas (ex forme plurielle « heure » / « heures ») les différences soient représentatives d'une différence conceptuelle qui est alors perdue (ex : instant ou date / durée période)
- Nous avons dans une expérience artificiellement « enrichi » les textes en ajoutant des synonymes et hyperonymes des termes des documents dans le corps du document même pour simuler l'ajout de connaissances ontologiques dans l'espace LSA. Les premiers résultats n'ont pas montré d'amélioration et parfois même une détérioration. La Figure 65 montre un exemple où nous cherchons à retrouver les mots-clefs associés à un rapport de recherche en fonction de son résumé.

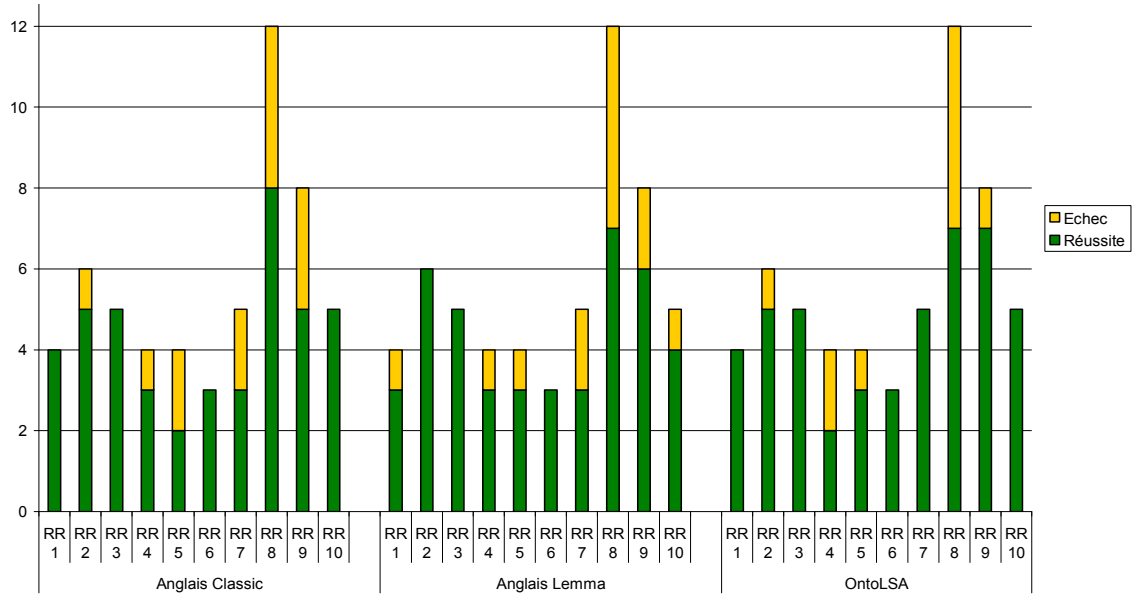


Figure 65. Performance à l'attribution de mots-clés pour 10 rapports de recherche de l'INRIA en fonction des différents espaces métriques : LSA classique, LSA sur un corpus lemmatisé et LSA augmenté par des répétitions de termes basées sur une ontologie. L'ontologie utilisée est basée sur le thésaurus ACM98.

4.9 Conclusion

Nous avons montré qu'une ontologie offre un support à d'autres types de raisonnement que la dérivation logique notamment à travers ses structures graphiques.

Ainsi la hiérarchie de notions qui sert de squelette à presque toute ontologie peut être vue comme un espace métrique permettant de définir des distances pour comparer la proximité sémantique de deux notions. Nous avons vu comment mettre en œuvre cette idée dans plusieurs scénarios comme l'allocation distribuée d'annotations, la recherche approchée ou le *clustering*.

Nous avons montré différentes utilisations que nous avons faites des distances sémantiques et les caractéristiques de ces métriques en discutant l'intérêt de certaines propriétés en fonction du comportement recherché.

En donnant les scénarios d'utilisation et les distances utilisées dans un échantillon représentatif de projets que nous avons menés, nous fournissons une première série d'expériences démontrant l'intérêt et le potentiel des distances et soulignant l'importance du travail restant à faire pour identifier et caractériser les familles de distances existantes et leur adéquation respective aux tâches pour lesquelles les utilisateurs souhaitent être assistés.

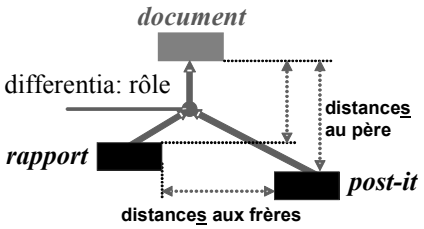
Nous avons enfin discuté l'appellation même de distance. En effet en intelligence artificielle symbolique, nous cherchons, par des systèmes symboliques et des manipulations automatiques, à simuler des comportements intelligents. L'appellation de 'distance' relève cependant plus de la métaphore que de la définition mathématique. Il nous parait donc plus important de nous assurer de l'adéquation, de la pertinence et de l'interprétation de ces formules plus que de chercher à en faire à tout prix des distances au sens mathématique. Il est évidement important de caractériser ces fonctions que nous obtenons mais rien ne prouve que les fonctions humaines, « les distances au naturel » que nous cherchons à simuler, soient des distances au sens mathématique.

4.10 Perspectives

Nous l'avons vu en introduction, il existe deux grandes approches pour calculer les distances : utiliser des informations extérieures aux modèles des connaissances (ex : statistiques sur des corpus de textes) ou reposer uniquement sur la structure des modèles (ex : la hiérarchie des types). Nous observons actuellement comment ces approches se déclinent sur l'utilisation d'informations extérieures (extraites d'une base d'annotations RDF) et sur des extensions de la structure habituellement exploitée dans les modèles (ici la hiérarchie de types en RDFS).

Les distances conceptuelles laissent de nombreuses questions de recherche nourries par un ensemble grandissant d'applications, en particulier:

- **Proximité naturelle vs. distance mathématique :** la distance utilisée par CORESE dans sa recherche approchée est une semi-distance *i.e.* elle ne vérifie pas l'inégalité triangulaire. Quelle est la valeur des conditions nécessaires de la définition mathématique des distances ? Devons-nous à tout prix essayer de les respecter ou est-ce simplement une limite de la métaphore des distances ? Quelles seraient sinon les caractéristiques définitionnelles d'une distance conceptuelle? Rappelons sur ce point, qu'il y a plus d'un siècle, William James en 1890 [123] décrit un contrexemple à l'inégalité triangulaire : une *flamme* est similaire à la *lune* par sa lumière, la *lune* est similaire à un *ballon* par sa rondeur, mais une *flamme* et un *ballon* n'ont rien de similaires. De même Tversky [112] remet en cause la symétrie de la similarité en donnant comme contrexemple le fait que la *Corée du Nord* est très similaire à la *Chine* alors que la *Chine* n'est pas très similaire à la *Corée du nord*. Tversky propose à l'époque une formule de similarité qui ne tient plus compte des axiomes définitionnels d'une distance comme vu en section 4.1.1.
- **Distance conceptuelle vs. espaces métriques :** les structures de la hiérarchie de types sont le terrain favori pour la définition des distances conceptuelles. Faut-il considérer des représentations plus riches [124] incluant, par exemple, des liens frère-frère, qui permettraient de mieux simuler de telles distances ? Comment mieux recenser et définir ces espaces métriques *i.e.* les espaces pertinents et les métriques adaptées ? Comment étudier les différentes familles de distances qui semblent cohabiter dans nos inférences au quotidien ? Comment les capturer, les apprendre, les maintenir, pour les utiliser dans des inférences de recherche d'information?



- **Distance a priori vs. distance contextuelle :** les distances employées par les humains semblent varier d'une tâche à l'autre, autrement dit le contexte semble paramétrer les critères utilisés pour l'évaluation de la similarité de deux notions. Comment peut-on capter ces critères et paramétrer les distances simulées ? Peut-

on apprendre ou adapter automatiquement une distance en analysant le comportement et le retour des utilisateurs ? Peut-on combiner dynamiquement ces distances ?

Nous sommes convaincus que la tâche devant être assistée par un système conditionne la ou les distances employées et leur combinaison éventuelle. Notre démarche actuelle est donc d'essayer les différents espaces métriques disponibles. Dans le cadre de plusieurs projets en cours, nous explorons actuellement de nouvelles distances ou extensions de distances.

S'il est clair que depuis près de trente ans, ces similarités ne cessent de resurgir dans l'exploitation de modèles et d'espaces de représentation, intuitivement, il nous semble qu'il reste encore beaucoup à faire pour identifier, étudier, caractériser et simuler ces similarités, dans la continuité des travaux de Blanchard *et al* [125]. Enfin, pour reprendre une remarque d'Amedeo Napoli, examinateur de cette habilitation, il paraît aussi intéressant de regarder si la notion d'espace topologique ne serait pas plus adéquate que celle d'espace métrique dans certains de nos scénarios.

Chapitre 4 : Récapitulatif de mes contributions personnelles.

Les travaux sur les distances sont l'un des sujets pour lesquels je suis moteur dans Edelweiss.

Dans cette section, j'ai résumé l'état de l'art que je maintiens sur ce domaine et les différences que j'ai identifiées entre les distances connues.

J'ai montré comment une distance peut être utilisée dans des protocoles distribués pour simuler des enchères sur la sémantique d'annotations à archiver [21].

J'ai caractérisé la dissimilarité du logiciel Corese comme une semi-distance sur une hiérarchie avec multi-héritage [74] et comme une distance dans le cas d'un arbre d'héritage simple [27].

J'ai construit et caractérisé des ultra-métriques à partir de ces distances et montré leur intérêt pour la génération de dendrogrammes permettant de contrôler les niveaux de détail d'une interface à base d'ontologies [27].

J'ai conduit des expériences statistiques avec des humains pour montrer des caractéristiques contrintuitives des distances réelles et les limites de nos espaces métriques [117].

J'ai suggéré une nouvelle dissimilarité basée sur les statistiques des instances pour intégrer l'usage effectif de nos primitives aux inférences de rapprochement [117] . Cette dissimilarité a trouvé une première utilisation dans nos interfaces. J'ai encadré le travail d'Ibrahima Diop sur ce sujet de définition de métriques sémantiques pour la recherche dans une base de cas de conception passés dans le cadre du projet européen SevenPro.

J'ai spécifié une nouvelle distance parcourant d'autres liens que ceux de la hiérarchie des types et qui a été validée dans une expérience de pilotage d'extraction d'annotations à partir de textes [39] dans le cadre du projet européen SeaLife.

5. Structures de Graphes pour la gestion de la distribution

Un web sémantique, qu'il soit public ou sur l'intranet d'une entreprise, repose généralement sur plusieurs serveurs web qui proposent chacun différentes ontologies et différentes bases d'annotations utilisant ces ontologies pour décrire des ressources.

Les scénarios d'usage amènent souvent un utilisateur à formuler des requêtes dont les réponses combinent des éléments d'annotation distribués entre plusieurs de ces serveurs. Ceci demande alors d'être capable :

(1) d'identifier les serveurs susceptibles d'avoir des éléments de réponse ;
(2) d'interroger des serveurs distants sur les éléments qu'ils connaissent sans surcharger le réseau ;
(3) de décomposer la requête et router les sous-requêtes vers les serveurs idoines ;
(4) de recomposer les résultats à partir des réponses partielles.

Nous avons, avec le web sémantique, les briques de base d'une architecture distribuée. Ce chapitre résume un certain nombre d'approches que nous avons proposées pour tenir compte de la distribution et gérer des ressources distribuées dans les webs sémantiques que nous concevons.

5.1 RDF et les graphes distribués

Les recommandations autour de RDF et les URIs forment un premier ensemble de briques de base. Le modèle RDF et sa syntaxe XML fournissent un format d'échange entre des acteurs distribués sur un réseau et les URI donnent des identifiants universels uniques permettant notamment de fusionner des métadonnées distribuées mais portant sur la même ressource.

La recommandation SPARQL du W3C a publié trois autres briques de base :

- un langage de requêtes permettant de décrire les graphes RDF recherchés sous forme de triplets (arcs du graphe) et de contraintes sur ces triplets (filtres) ;
- un format de réponse en XML donnant les valeurs des variables sélectionnées dans la requête pour chaque réponse trouvée ;
- un protocole permettant d'échanger les requêtes et leurs réponses avec un serveur distant, notamment en utilisant une architecture de services web.

Si la recommandation SPARQL permet effectivement d'interroger un serveur web sémantique distant, elle ne fournit pas de mécanisme permettant d'orchestrer la résolution collective d'une requête par plusieurs serveurs.

Parmi les travaux proches de ces problématiques, ARQtick[10] propose une clause supplémentaire de SPARQL permettant de distribuer des parties prédéterminées d'une requête à des serveurs prédéterminés ; chaque partie n'étant déléguée qu'à un seul serveur pré-choisi. L'extension "Federated SPARQL"[11] propose le complémentaire d'ARQtick en permettant d'associer à une requête des *bindings i.e.* des listes de candidats pré-calculés pour des variables de la requête. Ainsi des requêtes partiellement résolues peuvent transiter entre des serveurs. Ces deux initiatives se placent à un niveau plus élémentaire de la fédération de bases et fournissent des briques élémentaires pour des algorithmes de distribution de requêtes ou des cas de fédération très spécifiques.

DARQ[12] [126] permet de fédérer des bases RDF et utilise un routage au niveau des triplets en se basant sur le type de prédicat demandé. Nous allons nous intéresser au routage de sous-graphes complets et au problème d'indexation de ces sous-graphes pour caractériser des bases distribuées.

Nous ne considèrerons pas les approches pair-à-pair telles que [127] ou [128] car leurs échelles de déploiement et la nature éphémère de leurs réseaux ne leur permettent pas de reposer sur un échange préliminaire d'index comme nous l'envisageons et les orientent vers des problèmes de routage de requêtes entre pairs et des structures comme les tables de hachage distribuées. Nous ne considérons ici que des réseaux de *hubs* petits et stables comme on pourrait en trouver dans un système d'information d'entreprise.

---

[10] http://seaborne.blogspot.com/2007/07/basic-federated-sparql-query.html

[11] http://www.w3.org/2007/05/SPARQLfed/

[12] http://darq.sourceforge.net/

Enfin, l'optimisation des requêtes est un domaine connexe où les index sont aussi des structures centrales. Dans le cadre de RDF, nous citerons en particulier [129] qui proposent aussi un ensemble de structures d'index de littéraux et de graphes mais dédiées à l'accélération des accès à une base de triplets. On peut aussi trouver dans [130] trois familles d'indexation pour la subsomption:

- Codage binaire : chaque nœud d'une hiérarchie est associé à un vecteur de n bits avec n le nombre total de nœuds, la hiérarchie devenant une matrice $n*n$. A chaque nœud correspond un bit dans le vecteur qui est positionné à 1 pour ce nœud et pour tous ses descendants. Les vecteurs de bits permettent de vérifier la parenté de deux nœuds, de connaître leurs plus proches ancêtres ou descendants communs et utilisant des opérations sur les bits et donc en temps constant. La matrice creuse de la hiérarchie peut devenir énorme et son codage efficace est un problème en lui-même ; de plus, sa mise à jour est coûteuse.
- Chaines de préfixes : au label de chaque nœud est concaténé le label de son père et ce récursivement. Les comparaisons de subsomption deviennent alors des comparaisons de chaînes de caractères. Un nouveau fils peut toujours être ajouté à un père par la droite de l'arbre. Dans cette approche, les labels peuvent rapidement grossir en taille et la recherche devient alors coûteuse en temps et en mémoire.
- Division d'intervalles : le label d'un nœud est un intervalle qui est inclus dans l'intervalle de ses ancêtres. Le test de subsomption devient alors un calcul d'inclusion d'intervalles.

Si ces approches sont efficaces, elles se focalisent sur la représentation d'une hiérarchie basée sur une relation unique (ex : subClassOf) ne permettent pas toujours de coder l'héritage multiple (ex : les intervalles). De plus, elles sont souvent très consommatrices en mémoire et en temps de maintenance. Elles sont, par contre, de bonnes inspirations pour résoudre certains des problèmes que nous rencontrerons.

5.2 Architectures de distribution

Les progrès de la programmation ont notamment été obtenus grâce au développement d'abstractions de plus en plus puissantes permettant de modéliser et de développer des systèmes de plus en plus complexes. Nous avons pu vérifier dans deux projets (Comma / IST et myCampus / DARPA) que le paradigme des systèmes multi-agents (SMA), issu de l'intelligence artificielle distribuée, est un candidat possible pour disposer d'un nouveau niveau d'abstraction. Il peut être employé pour comprendre, modéliser, et développer une classe importante de systèmes distribués complexes.

Les agents sont des composants logiciels autonomes faiblement couplés. La famille des agents qui nous a intéressée dans ces deux projets est celle où les agents, en tant que composants logiciels, intègrent des méthodes de l'IA symbolique et échangent des messages au niveau sémantique pour collaborer dans des tâches collectives. Le niveau d'abstraction et le fait que la notion d'organisations artificielles soit au cœur du paradigme le rendent très proche de la réalité des communautés humaines dans lesquelles sont plongés nos systèmes.

Entre une mémoire organisationnelle fragmentée et hétérogène et une population d'utilisateurs dispersés et de profils variés, il est intéressant que l'interface soit elle-même distribuée et hétérogène, ce qui est le cas des SMA : les agents peuvent être de types différents, remplissant différents rôles et déployés sur les réseaux à travers lesquels ils communiquent pour coopérer. Nous verrons que ce raisonnement est aussi valable pour la dernière génération de services web.

Par leur collaboration, les agents logiciels peuvent ainsi être utilisés pour réaliser une intégration des connaissances de l'organisation et en permettre leur capitalisation :

- D'un côté, la collaboration intelligente des agents du SMA permet de réaliser une capitalisation globale de la connaissance de l'organisation.
- D'un autre côté, l'autonomie et l'individualité de chaque agent lui permettent de s'adapter localement à la spécificité des ressources et des utilisateurs individuels, tout en en faisant bénéficier l'ensemble de la société.

Parallèlement, le web n'est plus uniquement un lieu où trouver de l'information, il est aussi devenu un moyen d'action à travers la mise en ligne d'applications de plus en plus nombreuses. Dans une communauté, le paradigme des services web peut être utilisé pour faire de l'intégration d'applications. Comme nous le verrons plus loin, nous avons par exemple commencé à nous intéresser à l'intégration de ressources dynamiques dans un intraweb sémantique en utilisant les services web sémantiques : l'accès aux applications est encapsulé dans des services web annotés ; un moteur de recherche sémantique est utilisé comme registre UDDI sémantique pour générer un portail des services ; les pages du portail permettent notamment la découverte automatique, l'invocation dynamique de services et leur composition. Ce travail se poursuit d'ailleurs dans le cadre du projet RNTL eWok.

Nous définissons la mémoire organisationnelle comme la représentation persistante, explicite, désincarnée des connaissances et des informations dans une organisation, afin de faciliter leur accès, leur partage et leur réutilisation par les membres adéquats de l'organisation, dans le cadre de leurs tâches individuelles et collectives [1]. La mémoire d'entreprise correspond au cas particulier où l'organisation considérée est une entreprise. Les problèmes de recherche qui se posent dans ce chapitre sont :

- Quelles architectures logicielles choisir afin de pouvoir générer des configurations de déploiement suffisamment variées pour s'adapter aux différentes topographies organisationnelles ?
- Quelles inférences et capacités techniques pour les agents en charge d'une mémoire ? Comment exploiter les systèmes symboliques définis pour la modélisation des connaissances à l'intérieur d'un agent ou un service assigné à une tâche ?
- Comment permettre la communication et la coopération entre agents et/ou services afin de mettre en place des sociétés d'agents et des compositions de services dédiées à la gestion de la mémoire ?
- Comment gérer la distribution des connaissances par la distribution d'intelligences artificielles ? Comment passer des inférences telles qu'elles peuvent être effectuées dans des systèmes à base de connaissances monolithiques, à des inférences distribuées basées sur des sociétés artificielles ?
- Quelles inférences peuvent assister l'intégration d'applications d'entreprise si celles-ci deviennent des services web sémantiques *i.e.* des services dont l'interface est décrite par des ontologies ? Quels sont les scénarios et les mécanismes de composition ? Quelles interfaces pour l'invocation de ces services et de leurs compositions?
- Comment interagissent les connaissances formalisées du web sémantique et les services web ? Quelles connaissances sur les services et quels services sur les connaissances ? Quelles formes de composition doit-on prévoir entre accès à la connaissance (ex. SPARQL) et service web?
- Comment s'intègrent les nouvelles architectures de grilles, de services et d'agents entre elles?

5.3 Index d'arcs

Nous rappelons ici rapidement l'architecture obtenue en fin du projet CoMMA [21] car nous verrons qu'elle a servi de base à un certain nombre d'autres travaux et extensions présentés dans cette partie.

Nos premiers travaux sur la distribution de requêtes entre plusieurs bases de triplets RDF distantes remontent au projet CoMMA et ma thèse [21]. La distribution, au sens large, d'une mémoire organisationnelle est une caractéristique fondamentale dans le développement et le déploiement de systèmes de gestion des connaissances. Dans ce premier projet, le web sémantique nous offrait de nouvelles technologies pour matérialiser cette mémoire et les ontologies jouaient un rôle important. Parallèlement, les systèmes multi-agents proposaient un nouveau paradigme pour concevoir une plate-forme logicielle permettant de s'adapter à la topographie interne des organisations, et de capitaliser, en les intégrant dans une vue globale, les connaissances de l'organisation. Nous rappelons ici cette architecture de départ et un premier retour d'expérience.

5.3.1 Une mémoire distribuée

L'enjeu de la construction d'un système de gestion de mémoire organisationnelle est de réussir l'intégration cohérente de la connaissance dispersée dans l'organisation afin d'en promouvoir la croissance, la communication et la préservation [131]. *La mémoire organisationnelle est un système distribué au sens large du terme*. Elle n'est pas seulement distribuée entre des systèmes informatiques, sa distribution inclut d'autres artefacts (livres, panneaux d'affichage, etc.), mais aussi, et surtout, les membres de l'organisation. Les solutions de gestion de la mémoire, doivent prendre en compte cette « super-distribution ». Les modèles de la distribution doivent aller au-delà de la dimension technique et rendre compte de l'organisation, de son infrastructure, de son environnement et de ses caractéristiques psychosociologiques.

Une mémoire distribuée et hétérogène. La mémoire organisationnelle a cela en commun avec le web qu'elle est un paysage d'informations hétérogènes et distribuées. De ce fait, tous les deux partagent les problèmes de bruit et d'imprécision lors de la recherche d'information. Pour reprendre la formule de John Naisbitt, « les utilisateurs du web se noient dans l'information alors qu'ils ont soif de connaissance ». Parmi les initiatives visant à résoudre ces problèmes, le web sémantique est une approche prometteuse [4] ; l'idée est de rendre explicite la sémantique des documents au travers de méta-données ou d'annotations. Les intranets reposant sur la technologie internet, peuvent bénéficier des progrès du web sémantique, et la mémoire d'entreprise peut alors être étudiée comme un « web sémantique d'entreprise ».

Une population d'utilisateurs distribuée et hétérogène. La population des utilisateurs d'une mémoire organisationnelle est hétérogène car il existe plusieurs profils de membres concernés par la gestion et l'exploitation de la mémoire. Chacun a ses particularités, ses centres d'intérêts et son rôle dans l'organisation. Cette population est

aussi distribuée car les utilisateurs sont dispersés dans l'infrastructure de l'organisation. Ils ne sont pas forcément conscients de l'existence des autres, de leur fonction ou de leurs capacités. Ils ignorent parfois l'existence, l'emplacement et la disponibilité d'artefacts et de personnes avec lesquels ils partagent des centres d'intérêts. Pour s'adapter aux profils des utilisateurs et aux contextes d'utilisation de la mémoire, l'apprentissage symbolique propose des techniques permettant d'apprendre les particularités individuelles ou de faciliter l'apparition de communautés d'intérêt et la diffusion proactive d'information par une exploitation collective des profils.

Une gestion distribuée et hétérogène. La mémoire organisationnelle est distribuée et hétérogène. La population de ses utilisateurs est distribuée et hétérogène. Il semble donc intéressant que l'interface entre ces deux mondes soit elle-même de nature distribuée et hétérogène pour pouvoir d'une part se calquer sur le paysage d'information, et d'autre part s'adapter aux utilisateurs. Comme le note [132], les progrès de la programmation se sont faits à travers le développement d'abstractions de plus en plus puissantes permettant de modéliser et de développer des systèmes de plus en plus complexes. L'abstraction procédurale, les types de données abstraits, et plus récemment les objets et les composants sont tous des exemples de telles abstractions. Le paradigme des SMA [132] représente une nouvelle étape dans l'abstraction et il peut être employé pour comprendre, modéliser, et développer des systèmes distribués complexes. Ce paradigme apparaît particulièrement adapté au déploiement d'une architecture logicielle au-dessus de ce paysage d'information distribué.

La Figure 66 présente une vue globale du système CoMMA. D'un côté, les agents de CoMMA sont les acteurs d'une société qui, par leur collaboration et leur coopération basées sur l'échange de messages au niveau sémantique, résolvent des problèmes complexes nécessitant une vue globale et intégrée de la mémoire d'entreprise. *La collaboration intelligente des agents du SMA permet de réaliser une capitalisation globale de la connaissance de l'entreprise*.

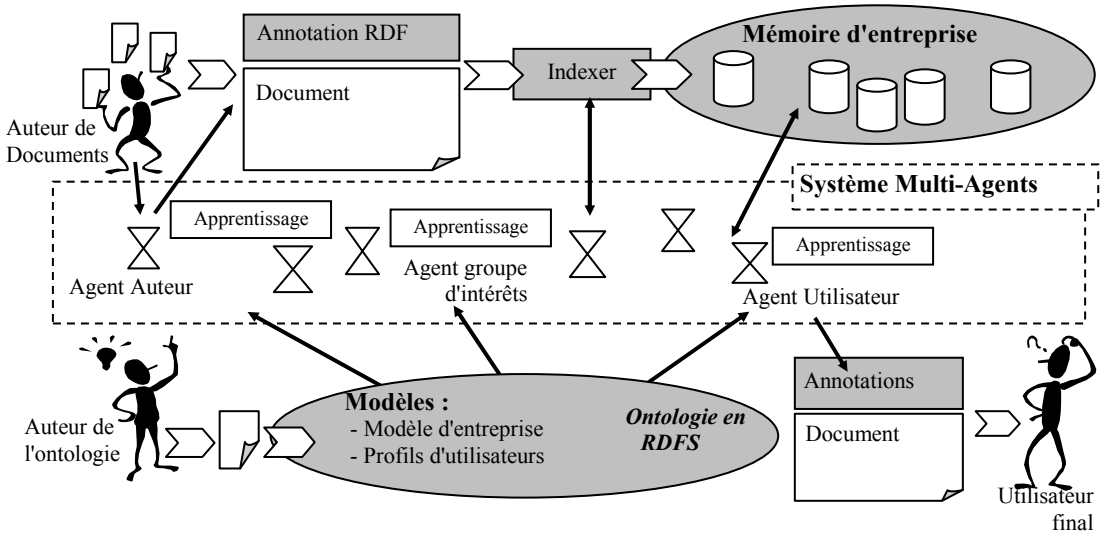


Figure 66. Schéma général du système CoMMA [21]

D'un autre côté, ces mêmes agents sont des entités localisées et affectées à un utilisateur pour l'assister dans son interaction avec la mémoire, ou dédiées à une ressource pour en permettre l'exploitation au bénéfice de l'ensemble de la société des agents. Pour ce faire, les agents possèdent des capacités d'inférence voire d'apprentissage utilisant les annotations, les modèles, l'ontologie et les profils utilisateurs ainsi que des interfaces graphiques indispensables pour gérer les interactions entre ces objets conceptuels et les préoccupations des utilisateurs finaux. Les agents intègrent ces capacités à leur comportement afin de remplir leur rôle social. *L'autonomie et l'individualité de chaque agent lui permet de s'adapter localement à la spécificité des ressources et des utilisateurs individuels, tout en en faisant bénéficier l'ensemble de la société*.

Les SMA proposent une approche sous la forme d'un réseau de composants logiciels faiblement couplés, les agents, qui fonctionnent ensemble comme une société visant à résoudre des problèmes qui seraient hors d'atteinte pour n'importe quel agent pris individuellement. Un SMA est hétérogène s'il inclut des agents d'au moins deux types différents. Un système d'information multi-agents (SIMA) est un SMA visant à fournir tout ou partie des fonctionnalités permettant de contrôler et d'exploiter un ensemble d'informations. L'architecture logicielle de CoMMA est un SIMA hétérogène et nous étudions la conception d'une telle architecture pour assister la distribution de l'information. La dualité de la définition du mot 'distribution' indique deux problèmes importants à traiter:

- Distribution signifie 'dispersion' : la propriété spatiale de ce qui est épars dans un espace. Le problème associé est de pouvoir gérer les données, l'information ou la connaissance naturellement distribuées dans l'organisation.
- Distribution signifie également l'action de 'propagation' et de 'répartition'. Le problème associé est de s'assurer que l'information appropriée arrive à l'agent intéressé.

Ces deux aspects de la distribution trouvent leur réponse technique dans l'utilisation de RDF/XML pour la matérialisation de la mémoire et l'utilisation d'une architecture multi-agents pour sa gestion et son exploitation.

Les ontologies fournissent, d'une part, des ressources conceptuelles et notionnelles pour formuler et expliciter un savoir ; d'autre part elles constituent un cadre partagé par les différents acteurs ; enfin, elles représentent le sens de différents contenus échangés dans les systèmes d'informations [124]. Ainsi, dans CoMMA, nous pouvons distinguer plusieurs raisons pour lesquelles nous avons besoin d'une ontologie:

- une venant de la nature même des SMA où les agents ont besoin d'un vocabulaire conceptuel consensuel pour exprimer les messages qu'ils échangent ;
- une autre venant de l'aspect système d'information utilisant des requêtes et des annotations basées sur un vocabulaire conceptuel consensuel ;
- enfin l'ontologie est une composante de la mémoire d'entreprise qui capture une connaissance potentiellement intéressante en elle-même pour l'utilisateur membre de cette communauté d'entreprise.

Nous n'avons développé qu'une seule ontologie (O'CoMMA) mais les différents agents et les différents utilisateurs n'exploitent pas les mêmes aspects de cette ontologie.

5.3.2 Une architecture logicielle distribuée : le paradigme agent

Lorsqu'il envisage une solution logicielle dans une perspective multi-agents, le concepteur doit gérer la relation entre :

- le niveau macroscopique du SMA (la société des agents), où se posent les problèmes d'ingénierie des interactions et d'organisation de la société du SMA afin d'obtenir, du point de vue global du système, les fonctionnalités correspondant aux exigences de l'utilisateur ;
- le niveau microscopique du SMA (les agents en eux-mêmes) où se posent les problèmes d'identification des rôles nécessaires, d'ingénierie des comportements tenant compte des interactions qui se produiront et fournissant les différentes compétences recherchées.

Notre approche partage avec AALAADIN [133] et GAIA [132] le souci de chercher à concevoir un SMA comme une organisation humaine, en identifiant les rôles nécessaires au fonctionnement des sociétés d'agents, les relations qui existent entre ces rôles et les interactions systématiques auxquelles ils participent suivant des protocoles institutionnalisés.

Les fonctionnalités souhaitées pour le système ne se transfèrent pas directement en fonctionnalités d'agents, mais influencent la conception et sont finalement distribuées dans les interactions sociales des agents et l'ensemble des capacités, des rôles et des comportements qui leur sont associés.

En considérant les fonctionnalités du système CoMMA, nous avons identifié quatre sous-sociétés : (1) une sous-société dédiée au modèle d'entreprise et à l'ontologie ; (2) une sous-société dédiée aux documents ; (3) une sous-société dédiée aux utilisateurs ; (4) une sous-société dédiée à l'interconnexion. Nous ne rappellerons ici que la sous-société dédiée aux documents qui sera exploitée et étendue dans les travaux présentés dans ce chapitre.

Les agents de cette sous-société sont en charge de l'exploitation des documents et des annotations distribués dans des bases de l'entreprise. Ces agents recherchent et extraient les références satisfaisant la requête de l'utilisateur avec l'aide des agents dédiés à l'ontologie et aux modèles. Parmi les différents types d'organisations d'une sous-société nous avons opté pour une société hiérarchique (Figure 67) appliquée à la gestion de bases d'annotations RDF.

La société hiérarchique distingue deux types de rôles :

(1) Les représentants : agents médiateurs entre leur société et le reste du SMA. Ils sont responsables du traitement des requêtes externes, de leur décomposition, du dialogue avec les exploitants et de l'intégration des résultats intermédiaires pour fournir une réponse appropriée au commanditaire externe ;

(2) Les exploitants : assignés à une ressource locale, ces agents contribuent autant que possible à la résolution des requêtes qu'ils reçoivent, avec cette ressource locale.

Dans cette société, l'information est répartie entre les agents assignés aux ressources. Ceci permet de conserver la répartition initiale des ressources et de distribuer et équilibrer la charge de travail : le travail de recherche locale est réparti entre les exploitants et le travail de fusion est effectué par les représentants. La spécialisation des agents et la distribution des rôles permettent la distribution de la charge de travail mais en contrepartie requièrent un volume plus important d'échanges sur le réseau.

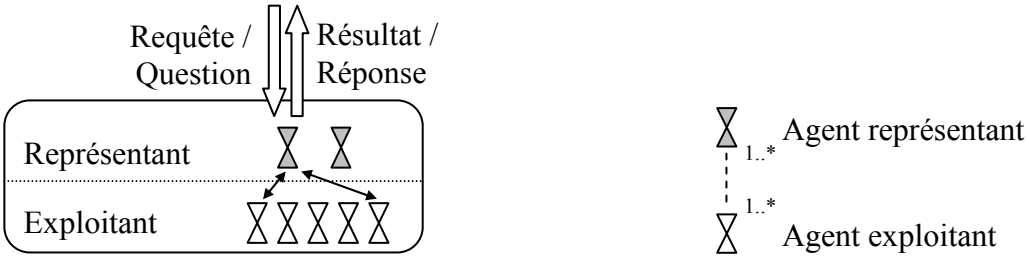


Figure 67. Société hiérarchique

Le rôle de *Médiateur* d'Annotations est de proposer ses services aux agents des autres sociétés pour résoudre leurs requêtes et archiver leurs annotations ; le médiateur engage les services des *Archivistes d'Annotations* pour effectivement archiver une annotation et résoudre une requête :

(1) Il décompose et distribue les requêtes aux *Archivistes* (2)
(2) Il compile les réponses partielles pour construire le résultat final.

Le rôle de l'*Archiviste* d'Annotations est affecté à l'exploitation d'une archive locale. Quand cet agent reçoit une requête, il essaie de la résoudre même partiellement pour permettre au *Médiateur* de retrouver des réponses distribuées.

Le premier prototype de CoMMA utilisait une implantation simplifiée de la distribution des annotations et des requêtes permettant de tester l'architecture et l'intégration. Des algorithmes plus poussés d'archivage et de résolution distribués ont ensuite été développés. Ils utilisent des statistiques sur les bases d'annotations exploitant l'ontologie pour l'allocation d'une nouvelle annotation et la répartition des tâches entre agents lors de la résolution d'une requête. [21]

5.3.3 Protocoles d'archivages et de requêtes : interactions sociales artificielles

L'enjeu de la société des agents archivistes et médiateur est de trouver des mécanismes pour décider où stocker les annotations nouvellement soumises et comment distribuer une requête pour ne pas laisser échapper des réponses simplement parce que l'information nécessaire est dispersée entre plusieurs bases d'annotations. Ces deux facettes de la distribution sont liées puisque le processus de résolution distribuée d'une requête dépend étroitement du choix fait pour la distribution des annotations. Dans cette société, le médiateur d'annotations (AM) est responsable de la gestion de la dispersion

des annotations ; ces annotations étant réparties entre des archivistes d'annotations (AAs).

Pour allouer une annotation nouvellement soumise, l'AM émet un appel à proposition à l'attention des AAs. Un protocole d'interaction correspondant à un *contract-net* [134] imbriqué est utilisé pour déterminer lequel des AAs gagnera l'annotation nouvellement soumise. En réponse, chaque AA mesure la proximité sémantique entre l'annotation et les types des concepts et de relations actuellement dans ses archives. L'AA le plus proche gagne le contrat. Ce protocole permet de spécialiser les bases d'annotations sémantiques de la mémoire et d'entretenir cette spécialisation. Pour ce protocole, nous avons donc défini une pseudo-distance en utilisant la hiérarchie de l'ontologie et une distance lexicographique pour les valeurs littérales. La distance utilisant la hiérarchie est basée sur le chemin entre deux notions, passant par le plus proche commun supertype ; les supertypes communs représentent ce que deux notions ont en commun (voir section 4.2). Nous l'employons pour comparer les offres des différents AAs. L'ontologie est ainsi utilisée comme un espace commun permettant de définir des (pseudo-)distances communes dont les résultats sont comparables. Le consensus ontologique fournit donc une base pour d'autres consensus (ex : un consensus sur le calcul d'une métrique partagée). L'objet 'ontologie' est donc la pierre de touche de tous les mécanismes distribués d'une gestion intelligente des connaissances dispersées.

La résolution d'une requête peut impliquer plusieurs bases d'annotations réparties entre plusieurs AAs ; le résultat est une fusion de résultats partiels. Pour déterminer si et quand un AA doit participer à la résolution d'une requête, les AAs calculent le recouvrement entre la liste des notions actuellement utilisées dans leur base et la liste de celles utilisées dans la requête (en prenant en compte les liens de subsomption dans l'ontologie). En utilisant ces descriptions de recouvrement, l'AM peut identifier à chaque étape de son algorithme de décomposition et pour chaque requête intermédiaire qu'il produit, les AAs à consulter. L'ontologie partagée fournit ici les primitives permettant de décrire des connaissances allouées à chaque agent et permet ainsi de statuer sur la pertinence de la participation d'un agent à une tâche donnée.

Une fois que les rôles d'AA et d'AM ont été spécifiés avec leurs interactions, des modules de CORESE ont été intégrés dans les types d'agent implantant ces rôles afin de leur fournir les compétences nécessaires.

J'ai détaillé l'exemple de cette société non seulement parce qu'il est le point de départ des sections suivantes, mais aussi pour montrer comment les techniques de l'ingénierie des connaissances pouvaient fournir les outils nécessaires pour que d'autres domaines puissent apporter leur contribution à la résolution de problèmes complexes. Ainsi, l'approche reposant sur la mise en place d'une ontologie partagée permet à un SMA de participer à la gestion de connaissances distribuées.

L'interaction entre agents est plus que le simple envoi d'un message isolé. Le modèle d'une conversation doit être spécifié par des protocoles que les agents doivent respecter pour que le SMA fonctionne effectivement. Les protocoles sont des codes de conduite

sociale pour l'interaction ; ils décrivent des procédures standards régulant l'échange d'informations entre agents en institutionnalisant des modèles de communications pouvant survenir entre des rôles identifiés. La spécification d'un protocole part d'un graphe d'accointances au niveau des rôles, qui représente par un graphe orienté les voies de communication existant entre les agents jouant ces rôles. A partir de ce graphe, on spécifie la séquence possible des messages échangés pendant l'interaction. La Figure 68 montre, par exemple, une sous-partie d'un diagramme d'interaction documentant les échanges ayant lieu lors de l'allocation d'une nouvelle annotation ; on pourrait décrire ce protocole comme un *contract-net* à deux niveaux.

La Figure 69 montre le diagramme d'interaction documentant les échanges ayant lieu lors de la résolution d'une requête distribuée. Dans CoMMA, le découpage des requêtes se faisait au niveau des triplets. Avec la section suivante, nous allons voir une première extension qui s'intéresse à des motifs de graphes.

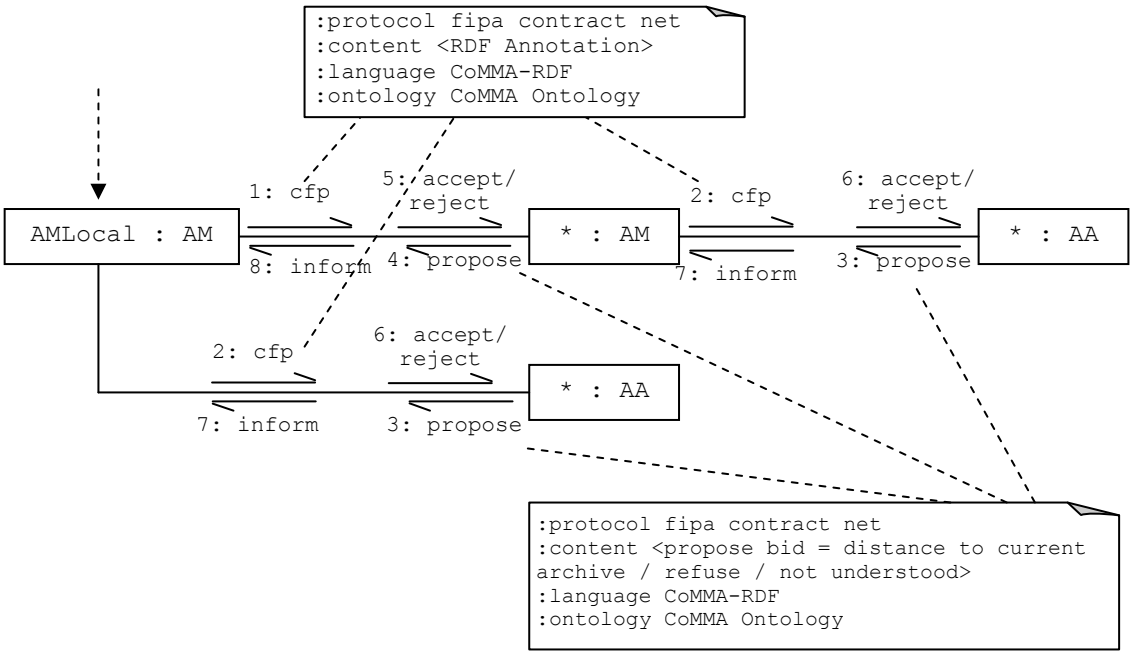


Figure 68. Sous-partie d'un diagramme d'interaction pour l'allocation d'une annotation

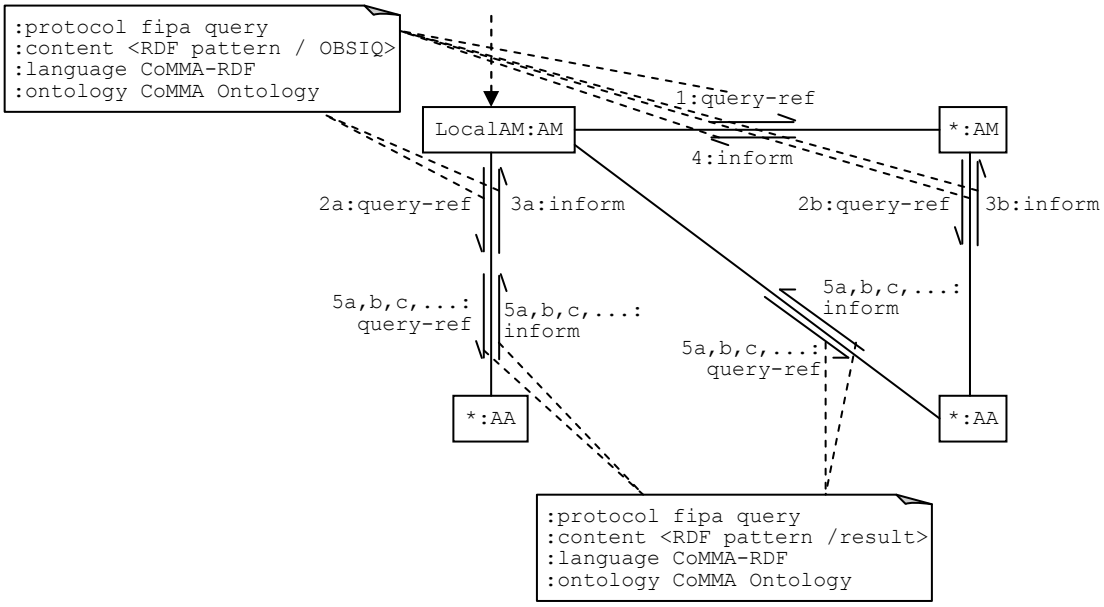


Figure 69. Sous-partie d'un diagramme d'interaction pour la résolution d'une requête

5.4 Index de motifs de graphes

Le travail présenté dans cette section est une extension du travail sur CoMMA [21] motivée par l'article de Stuckenschmidt *et al.* [86] dont la structure hiérarchique d'index et ses définitions a servi de base aux définitions et extensions faites ici. Notre principal ajout est l'introduction de la structure d'étoile comme deuxième motif de décomposition des bases. Le travail présenté dans Stuckenschmidt *et al.* [86] capitalisait déjà sur des travaux sur les structures d'index de modèles objets de Bertino *et al.* [135] [136] [137]. Ces travaux sont les plus pertinents en ce sens qu'ils bâtissent des propositions à la fois sur les résultats dans le domaine des bases de données distribuées et dans le domaine des bases de données ayant un schéma objet.

Notre scénario motivant ici, vient du projet européen *SevenPro* (voir section 2.4.1) visant à développer des outils et des technologies afin d'aider un ingénieur à concevoir de nouveaux objets en lui fournissant des rendus 3D de l'objet à concevoir ou d'objets similaires conçus précédemment. Il utilise ces vues pour accéder aux informations sur ces objets quelle que soit la provenance de ces informations. Le scénario inclut notamment l'extraction d'annotations à partir de documents textuels et de fichiers CADs (conception et dessin industriel assistés par ordinateur). Les sources d'information et d'annotations étant potentiellement distribuées, c'est dans ce cadre que nous nous intéresserons à la résolution de requêtes dont les réponses combinent des éléments d'annotation distribués entre plusieurs serveurs.

Dans cette section, nous allons donc nous intéresser à un scénario particulier où les standards du web sémantique sont utilisés pour implanter un système d'information dans une organisation et où chaque serveur possède le même jeu d'ontologies (RDFS, OWL) que les autres mais une base d'annotations (RDF) différente qu'il met à disposition sous la forme d'un service web (SPARQL). Autrement dit, dans ce système à base de connaissances, le vocabulaire conceptuel est commun à tous les serveurs mais les bases de faits sont spécifiques à chaque serveur.

Dans un premier temps, nous présentons l'architecture du système qui repose sur des serveurs pairs appelés *hubs*. Nous décrivons ensuite le modèle d'index proposé pour caractériser les contributions possibles d'un *hub* ainsi que la méthode de résolution distribuée d'une requête.

5.4.1 Hubs : des services web pairs

Notons que si ici le paradigme de distribution utilisé est celui des services web et non des agents, le protocole d'interaction pour la résolution d'une requête est équivalent.

Notre système repose sur des *hubs* ou serveurs pairs-à-pairs dont l'architecture est la même partout où ils sont déployés. Un *hub* contient systématiquement:

- une interface utilisateur : un serveur web proposant des applications accessibles aux utilisateurs par leur navigateur web ;
- une interface programmatique : des services web proposant un accès distant aux applications du *hub* pour d'autres applications et, notamment, d'autres *hubs*.

Dans cette architecture, illustrée dans la Figure 70, un utilisateur peut se connecter à n'importe quel *hub* pour utiliser une application web et en particulier pour soumettre une requête. Pour cet utilisateur et sa requête, ce *hub* est alors chargé d'identifier les autres *hubs* susceptibles de l'aider à répondre à la requête et il doit ensuite orchestrer la résolution distribuée de la requête avec les *hubs* identifiés. Pour ce scénario, nous avons fixé deux hypothèses fortes :

- chaque *hub* a les mêmes ontologies que les autres *i.e.* les mêmes schémas RDFS et OWL sont répliqués sur chaque *hub*.
- les *hubs* sont en nombre restreint et leur connectivité est stable.

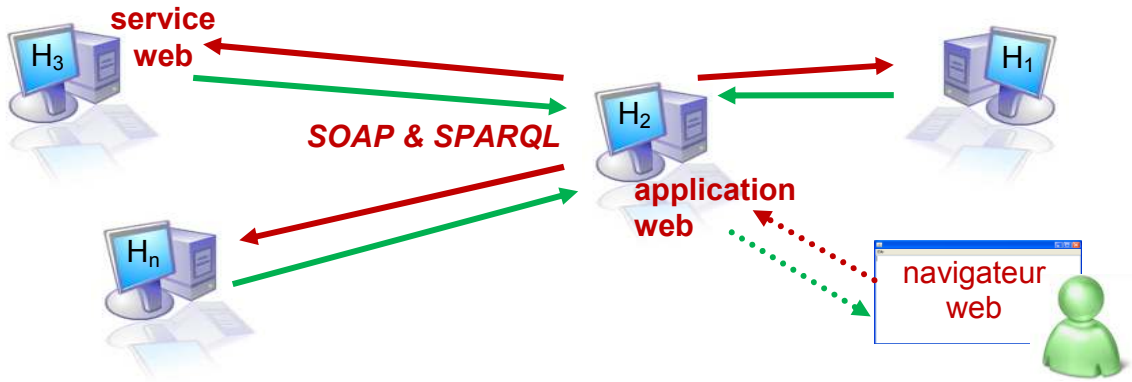


Figure 70. Architecture globale d'un système à base de *hubs*.

Un *hub* implante les interfaces SPARQL ainsi que les interfaces nécessaires à la gestion de la distribution et décrites dans les sections suivantes. Chaque *hub* propose aussi une application web permettant de déclarer les autres *hubs* et en particulier l'URL du point d'accès à leurs services web. Cette description est elle-même une annotation RDF du *hub* qui pourrait, dans une extension de ce travail, être générée à partir d'une description SAWSDL par exemple en utilisant le mécanisme GRDDL.

Avec cette architecture, chaque *hub* peut donc envoyer des requêtes à n'importe quel autre *hub*. Afin d'éviter l'envoi systématique de toutes les requêtes à tous les *hubs* (*broadcast*) et afin de trouver les réponses à une requête dont les triplets sont distribués sur différents serveurs nous allons nous intéresser dans les sections suivantes à

sélectionner les *hubs* pouvant contribuer à la résolution d'une requête (*multicast* sélectif). Le premier problème à résoudre est de pouvoir caractériser le contenu de chaque serveur afin de pouvoir prédire s'il peut ou non contribuer à la résolution d'une requête donnée.

La caractérisation du contenu d'un *hub* est un compromis entre la précision de la description (plus précise est la description, meilleure sera la sélection) et la concision de la description (la meilleure description du contenu, c'est le contenu lui même ; une description doit avoir une taille qui permet de l'échanger et de l'exploiter efficacement).

Nous appellerons index d'un *hub* la caractérisation du contenu de sa base d'annotation. Notre domaine d'application, la conception, va nous faire considérer la décomposition suivante ; une annotation d'un objet de conception contient souvent deux types de structures:

- des étoiles : une ressource qui est le sujet de plusieurs triplets, par exemple : un boulon qui a un diamètre, une longueur, une référence, un acier, etc.; 
- un chemin : une chaîne de propriétés qui relie plusieurs ressources par exemple : un chemin de partonomie/méronymie où une voiture inclut une portière, qui inclut une charnière, qui inclut un boulon, qui inclut une vis.

Tout graphe a évidemment ces structures, mais ce qui nous intéressera ici, c'est de considérer le typage des ressources et des liens dans ces deux structures pour abstraire dans l'index les types de ressources et les types de liens impliqués et récurrents.

Définition 1 : un chemin. Soient x et y deux ressources annotées dans une base. On dit qu'il existe un chemin $C(x, y) = <r_0, p_0, r_1, p_1, r_2, ..., p_{n-1}, r_n>$ de x vers y, si et seulement si il existe un nombre fini $n+1$ de ressources $r_0, r_1, ..., r_n$ avec $r_0=x$ et $r_n=y$ telles que pour tout $0 \leq i \leq n-1$ il existe dans les annotations un triplet (r_i, p_i, r_{i+1}). Notons que $r_n=y$ peut être un littéral.

Définition 2 : un chemin d'index. Un chemin d'index $CI = <t_0, p_0, t_1, p_1, t_2..., p_{n-1}, t_n>$ est un chemin mis dans l'index d'un *hub* pour indiquer qu'il existe dans la base d'annotations de ce *hub* au moins un chemin $C = <r_0, p_0, r_1, p_1, r_2, ..., p_{n-1}, r_n>$ tel que r_i est du type t_i avec $0 \leq i \leq n$.

Définition 3 : une étoile. Soit x une ressource annotée dans une base. On dit qu'il existe une étoile $E(x) = ((x, p_1, r_1), ..., (x, p_n, r_n))$ autour de x, si et seulement si il existe un nombre fini n de ressources ou littéraux $r_1, ..., r_n$ tels que pour tout $1 \leq i \leq n$ et $n \geq 2$ il existe dans les annotations un triplet (x, p_i, r_i).

Définition 4 : une étoile d'index. Une étoile d'index $E = ((t_x, p_0, t_0), (t_x, p_1, t_1)... (t_x, p_n, t_n))$ est une étoile mise dans l'index d'un *hub* pour indiquer qu'il existe dans la base d'annotations de ce *hub* au moins une étoile $E = ((x, p_1, r_1), ..., (x, p_n, r_n))$ telle que r_i est du type t_i avec $1 \leq i \leq n$ et x est de type t_x.

Le calcul des chemins d'index (resp. étoiles) se fait par requêtes récursives à partir de chacun des chemins d'index (resp. étoiles) de taille 1 trouvés dans la base. Etant donné que RDF est un modèle de graphes orientés étiquetés qui peut contenir des cycles, nous contrôlons l'exploration récursive avec une profondeur de recherche maximale.

La Figure 72 donne deux exemples de la génération d'un index respectivement pour les chemins (partie haute) et pour les étoiles (partie basse). Les exemples sont issus de bases d'annotations du projet *SevenPro*.

En regardant cet exemple, on peut s'interroger sur la concision de l'index puisqu'ici il est plus grand que l'annotation de départ. Cependant, le "facteur de compression" de l'index est directement lié à la spécialisation des bases : si une base contient des descriptions concernant un même domaine (par exemple : les stocks de pièces d'assemblage) ces descriptions vont rapidement utiliser les mêmes structures d'étoiles et de chemins ; dès lors le nombre d'instances utilisant les mêmes structures d'index peut augmenter sans que l'index lui ne change de taille.

L'index, calculé par un *hub* à partir de ses annotations, est lui même sauvé comme une annotation RDF (Figure 71) en utilisant les primitives d'une ontologie système pour représenter les notions d'index, d'étoile et de chemin.

Chaque chemin (resp. étoile) d'index est représenté par un chemin (resp. étoile) où des nœuds anonymes (*blank nodes*) sont typés à l'identique du chemin (resp. étoile) à représenter. Ainsi le mécanisme de la projection de graphe et le langage de requête SPARQL peuvent être utilisés directement pour interroger ces index.

```
<rdf:RDF (...) >
<CoreseServer rdf:about="#serverA12">
<contains>
 <Path>
  <numberInstance>18</numberInstance>
  <length>3</length>
  <content>
   <es:ShortLifter><es:partOf><es:IntermediateDiaphragmMill>
    <es:partOf><es:CementMill><es:capacityMill><rdfs:Resource/>
     </es:capacityMill></es:CementMill></es:partOf>
   </es:IntermediateDiaphragmMill></es:partOf></es:ShortLifter>
  </content>
 </Path>
</contains>
(...)
</CoreseServer>
</rdf:RDF>
```

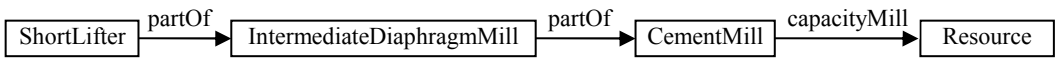


Figure 71. Exemple d'annotation représentant un chemin d'index de longueur 3.

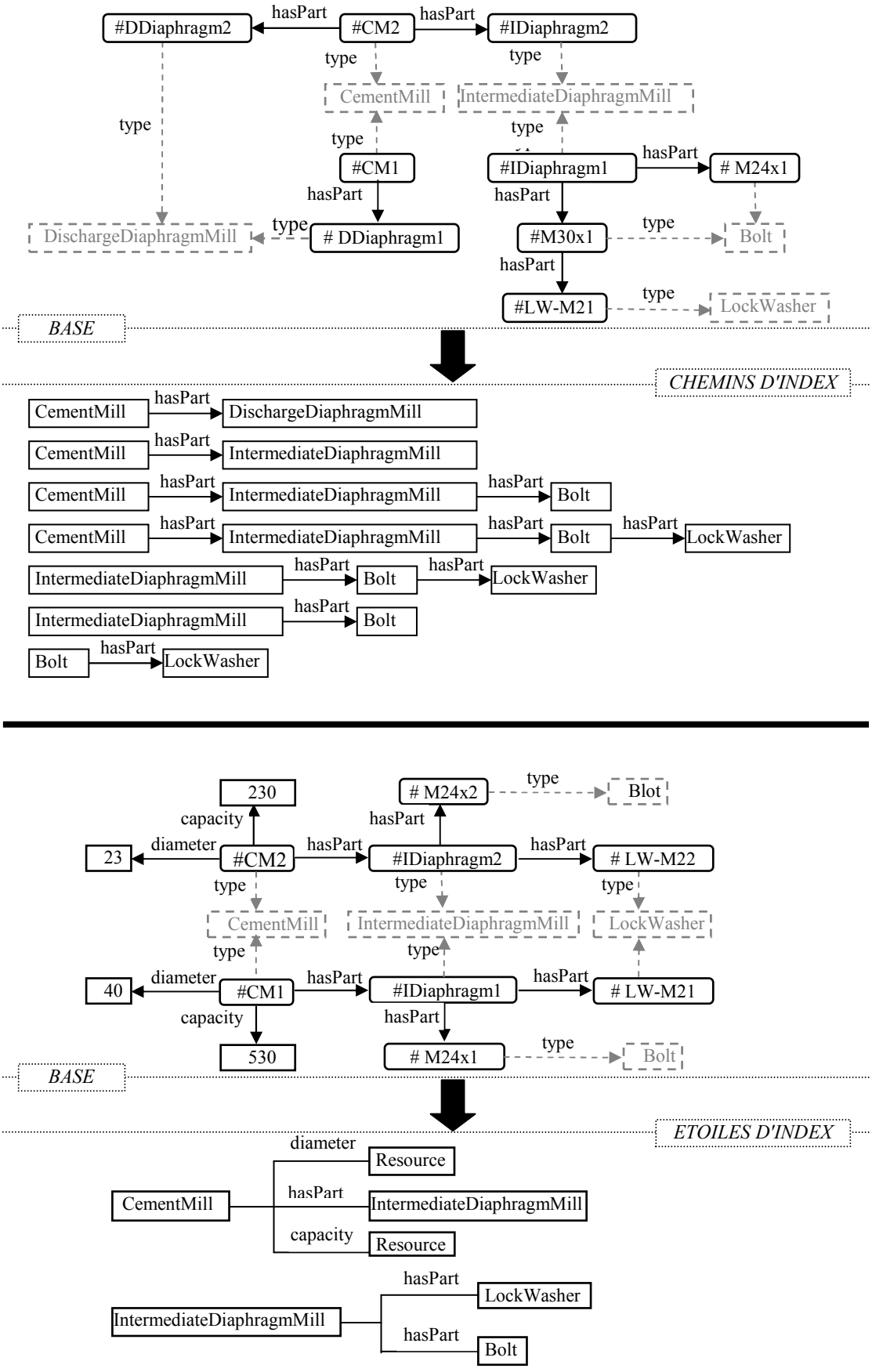


Figure 72. Exemples de chemins et d'étoiles d'index générés à partir d'une annotation.

Les index sont calculés par des requêtes au moteur d'inférence de Corese [72]. Ce moteur implante la sémantique de RDF, RDFS, ainsi que les *datatypes*, les propriétés transitives, symétriques et inverses de OWL Lite. Corese inclut aussi un moteur de règles en chainage avant, utilisé en particulier pour implanter la vérification des restrictions en OWL DL. Ces inférences, et les règles de production éventuellement ajoutées par l'application utilisant Corese, sont appliquées jusqu'à saturation, avant le calcul des index et la résolution des requêtes. Tous les traitements présentés ici se font donc sur une base saturée par ces inférences.

Les annotations représentant les index étant ajoutées à la base, le lecteur pourrait penser qu'elles risquent de polluer les résultats des requêtes sur cette base en ajoutant de nouveaux chemins et étoiles. En utilisant les mécanismes de provenance[13] (section 3.3.3 et [87] [88]) associant une source à chaque annotation, nous pouvons identifier les annotations ayant pour source le calcul des index et les ignorer dans les traitements autres que le pilotage de la résolution distribuée d'une requête.

Les sérialisations en RDF/XML des index sont échangées entre les *hubs* à chaque fois que l'un d'entre eux se déclare auprès des autres, par l'intermédiaire d'un service web dédié à ces échanges.

Une fois échangés, ces index permettent à un *hub* de connaître le type de contribution qu'il peut attendre d'un autre *hub* ; chaque chemin (resp. étoile) d'index inclus dans l'index d'un *hub* indique que ce *hub* possède au moins un chemin (resp. étoile) de triplets utilisant des ressources et des propriétés de types donnés par ce chemin (resp. étoile).

Lorsqu'une requête est soumise à un *hub* par un utilisateur ou une application cliente celui-ci devient responsable de l'orchestration de sa résolution distribuée. Les grandes étapes de la résolution sont les suivantes:

(1) décomposer la requête en chemins et en étoiles. Le principe est le même que pour les bases d'annotation : une construction récursive à partir de chacun des chemins (resp. étoiles) de taille 1 trouvés dans la requête.
(2) projeter sur les index des *hubs* connus chaque chemin et étoile de la requête pour savoir quel *hub* est susceptible de contribuer et pour quelle partie de la requête. Les index étant des annotations comme les autres, la projection d'une étoile (resp. chemin) sur l'index est une requête SPARQL classique dont le résultat est une liste de *hubs* archivant des annotations susceptibles de contenir les étoiles (resp. chemin) recherchées.
(3) émettre les sous-requêtes correspondant à chaque étoile (resp. chemin) vers les *hubs* identifiés comme des contributeurs pour cette étoile (resp. chemin) en utilisant les services web publiés par chacun de ces *hubs*.

[13] http://www.w3.org/Submission/rdfsource/

(4) recevoir les résultats partiels et les charger dans une base temporaire sur laquelle sera résolue la requête globale. Ce chargement dans une même base réalise l'union des résultats temporaires et leur jointure sur les URI communs.

La génération de sous-requêtes peut être optimisée, comme nous le discuterons en conclusion. La version actuelle prend déjà en compte le problème des *blank nodes* aux extrémités des sous-graphes requêtes : un *blank node* est un sommet anonyme dans le graphe RDF ; un sommet anonyme ne peut être joint à aucun autre sommet sauf cas particuliers en OWL non traités ici (ex : propriétés fonctionnelles).

Par conséquent, les sous-requêtes construites incluent des contraintes pour éviter de renvoyer des résultats partiels qui ne pourraient pas être joints aux autres lors de la résolution de la requête globale. En d'autres termes, tout sommet d'un chemin ou d'une étoile devant être joint à d'autres sommets ne peut être un sommet anonyme.

En SPARQL, un corps de requête (Figure 73) va donc être décomposé en plusieurs sous-requêtes SPARQL correspondant aux étoiles et chemins qu'elle contient (Fig 4 - les lignes 1-6 et 1;7-9 génèrent respectivement des chemins Figure 74 et des étoiles Figure 75). Toutes les opérations de résolutions de requêtes sur une base ou un index sont implantées en utilisant le moteur CORESE [72].

```
1  ?x rdf:type es:CementMill .
2  ?x es:includes ?y .
3  ?y rdf:type es:Diaphragm .
4  ?y es:includes ?z .
5  ?z rdf:type es:Lifter .
6  ?z es:fixedBy  ?u .
7  ?x es:diameter ?d .
8  ?x es:capacity ?c .
9  ?x es:hasPart  ?t .
```

Figure 73. Exemple de corps de requête.

```
?x rdf:type es:CementMill .
?x es:includes ?y .
?y rdf:type es:Diaphragm .
?y es:includes ?z .
?z rdf:type es:Lifter .
filter(!isBLANK(?x))
filter(!isBLANK(?z))
```

Figure 74. Exemple de sous requête générée pour un chemin de la Figure 73.

```
?x rdf:type es:CementMill .
?x es:diameter ?d .
?x es:capacity ?c .
?x es:hasPart  ?t .
filter(!isBLANK(?x))
```

Figure 75. Exemple de sous requête générée pour une étoile de la Figure 73.

La Figure 76 montre une décomposition en étoiles et en chemins correspondant à l'index d'une base d'annotations de documents de CAO. L'un des objectifs de *SevenPro* est d'obtenir par ailleurs ces annotations à partir de la fouille de la structure des documents de CAO. La Figure 77 montre le suivi de la résolution distribuée d'une requête sur les trois *hubs* de test de *SevenPro* et la sélection des serveurs pertinents à chaque étape de la décomposition. La requête interroge les bases sur les grands types de composants utilisés dans le moulin (ex : diaphragme) en fonction des caractéristiques du moulin (ex : diamètre, produit mouliné, capacité). Les réponses peuvent être intégrées à des visualisations 3D (Figure 78) pour l'aide à la conception, les tests, l'aide au montage, etc.

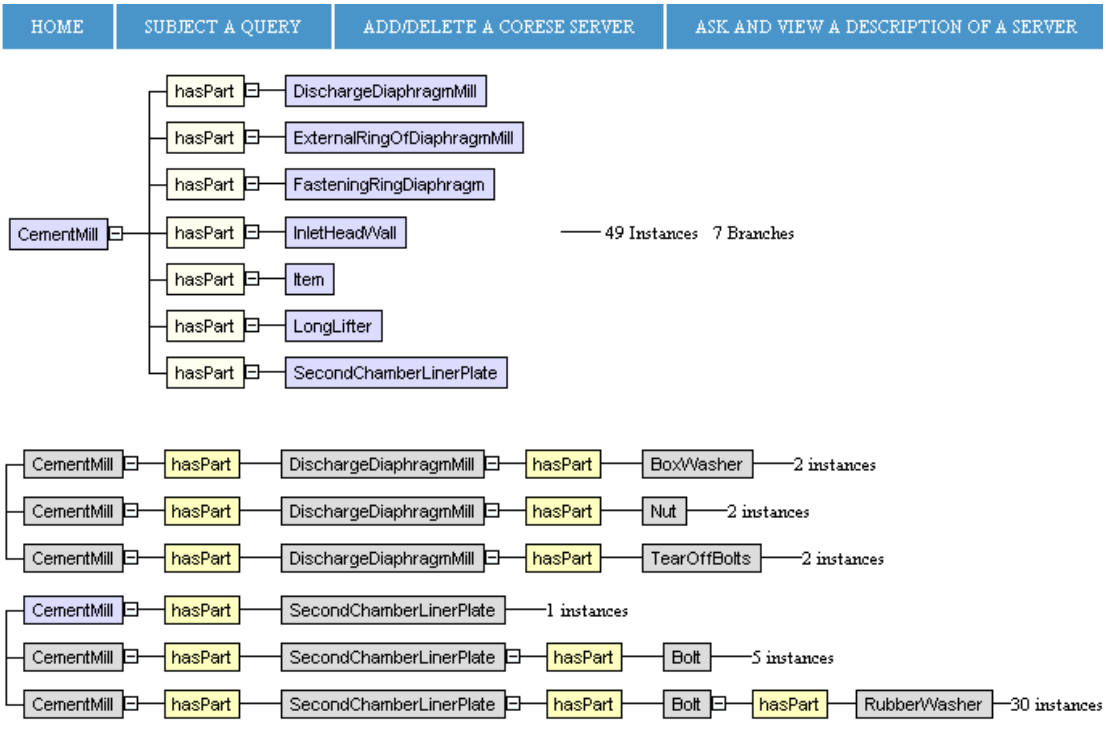


Figure 76. Copie d'écran de l'interface de visualisation des index dans *SevenPro*.

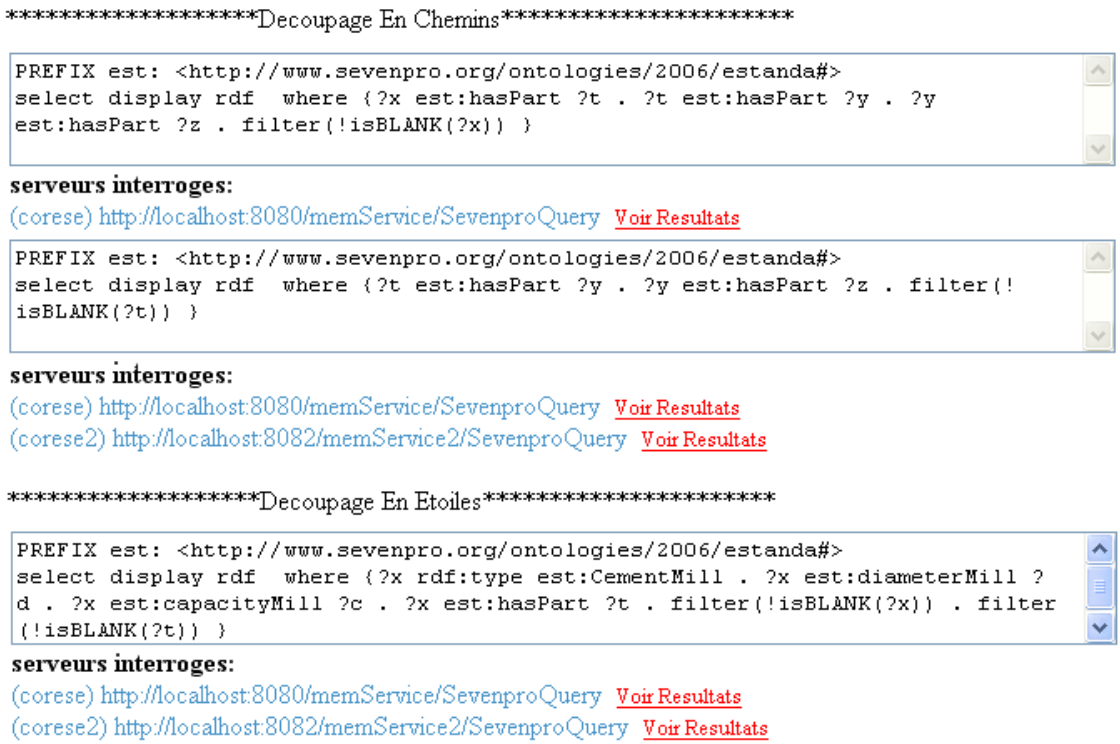


Figure 77. Copie d'écran de l'interface de suivi de la résolution distribuée.

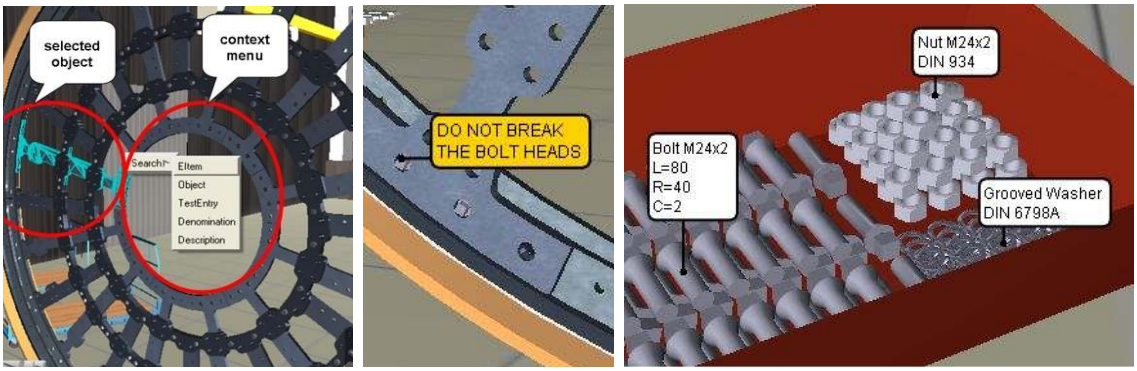


Figure 78. Visualisation 3D enrichie par des requêtes SPARQL dans *SevenPro*.

Le cas d'usage de *SevenPro* visualisé ici est celui d'une entreprise de fabrication de moulins industriels ayant plusieurs sources d'information sur chaque produit : bases de CAO, documents textuels techniques, catalogues numériques, base de normes publiques, etc.

Nous verrons en conclusion que ce travail a laissé un certain nombre de perspectives notamment en ce qui concerne l'optimisation de la génération des index et de ses motifs.

5.5 Sources externes

Aucune organisation n'est isolée, elle vit dans une culture, un pays, une société, un marché, etc. et beaucoup d'informations intéressantes (car se rapportant à l'environnement de l'organisation, aux activités dans son domaine, etc.) sont disponibles sur le web ouvert. Ces ressources extérieures, mais pertinentes pour l'activité de l'entreprise, peuvent être annotées pour être intégrées dans le web sémantique interne d'une entreprise par exemple pour une activité de veille.

Nous considérons ceci comme le problème dual de la vision habituelle d'un portail d'entreprise. Un portail d'entreprise offre des services de l'organisation sur le web externe ; c'est une vitrine, un portail externe d'accès à des services internes. Réciproquement les connaissances de l'organisation peuvent être employées pour filtrer et extraire des informations du web extérieur et fournir aux communautés d'intérêt internes un portail leur permettant d'y accéder de façon choisie et validée ; c'est un portail interne d'accès à des services externes.

En étendant l'architecture de CoMMA, une implémentation possible pour un tel portail consiste à introduire une société d'extracteurs [22]. Les agents de cette société automatisent l'extraction de certaines informations pertinentes et leur intégration dans la mémoire de l'organisation. Nous travaillons ici sur deux niveaux de distribution :

- la distribution de ressources pertinentes pour une entreprise mais disponibles sur le web ouvert
- l'intégration de ces ressources à la distribution du système d'information interne de l'entreprise.

La tâche d'annotation peut rapidement devenir fastidieuse si un vaste ensemble de documents pertinents est découvert et l'hypothèse de l'annotation manuelle devient alors peu réaliste. Or certains sites web ont une structure plutôt statique qui, même si elle est implicite, fournit des indices structurels (type de police, tableaux, séparateurs, etc.) qui peuvent être exploités pour automatiser des règles d'extraction. Ceci permet à l'utilisateur de produire automatiquement des annotations à partir du contenu d'un ensemble de pages structurées sur le même principe (ex, un catalogue, une bibliothèque, des normes, des brevets, etc.) Nous décrivons, dans la suite, une société d'agents fournissant ce type de service.

Le web ouvert est destiné aux humains et un grand nombre de ressources du web sont non structurées ou semi-structurées. L'extraction d'annotations sémantiques fournissant à l'organisation des pointeurs internes vers des ressources externes hétérogènes, pose donc le problème du développement d'extracteurs spécifiques pour chaque source d'information jugée pertinente.

Les recherches sur l'extraction d'information visent à développer des systèmes de génération d'«extracteurs» : des procédures qui extraient automatiquement des données à partir de sites web et convertissent l'information en un format structuré.

Alors que TSIMMIS [138] permet de questionner des sources multiples (sources du web, systèmes de base de données et système de fichiers), STALKER [139] se concentre sur les sources web exclusivement.

Dans TSIMMIS, les extracteurs sont écrits dans un langage de programmation procédural et sont compilés en code exécutable, tandis que dans STALKER [139] les agents extracteurs utilisent une structuration hiérarchique de l'information pour apprendre des règles d'extraction sous la forme d'automates finis.

XWRAP [140] est un générateur semi-automatique d'extracteurs qui repose sur la signification des balises spécifiques de HTML (par exemple des titres, des tableaux). Les extracteurs ainsi générés dépendent de l'imbrication et de l'orientation des tableaux et de tout autre élément, donc ils ne fonctionnent bien qu'avec des sites tabulaires.

Le WysiWyg Web Wrapper Factory (W4F) [141] est une boîte à outil pour produire des extracteurs du web. Il contient un langage d'identification de sites web et de navigation dans ces sites (règles de recherche) et un langage déclaratif pour extraire des données à partir des pages web (règles d'extraction). Il fournit également un mécanisme pour la projection des données extraites sur une structure cible. La méthode d'extraction employée dans W4F et notre méthode sont semblables à beaucoup d'égards, mais la principale différence est que W4F utilise un langage propriétaire pour l'extraction de données et des règles de projection. Comme nous allons le voir, notre approche repose sur les technologies XML et est guidée par une ontologie.

Pour certains sites du web tels que les catalogues en ligne ou les bibliothèques électroniques, nous avons observé qu'une majeure partie de l'information utilisée pour annoter les pages web est présente dans le contenu du document lui-même. Aussi nous avons conçu une solution pour produire automatiquement des annotations RDF en extrayant des données semi-structurées à partir des pages web.

La solution se base entièrement sur les technologies XML (XHTML, de XSLT et RDF) et est guidée par une ontologie formalisée en RDFS. L'outil que nous avons développé permet de produire des extracteurs qui téléchargent les documents d'un site web et en extraient des données en utilisant un script XSLT basé sur l'ontologie O'CoMMA afin de produire des annotations sémantiques.

Dans l'implémentation de l'extraction, nous avons le choix entre deux options pour le fonctionnement de l'extracteur:

- La conversion au vol, où l'extracteur emploie ses modèles pour convertir l'information qui lui est demandée à chaque fois qu'il est sollicité. Cette approche a l'avantage de toujours fournir au demandeur l'information la plus à jour mais le processus de conversion peut avoir un impact sur le temps de réponse de l'agent.
- La maintenance d'une image locale, où l'extracteur contrôle la source à intervalles réguliers et, si nécessaire, applique des mécanismes de conversion pour mettre à jour une base d'annotations. Cette approche a l'avantage de fournir un accès rapide et d'être disponible même si le site web extérieur n'est plus

accessible. Cependant, selon les cas, les données peuvent ne pas être à jour et de plus la quantité d'informations recopiées en interne peut être importante.

Nous avons choisi la deuxième option pour générer des annotations car elle permet que les annotations soient déjà disponibles lors du traitement d'une requête. De plus, elle découple et isole l'intranet de l'Internet, ce qui est une caractéristique appréciable pour la sécurité et la disponibilité. La transformation que nous opérons va donc produire des annotations qui seront ensuite gérées par un archiviste comme n'importe quelle autre base d'annotations.

Notre approche se concentre sur des sites web bien structurés où l'information peut être fréquemment mise à jour, mais où la disposition et l'organisation de ces pages demeurent les mêmes, ou du moins ne changent pas trop. Beaucoup de sites de ce type existent sur le web : les bibliothèques électroniques, les sites de la météo, des catalogues de produits, des sites boursiers, les sites d'équipes de recherches, etc.

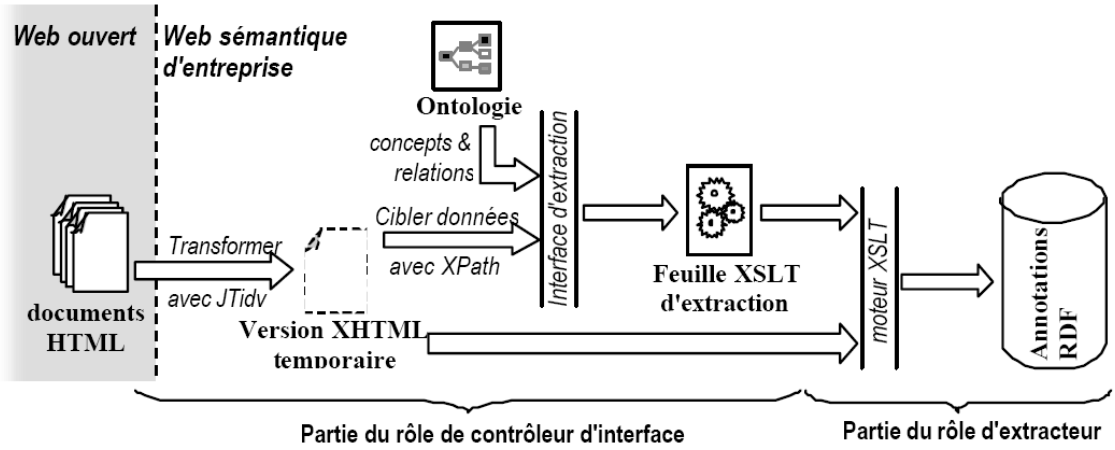


Figure 79. Processus d'extraction d'annotations par des feuilles XSLT

Comme illustré sur le Figure 79, nous procédons en trois étapes:

1. **Le système télécharge le code source en HTML et le convertit en XHTML.** Le modèle d'extraction est établi en utilisant une page web choisie parmi l'ensemble des pages à annoter et représentative de leur structure répétitive. Nous supposons que les pages sont repérées par des URL contenant un préfixe, un suffixe, et un compteur au milieu. Pour extraire des données à partir d'une page web, le système doit pouvoir désigner et accéder aux données contenues dans le document source. Une grande partie des documents HTML actuellement sur le web sont mal formés (balises manquantes, imbrication incorrecte, etc.) en raison de la tolérance des « parsers » des navigateurs web. Par conséquent, le système convertit les documents HTML en documents XHTML suivant la structure hiérarchique et corrigeant les erreurs grâce à JTidy, une API Java de HTML-Tidy, qui est un outil recommandé par le W3C. Ainsi, les étapes ultérieures peuvent utiliser des outils XML pour manipuler la page web comme un arbre DOM (modèle objet du document).

2. **Le système assiste la conception d'un modèle d'extraction basé sur l'ontologie O'CoMMA**. Ici, une annotation contient les informations extraites à partir du document ; elle est structurée par des concepts et des propriétés choisis dans l'ontologie O'CoMMA. Pour annoter un document web, l'utilisateur navigue dans l'ontologie pour choisir les concepts et les relations (propriétés) qui vont décrire la sémantique du document. Les valeurs de ces propriétés sont des littéraux qui apparaissent quelquefois dans le contenu de la page web. Le rôle du processus d'extraction est de rattacher ces informations utiles aux concepts et relations correspondants. Dans notre travail, nous supposons que les concepts et les relations dans l'ontologie O'CoMMA sont suffisants pour des annotations. Nous utilisons XSLT pour décrire les règles d'extraction dans le document XML, en employant des chemins XPath pour localiser les données à extraire. L'utilisation des expressions XPath est particulièrement efficace pour l'extraction de données significatives bien localisées, comme le prix des produits, le nom d'un auteur, les mots-clés d'un article, des tableaux sur les cours de la Bourse, etc. Pour assurer la précision et l'automatisation du processus d'extraction, nous avons créé des modèles XSLT génériques fournissant des fonctions de haut niveau comme dans la Figure 80 pour l'extraction récursive d'une liste de données délimitées par un séparateur fixe (ex. une liste d'auteurs), l'extraction itérative d'une liste de données consécutives dans une structure balisée (ex. les données d'une même colonne d'un tableau), le remplacement de données extraites par des concepts correspondants dans l'ontologie (ex. extraction de mots-clés). Ces modèles sont transparents aux utilisateurs du système, et sont inclus dans les modèles d'extraction qu'ils produisent. L'ensemble est facilement extensible. Enfin, le mécanisme d'extension de XSLT permet de combiner des expressions régulières dans XPATH, et donc de donner la possibilité d'extraire plus efficacement des données.
3. **Application du modèle d'annotation XSLT aux sources XHTML pour établir une base d'annotations.** Une fois que le modèle est créé, il est employé par un moteur de feuilles de styles XSLT pour transformer toutes les pages en une archive d'annotations pointant vers ces documents.

```
<xsl:template name="getListItem">
<xsl:param name="list" />
<xsl:param name="delimiter"/>
<xsl:param name="openning"/>
<xsl:param name="closing"/>
<xsl:choose>
 <xsl:when test ="$delimiter = 'br'">
   <xsl:for-each select="$list">
 <xsl:value-of select="$openning" disable-output-escaping="yes"/>
   <xsl:value-of select="normalize-space()" />
 <xsl:value-of select="$closing" disable-output-escaping="yes"/>
 </xsl:for-each>
 </xsl:when>
 <xsl:otherwise>
  <xsl:choose>
  <xsl:when test="string-length($list) = 0" />
  <xsl:otherwise>
     <xsl:choose>
        <xsl:when test="contains($list, $delimiter)">
        <xsl:value-of select="$openning"
                        disable-output-escaping="yes"/>
          <xsl:value-of select="concat(normalize-space
         (substring-before($list, $delimiter)),'&#10;')" />
        <xsl:value-of select="$closing"
                            disable-output-escaping="yes"/>
        </xsl:when>
        <xsl:otherwise>
       <xsl:value-of select="$openning"
                            disable-output-escaping="yes"/>
       <xsl:value-of select="concat(normalize-space
                                        ($list),'&#10;')" />
       <xsl:value-of select="$closing" disable-output-
                                            escaping="yes"/>
        </xsl:otherwise>
     </xsl:choose>
   <xsl:call-template name="getListItem">
    <xsl:with-param name="list" select="substring-
                               after($list, $delimiter)" />
  <xsl:with-param name="delimiter" select="$delimiter" />
    <xsl:with-param name="openning" select="$openning" />
    <xsl:with-param name="closing" select="$closing" />
   </xsl:call-template>
  </xsl:otherwise>
  </xsl:choose>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
```

Figure 80. Template de haut niveau réutilisable pour l'extraction de listes.

Pour aider les deux premières étapes du scénario, nous avons développé un outil graphique appelé WebAG (Générateur d'Annotations du Web) qui a été inséré dans l'interface existante de l'agent d'interface (IC) de CoMMA. La Figure 81 est une copie d'écran. L'utilisateur peut indiquer la page web échantillon dans la zone de texte en haut du cadre (1). La page web est alors déchargée et convertie en XHTML. La structure hiérarchique résultante (DOM) est visualisée et permet à l'utilisateur de choisir les données à extraire et d'indiquer leur XPath simplement par «drag and drop ». Le cadre (2) permet de naviguer dans l'ontologie et de choisir les concepts et les relations à utiliser dans les annotations. Le cadre (3) permet de définir le modèle d'annotation. Les données sélectionnées dans le cadre (1) et les concepts et les relations choisies dans le cadre (2) sont déposés dans ce cadre. Le cadre (4) montre le code de la feuille XSLT, comme il a été dérivé par le système. S'il le veut, un expert peut intervenir ici pour améliorer le résultat du processus d'extraction au cas où une manipulation particulière serait nécessaire. L'utilisateur visualise et contrôle l'annotation résultante dans le cadre (5) et dans le cadre (6) il peut demander la création d'un AWA (agent *wrapper* et archiviste) en indiquant l'ensemble des pages auxquelles ce modèle s'applique et en soumettant une requête au Wrapper Manager avec en paramètres le motif des URLs et la feuille XSLT.

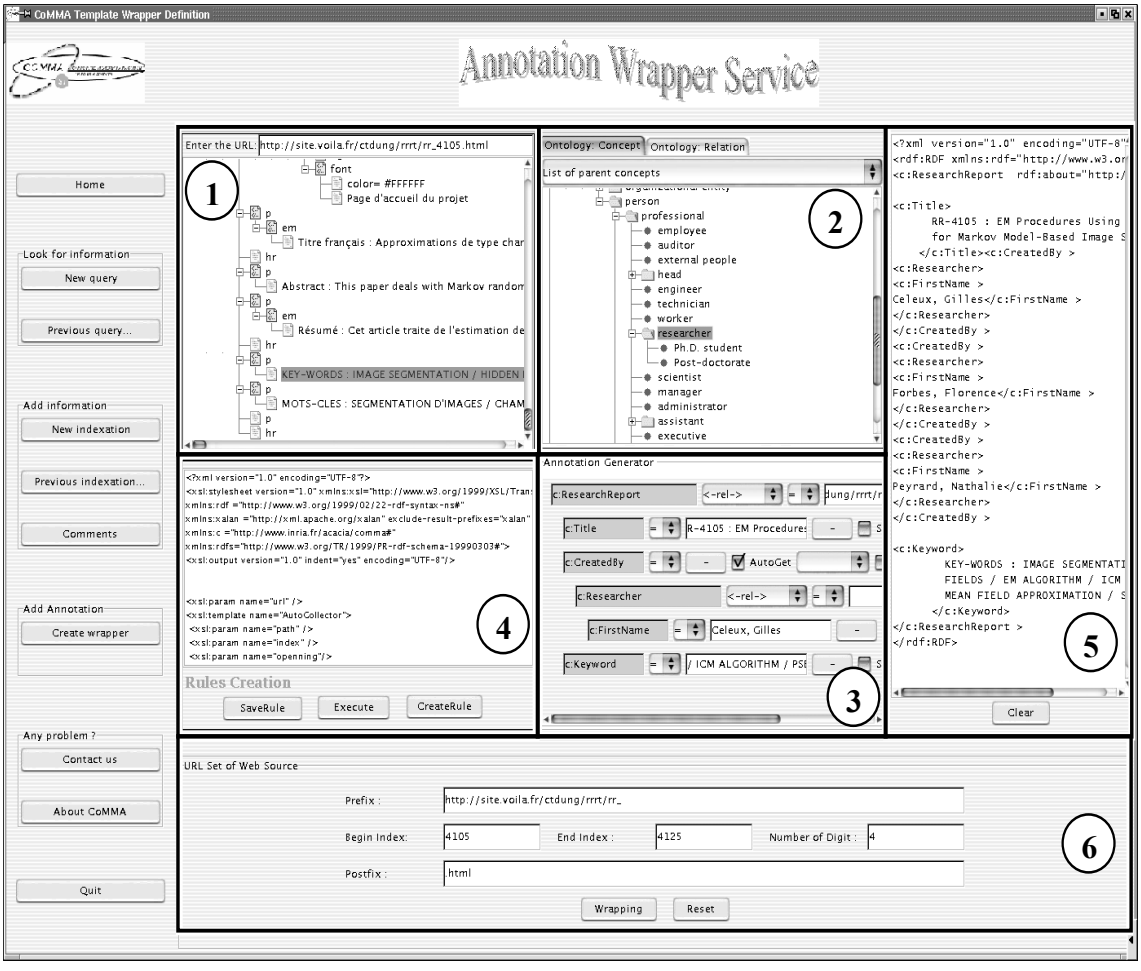
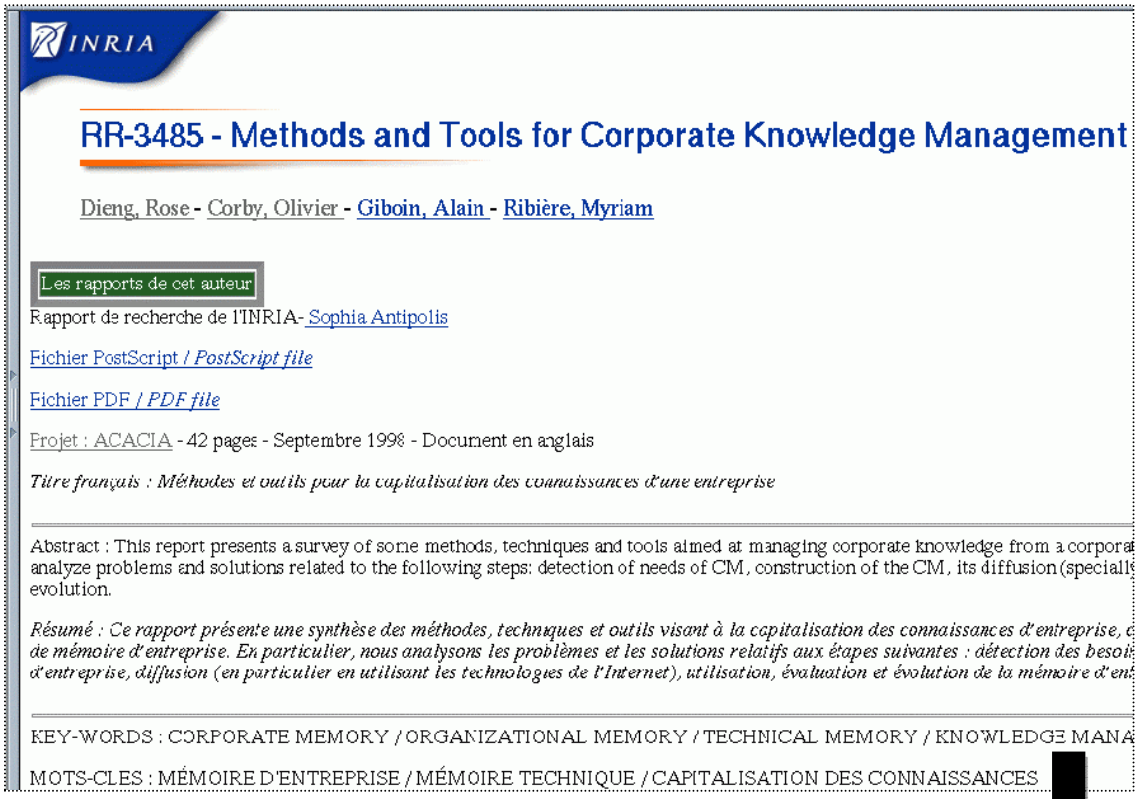


Figure 81. Interface d'édition de feuilles d'extraction d'annotation.

Nous avons testé la société d'extraction essentiellement sur trois sites web : la bibliothèque électronique des rapports de recherches de l'INRIA, la bibliothèque électronique des rapports techniques de l'Institut Technologique de Programmation de l'Université de Carnegie Mellon et la bibliothèque électronique des médicaments PubMed. Les résultats sont encourageants : les extracteurs ont correctement annoté ces sites en utilisant un modèle produit sur une page typique. La Figure 82 donne un exemple simple d'une annotation extraite.

Plusieurs améliorations peuvent être envisagées. Tout d'abord, dans les feuilles d'extraction, l'emplacement des données du document est représenté comme un chemin XPath absolu, cette solution est moins robuste lors d'un changement de la structure des pages web que des chemins relatifs. Nous envisageons d'utiliser un chemin XPath relatif, de sorte que l'emplacement des données puisse être indiqué en fonction de l'emplacement d'autres données et pas systématiquement à partir de la racine du document. De plus, pour améliorer la génération et la mise à jour des feuilles d'extraction, l'application de techniques d'apprentissage symbolique peut être envisagée à plus long terme.

En utilisant des technologies XML, l'implantation du processus d'extraction a été simplifiée par l'utilisation des outils disponibles pour la manipulation de XML, en particulier pour le prétraitement et l'analyse des documents. Cela devrait également réduire les coûts de maintenance et rendre l'outil plus pérenne. L'utilisation des feuilles XSLT pour la représentation des règles d'extraction, nous permet de reposer sur une norme et les composants développés pour ce système peuvent ainsi être réutilisés. De plus, en utilisant XSLT et XML, il est possible de fusionner des informations extraites de plusieurs ressources dans un nouveau résultat, donc cette approche peut être utilisée avec des sources hétérogènes.




```
<Comma:Article rdf:about="http://www.inria.fr/rrrt/rr-3845.html">
 <Comma:Title>
  RR-3485 - Methods and Tools for Corporate Knowledge Management
 </Comma: Title>
 <Comma:createdBy>
  <Comma:Researcher>
   <Comma:Name>Dieng, Rose</ Comma:Name>
  </Comma:Researcher>
 </Comma:createdBy>
 <Comma:createdBy>
  <Comma:Researcher>
   <Comma:Name>Corby, Olivier</Comma:Name>
  </Comma:Researcher>
 <Comma:createdBy>
 <Comma:createdBy>
  <Comma:Researcher>
   <Comma:Name>Giboin, Alain</Comma:Name>
  </Comma:Researcher>
 </Comma:createdBy>
 <Comma:createdBy>
  <Comma:Researcher>
   <Comma:Name>Ribière, Myriam</Comma:Name>
  </Comma:Researcher>
 </Comma:createdBy>
 <Comma:Keywords>CORPORATE MEMORY</Comma:Keywords>
 <Comma:Keywords>ORGANIZATIONAL MEMORY</Comma:Keywords>
 <Comma:Keywords>TECHNICAL MEMORY</Comma:Keywords>
 <Comma:Keywords>KNOWLEDGE MANAGEMENT</Comma:Keywords>
</Comma:Article>
```

Figure 82. Exemple d'extraction d'annotation sur un catalogue de rapports de recherche

Des fonctionnalités supplémentaires ont été suggérées pour le gestionnaire d'extracteurs tel que tuer ou ressusciter un AWA afin de gérer la population des extracteurs. De même des fonctionnalités supplémentaires sont envisagées pour l'AWA, en particulier des services de notification : informer d'un changement de la base d'annotations, informer de l'échec de l'application d'un modèle, informer d'un mot-clé qui n'a pas pu être traduit en un concept de l'ontologie, etc. Même si les sources initiales sont structurées de différentes façons, nous pouvons les traduire et les restructurer selon notre ontologie d'une façon qui peut être complètement différente de la structure initiale de l'information.

Etant guidé par une ontologie, le processus d'extraction de données peut exploiter un modèle du domaine pour produire non seulement la structure des annotations mais aussi, par exemple, des concepts se substituant à des mots-clés, afin d'être utilisés dans des inférences. La qualité du procédé de recherche dans la mémoire dépend de la qualité des annotations puisque CoMMA emploie des inférences pour exploiter la sémantique des annotations, par exemple pour généraliser ou spécialiser des requêtes de l'utilisateur. Reposer sur des annotations produites automatiquement peut ne pas fournir la qualité exigée et un environnement semi-automatisé dans lequel un opérateur humain est encore impliqué, comme proposé ici, est la garantie d'une qualité acceptable.

Notons en conclusion que cette approche réalisée en 2002 [22] préfigurait l'utilisation de XSLT dans GRDDL en 2007 [36] pour l'extraction de RDF depuis d'autres ressources web.

5.6 Graphes représentant des services et leurs compositions

Jusqu'à la fin des années 90, la modélisation d'entreprise fut principalement employée comme outil d'ingénierie organisationnelle. La prise de conscience de l'importance de la capitalisation des connaissances a aussi fait prendre conscience aux entreprises du fait que le modèle d'entreprise avait un rôle à jouer dans la gestion de la connaissance. Les recherches sur les systèmes d'information montrent, qu'une solution d'intégration d'informations peut bénéficier de modèles au niveau organisationnel. De même, une solution au problème d'intégration des technologies et applications d'entreprise peut reposer sur ces mêmes modèles. Cette idée est intéressante pour les praticiens de l'intégration d'applications d'entreprise mais exige un paradigme de programmation à un niveau d'abstraction assez élevé pour en rendre l'implantation réaliste.

Comme nous l'avons vu, dans le projet CoMMA, l'équipe Acacia avait déjà expérimenté des architectures multi-agents pour une gestion des connaissances distribuées [21]. A cette époque, le web sémantique n'avait pas encore rencontré les services web et, nos architectures et protocoles étant spécifiés au niveau connaissance, les systèmes multi-agents offraient le meilleur paradigme d'implantation.

Avec l'apparition d'environnements pour annoter sémantiquement les services web [142] [143] [144], un nouveau paradigme peut être employé pour intégrer des applications d'entreprise dans une mémoire basée sur un modèle : les services web sémantique d'entreprise ; c'est ce travail que nous présentions en [145] et que nous rappelons dans cette section. Nous verrons d'abord les scénarios motivants de cette approche et nous nous positionnerons dans l'état de l'art. Puis nous détaillerons les apports d'une telle intégration.

« Les organismes qui peuvent intégrer leurs applications et leurs sources de données ont un avantage concurrentiel différentiateur : l'utilisation stratégique des données et des technologies d'une compagnie pour une plus grande efficacité et un meilleur profit. Cependant, les directeurs de systèmes d'information s'essayant au défi de l'intégration sont confrontés à des problèmes décourageants : systèmes légataires disparates ; mélange incongru de matériel, de logiciels d'exploitation, et de technologies réseau ; applications et solutions propriétaires ; etc. L'intégration d'applications d'entreprise offre une solution à ce besoin de plus en plus pressant des entreprises. Elle intègre des technologies qui permettent à des processus et à des données d'affaires de communiquer et d'être échangés au travers des applications, et ce en intégrant un grand nombre de systèmes différents en un tout sans fractures. » [146]

De plus en plus souvent, notre équipe doit faire face à des scénarios exigeant non seulement une amélioration de l'accès à la connaissance mais également l'aide au calcul, à la décision, à l'aiguillage, à la transformation, etc. de cette connaissance. Jusqu'ici, les webs sémantiques que nous concevions se focalisaient sur la réalisation d'un accès unifié et intégré à un ensemble de sources de connaissances. Récemment nous avons noté une demande croissante pour obtenir la même facilité d'accès aux différentes

applications et services de l'entreprise et pour intégrer les deux mondes : intégrer la mémoire de l'entreprise et les applications de l'entreprise. Ces applications sont une nouvelle classe de ressources distribuées à gérer dans un intraweb sémantique.

Les utilisateurs s'attendent à ce que les responsables informatique permettent à des outils d'information très différents (station de travail, téléphone mobile, agenda électronique, fermes de calcul, etc.) de communiquer et, plus difficile encore, permettent à la diversité d'applications qui fonctionnent sur ces systèmes d'inter-opérer.

Les utilisateurs ne veulent pas seulement obtenir l'accès à l'information nécessaire, ils veulent cette information dans le format auquel ils sont habitués, avec une certaine assurance de qualité ou de provenance et avec les outils appropriés pour l'analyser, la modifier, la combiner, etc.

Les scénarios d'utilisation s'élargissent donc d'un accès unifié à l'information à un accès unifié à l'information *et* aux applications. Les mémoires organisationnelles incluent alors non seulement des médias de l'information mais plus généralement:

- Des services de *stockage* de l'information comprenant : des systèmes de diffusion de l'information (bibliothèques numériques, listes de diffusion, forums, blogs, etc.) et des systèmes dédiés (bases de données de l'entreprise ou publiques, ERP, entrepôt de données, etc.);
- Des services de *création* de l'information comprenant : des capteurs (ex : localisation, présence et disponibilité), des systèmes de calcul et d'inférence (ex : outils d'analyse de données);
- Des services de *gestion des flux* d'information comprenant : des canaux de communication sécurisés, des moteurs de règles d'affaires et *workflow*, une gestion de la connectivité, une gestion de la confidentialité et de la confiance ;
- Des services de *médiation* de l'information comprenant : des annuaires, des traducteurs, une surveillance de la qualité de services et du respect des contrats de service ;
- Des services de *présentation* de l'information comprenant : la transformation et la traduction inter-média, l'adaptation contextuelle, la personnalisation dynamique et les interfaces de manipulation.

Tous ces services peuvent être internes ou externes à l'organisation commanditaire, pourtant les utilisateurs veulent qu'ils inter-opèrent sans problèmes et, plus encore, qu'ils intègrent automatiquement les workflows définis dans les modèles d'affaires (business layer).

5.6.1 Services web sémantiques

Les services web sont une pratique normalisée pour intégrer des applications au travers du web. Ils se basent sur un ensemble de normes ouvertes utilisées pour échanger des données XML entre des applications en utilisant les protocoles d'Internet et en particulier ceux du web.

Un service web peut être défini comme un service en ligne, éventuellement commercial, basé sur XML, adapté à une tâche ciblée, simplement accessible et qui peut être composé. Un service web fournit une interface réseau fixe qui peut être appelée par n'importe quel client ; il est habituellement décrit par des métadonnées utiles aux consommateurs de services (nom, description, version, qualité, etc.). A la différence des modèles client/serveur web traditionnels, les services web ne fournissent pas à l'utilisateur une interface graphique mais une interface programmatique pour partager des données et des processus d'affaires (*business process*) à travers un réseau. Ainsi, typiquement, un appel à un service web peut être inclus dans une application offrant l'interface d'interaction avec les utilisateurs. Utilisés principalement dans le B2B en tant que moyens de communication pour les entreprises, les services web permettent à des organismes de relier leurs systèmes d'information pour un ensemble d'interactions focalisées et sans nécessiter une connaissance intime des systèmes de chacun. Le format de données et les protocoles normalisés des services web ne sont pas attachés à un langage de programmation ni même un système d'exploitation ou une plate-forme et permettent donc à des applications très hétérogènes de communiquer.

La solution la plus adoptée jusqu'ici dans l'industrie pour localiser, décrire, et invoquer des services web est le trio [147] SOAP, WSDL et UDDI. SOAP est un protocole simple basé sur XML pour la transmission de messages et l'invocation de procédures à distance (RPC); il repose sur les protocoles standards de transport tels que HTTP et SMTP. WSDL est un langage de description pour les interfaces des services web, c'est-à-dire, les points d'accès à travers lesquels les applications communiquent avec lui ; cette description inclut les séquences de messages échangés, la structure des messages, la structure des données incluses dans ces messages, etc. UDDI définit des services et langages permettant l'indexation et la recherche de services mis en ligne.

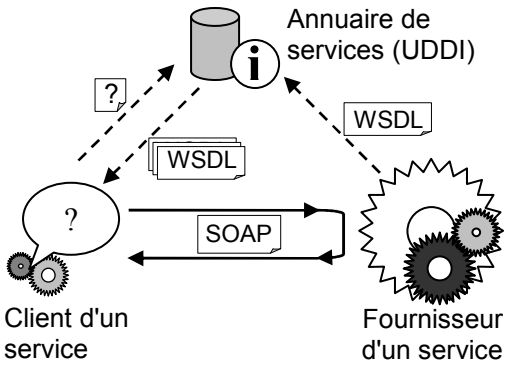


Figure 83. Triptyque de l'architecture de service [17]

Un certain nombre d'extensions ont été proposées pour palier aux limitations de ce trio de base. Citons : BPEL4WS [148] qui permet la spécification formelle de protocoles et de processus d'affaires utilisant les services web ; BPML [149] lui aussi pour la modélisation des processus d'affaires ; WSCI [150] pour la description d'une chorégraphie de services *i.e.* la description d'une séquence d'interactions entre des services ; XLANG [151] pour spécifier des protocoles et flots d'interaction complexes ; WSCL [152] pour spécifier une interface abstraite d'un service *i.e.* au niveau affaires ; WSFL [153] pour décrire la composition de services ; etc.

Il est un sous ensemble particulièrement intéressant pour nous parmi ces extensions : celui des extensions intégrant le web sémantique et notamment trois contributions OWL-S, WSMO et WSDL-S:

- OWL-S [154] [143] est un ensemble de trois ontologies en OWL pour décrire les services web : (i) *Service Profile* est utilisé pour décrire essentiellement ce que fait le service (nom, catégorie, qualité, etc.) ; (ii) *Service Process* indique comment fonctionne le service (opérations, entrées et sorties) ; (iii) *Service Grounding* précise comment accéder au service (interface et messages d'interaction).
- SWWS [155] [156] vise à fournir une infrastructure pour la description, une infrastructure pour la découverte et une plate-forme de médiation pour les services web. L'infrastructure WSMF (*Web Service Modeling Framework*) fournit un modèle conceptuel pour développer et décrire les services web et leur composition. WSMO (*Web Service Modeling Ontology*) est une ontologie pour décrire les services.
- WSDL-S [142] étend l'expressivité de WSDL avec des annotations, basées sur des ontologies externes, pour décrire les compétences et les besoins d'un service (entrées/sorties, pré-conditions, effets, opérations).

OWL-S étant directement exprimé en OWL nous l'utilisons pour l'intégration de ressources dynamiques dans les mémoires d'entreprise. Cependant, dans nos scénarios actuels, nous n'utilisons que le *profile* et le *grounding* ainsi que la description des entrées/sorties du *process*. Cela correspond dans WSMO à la description des capacités des services et des entrées/sorties et à la sémantique ajoutée à WSDL par WSDL-S.

La Figure 84 résume la pile de standards sur laquelle nous reposerons pour l'ensemble des contributions présentées dans cette section. A l'époque où ce travail était réalisé, SAWSDL [157] n'existait pas et nous reviendrons sur ce point en conclusion.

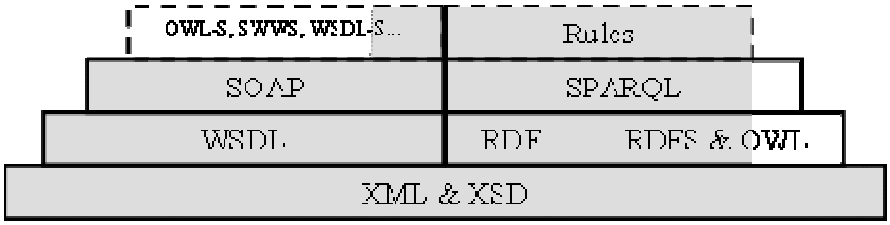


Figure 84. Partie implantée de la pile des standards

5.6.2 Description et identification de services

Dans les mémoires d'entreprise que nous développions jusque là, les annotations décrivaient généralement des ressources documentaires ou des structures d'entreprise. Cependant, en s'appuyant sur un schéma comme OWL-S, ces annotations peuvent aussi décrire les services web disponibles en ligne (intranet, extranet, Internet). A titre d'exemple, notre première annotation était celle d'un service basé sur une application LDAP de l'INRIA permettant aux employés d'obtenir le numéro de téléphone de l'assistante d'un autre employé. Cette annotation donne dans une structure RDF le type du service, l'URL où l'invoquer, les paramètres d'entrée et de sortie ainsi que leurs types.

Une fois les services annotés, Corese [84] nous permet d'automatiser l'identification des services disponibles. Selon une architecture orientée service (Figure 83) Corese se positionne comme un registre UDDI sémantique (Figure 85). Dans cette nouvelle architecture, nous sommes passés de l'utilisation d'une recherche UDDI textuelle à une recherche sémantique pour résoudre les requêtes sur les descriptions des services, en tenant compte des ontologies utilisées pour les caractériser. Avec cette architecture, les annotations des services correspondant aux applications d'entreprise sont stockées dans le web sémantique d'entreprise. Dès lors, les services peuvent être automatiquement découverts et invoqués dynamiquement par les utilisateurs finaux sans connaissance préalable de leurs descriptions.

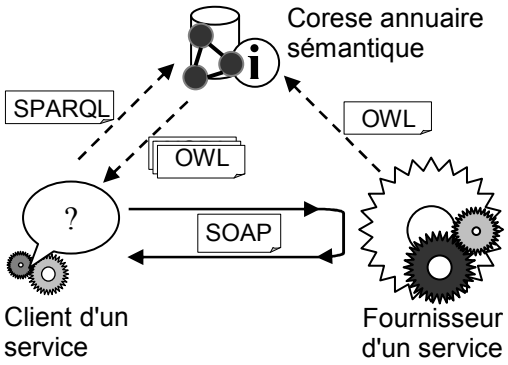


Figure 85. Corese utilisé comme un registre UDDI sémantique

Notre implémentation a été intégrée à l'architecture de serveur web sémantique de Corese. Le portail s'est donc enrichi d'un volet pour les services avec trois principales fonctionnalités : découverte de services web, invocation dynamique de services web et composition de services web.

5.6.3 Découverte et invocation de services

En utilisant le langage de requête de Corese et les fonctionnalités de rendu du serveur web sémantique, nous avons construit des pages d'annuaire et de recherche de services. A titre d'exemple, la requête suivante permet d'identifier les services de messagerie et leurs paramètres d'entrée.

```
?s   rdf:type              ex:Messaging
?s   service:describedBy   ?p
?p   proc:hasInput         ?param
```

La recherche exploitant l'ontologie, les résultats incluent les sous-types de ce type de service (ex : email, messagerie instantanée, messages vocaux, etc.) et les types précis de leurs paramètres (ex : adresse mail, téléphone, etc.):

En utilisant des patrons de requêtes génériques paramétrables par des formulaires, nous construisons des services d'annuaires et de recherche simples tel que celui de la Figure 86 où l'utilisateur peut sélectionner un service selon sa catégorie, les informations qu'il requiert, les sorties qu'il produit, etc.

Corese résout ces requêtes et le résultat est dynamiquement rendu sous formes de pages dans le portail comme pour la recherche d'une ressource documentaire. Comme le montre la Figure 87 lorsqu'un service est sélectionné par un utilisateur via le portail nous générons dynamiquement un formulaire offrant une interface pour appeler ce service. La génération de ce formulaire est rendue possible par l'annotation du service qui permet en particulier de connaître précisément les paramètres d'entrée, leurs types, leur désignation.

A la soumission d'un tel formulaire, les entrées sont utilisées pour générer un client dynamique et l'appel au service. Le résultat est alors simplement affiché comme une page intégrée au portail.

Figure 86. Extrait de l'interface de recherche d'un service

La Figure 87 montre deux fenêtres :

- La fenêtre de gauche montrant le résultat de la requête de recherche d'un service dans laquelle l'utilisateur a sélectionné le service d'envoi d'emails ;
- La fenêtre de droite fournissant un formulaire pour spécifier les entrées. Une fois soumis, ce formulaire déclenche un appel au service qui est alors dynamiquement exécuté et affiche les sorties possibles dans l'interface web.

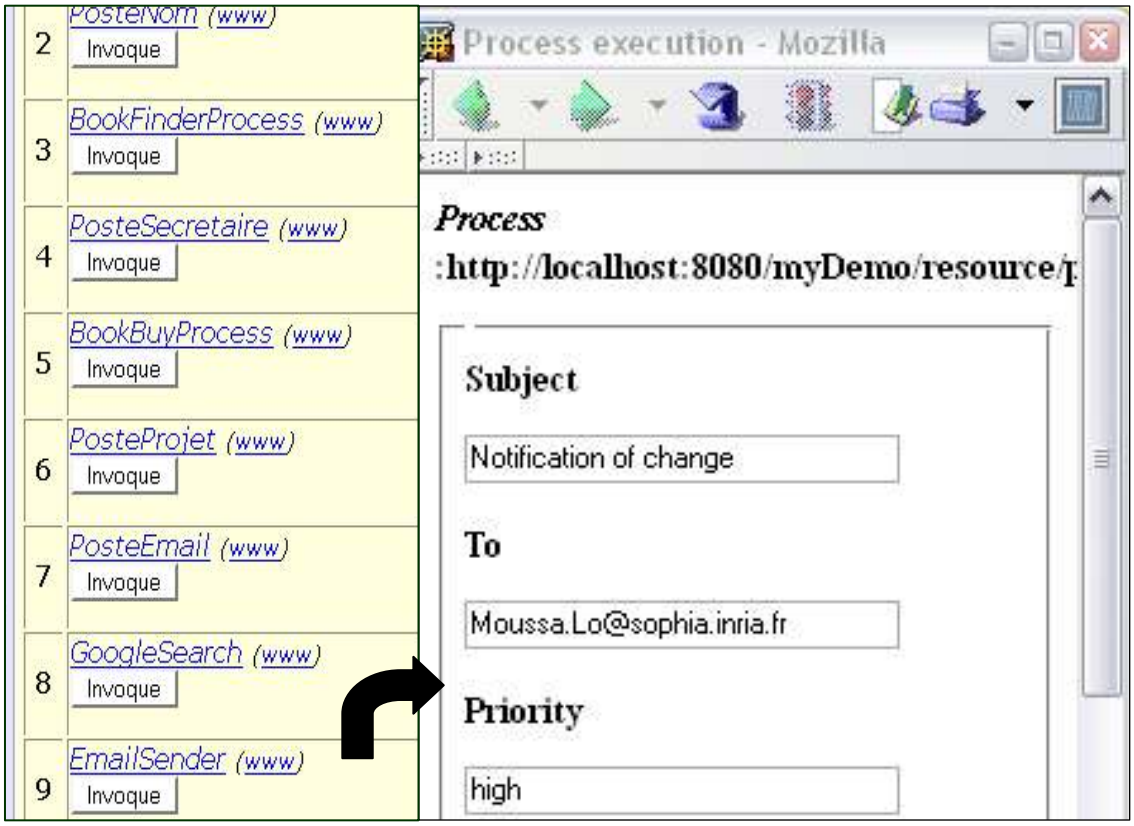


Figure 87. Découverte et invocation d'un SWS d'entreprise

Cette première étape de l'intégration de services web et web sémantique d'entreprise permet donc d'intégrer dans les portails d'information d'un web sémantique d'entreprise des accès à des applications d'entreprise avec les avantages suivants:

- Un environnement unique pour l'accès à l'information et aux services aussi bien d'un point de vue coût cognitif (un point d'accès unifié, une seule et même interface, etc.) que d'un point de vue coût d'infrastructure (réutilisation des clients web et serveurs web déjà en place, réintégration d'applications légataires) ;
- La capacité à ajouter dynamiquement des services simplement en enregistrant de nouvelles annotations les décrivant ;

- La capacité à dissocier application *vs.* service permettant ainsi : d'éclater une application en différents services pouvant s'adresser à différents groupes d'utilisateurs ; de regrouper plusieurs applications dans un service pour former des services intégrés à forte valeur ajoutée ; d'adapter différents services sur une même application pour réaliser une même tâche mais de différentes façons.

Ce dernier point conduit naturellement à l'intérêt de l'aide à la composition de services.

5.6.4 Composition de services

Le service recherché par un utilisateur peut ne pas être disponible en tant que tel mais être équivalent à la composition de plusieurs services disponibles. Nous utilisons Corese pour proposer deux façons d'obtenir une composition de service : la composition interactive et la génération automatique de séquences.

Dans la composition interactive, l'utilisateur part d'une entrée ou d'une sortie souhaitée ; le moteur lui propose itérativement des services consommateurs ou producteurs de ce type d'information en réitérant à partir des entrées ou sorties du dernier service sélectionné et ce jusqu'à ce que l'utilisateur soit satisfait de la composition. Cette méthode permet à l'utilisateur qui connaît le type de combinaison qu'il souhaite faire de rapidement localiser les services qui composent la chaîne et de les composer pour former un nouveau service directement utilisable.

La Figure 88 montre quatre étapes d'une composition assistée où l'utilisateur part d'une preuve de paiement comme sortie souhaitée (*BookBuyNotification*) et sélectionne en trois étapes les trois services qui, composés en séquence, lui permettent d'obtenir cette notification à partir d'un titre de livre (*BookName*). Cette composition peut alors être enregistrée comme un nouveau service (complexe) permettant d'acheter un livre à partir de son titre.

Dans la composition automatique de séquences, l'utilisateur spécifie une entrée *et* une sortie souhaitées et Corese propose automatiquement des séquences de services reliant ces deux types d'information.

Pour cela, nous exploitons l'algorithme de Corese permettant de trouver des chemins de longueur variable et de type contrôlé entre deux ressources. Ainsi la requête suivante peut être utilisée pour trouver des liens de famille (chemins de longueur maximale 6) entre deux personnes (`Fabien` et `Laura`):

```
Fabien ex:lienDeFamille[6] Laura
```

Les liens trouvés peuvent être des sous-types de `lienDeFamille` et donc une réponse possible est:

```
Fabien → (filsDe) → Michel → (frèreDe) → JeanLuc → (pèreDe) → Fabrice →
(pèreDe) → Laura
```

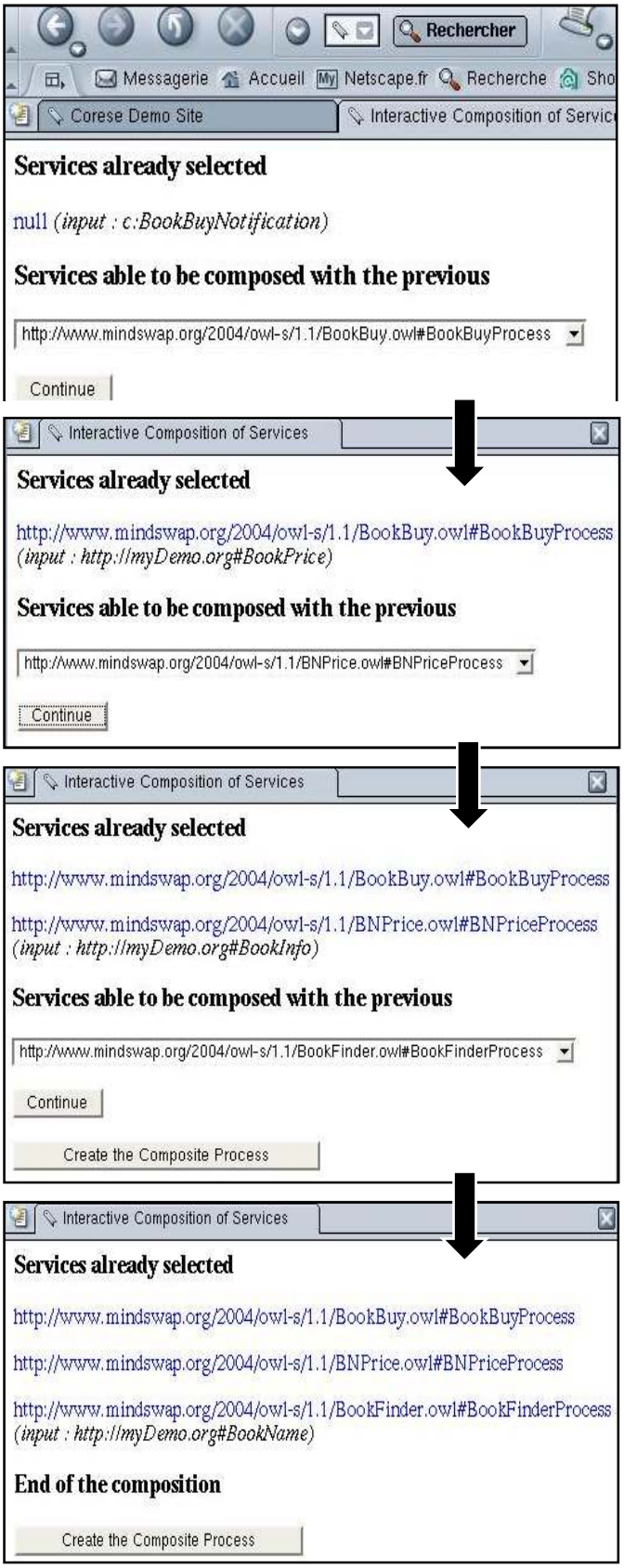


Figure 88. Exemple de composition interactive de services

Utiliser cette fonction pour générer des séquences de services se fait donc en deux temps:

1. Définir formellement la notion de services directement composables dans une séquence : deux services sont directement composables dans une séquence *ssi* les entrées de l'un correspondent ou subsument, au sens ontologique, les sorties de l'autre. Cette connaissance est codée dans une règle de production qui complète alors automatiquement la base d'annotation des services avec les instances de la propriété *composable* marquant les couples de services compatibles (Figure 89);
2. Demander un chemin de services composables qui relie deux services tels que l'entrée du premier service et la sortie du dernier service correspondent aux entrée et sortie demandées par l'utilisateur.

```
<cos:rule>
    <cos:if>
      ?s1 rdf:type proc:Process
      ?s2 rdf:type proc:Process
      ?s1 proc:hasInput ?input
      ?s2 proc:hasOutput ?output
      FILTER(?s1 != ?s2)
      ?input proc:semanticType ?inType
      ?output proc:semanticType ?outType
      ?outType rdfs:subPropertyOf ?inType
    </cos:if>
    <cos:then>
      ?s2 proc:composable ?s1
    </cos:then>
</cos:rule>
```

Figure 89. Règle pour la détection de services composables

En reprenant l'exemple précédent sur l'achat d'un livre, la requête est la suivante et la séquence proposée par le portail en résultat est donnée en figure (Figure 90) :

```
?s1 all::proc:composable[4] ?s2
?s1 proc:hasInput ?param1
?s2 proc:hasOutput ?param2
?param1 proc:semanticType c:BookName
?param2 proc:semanticType c:BookBuyNotification
```

Ce résultat donne la chaîne de composition de trois services suivante :

BookFinderProcess (input : *BookName*, output : *BookInfo*)

BNPriceProcess (input: *BookInfo*, output: *BookPrice*)

BookBuyProcess (input:*BookPrice*, output: *BookBuyNotification*)

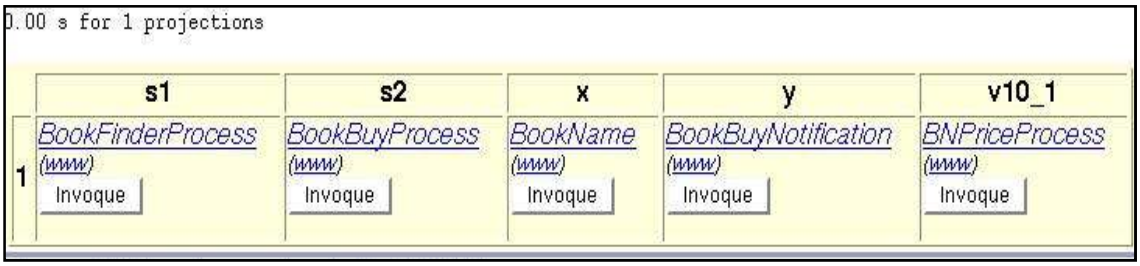


Figure 90. Séquence de services pour l'achat d'un livre

Les deux méthodes permettent d'obtenir des compositions qui peuvent être enregistrées et annotées comme de nouveaux services et ainsi directement réutilisables par d'autres utilisateurs sans repasser par une nouvelle phase de composition.

5.6.5 Composition entre les services et la mémoire

Dans un web sémantique et *a fortiori* dans celui d'une entreprise, des connaissances sont disponibles tout autour des services. Nous avons donc envisagé deux formes d'interactions entre les services et ces connaissances:

- *Assistance à l'invocation de services* : en utilisant le typage des entrées des services nous dérivons des requêtes permettant d'isoler les valeurs possibles pouvant être utilisées comme paramètres d'un appel à un service ;
- *Intégration des accès en lecture et écriture de la mémoire comme un service web* : en publiant des interfaces d'accès à Corese sous la forme de services web, ces nouveaux services d'accès à la mémoire peuvent être composés avec d'autres services/applications de l'entreprise pour proposer de nouveaux services complexes intégrant les séquences d'interaction avec la mémoire.

Pour l'assistance à l'invocation de services, nous associons un prédicat à chaque entrée de service et, au besoin, nous définissons ce prédicat dans une règle qui permet d'en générer automatiquement les valeurs. La Figure 91 montre l'exemple d'un service LDAP de l'INRIA qui, encapsulé dans un service web, permet d'obtenir le numéro de téléphone de l'assistante d'un employé.

Le paramètre d'entrée "nom" de ce service a été associé au prédicat *EmployeeName* défini par la règle de production:

```
?x rdf:type c:Employee
?x c:Name ?n
⇒
?x c:EmployeeName ?n
```

Dès lors, comme le montre la Figure 91, lorsque ce service est sélectionné, son interface est pré-remplie avec des valeurs candidates issues de requêtes sur les prédicats associés aux entrées. Ainsi, dans notre exemple, l'interface propose par défaut une liste de noms d'employés prêts à être utilisés.

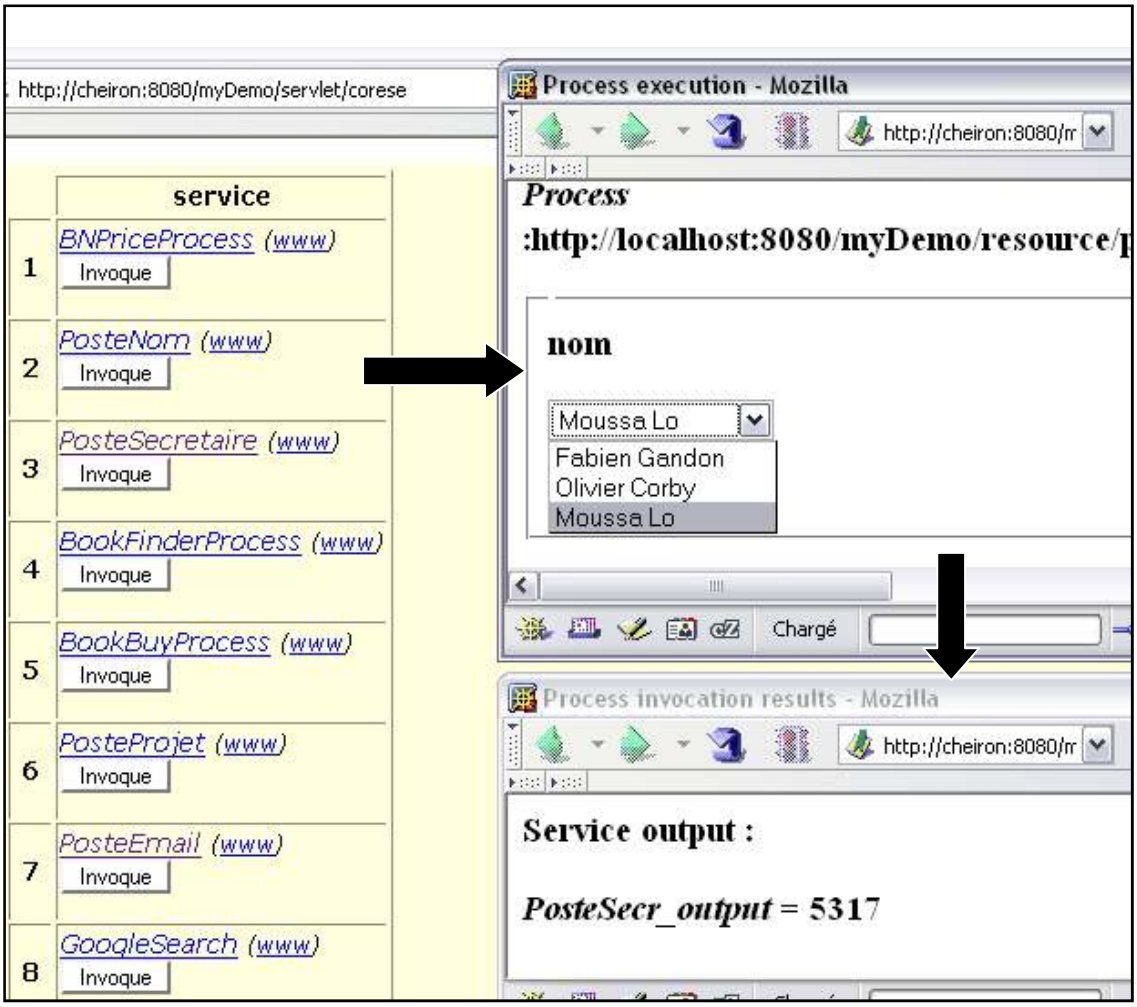


Figure 91. Complétion automatique des entrées possibles

L'intégration des accès en lecture et écriture de la mémoire comme un service web est plus complexe mais permet d'aller plus loin : non seulement le service attaché à Corese peut permettre d'utiliser le résultat de requêtes complexes sur la mémoire pour nourrir des services, mais il peut aussi être utilisé pour enregistrer dans la mémoire des annotations construites à partir du résultat d'un ou plusieurs services. Pour ce faire nous avons conçu un prototype basé sur trois services:

- *Corese Query Service* qui prend en entrée une requête et donne en sortie sa réponse en XML ;
- *Corese Save Service* qui prend en entrée une annotation, la sauvegarde et fournit en sortie la confirmation de son enregistrement ;
- *XSLT Service* qui est utilisé pour transformer les formats XML des données qui transitent entre les services (médiateur syntaxique) ; par exemple pour extraire le paramètre d'un service à partir du résultat d'une requête.

Ce dernier prototype ouvre l'intéressante perspective d'une plate-forme de gestion de flots de données et d'applications entièrement intégrées au web interne de l'entreprise. Elle demande cependant encore beaucoup de réflexion.

5.6.6 Discussion

Nous avons présenté notre approche d'intégration de ressources dynamiques dans un web sémantique d'entreprise en utilisant les services web sémantiques. L'accès aux applications est encapsulé dans des services web annotés. Le moteur de recherche sémantique Corese est utilisé comme registre UDDI sémantique pour prototyper un portail de services de l'entreprise. Il permet notamment la découverte automatique, l'invocation dynamique et la composition de services. Les aspects techniques sont détaillés dans un rapport de recherche [158].

Il y a un grand nombre d'améliorations possibles : améliorer l'ergonomie des interfaces de création et d'exécution d'une composition ; assister l'annotation sémantique d'un nouveau service (atomique ou composé) ; dépasser la composition sous forme de séquence ; approfondir les méthodes de composition de la mémoire et des services ; introduire les contraintes de sécurité ; etc.

Cependant, ces premiers essais nous ont aussi permis de vérifier l'importance et la faisabilité de plusieurs scénarios des services web sémantiques. En particulier, une question récurrente de ce nouveau domaine est celle de la composition manuelle vs. semi-automatique vs. automatique. Pour l'instant, nous n'avons pas trouvé dans nos scénarios un cas où les besoins d'un utilisateur peuvent être captés facilement et d'une façon si complète qu'elle permette d'envisager une composition entièrement automatique. Une telle fonctionnalité semble toujours requérir une quantité importante de connaissances du domaine et de connaissances de sens commun. Nous trouvons plus réaliste de considérer par exemple la requête d'un utilisateur souhaitant un environnement l'assistant dans la description de l'implantation des workflows de son organisation ou des méthodes de composition dédiées.

Parmi nos perspectives, nous étudions actuellement les relations possibles entre les services web sémantiques et d'autres standards tels que SPARQL (requêtes et protocoles), les règles, la sécurité, etc. Nous pensons aussi que, comme pour le web sémantique, il y a un besoin d'étendre ou compléter les formalismes actuels pour intégrer des primitives de gestion du niveau sémiotique. En effet, qu'il soit atomique ou composé, un service web dynamiquement découvert, tout comme une requête ou sa réponse, vient très souvent au contact de l'homme en devant, par exemple, intégrer ses entrées ou sorties à une interface utilisateur. S'il est vrai que les services web sont principalement conçus pour des interactions programmatiques dans des scénarios de B2B, il n'en est pas moins vrai que ces services, lorsqu'ils sont dynamiquement découverts et intégrés à des applications, posent le problème de la génération dynamique d'une interface d'interaction adaptée. En d'autres termes, tout comme le web sémantique doit se doter de primitives au service d'une logique sémiotique et pragmatique de son rendu [159], les services web sémantiques doivent intégrer ces mécanismes pour qu'à l'issue d'une découverte ou d'une composition l'invocation dynamique de ce service puisse se faire dans une interface ergonomique dynamiquement générée pour le contexte et les besoins de l'utilisateur.

Enfin, depuis la réalisation de ce travail, il faut noter deux évolutions :

- La recommandation SAWSDL [157] émise par le W3C en aout 2007 qui correspond à peu près à l'expressivité de WSDL-S et celle que nous utilisons ici.
- Le projet e-Wok [28] dans lequel nous participons actuellement à l'annotation des services des partenaires en SAWSDL [157] et à la création d'un profile GRDDL [38] permettant d'obtenir automatiquement l'annotation RDF/XML d'un service décrit en WSDL 2.0 + SAWSDL.

5.7 Vers des écosystèmes de l'information

Dans cette dernière section, nous voulons reprendre une perspective que nous avions proposée en 2003 sur la combinaison d'agents réactifs et délibératifs pour peupler les écosystèmes des paysages d'information (ou infosphère) [160]. Notons que cette année sera créé l'atelier NatuReS[14] à ISWC en 2008 qui s'intéresse aux approches émergentes pour le web sémantique.

La diversité des ressources et de l'information des infosphères appelle des écosystèmes artificiels avec une diversité d'agents interagissant et allant du paradigme des agents réactifs au paradigme des agents délibératifs afin de maintenir les écologies de l'information. Dans l'article [160], nous discutions de la notion d'infosphère et de certains intérêts de la famille XML pour un tel monde, et nous fournissions des exemples montrant l'intérêt de ces systèmes hybrides.

L'avènement des réseaux d'information rend le cyberespace de William Gibson plus proche d'une formidable prévision que d'une fiction. Toutefois, le résultat actuel est un monde «sauvage», un *world 'wild' web*, où les ressources d'information et les services sont dispersés, toujours changeants et croissants, ce qui rend de plus en plus difficile pour les humains de localiser et d'accéder à une ressource pertinente. Parce que ces paysages d'informations sauvages sont non organisés et hétérogènes dans leur forme, leur contenu et leur qualité, il reste extrêmement difficile d'automatiser des tâches et de fournir une aide par des outils intelligents. En fait, à bien des égards, ces paysages peuvent être comparés à notre propre monde : ils sont vastes, distribués, hétérogènes, ils fournissent un sol fertile riche en ressources d'information ; les acteurs de ces espaces peuvent y être situés ; les acteurs peuvent y percevoir et agir avec leurs ressources locales ; les acteurs peuvent y interagir par le biais de leur environnement.

De cette similitude découle la métaphore de l'infosphère : l'équivalent pour le monde de l'information de notre biosphère et de son écologie. La biosphère est la sphère d'action de la vie sur Terre qui englobe les êtres vivants avec leur environnement. C'est un écosystème fermé, autorégulé à travers des cycles complexes impliquant de multiples interactions dans une grande variété d'êtres vivants, et entre ces derniers et une grande variété d'environnements. Ainsi, l'idée principale est de mettre en place des écosystèmes de l'information ou infosphères ; l'infosphère est constitué par l'ensemble des entités d'information - y compris les agents - les procédés, leurs propriétés et relations mutuelles. [161]

L'intelligence artificielle distribuée est en train d'élaborer des agents d'information afin de peupler ces infosphères. Les agents sont définis comme des entités artificielles clairement identifiables avec des frontières et des interfaces définies. Ils sont situés dans un environnement qu'ils perçoivent par le biais de capteurs, sur lequel ils agissent et auquel ils réagissent à travers des effecteurs. Ils ont des aptitudes sociales pour interagir

[14] http://natures.few.vu.nl/

avec d'autres agents ou des humains, mais tout en gardant le contrôle de leur comportement. Contrairement à l'homme, les agents ont une infinie patience et persévérance, et ils peuvent exploiter et gérer d'énormes quantités d'informations très rapidement. L'importance et l'intérêt d'une convergence entre les agents et les communautés web sont maintenant reconnus [4]. Ainsi, en parallèle au développement des agents d'information permettant l'automatisation de tâches, le W3C a publié des recommandations pour doter le web d'une structure (XML) et de sémantique (RDF/S et OWL). Ensemble, ces deux domaines contribuent au développement complet des infosphères.

Beaucoup de personnes en sont déjà à construire l'économie, l'éthique, la confiance, etc. de ces infosphères, mais il leur manque toujours une écologie stable. En fait, au lieu d'écologie, les travaux actuels dans le domaine des agents ressemblent plutôt à de l'autécologie (c'est-à-dire, l'étude d'un organisme individuel ou d'une seule espèce), alors que nous devrions vraiment aller vers la synécologie (c'est-à-dire, l'étude des relations écologiques entre les communautés et les espèces d'organismes) et l'écologie complète (c'est-à-dire, l'étude des relations entre les communautés et les espèces et de leurs relations à leur environnement). Un symptôme de ce manque est qu'il reste toujours une dichotomie entre les agents intelligents d'information (tels que présentés dans [162]) et les agents légers ou réactifs [163] formant des essaims et une intelligence sociale [164] dans des espaces d'information comme Anthill [165] et son application Gnutant.

Pour la plupart des problèmes, ni une architecture purement délibérative, ni une architecture purement réactive n'est complètement appropriée. Les approches conciliant une architecture réactive et des comportements délibératifs se placent souvent au niveau de l'architecture d'un agent [166], ce qui conduit à des agents hybrides avec une architecture de traitement à la fois réactive et délibérative pour son comportement, et habituellement basée sur une architecture en couches hiérarchiques ou parallèles. S'il est vrai que le corps humain est composé de cellules que l'on peut voir comme des petits organismes et que, par conséquent, une perspective holonique[15] des agents peut être intéressante, il est également vrai que les humains ne sont pas composés d'insectes alors qu'ils bénéficient des nombreux rôles écologiques des insectes et vice-versa. La métaphore de ces organisations doit être étendue à des systèmes hybrides complexes composés d'agents et d'organisations hétérogènes.

La complexité des espaces d'information sera bientôt telle qu'elle devra faire appel à des systèmes de régulation complexes et à de nouvelles approches telles que l'informatique autonomique [167] ou l'écologie des infosphères. Nous devons chercher à développer des communautés optimales de l'information c'est-à-dire des agents qui peuvent atteindre un stade stable grâce à un processus en chaîne, dans lequel les communautés relativement simples fournissent une base pour les communautés plus complexes. L'idée est de développer dans ces mondes d'information l'équivalent, par exemple, des chaînes

[15] approche dans laquelle un agent peut se constituer à partir d'un regroupement d'autres agents.

alimentaires et des toiles alimentaires (*i.e.*, un chevauchement de chaînes) pour construire des chaînes d'informations et des toiles d'informations, fournissant au final une grande valeur ajoutée, par rapport au terreau fertile mais sauvage des terrains d'information de départ.

La diversité des infosphères suggère que nous avons besoin d'un large spectre d'agents (depuis les types purement réactifs jusqu'aux types délibératifs les plus complexes en passant par tout le spectre intermédiaire) en réponse au large éventail des tâches de gestion de l'information et des services. Ceci pose la question de leur cohabitation et des interactions entre eux, et entre ces derniers et leur environnement. Notre propre écologie a besoin d'un spectre complet des êtres et des organisations de la vie ; il est en équilibre complexe basé sur des relations directes (par exemple proie-prédateur) ou des chaînes indirectes (par exemple, les insectes dégradent les détritus organiques en matière fertile, les plantes utilisent ce terrain fertile à la croissance de matière végétale par photosynthèse, les herbivores se nourrissent de ces plantes pour produire une matière organique animale, etc.). De même, de nombreuses formes d'interactions peuvent être envisagées entre les différentes espèces d'agents d'information. Cependant actuellement, les interactions se font généralement uniquement au sein d'une même famille d'agents par stigmergie [168] (agents réactifs communiquant indirectement en modifiant leur environnement local), par communication au niveau des connaissances [169] (agents délibératifs communiquant en utilisant des langages, des ontologies et des protocoles) ou par approches holoniques (où les holons forment des communautés qui sont réifiées comme de nouveaux agents capables de former de nouvelles communautés). Un écosystème d'information complexe comprend des chaînes et des toiles d'interactions qui transcendent les familles d'agents pour construire des cycles stables et maintenir la pyramide des espèces où chaque niveau apporte une valeur ajoutée (plus de structure, l'analyse des résultats, etc.) et permet d'extraire, d'affiner, d'exploiter et de gérer les minerais riches présents dans les ressources d'information.

Dans un article de 2003 [160], nous mettions l'accent sur une position technologique précise : l'utilisation des technologies XML pour les paysages d'informations. Puis nous présentions deux points de vue sur les interactions :

- La personnalisation d'un comportement holonique, où les tâches atomiques incluses dans les plans d'un agent sont définies dans des scripts extérieurs et échangés comme de simples agents réactifs.
- L'élevage de populations d'agents réactifs par des agents intelligents pour propager des tâches sur le réseau.

Dans le sillage de XML, une famille d'extensions et de modules se développe parmi lesquels XSLT [170] (*Extensible Stylesheet Language Transformation*) est d'un intérêt particulier pour nous : ce langage permet la transformation d'un arbre XML en un autre arbre XML ou en tout autre format textuel. Il est possible, par exemple, de générer une table des matières, d'adapter le tri des listes, etc. Ainsi, un document peut être considéré différemment et transformé en d'autres documents afin de s'adapter aux besoins et aux

profils des agents et des utilisateurs tout en étant stocké et transféré dans un format unique.

XSLT est un langage de règles, où les règles de formatage appelées *templates*, transforment un arbre source en un arbre résultat. La transformation est réalisée en comparant les conditions d'application de chaque *template* à l'arbre source et instanciant le cas échéant le contenu du *template* pour créer l'arbre résultat.

Plusieurs *templates* peuvent correspondre à un élément donné dans l'arbre source, mais seul le plus précis sera appliqué. Des opérateurs permettent d'accéder aux valeurs des nœuds, des instructions de test et d'embranchements sont disponibles. Les *templates* sont appliqués récursivement sur le document XML, en recherchant, après l'avoir traité, les *templates* correspondants aux enfants du nœud courant afin de les appliquer à leur tour. Il y a des opérateurs pour le tri et le comptage des éléments, l'importation d'autres feuilles XSLT, la définition de variables et de paramètres, des *templates* nommés et paramétrés, etc.

Les conditions d'application des *templates* et les tests dans les instructions de branchement utilisent XPath, un langage qui permet de décrire un chemin dans une structure XML, d'exprimer des contraintes de sélection et de récupérer les valeurs des éléments. Un chemin est composé d'étapes comme :

```
/livre/introduction/paragraphe[position()=2].
```

Les chemins peuvent utiliser des conditions et le résultat est un ensemble de nœuds sélectionnés par ces contraintes. Le chemin et les conditions sont exprimés sur des axes qui orientent la navigation depuis un nœud vers un autre, par exemple : l'axe des nœuds ancêtre. Des fonctions sont utilisées pour construire les chemins de sélection et manipuler les valeurs.

Dans [171], un modèle formel d'un sous-ensemble de XSLT est analysé et les auteurs montrent que, du point de vue de la théorie des langages, l'expressivité de XSLT correspond à celle d'un transducteur à parcours d'arbre avec des registres et que son expressivité est meilleure qu'un certain nombre de langues de requête pour XML. En outre, XSLT bénéficie de deux moyens d'extension : une méthode d'extension pour le jeu d'éléments d'instructions utilisés dans les *templates* et une méthode d'extension pour l'ensemble des fonctions utilisées dans des expressions XPath. Pour ces raisons, et parce que XSLT fait partie de la famille XML, nous avons proposé de l'utiliser pour créer et déployer des agents de script simples.

Beaucoup de langages existent pour les agents mobiles et les scripts, allant de Java qui fournit simplement le chargement dynamique de classes, à l'une des plates-formes les plus anciennes et les plus connues : Telescript de General Magic Inc. Cependant, il serait hors-sujet de comparer les différentes contributions, nous nous contenterons de considérer ici que du fait que XML est un format d'échange universel de données, XSLT est en train de devenir un format d'échange universel pour la manipulation de

données ; XML est utilisé pour fournir un langage déclaratif pour la communication entre agents ; XSLT peut être utilisé pour fournir un langage de traitement indépendant de la plate-forme pour la communication entre agents.

Les agents, qu'ils soient réactifs ou délibératifs ont en commun d'être des entités artificielles clairement identifiables, situées dans un espace d'information (comme un réseau) dans lequel elles peuvent percevoir, réagir et agir. Les interactions chez les agents réactifs se résument habituellement à un simple signal modifiant l'environnement. Les agents délibératifs, eux, peuvent communiquer au niveau des connaissances [169]. Tous deux ont le contrôle de leur comportement. Le comportement d'un agent réactif est généralement simple et fondé sur des mécanismes de réflexe c'est-à-dire une réaction automatique à un stimulus. Alors que les agents délibératifs peuvent combiner des comportements plus complexes (systèmes à base de connaissances, BDI, techniques d'apprentissage automatique, etc.) qui sont en fin de compte, également composés d'une séquence de tâches simples. Sur la base de ces distinctions, nous allons envisager ici deux points de vue sur l'utilisation de XSLT dans les interactions d'un système multi-agent pour un système d'information exploitant XML. Dans les deux cas XSLT est utilisé pour propager des comportements simples de manipulation du XML.

La première option est d'utiliser des scripts XSLT pour dynamiquement adapter des rôles génériques d'agent d'information au moment de l'exécution. Certaines des actions atomiques composant les plans et les comportements de ces agents délibératifs peuvent être décrites par des modèles XSLT externes et échangés entre agents ce qui les rend facile à personnaliser et à maintenir afin de s'adapter aux utilisateurs et aux évolutions des ressources d'information. Les comportements des agents intelligents peuvent ainsi être conçus à un niveau assez générique en s'appuyant sur XSLT pour les réglages finaux du comportement. Cette perspective peut être considérée comme une approche holonique où le comportement global d'un agent délibératif d'information repose sur la composition intelligente de certains agents simples.

Techniquement parlant, la deuxième option est équivalente. Toutefois, d'un point de vue conceptuel, elle est plus proche de la vision en pyramide des espèces où les agents exploitent les résultats et même élèvent des agents d'autres couches pour maintenir l'écologie de l'infosphère. Comme le montre la Figure 92, XSLT propose un certain nombre de constructeurs intéressants pour décrire et propager des agents réactifs simples:

- Les capteurs sont fournis par les contraintes d'application d'un *template* ou les instructions de test, les deux utilisant des expressions XPath.
- Les effecteurs sont les instructions de manipulation de valeurs qui permettent à l'agent de créer l'arbre résultat.
- Les réactions sont codées à travers des appels récursifs et des instructions d'embranchement.

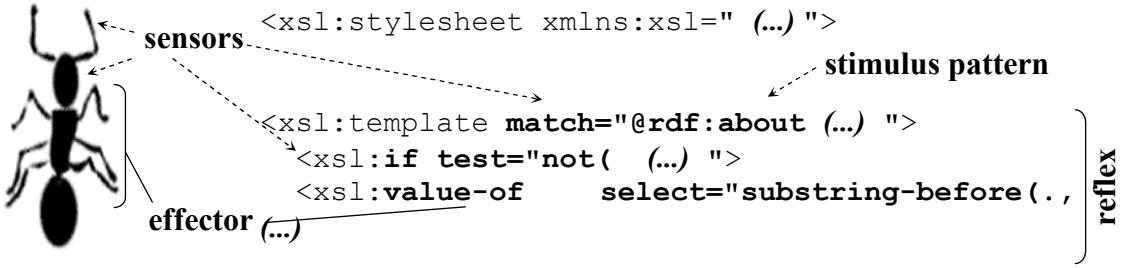


Figure 92. Anatomie d'un agent XSLT réactif

Dans notre plate-forme, le principe d'exécution de ces agents est que tout arbre XML qu'ils rencontrent est remplacé par la sortie de leur modèle. Par conséquent, la règle n ° 1 de ces agents est de respecter leur environnement en laissant intact les morceaux de XML qui ne les intéressent pas. Ainsi, tout agent réactif inclut au moins le *template* suivant qui, par défaut, s'applique à tout nœud et le recopie intact.

```
<xsl:template match="@*|node()">
  <xsl:copy>
    <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>
```

Figure 93. Règle minimale de recopie de l'information rencontrée

Si un agent ne contient que ce *template*, alors il passera sur les documents XML les laissant inchangés. Puisque qu'un moteur XSLT choisit toujours le plus précis des *templates*, le modèle par défaut de la règle n°1, assure que si un agent n'a rien de mieux à faire, il va au moins respecter son environnement d'information. Cette règle peut être ajoutée dans le corps des agents réactifs par les agents délibératifs, par exemple pour empêcher tout agent nocif intentionnel ou accidentel de propager une suppression. D'autres règles plus précises sont ensuite ajoutées pour adapter les réactions de cette population d'agents. Ces modèles sont générés par les agents d'information délibératifs pour propager des actions qui les intéressent (tâches de maintenance, mises à jour périodiques, etc.)

Des protocoles génériques sont utilisés dans les deux perspectives. Par exemple et dans le cas de la propagation d'extracteurs de pages comme en section 5.5 le protocole *FIPA-Request* peut être utilisé pour propager une population, créer un agent d'information paramétré par un ou plusieurs *templates*, déclencher des mises à jour, etc.

Dans les deux cas, nous pouvons associer des métadonnées à des *templates*, en utilisant RDF pour déclarer le type de l'agent, identifier l'expéditeur pour les contrôles de sécurité (le *template* peut être encodé en utilisant une clé privée correspondant à la clef publique de l'émetteur), fournir des paramètres à utiliser lors de l'appel de la feuille de style, gérer un TTL (*Time To Live*), etc.

La section 5.5 de ce chapitre a déjà donné des exemples de la première perspective où XSLT est utilisé pour paramétrer des comportements. Nous ne parlerons donc ici que d'un essai sur la deuxième perspective : la propagation de tâches de bases pour assurer des traitements bas niveaux dans l'écologie d'une infosphère. Nous donnerons deux exemples : l'entretien et la fermentation.

Entretien. Les inévitables problèmes posés par le cycle de vie des connaissances sont une source inépuisable d'exemples où la propagation de simples agents réactifs peut être utile. Ces problèmes exigent des tâches telles que : la correction d'annotations archivées, la maintenance de la cohérence des annotations après un changement dans l'ontologie, l'effacement d'annotations obsolètes, etc.

A titre d'exemple, considérons le cas où la mise à jour est due au changement de l'URI d'une ressource. La Figure 94 présente trois cadres : un exemple de l'annotation à rectifier ; un extrait du XSLT pour la correction ; l'annotation après correction. Ce cas assez simple illustre l'utilisation d'un type spécialisé d'agents réactifs, adaptés par un agent d'information pour propager une tâche sur l'ensemble de la mémoire. Une population de ces agents XSLT peut être générée automatiquement par un agent intelligent ou par un utilisateur doté d'une 'interface générique et de bibliothèques de *templates*.

```
(…)
 <CoMMA:WebPage rdf:about="http://www.inria.fr/acacia/ ">
  <CoMMA:Title>Web page of ACACIA</CoMMA:Title>
  <CoMMA:CreatedBy>
   <CoMMA:Person rdf:about="http://www.inria.fr/people#rdieng/" />
  </CoMMA:CreatedBy>
 </CoMMA:WebPage>
(…)
```

```
(…)
<xsl:template match="@rdf:about[contains(.,'www.inria.fr')]">
<xsl:attribute name="rdf:about">
<xsl:value-of select="substring-before(.,'www.inria.fr')"/>
www-sop.inria.fr
<xsl:value-of select="substring-after(.,'www.inria.fr')"/>
</xsl:attribute>
</xsl:template>
(…)
```

```
(…)
<CoMMA:WebPage rdf:about="http://www-sop.inria.fr/acacia/ ">
  <CoMMA:Title>Web page of ACACIA</CoMMA:Title>
  <CoMMA:CreatedBy>
   <CoMMA:Person rdf:about="http:// www-sop.inria.fr/people#rdieng/" />
  </CoMMA:CreatedBy>
 </CoMMA:WebPage>
(…)
```

Figure 94. Un agent XSLT propageant un changement d'URI

Fermentation. *Swarm intelligence* est la propriété d'une société où des comportements individuels peu sophistiqués interagissent localement avec leur environnement et provoquent l'émergence d'une activité fonctionnelle globale cohérente.

En utilisant plusieurs populations de ces agents et des mécanismes de stigmergie on peut obtenir des résultats complexes : une population permet de détecter et traiter un aspect et une autre détecte cet ajout et le combine avec d'autres aspects pour produire un autre résultat et ainsi de suite. Par exemple, pour assister les différentes étapes d'un processus de traitement d'informations une espèce peut être générée pour chaque étape et effectuer un lien dans cette chaîne de traitement de l'information. Dans nos paysages d'information riches en connaissances latentes, des essaims d'agents peuvent entretenir une fermentation des informations (analyse simple et petits prétraitements) pour maintenir la dérivation de structures de connaissances basiques mais exploitables ensuite par d'autres agents.

Ainsi, ce dernier exemple va illustrer comment des agents réactifs peuvent être utilisés pour favoriser l'émergence de simples chemins de consultation, au moyen d'une relation « voir-aussi » pour construire des chemins et des groupes de ressources. Etant donnée une nouvelle annotation d'un document, une nouveau script XSLT peut être généré pour propager des pointeurs vers cette ressource dans d'autres annotations. Ce script teste si une annotation porte sur une ressource qui a les mêmes mots-clés, auteurs, etc. que celle à partir duquel il a été produit. Cet agent réactif est ensuite propagé pour enrichir les annotations avec des suggestions « voir-aussi ». Lorsque l'utilisateur consulte une de ces annotations modifiées, le système peut afficher les suggestions associées et ainsi l'utilisateur peut commencer à suivre un chemin construit par le déploiement de plusieurs espèces de ces agents réactifs ; de cette façon le système peut proposer des voies de navigation ou des requêtes qui n'ont pas forcément été envisagées par l'utilisateur ou qui sont si usuelles que la fourniture de ces raccourcis en est naturelle. La même procédure peut être utilisée pour lier les utilisateurs et construire des réseaux d'accointances et aussi pour proposer des voies de routage dans les systèmes où la résolution de requête est basée sur la propagation de la requête ou des index distribués.

Pour ce dernier essai, nous avons utilisé les annotations extraites de PubMed[16] en utilisant les extracteurs présentés en section 5.5. PubMed est un service de la Bibliothèque nationale de médecine qui donne accès à plus de 12 millions de citations MEDLINE et autres revues états-uniennes des sciences de la vie. La base d'essai contient 9981 annotations extraites et le comportement testé était la pollinisation croisée d'annotations (appelée entre nous bee2bee ☺): un script XSLT est généré à partir d'une annotation sur un rapport et de sa liste des auteurs, puis pour chaque annotation visitée, il laisse une « phéromone » si elle partage un auteur avec son annotation de départ.

[16] http://www.ncbi.nlm.nih.gov/entrez/

```
<c:ResearchReport rdf:about="URL in visited annotation">
 (…)
<c:sameAuthorAs>
 <c:SameAuthorDoc c:nbSharedAuthors="nb shared authors"
                  rdf:about="url initial document" />
 </c:sameAuthorAs>
</c:ResearchReport>
```

Figure 95. Phéromone laissée par un agent de bee2bee

C'est une sorte de relation "voir aussi", qui pointe vers la première annotation et donne le nombre d'auteurs en commun qui est utilisé à la fois pour classer les pointeurs et pour limiter la liste des pointeurs vers les plus pertinents. Sur les 9981 annotations, 7724 liens `sameAuthorAs` ont été générés reliant ainsi 2728 rapports c'est-à-dire 27% de la base. La Figure 96 montre le nombre de rapports par rapport au nombre des auteurs qu'ils partagent en utilisant une échelle logarithmique. Comme c'est habituellement le cas en biologie, la liste des auteurs pour une publication peut être très long et, comme indiqué sur la figure, deux des rapports partagent effectivement 191 auteurs.

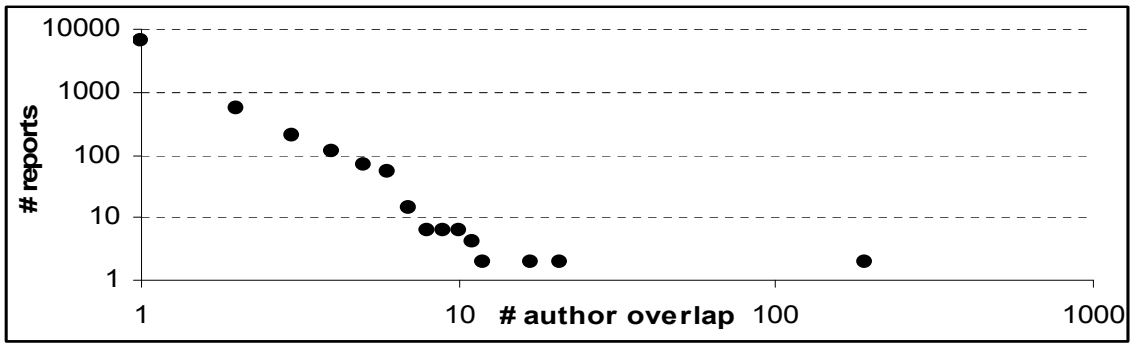


Figure 96. Nombre de rapports vs. Recouvrements trouvés entre les auteurs

Ce dernier exemple est plongé dans l'architecture de CoMMA et nous voyons un cycle complet impliquant plusieurs espèces d'agents : les *wrappers* sont des agents logiciels personnalisés pour extraire des annotations ; les agents de pollinisation sont des agents réactifs enrichissant les annotations ; les agents archivistes maintiennent les bases d'annotations et répondent aux requêtes autant que possible avec leurs ressources locales ; les médiateurs d'annotation gèrent le processus des requêtes distribuées entre les archivistes et assurent un résultat intégré. Ainsi, lorsqu'un utilisateur soumet une requête, sa résolution effective bénéficie du travail accompli par l'ensemble de la chaîne d'agents de l'infosphère.

Au-delà de l'utilisation de XSLT pour des agents réactifs dans des infosphères, ce que nous montrions dans cette contribution c'est l'intérêt et la nécessité de chaque type d'agent et l'idée d'aller au-delà d'architectures axées soit sur les agents délibératifs soit sur les agents réactifs. Nous promouvions la métaphore de l'écosystème avec des interactions riches entre les différentes espèces pour maintenir et exploiter le paysage d'information.

L'idée défendue est de compléter les nouvelles métaphores d'intelligence émergente et de sociétés d'agents délibératifs en réutilisant les mêmes chaînes écologiques qui se produisent dans la nature c'est-à-dire des espèces plus organisées ou intelligentes exploitant et même cultivant les résultats d'autres espèces (par exemple, les fourmis et les pucerons) et vice-versa.

Dans l'avenir, les agents intelligents pourraient utiliser des heuristiques, des règles ou des normes qui leur ont été explicitement données ou qu'ils ont obtenues, sur la façon d'utiliser l'intelligence en essaim pour atteindre leurs objectifs.

Ce chapitre a montré plusieurs exemples d'interactions entre les agents réactifs et les agents délibératifs qui ont été utilisées avec succès pour mettre en œuvre des applications du web sémantique.

5.8 Conclusions

Nous venons de voir qu'en utilisant les standards du web sémantique comme briques de base nous pouvons proposer des architectures distribuées qui dépassent largement le simple paradigme client-serveur.

Ce chapitre a résumé un certain nombre d'approches que nous avons proposées pour tenir compte et gérer la distribution des ressources distribuées dans les webs sémantiques que nous concevons.

Nous avons présenté des méthodes et formalisé des structures d'indexation d'annotations RDF pour gérer leur archivage et utiliser leur sémantique pour router les décompositions de requêtes lors de la résolution distribuée dans des architectures multi-agents ou multi-services.

Nous avons montré comment intégrer des sources externes dans ces architectures comme des bases d'archivage particulières produites par des mécanismes d'extraction spécifiés par les utilisateurs.

Nous avons aussi rappelé que les ressources distribuées ne se limitent pas à des documents et des annotations mais incluent aussi des services. Nous avons alors montré comment ces services distribués pouvaient être eux aussi annotés pour être par la suite identifiés, retrouvés, invoqués et composés entre eux et avec la mémoire.

En prenant du recul sur ces architectures, nous avons suggéré que les différents paradigmes actuels s'étalant du service web répondant aux sollicitations qu'il reçoit jusqu'à l'agent délibératif doté d'un comportement autonome et de buts personnels, sont en fait les maillons d'une chaîne écologique de gestion de l'information et que les solutions complètes de gestion de connaissances bénéficient de chacun de ces acteurs, mais aussi de leurs interactions.

5.9 Perspectives

Nos travaux actuels se concentrent sur les perspectives des index:

- il existe une redondance entre les index de chemins et les index d'étoiles notamment pour les tailles 1 et 2. Nous étudions une meilleure intégration des algorithmes d'indexation pour obtenir un index unique sans redondance.
- il existe une redondance entre les réponses à une sous-requête correspondant à un chemin (resp. étoile) et les réponses à une sous-requête correspondant aux chemins (resp. étoiles) de taille inférieure. Nous cherchons actuellement à combiner une génération plus concise des index et des contraintes supplémentaires dans les sous-requêtes de façon à éviter cette redondance.
- les valeurs littérales sont correctement traitées dans la résolution, mais ne participent pas à l'optimisation de l'émission des sous-requêtes car leurs particularités ne sont pas exploitées par les index au contraire de [21] et [129].
- il serait intéressant d'exploiter des résultats partiels déjà reçus pour contraindre les sous-requêtes restantes en envisageant une extension comme ARQtick.

Avec la thèse d'Adrien Basse, nous envisageons l'extension de l'hypothèse de la décomposition en étoiles et chemins des index pour considérer d'autres structures par exemple en cherchant à détecter les graphes minimaux des bases *i.e.* le graphe en étoile autour d'une ressource dont toutes les branches se terminent soit sur une ressource nommée soit sur un littéral soit sur une ressource anonyme sans autre relation ; voir, par exemple, les MSG (*Minimal self-contained graph* [172]).

Nous envisageons aussi l'utilisation de statistiques sur les motifs composant un index afin d'ordonner les priorités de routage entre les *hubs* par exemple en favorisant les requêtes les plus contraintes pour diminuer la combinatoire. Une autre extension consisterait à permettre des résultats paginés pour contrôler le flot des données sur le réseau. Enfin nous envisageons d'étendre l'infrastructure à des producteurs d'annotations dynamiques par exemple en incluant des *wrappers* de base de données correspondant à des *hubs* ayant un index fixé et spécialisé.

Ces *wrappers* nous ramènent aussi au problème de sources externes qui a maintenant trouvé des solutions dans les standards GRDDL [38] et RDFa [37] pour les sources structurées mais qui appelle toujours des travaux dans le cadre général notamment en couplant des techniques de l'analyse de la langue naturelle, du traitement du signal (images, vidéo, son) et de la fouille de données à des techniques de génération d'annotations pour automatiser l'indexation des ressources multimédias qui se multiplient.

Enfin, en ce qui concerne les services et les architectures distribuées en général, nous travaillons à la jonction entre des résultats de nos outils (ex : un *binding* SPARQL) et des standards d'exécution de *workflow* (par exemple en étendant BPEL4WS [148] avec des opérateurs SPARQL) afin que nos résultats soient réutilisables dans d'autres domaines.

Chapitre 5 : Récapitulatif de mes contributions personnelles.

J'ai été l'architecte principal du système multi-agents de CoMMA, de ces représentations, protocoles et inférences [21] et j'ai encadré les extensions [22] faites pendant le DEA et la thèse de Tuan-Dung Cao.

J'ai formalisé, spécifié et encadré le travail de Cheikh Niang sur les index de chemin et d'étoiles [24] dans le cadre du projet européen SevenPro.

J'ai spécifié et encadré le stage de perfectionnement post-doctoral de Moussa Lo sur l'utilisation de nos représentations pour la description, la recherche et la composition de services [158]. Ce travail a été présenté lors de d'un atelier du W3C qui a débouché sur la création du groupe ayant standardisé SAWSDL [157].

J'ai conçu, développé et testé le prototype de l'article où nous discutons de la notion d'infosphère [160] et de certains intérêts de la famille XML pour un tel monde.

6. Graphes RDF et interactions

Les ontologies et les représentations de connaissances sont souvent dans le cœur technique de nos architectures, notamment lorsqu'elles utilisent des représentations formelles.

Pour interagir avec le web sémantique et ses applications nous avons besoin d'interfaces qui les rendent intelligibles pour les utilisateurs finaux. Dans nos systèmes d'inférences, des éléments de connaissances sont manipulés et combinés, et même si les éléments de départ étaient intelligibles, l'intelligibilité des résultats, elle, n'est pas préservée au mieux par ces transformations.

Actuellement, et dans le meilleur des cas, les concepteurs d'interfaces mettent en œuvre des transformations *ad hoc* des structures de données internes en représentations d'interface en oubliant souvent les capacités de raisonnement que pourraient fournir ces représentations pour construire de telles interfaces. Dans le pire des cas, et encore trop souvent, les structures de représentation normalement internes sont directement mises à nu dans des *widgets* sans que cela soit justifié et, au lieu d'assister l'interaction, ces représentations alourdissent les interfaces.

Puisqu'elles reçoivent les contributions d'un monde ouvert, les interfaces du web sémantique devront être, au moins en partie, générées dynamiquement et rendues pour chaque structure devant rentrer en contact avec les utilisateurs. Il y a une opportunité croissante d'utiliser des systèmes à base d'ontologies dans l'assistance aux interactions avec les utilisateurs.

Dans ce chapitre, nous commençons par un exemple relativement simple qui introduit l'idée d'utiliser des représentations et raisonnements sur les connaissances pour des interfaces. Puis nous identifierons deux aspects du projet myCampus liés à l'utilisation de connaissances dans les interactions : le contrôle de la confidentialité et le respect de la vie privée ; l'utilisation des connaissances de contexte pour la simplification des interactions avec des services.

Dans la section suivante, nous nous intéresserons à un problème clef de ces interactions : la représentation visuelle d'une connaissance sur une ressource. Nous introduirons la notion de substitut qui, en recherche d'information, a deux sens pour rendre compte de deux besoins : la représentation de nos résultats en machine ; la présentation de nos résultats à l'utilisateur.

Dans les trois dernières parties nous aborderons la dimension sociale des interactions à travers une expérience de sémantisation d'un wiki et deux prises de position sur l'intérêt et les enjeux d'utiliser la représentation des connaissances dans des applications du web social.

6.1 Formalisation et (Re)présentations

Nous commençons cette section par un exemple relativement simple et limité mais qui introduit assez bien cette idée de l'utilisation des représentations et raisonnements sur les connaissances pour d'autres composants que le cœur algorithmique des systèmes d'information et notamment pour des interfaces.

Nous avons vu en section 4.4 comment l'ontologie était utilisable pour définir un espace métrique pour une ultra-métrique et la Figure 97 montre une interface affichant les clusters des arbres donnés en exemple section 4.4.

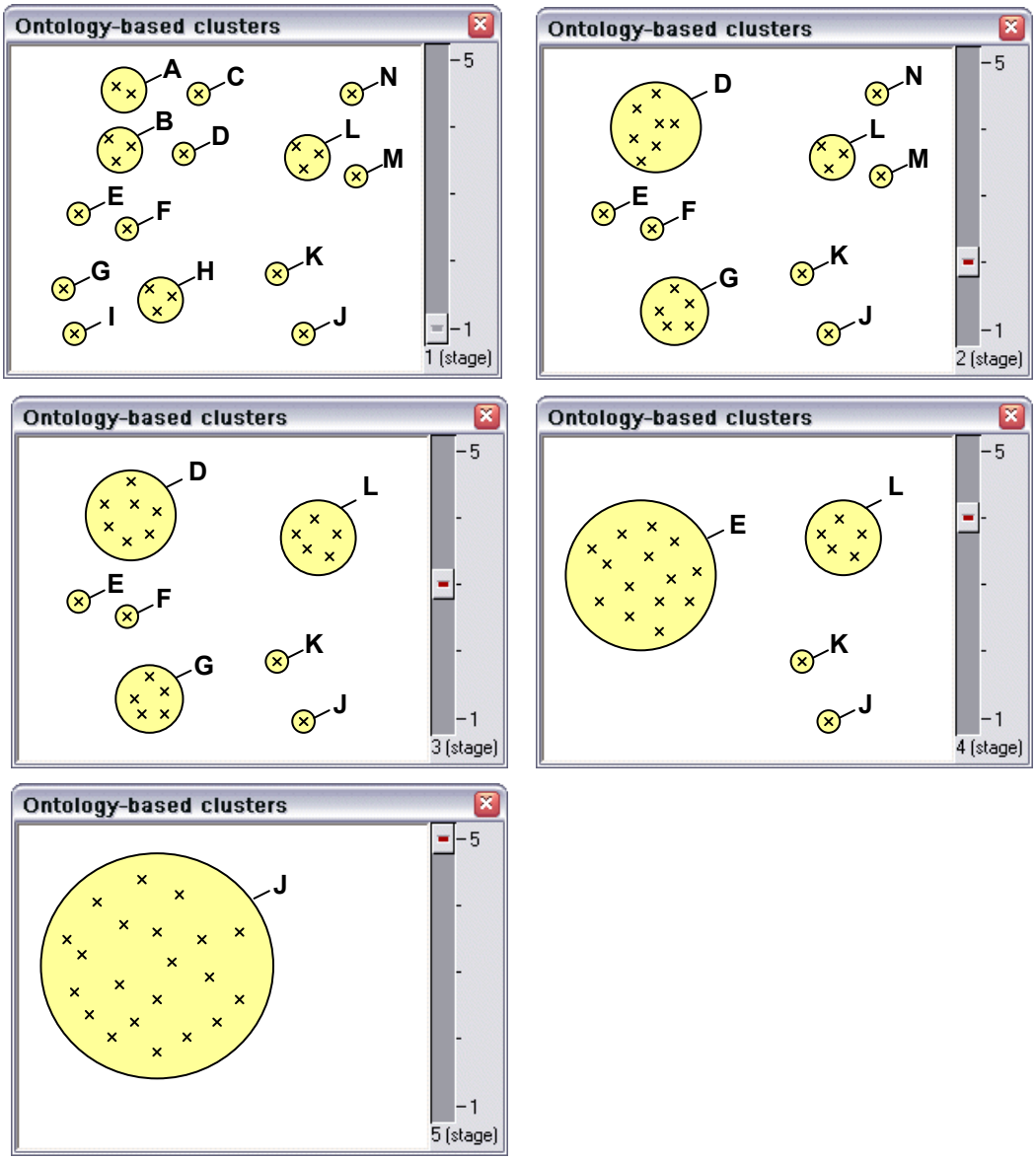


Figure 97. Clusters comme obtenus en section 4.4

Si l'on met en regard les représentations (Figure 98) auxquelles sont habitués les commanditaires du projet KmP (section 2.4.2) on voit immédiatement qu'il reste encore une étape importante à franchir : générer une présentation intelligible de ces représentations internes que nous avons construites.

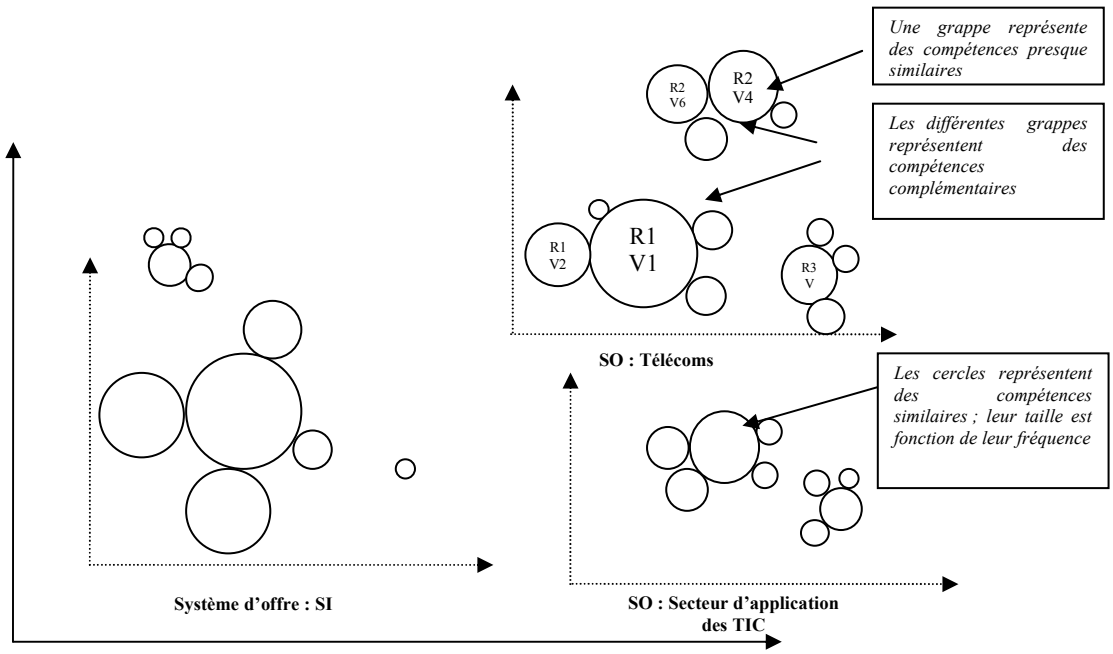


Figure 98. Représentation des clusters de compétences sur la Telecom Valley de Sophia Antipolis telle que souhaitée par les utilisateurs de KmP

Beaucoup de systèmes à base d'ontologies tombent dans deux pièges :

1. L'ontologie, qui est une représentation interne des connaissances, est directement intégrée à l'interface souvent sous la forme d'un arbre que l'on parcourt, déroule, etc. ; mais souvent rien ne justifie d'exposer ainsi l'utilisateur à la complexité et la complétude des ontologies. En tant qu'humain nous de mobilisons pas nos conceptualisations dans leur entièreté lorsque nous réalisons une tâche : nous nous focalisons et les interfaces doivent se focaliser avec nous.
2. L'ontologie et les raisonnements qu'elle permet restent enfouis dans les serveurs tiers de nos architectures et nous oublions qu'elles peuvent aussi être utiles pour opérationnaliser des rendus et des comportements d'interfaces.

Dans le cas de KmP nous avons intégré la représentation et le raisonnement à base d'ontologies à plusieurs étapes de la génération d'interfaces. Nous avons vu en section 4.4 que l'ontologie peut être utilisée pour proposer des niveaux de détails dans nos vues, par exemple en Figure 99 nous voyons une analyse de répartition projetée sur trois niveaux de détails différents.

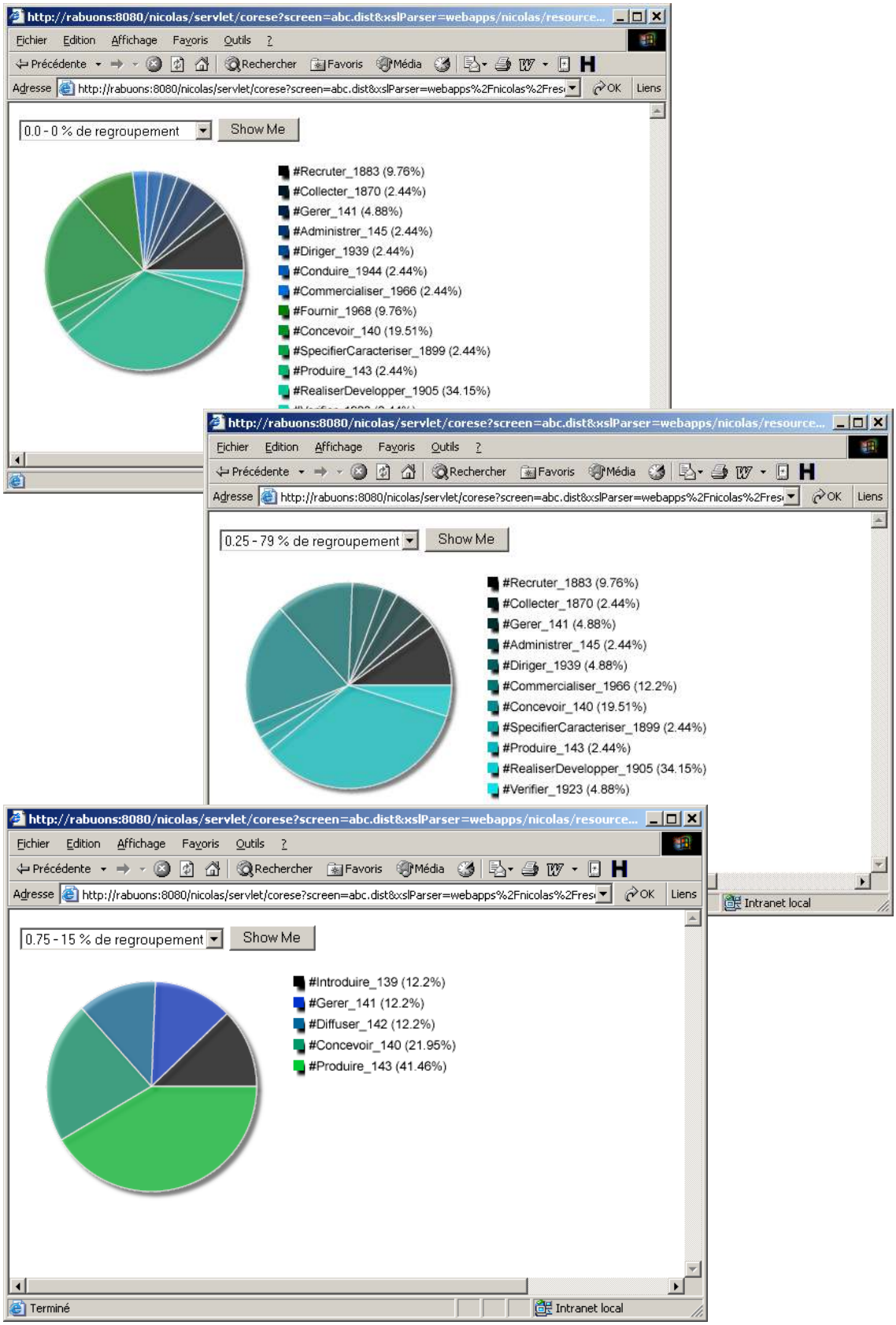


Figure 99. Analyse de répartition projetée sur trois niveaux de détails différents dans KmP

Le cas de la représentation demandée en Figure 98 a permis d'utiliser à nouveau l'ontologie comme un espace métrique, mais cette fois nous avons utilisé un système de coordonnées angulaires.

En effet, les utilisateurs sont intéressés par deux aspects dans la comparaison visuelle des clusters : la taille du cluster et le type de ressources utilisées par les compétences de ce cluster. Ils combinent ces deux dimensions pour obtenir ce qu'ils appellent une "vue radar" du pôle technologique comme en Figure 98. Le radar utilise une vue angulaire où l'angle est dérivé du typage par les classes de l'ontologie de la ressource utilisée dans le cluster et le rayon est dérivé de la taille du groupe. Deux compétences proches seront dans le même secteur angulaire. Deux compétences de même présence seront à même distance du centre.

La Figure 100 montre en vis-à-vis deux approches de génération de positions angulaires en utilisant l'ontologie:

- une approche en division descendante où les enfants partagent l'angle alloué à leurs parents (partie gauche de la figure),
- une approche ascendante où les feuilles se partagent équitablement l'angle maximal disponible et les parents obtiennent l'union des angles de leurs enfants (partie droite de la figure).

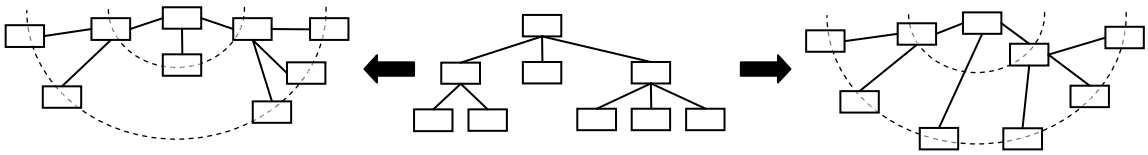


Figure 100. Méthodes ascendantes et descendantes d'attribution d'un angle à une classe de l'ontologie

Ici, l'angle initial était de 180 °. Sur la gauche (division ascendante), nous avons une première division en trois pour le premier niveau (60 ° chacun), puis le premier et le dernier tiers ont été respectivement divisés en deux (30 °) et trois (20 °) pour les deuxièmes descendants. Sur la droite (fusion ascendante), les 180 ° ont été répartis en 6 pour les feuilles de l'arbre (30 ° chacun), puis les nœuds du premier niveau sont respectivement affectés à des angles de 60 ° et 90 °.

La division descendante maintient l'égalité entre les frères et favorise la structure haute de l'ontologie, tandis que fusion ascendante maintient l'égalité entre les feuilles et favorise les branches détaillées.

Dans notre cas, la division descendante est plus intéressante car elle divise également l'espace entre les principaux domaines des ressources et, comme le montre la Figure 101, l'ontologie est beaucoup plus détaillée dans certaines régions (par exemple, les ressources informatiques) que dans d'autres (par exemple, les ressources de gestion). Sur une telle ontologie, la fusion ascendante biaiserait le point de vue tandis que l'approche descendant sur 360 ° en Figure 101 maintient un angle égal pour les frères, ce qui est utilisé pour avoir une position égalitaire des clusters d'entreprises s'appuyant sur leurs ressources représentatives.

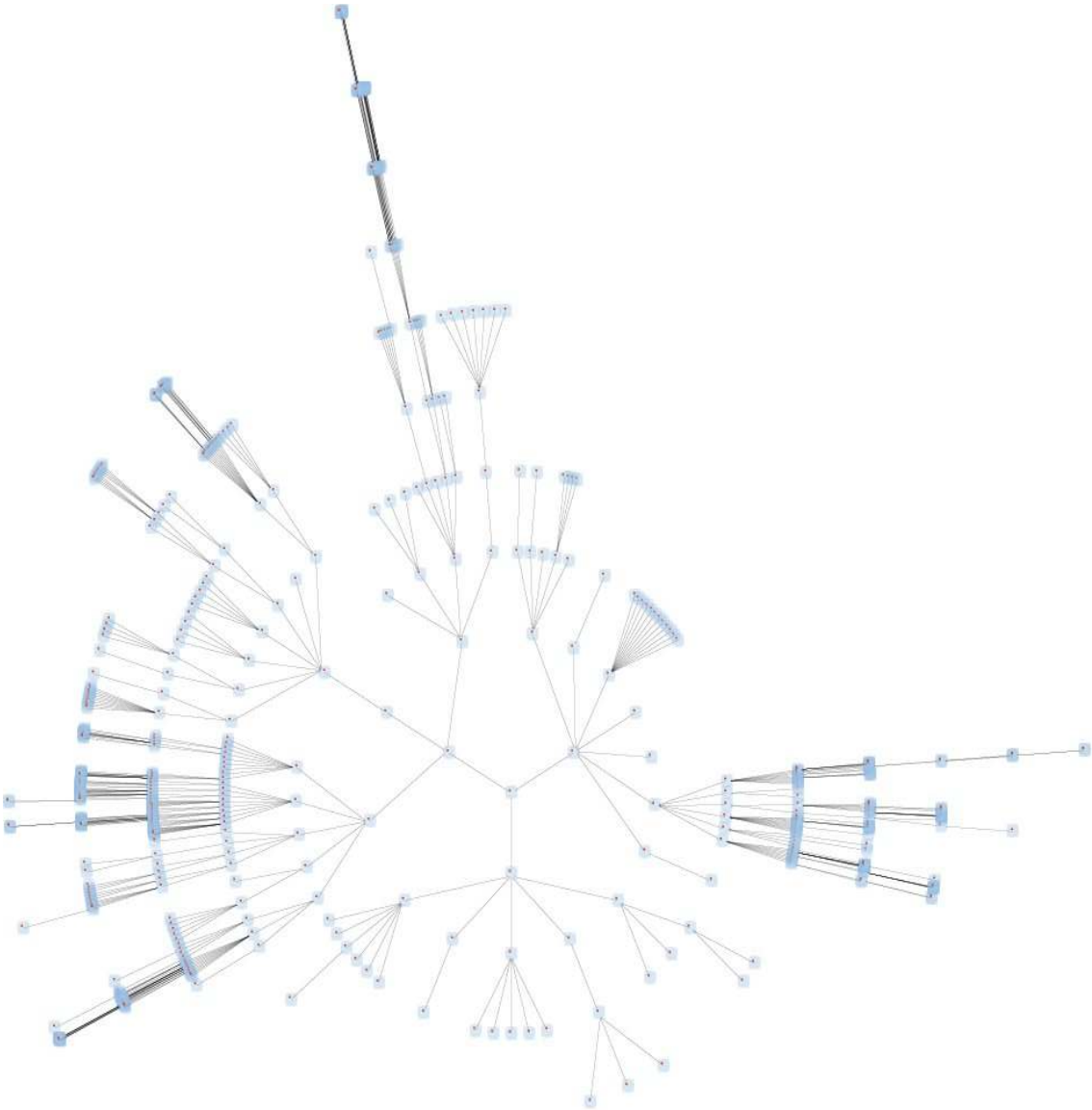


Figure 101. Division descendante de l'ontologie de KmP appliquée sur 360°

La Figure 102 montre le résultat de cette approche appliquée à la liste des clusters obtenus avec l'inférence précédemment décrite en section 4.4 : la position angulaire est donnée par la place de la ressource représentative de la compétence dans l'ontologie, et le rayon correspond à la taille du cluster. Nous pouvons voir que l'activité du pôle technologique est principalement axée dans un secteur Réseau et Logiciel (Figure 103).

Ce premier exemple, bien que simple, illustre bien l'esprit de ce chapitre qui est de s'intéresser à l'utilisation des systèmes à base d'ontologies dans les interactions avec les utilisateurs.

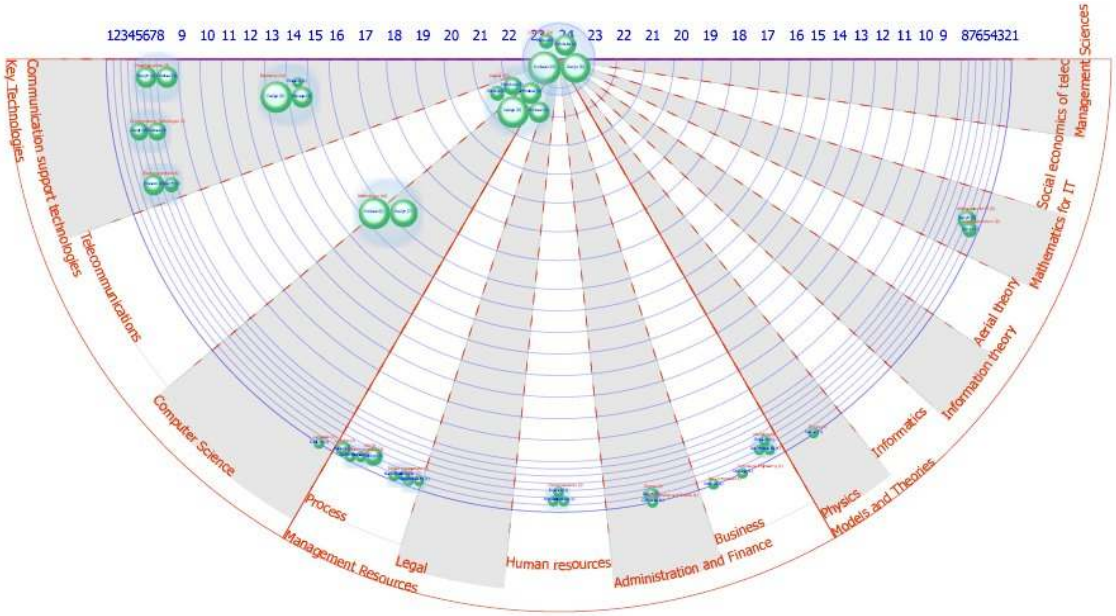


Figure 102. Placement des clusters selon leur angle ontologique et leur cardinal

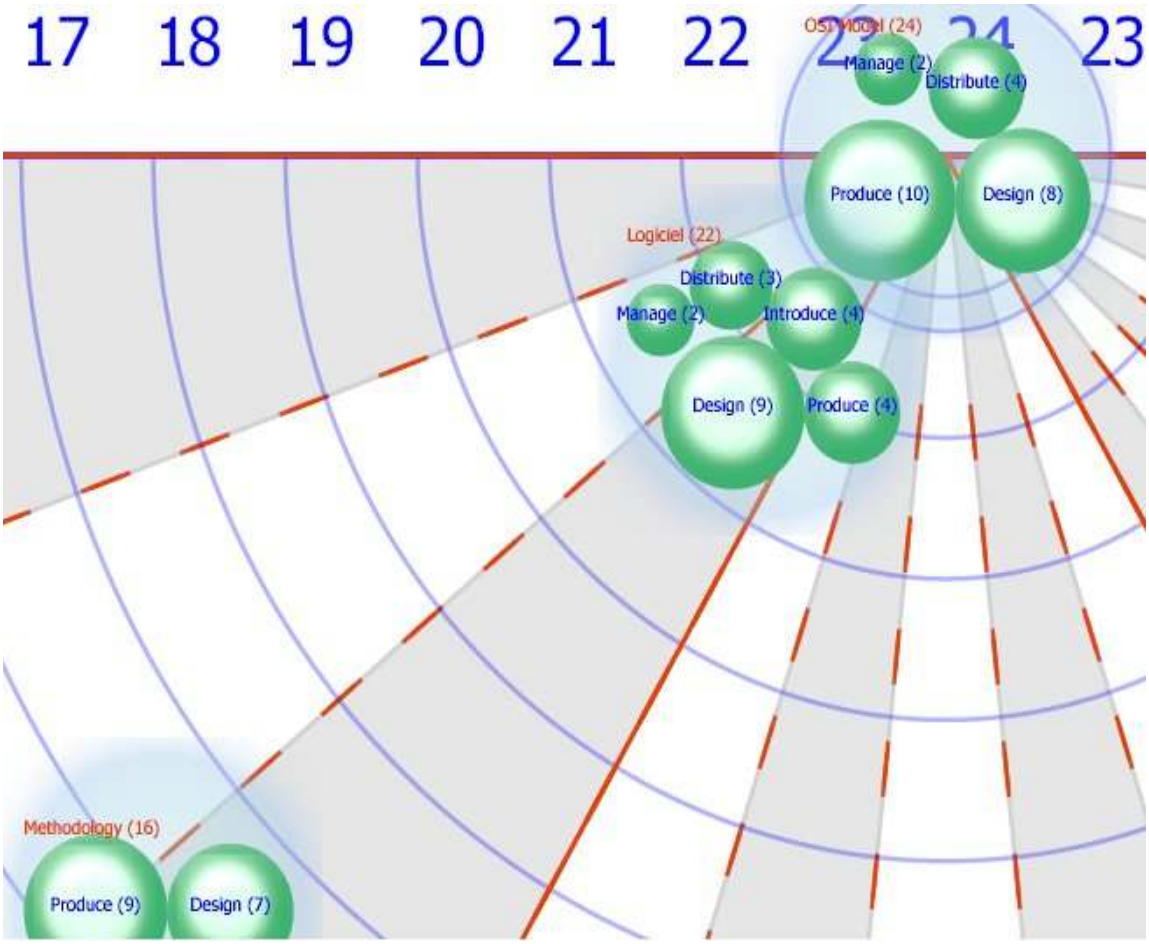


Figure 103. Détail du cluster de compétences central à la Telecom Valley de Sophia Antipolis

6.2 Informations personnelles et vie privée

Beaucoup de personnes parlent de ces futurs réseaux où les connaissances formalisées nous permettront d'identifier et d'accéder aux informations et aux services de façons beaucoup plus efficaces. Mais qui accepterait de se confier à de tels réseaux et services si les mécanismes de sécurité et de confidentialité ne suivent pas la même évolution ? Qui utiliserait un web sémantique qui puisse être employé et abusé, par exemple pour du marketing non sollicité ou autre profilage indiscret ?

Peu de personnes parlent de futurs réseaux où des connaissances formalisées permettront aussi de cacher ou de restreindre l'accès à ces services et à ces informations. Pourtant, cela doit être abordé rapidement, ou les webs sémantiques et leurs services risquent de ne jamais recevoir la confiance dont ils ont besoin pour fonctionner à pleine puissance.

Dans les articles [12] [13] [14] [15] nous montrons comment une approche ontologique, améliore la gestion des connaissances privées et de la confidentialité. La modélisation orientée ontologie permet des inférences pour contrôler l'accès et ajuster la précision des réponses, voire même mentir lorsque cela s'avère moins dangereux que de refuser de répondre. Pour cela, nous détaillerons l'approche suivit dans le projet *my*Campus et en particulier son composant central : le *e-Wallet*.

Le *e-Wallet* du projet *myCampus* offre un accès sécurisé aux ressources personnelles d'un utilisateur, c'est donc un élément central de notre architecture web sémantique pour la prise en compte du contexte et le respect de la vie privée. Il représente *une interface sémantique unifiée et sécurisée pour les ressources personnelles d'un utilisateur* et permet ainsi aux agents-services de les mobiliser. La connaissance d'un utilisateur gérée dans son *e-Wallet* peut se diviser en quatre catégories :

- *La connaissance statique*. Cette connaissance inclut typiquement des connaissances indépendantes du contexte (ex : nom, courriel, *etc.*).
- *La connaissance dynamique*. C'est la connaissance sensible au contexte de l'utilisateur (ex : en conduisant, je ne veux pas recevoir de messages).
- *Les règles d'invocation de services*. Ces règles permettent d'intégrer au *e-Wallet* des ressources d'information externes, personnelles (ex : agenda) ou publiques (ex : météorologie). Elles ajoutent au *e-Wallet* les capacités d'un annuaire sémantique de services qui peuvent être dès lors automatiquement identifiés et invoqués pour traiter des requêtes. Dans un premier temps, chaque ressource personnelle est transformée en un service web sémantique (ex : une règle peut indiquer qu'une question au sujet de l'activité courante de l'utilisateur peut être répondue en invoquant préalablement le service web correspondant à son agenda Outlook). Dans un deuxième temps, des règles d'invocation de services relient les types de connaissances contextuelles et des services web disponibles pour obtenir ces connaissances. Plusieurs règles peuvent fournir la même information, par exemple si le portable de l'utilisateur est allumé et est sur le réseau sans-fil,

c'est une façon de le localiser sinon, son agenda peut suggérer une réponse (ex : une salle de réunion). Enfin, pour répondre à des questions sur l'utilisateur, des services publics peuvent également être nécessaires. Par exemple, une question comme "l'utilisateur est-il dans un endroit ensoleillé" exigera typiquement la localisation de l'utilisateur et l'accès à un service public de météorologie.

- *Préférences de confidentialité*. Ces préférences explicitent quelles informations l'utilisateur est disposé à révéler, à qui et sous quelles conditions. Ces préférences se divisent en deux catégories :
    - *règles de contrôle d'accès* : ces règles disent qui peut voir quelle information et sous quelles conditions (ex : "ma localisation est accessible à mes collègues seulement pendant les jours de travail entre 8H et 20H")
    - *règles de révision* : souvent les préférences de confidentialité d'un utilisateur ne sont pas booléennes mais impliquent différents niveaux d'exactitude ou d'inexactitude. Une règle de *révision par abstraction* permet de contrôler le niveau de détails d'une réponse en fonction du contexte (ex : indiquer que vous êtes en ville sans donner le lieu exact). Une règle de *révision par falsification* est utilisée dans des scénarios où l'utilisateur ne veut pas que l'on sache qu'il cache une information et préfère fournir une réponse fausse (ex : un utilisateur peut ne pas vouloir indiquer son véritable courriel à un service par crainte de recevoir des publicités non sollicitées).

Tous les types de connaissances présentés ci-dessus (règles y comprises) sont représentés en OWL. Ils requièrent un certain nombre d'ontologies appropriées, par exemple : ontologies des attributs contextuels, des ressources personnelles, ainsi que pour des connaissances de domaine comme les types de cuisines et les préférences culinaires ou les types de messages et les préférences de filtrage

6.2.1 Un exemple de scénario d'interrogation du *e-Wallet*

Avant de descendre dans les détails techniques du *e-Wallet*, un scénario nous aidera à exemplifier les étapes principales du traitement d'une requête. Imaginons qu'une question soit soumise par un service invoqué par un utilisateur (Norman) au *e-Wallet* d'un autre utilisateur (Fabien) requérant l'endroit où se trouve actuellement ce dernier. Les étapes principales du traitement d'une requête sont les suivantes :

Déclarer le contexte de la requête : dans un premier temps, un maximum de faits décrivant le contexte de la requête sont affirmés *i.e.* ils sont chargés dans le moteur d'inférence du *e-Wallet* pour être éventuellement utilisés par les inférences qui traiteront la requête. Dans notre scénario, un exemple est que "l'expéditeur de la requête est Norman" ou que "la requête est arrivée à 15H34".

Déclarer les besoins élémentaires en information et les autorisations nécessaires : ici la requête est scindée en (a) une conjonction de besoins élémentaires en information, ex : "l'endroit où se trouve Fabien" ; et (b) la nécessité d'obtenir une autorisation d'accès

pour chacun de ces besoins en information. Le processus d'autorisation est ensuite effectué en deux temps aux étapes 3 et 6 de la Figure 104.

Pré-vérification des autorisations : un premier contrôle est effectué pour voir si la question est acceptable dans la limite des connaissances disponibles à ce stade. Dans notre exemple, le *e-Wallet* cherche à vérifier que Norman peut demander à localiser Fabien. Le *e-Wallet* de Fabien inclut une règle de confidentialité indiquant que ses collègues de travail peuvent connaître le bâtiment où il se trouve, lorsqu'il est sur le campus. Le *e-Wallet* cherche alors à prouver que Norman est effectivement un collègue de Fabien en utilisant une base de connaissances sur l'organisation. S'il prouve le contraire, l'*e-Wallet* rejette la question. Dans notre exemple Fabien est un collègue de Norman et la procédure continue car à ce stade, et parce qu'il n'a pas encore déterminé si Fabien est sur le campus, le *e-Wallet* n'a aucune raison de refuser la requête.

Faire appel à la base de connaissances locale du e-Wallet : certaines requêtes utilisent des faits de la base de connaissance locale au *e-Wallet*, qui contient des connaissances statiques (nom, courriel, *etc.*) et des connaissances sensibles au contexte ("je ne veux pas de messages quand je conduis"). Dans notre exemple particulier, une telle connaissance n'est pas utile.

Faire appel à des services personnels ou publics : quand la connaissance locale n'est pas suffisante pour répondre à une question, le *e-Wallet* se tourne vers ses règles d'invocation de services pour identifier les ressources externes qui pourraient l'aider à répondre. Ceci peut impliquer d'accéder à une ou plusieurs ressources personnelles de l'utilisateur (ex : son agenda) et/ou à un ou plusieurs services publics auxquels il puisse faire confiance. Dans notre exemple, le campus où travail Fabien a un réseau sans-fil qui permet la localisation des appareils connectés. Cette fonctionnalité est invoquée par le *e-Wallet* pour obtenir la localisation de Fabien et ajouter ce nouveau fait à sa base de connaissances.

Post-vérification des autorisations : avec cette nouvelle connaissance, le *e-Wallet* peut maintenant finir de vérifier que la question est autorisable. Dans notre exemple, on permet aux collègues de Fabien de voir sa localisation uniquement lorsqu'il il est sur le campus. En supposant que Fabien soit sur le campus, la demande est maintenant considérée comme permise.

Application des règles de révision : la localisation par réseau sans-fil peut avoir renvoyé la salle exacte où se trouve Fabien, cependant, comme mentionné en 3, Fabien n'est disposé à révéler que le bâtiment où il se trouve. Cette dernière condition est capturée par le *e-Wallet* sous la forme d'une règle de révision qui n'autorise que la connaissance du bâtiment à être utilisée pour la résolution de la question. L'application de cette règle implique typiquement d'accéder à des ontologies décrivant les concepts de salles et de bâtiments ainsi que des annotations au sujet du campus où Fabien travaille.

Réponse : la question ayant été traitée, une réponse est produite ("Fabien est dans le bâtiment Borel") et envoyée au service invoqué par Norman.

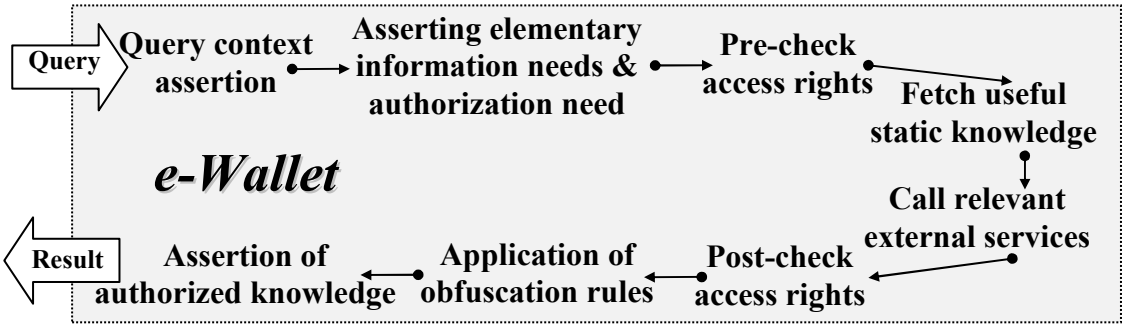


Figure 104. Etapes de traitement dans le *e-Wallet*

6.2.2 Architecture interne du *e-Wallet*

Nous avons conçu et implanté une architecture en trois couches (Figure 105) :

La *couche noyau* inclut un méta-modèle de OWL et maintient la connaissance statique (indépendante du contexte) et les préférences dynamiques (dépendantes du contexte). Ces connaissances sont obtenues en chargeant des descriptions disponibles au sujet de l'utilisateur ainsi que les ontologies appropriées et en appliquant un algorithme de chaînage-avant pour compléter la base et éviter de devoir inférer les mêmes faits à plusieurs reprises. Dans cette couche, la connaissance est représentée en utilisant le modèle de triplets RDF classique.

La *couche service* complète la connaissance du noyau avec des règles d'invocation de services qui décrivent une correspondance entre des types de connaissances et des services externes permettant de les obtenir. Celles-ci sont représentées par des règles de chaînage-arrière qui ne sont donc invoquées qu'en cas de nécessité. La connaissance dans cette couche est représentée en utilisant un type de triplets particulier appelé "triplet de service". Les triplets résidents dans la couche de service peuvent résulter de la migration d'un triplet de la couche noyau ou de l'activation d'une règle d'invocation. La migration se fait par chaînage-arrière.

Dans la couche externe ou *couche de confidentialité*, les faits sont représentés par un autre type de triplets appelé "triplet autorisé". Lors de la réception d'une requête, le *e-Wallet* la décompose en un certain nombre de "besoins en triplets autorisés" *i.e.* seuls des triplets autorisés peuvent être utilisés pour répondre aux questions. Ces triplets sont produits par les règles d'autorisation et de révision à partir des triplets de service : les règles de confidentialité sont appliquées en chaînage-arrière et mettent en correspondance des besoins de triplets autorisés et des besoins de triplets de service ; une fois produits (par accès aux connaissances internes ou aux services externes), ces triplets sont révisés avant d'être recopiés dans la couche autorisée.

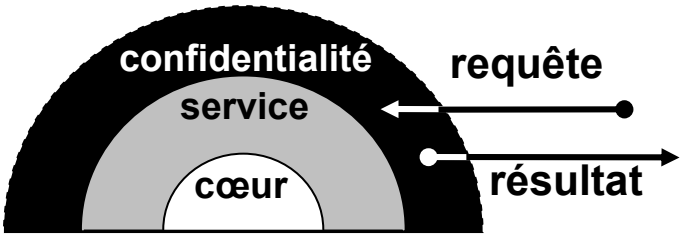


Figure 105. Architecture en couches du *e-Wallet*

En résumé, à travers un mécanisme de chaînage-arrière, les "besoins en triplets autorisés" d'une question engendrent des "besoins en triplets de service et en triplets du noyau", résultant soit (a) en la génération des triplets autorisés qui peuvent être retournés en réponse à la question soit (b) en la levée d'une exception, si la question est interdite. A travers ces trois types de triplets, la sécurité repose directement sur les mécanismes de typage du langage d'implantation.

6.2.3 Implantation du *e-Wallet*

L'implantation du *e-Wallet* repose sur JESS [61], un moteur de règles en chaînage-avant et son langage CLIPS. Ce moteur permet aussi le chaînage-arrière en réifiant les "besoins en faits" comme des faits eux-mêmes, qui peuvent à leur tour déclencher des règles en chaînage-avant chargées de répondre à ces besoins.

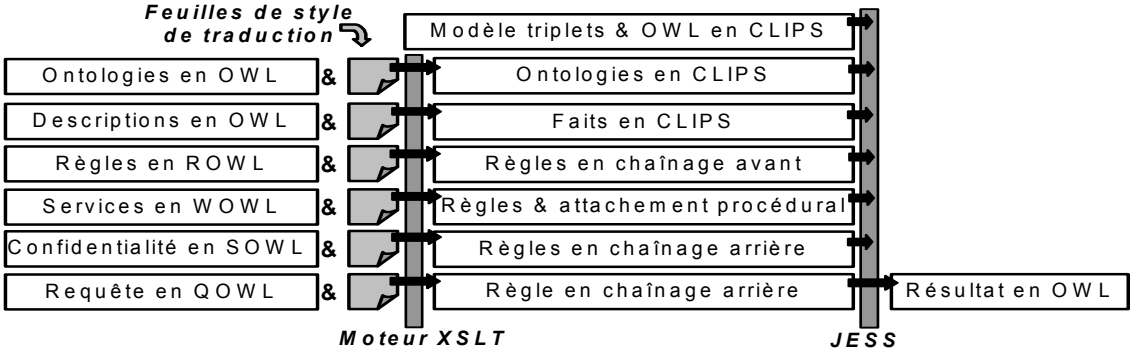


Figure 106. Architecture du *e-Wallet* pour la traduction et traitement.

Comme le montre la Figure 106, la base de connaissances du *e-Wallet* est initialisée avec : (a) un modèle des triplets [40], (b) un modèle des triplets spécialisés utilisés pour créer les couches du *e-Wallet* et les règles de migration entre la couche noyau et la couche service, et (c) un méta-modèle de OWL. Dans un deuxième temps, les connaissances additionnelles sont chargées en traduisant des descriptions OWL en des faits et règles CLIPS pour JESS. Les descriptions en OWL incluent des ontologies et leurs instances qui sont transformées en triplets classiques, des règles en chaînage-avant utilisées pour compléter la base (ex : définitions, préférences dynamiques) ainsi que des règles d'invocation de services et des règles de confidentialité en chaînage-arrière. Les langages ROWL, WOWL, SOWL et QOWL sont des extensions de OWL pour décrire respectivement, des règles de déductions de faits nouveaux, des règles d'invocation de services, des règles de confidentialité et des requêtes. Toutes les traductions sont des feuilles de styles [170]. Le système a été conçu pour rester efficace : la résolution des

requêtes s'interrompt dès qu'une violation d'autorisation peut être prouvée ; la couche service s'assure que la connaissance n'est pas disponible dans le noyau avant d'appeler un service ; *etc*.

Pour représenter nos règles, nous avons proposé et implanté un langage basé sur RDF-S/OWL pour décrire des clauses de Horn avec variables. Comparé à RuleML [173], nous n'avons pas voulu réifier les relations et les rôles de leurs arguments. Les relations étant toujours binaires en RDF, nous sommes restés sur le modèle des triplets et la syntaxe XML de RDF avec sa syntaxe de typage classique. Les seules extensions sont de nouvelles balises pour décrire les règles et un espace de nommage (*namespace*) spécial pour identifier des variables.

Le modèle des triplets RDF est défini comme un modèle de faits non ordonnés et utilisé dans des règles de chaînage-avant. Le méta-modèle de OWL est alors décrit par une liste de faits non ordonnés instanciant ce modèle de faits non ordonnés et la sémantique des propriétés du modèle est traduite par des règles [14]. Nous nous sommes limités aux aspects de OWL-Lite utiles à nos scénarios d'application. Le code et les résultats obtenus sur la base de tests officielle sont disponibles en ligne[17]. La Figure 107 donne en exemple un triplet typant la relation d'équivalence entre deux propriétés comme étant une relation symétrique (si $p_1 \Leftrightarrow p_2$ alors $p_2 \Leftrightarrow p_1$) et une règle décrivant sa sémantique (si $p_1 \Leftrightarrow p_2$ et $p_1(s,o)$ alors $p_2(s,o)$).

```
(triple   (predicate "http://www.w3.org/1999/02/22-rdf-syntax-ns#type")
          (subject   "http://www.w3.org/2002/07/owl#equivalentProperty")
          (object    "http://www.w3.org/2002/07/owl#SymmetricProperty") )

(defrule equivalent-property (declare (salience 100))
   (triple  (predicate "http://www.w3.org/2002/07/owl#equivalentProperty")
            (subject   ?p1)
            (object    ?p2))
   (triple (predicate p1?) (subject ?s) (object ?o))
   =>
   (assert (triple (predicate p2?) (subject ?s) (object ?o))) )
```

Figure 107. Déclarer en CLIPS la symétrie des relations et sa sémantique en OWL

Les triplets issus des ontologies et des descriptions sont aussi représentés par des faits non ordonnés. Des règles du domaine en ROWL sont utilisées pour la complétion de la base. Une règle définit par exemple les collègues comme des membres de la même équipe. Cette règle permet de représenter et d'interpréter des préférences sensibles au contexte telles que "mes collègues peuvent voir ma localisation quand je suis au travail". Le moteur d'inférence est utilisé pour compléter la base en appliquant toutes

[17] http://mycampus.sadehlab.cs.cmu.edu/public_pages/OWLEngine.html

ces règles et le résultat est gardé en mémoire, fournissant un point de repli et évitant ainsi de répéter cette complétion à chaque requête.

La Figure 108 montre une règle d'invocation de services permettant d'accéder à Pocket Outlook sur l'agenda électronique de l'utilisateur. Ces règles peuvent avoir jusqu'à trois sous-parties : la balise *output* décrit la connaissance que ce service peut produire ; la balise *precondition* décrit la connaissance requise pour appeler le service ; la balise *call* décrit comment faire appel au service web et avec quels paramètres.

```
<wowl:ServiceRule>
  <wowl:output>
    <mc:Person rdf:ID="&variable;#person">
      <mc:has_activity rdf:resource="&variable;#activity" /> </mc:Person>
  </wowl:output>
  <wowl:precondition>
    <mc:Person rdf:ID="&variable;#owner">
      <mc:PDA_endpoint>&variable;#endpoint</mc:PDA_endpoint> </mc:Person>
  </wowl:precondition>
  <wowl:call>
   <wowl:Service wowl:name="call-web-service">
     <wowl:qname>http://mycampus/PDAService#</wowl:qname>
     <wowl:endpoint>&variable;#endpoint</wowl:endpoint>
     <wowl:method>GetCurrentWeekAppointments</wowl:method>
     <wowl:user_id>&variable;#owner</wowl:user_id>
   </wowl:Service>
  </wowl:call>
</wowl:ServiceRule>
```

Figure 108. Règle d'invocation d'un service pour accéder à l'agenda de l'utilisateur.

La Figure 109, montre une règle de confidentialité donnant l'accès en lecture à la localisation de l'utilisateur mais limitant la précision de la réponse à la présence ou l'absence sur le campus. Ces règles peuvent avoir jusqu'à trois sous-parties : la balise *target* décrit le type de connaissances auquel cette règle s'applique ; la balise *check* décrit les conditions requises pour accorder l'accès ; la balise *revision* décrit les révisions à apporter avant que la connaissance ne soit recopiée dans la couche autorisée. Un système équivalent a été implanté pour les accès en écriture.

```
<sowl:ReadAccessRule>
  <sowl:target>
    <mc:Person rdf:about="&variable;#owner">
      <mc:location rdf:resource="&variable;#location"/> </mc:Person>
  </sowl:target>
  <sowl:check>
   <rowl:And>
     <rowl:condition>
       <mc:E-Wallet>
         <mc:owner rdf:resource="&variable;#owner"/> </mc:E-Wallet>
     </rowl:condition>
     <rowl:condition>
       <mc:Place rdf:about="http://www.cmu.edu">
          <mc:include rdf:resource="&variable;#location" /> </mc:Place>
     </rowl:condition>
     <rowl:not-condition>
       <qowl:Query>
         <qowl:sender rdf:resource="&variable;#owner" /> </qowl:Query>
     </rowl:not-condition>
   </rowl:And>
  </sowl:check>
  <sowl:revision>
    <mc:Person rdf:about="&variable;#owner">
      <mc:location rdf:resource="http://www.cmu.edu"/> </mc:Person>
  </sowl:revision>
</sowl:ReadAccessRule>
```

Figure 109. Règle de confidentialité limitant la localisation à la présence sur le campus.

6.2.4 Synthèse et discussion

Cette section a résumé une architecture web sémantique permettant l'accès à des connaissances personnelles et contextuelles tout en respectant la confidentialité. L'élément central présenté ici est le *e-Wallet* permettant la découverte et l'accès automatisés aux ressources personnelles d'un utilisateur et le respect des règles de confidentialité. Les ressources personnelles et publiques sont implantées sous la forme de services web. Des règles d'invocation de services basées sur des ontologies permettent au *e-Wallet* d'identifier dynamiquement les ressources susceptibles de l'aider à répondre à une requête. De plus, le *e-Wallet* maintient et utilise des préférences de confidentialité et des règles de révision permettant d'ajuster l'exactitude ou l'inexactitude des réponses fournies et ceci selon le contexte de chaque requête. Nous avons décrit l'architecture en trois couches du *e-Wallet* et son implantation reposant sur JESS, OWL-Lite et XSLT. Nous avons évoqué les évaluations menées sur le campus de Carnegie Mellon et le système a subi, durant le travail reporté ici, deux itérations de son cycle de développement par prototypes.

Les mécanismes de sécurité et de confidentialité courants ignorent la richesse des descriptions des ressources que propose la vision d'un web Sémantique. La communauté de la représentation des connaissances a su montrer comment de nouvelles inférences de recherche, de classification, de composition, *etc.* pouvaient utiliser ces connaissances pour rendre les outils plus intelligents dans leur manipulation du web. De la même façon, il est important de voir que d'autres tâches, telles que la sécurité et la confidentialité, peuvent elles aussi bénéficier d'inférences basées sur des ontologies et que cela est même nécessaire à la viabilité d'un paysage d'informations aussi riche dans ses représentations et dynamique dans ses évolutions.

On peut craindre des systèmes exploitant la connaissance du contexte qu'ils deviennent des systèmes de surveillance des individus. C'est pourquoi notre approche distribuée, basée sur les *e-Wallets*, impose au cœur même de l'architecture que le contrôle des accès aux ressources personnelles reste lui-même entre les mains de leurs propriétaires. Il ne s'agit pas ici de mettre en ligne toutes les ressources personnelles d'un utilisateur ; il s'agit simplement de permettre aux utilisateurs d'ouvrir un accès aux ressources nécessaires à des services qui leurs sont utiles et ceci en leur permettant de contrôler cet accès au même niveau de modélisation que les échanges effectués *i.e.* le niveau des connaissances et leur représentation orientée ontologie. C'est, à notre avis, une condition *sine qua non* d'une automatisation de l'accès et de la composition de services à l'échelle du World Wide Web.

Dans cette section, nous avons montré comment une approche ontologique, peut permettre d'améliorer les systèmes de sécurité et de gestion de la confidentialité. De nouvelles inférences ont été intégrées aux mécanismes de sécurité, pour contrôler l'accès en utilisant des règles très expressives et ajuster la précision des réponses, voire même ‘mentir’ lorsque cela s'avère moins dangereux que de refuser de répondre.

Nous avons clairement identifié les avancées du *e-Wallet* par rapport aux travaux existants en montrant ses innovations : une intégration en temps réel de sources d'informations personnelles et publiques grâce à leurs descriptions sémantiques ; une spécification des accès aux ressources personnelles pour chaque service ; un contrôle de l'exactitude et de la précision des réponses exploitant pleinement l'expressivité des ontologies mobilisées.

Ce nouveau concept de *e-Wallet* pose cependant un certain nombre de questions : Comment contrôler les répercussions et la diffusion des révisions apportées aux réponses ? Comment assurer la cohérence des réponses après recoupement ? *Etc.* Enfin, un autre défi demeure, héritage de toutes les approches utilisant des représentations riches : réconcilier l'expressivité des langages avec les exigences ergonomiques des utilisateurs finaux [16].

6.3 Localisation et contexte

Dans le projet *my*Campus, nous concevions une architecture distribuée facilitant l'accès aux services d'un campus au quotidien. L'environnement (Figure 110) est ouvert et basé sur les technologies des systèmes multi-agents [17] et du web sémantique [4] pour réaliser une plate-forme d'accès mobile [11] à des services en ligne [13]. Les utilisateurs souscrivent à différents types d'*agents-services* qui répondent à une attente et aident à différentes activités (ex : établir une réunion, filtrer des messages). Pour opérer, chaque agent a besoin d'informations sur son propriétaire et éventuellement sur d'autres utilisateurs. L'accès à une information sur un utilisateur est contrôlé par son agent *e-Wallet* et les règles de confidentialité qu'il contient.

Les agents ne sont pas limités aux ressources personnelles des utilisateurs. Ils accèdent également à des services web publics, à des documents web, à des ontologies et des descriptions du web sémantique, *etc.* Dans *m*yCampus, ceci inclut l'accès à une variété de ressources et services tels que des services web de localisation, de météorologie, *etc.* Dans *my*Campus l'accès au système se fait à partir d'assistants personnels numériques (PDA) reliés au réseau sans-fil de l'université de Carnegie Mellon. Cependant, cette architecture fonctionne aussi pour des scénarios à postes fixes et plus généralement dans des environnements où les utilisateurs se connectent par différents canaux et dispositifs d'accès. Les informations sur le dispositif et le canal peuvent être traitées comme des attributs du contexte et rendues disponibles via le *e-Wallet*. Nous avons suivi une architecture FIPA [17] incluant en particulier des *Agents annuaires* qui permettent à un agent/utilisateur à la recherche d'un service d'obtenir une liste d'agents potentiellement fournisseurs de ce service. Des *Agents d'Interaction* sont responsables des échanges avec l'utilisateur ; ils gèrent les sessions de connexion et les interactions avec d'autres agents. Nous nous concentrerons sur des scénarios impliquant des utilisateurs individuels, mais cette architecture s'étend à des scénarios où les utilisateurs sont des organismes entiers ayant chacun un ou plusieurs *e-Wallet*. Pour plus de détails sur l'architecture et sa mise en œuvre, se reporter à [15].

Dans cette section nous présentons des *agents-services* développés et testés dans *my*Campus. Les deux premiers agents présentés sont issus de la première version testée début 2003 sur le campus de Carnegie Mellon pour valider le concept. Les agents suivants se divisent en deux initiatives : (1) InfoBridge : un projet avec quatre étudiants en IHM se focalisant sur l'identification et l'évaluation d'un service utile et utilisable (2) la deuxième version de *my*Campus : se focalisant sur des services et des scénarios démontrant les potentialités du *e-Wallet*. Dans les deux cas, les représentations de connaissances sont utilisées pour réduire les interactions au stricte nécessaire.

Comme le montre la Figure 111, une session commence par un écran d'accueil et en cliquant sur les onglets, l'utilisateur peut consulter son *e-Wallet* ou la liste des services qui lui sont accessibles.

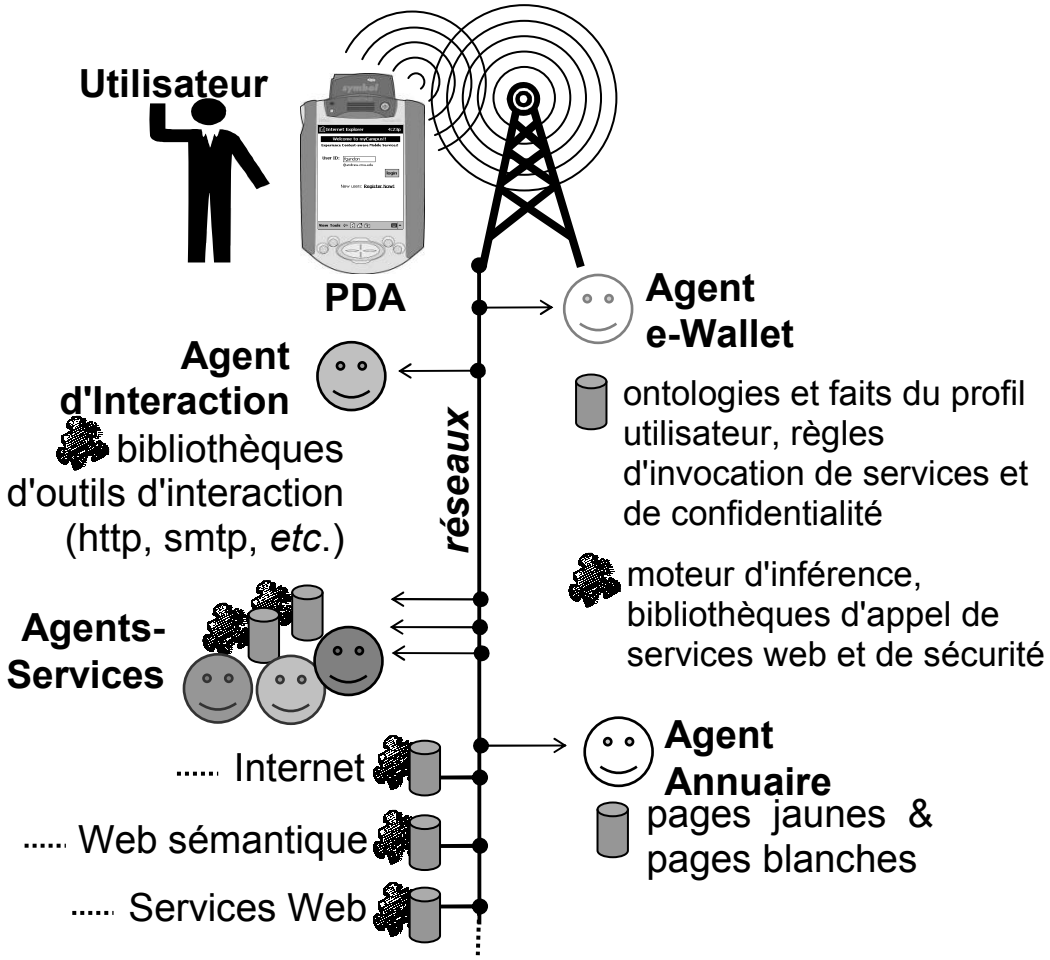


Figure 110. Vue d'ensemble du système *my*Campus.

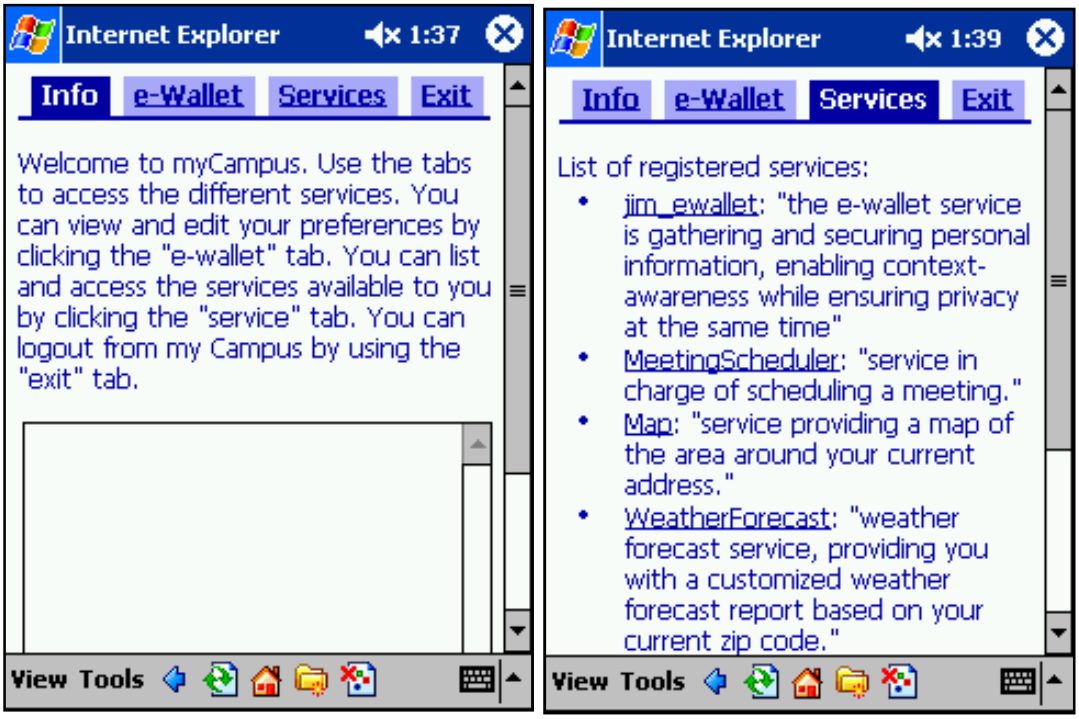


Figure 111. Ecran d'accueil et liste des services disponibles.

6.3.1 Recommander des restaurants

Testé dans la première version de *my*Campus, le "Concierge" est un agent suggérant aux utilisateurs où prendre leurs repas, en tenant compte de leurs préférences culinaires, de l'endroit où ils se trouvent et du temps qu'il fait. Pour cela, l'agent s'adresse au *e-Wallet* de l'utilisateur qui l'a invoqué, afin d'obtenir ses préférences (goûts culinaires, budget, distance, *etc.*) et sa localisation.

Les préférences font partie de la connaissance gardée au cœur du *e-Wallet* et éditables à travers des interfaces dédiées Figure 112. La position de l'utilisateur sur le campus est obtenue à travers le *e-Wallet* qui invoque le service de localisation par le réseau sans-fil.

Le Concierge se connecte aussi à un service web public donnant la situation météorologique. Il utilise ensuite UDDI [174] pour obtenir une liste de services web correspondant aux restaurants des environs ; nous avons créé cette liste en utilisant la base de test d'IBM permettant aux développeurs d'enregistrer des services pour leurs tests. Le concierge en retire pour chaque restaurant l'URL d'une description formelle (cuisine, localisation, prix, *etc.*) qu'il utilise pour suggérer une liste ordonnée de restaurants (Figure 112).

Figure 112. Deux écrans de l'interface du Concierge.

6.3.2 Filtrer et router des messages

Le "Messager", aussi testé dans la première version de *my*Campus, a été particulièrement populaire. Son rôle est de filtrer les messages envoyés à l'utilisateur en tenant compte de ses centres d'intérêt et de sa disponibilité (ex : "quand je suis occupé, retarde les messages jusqu'à ce que mon activité soit terminée").

Là encore, les centres d'intérêt et les préférences sont extraits du *e-Wallet* et modifiables à travers des interfaces dédiées (Figure 113- à gauche). L'activité courante de l'utilisateur est nécessaire à la prise de décision et est obtenue à travers le *e-Wallet* qui se connecte à un service web correspondant à l'agenda Outlook sur le PDA de son propriétaire. Le Messager utilise sa connaissance des contextes et des préférences des utilisateurs pour trier, retarder et router les messages afin que seuls les messages jugés pertinents arrivent à l'utilisateur et ce au moment le plus propice.

Le service est évalué en utilisant des interfaces de retours telles qu'en Figure 113 à droite.

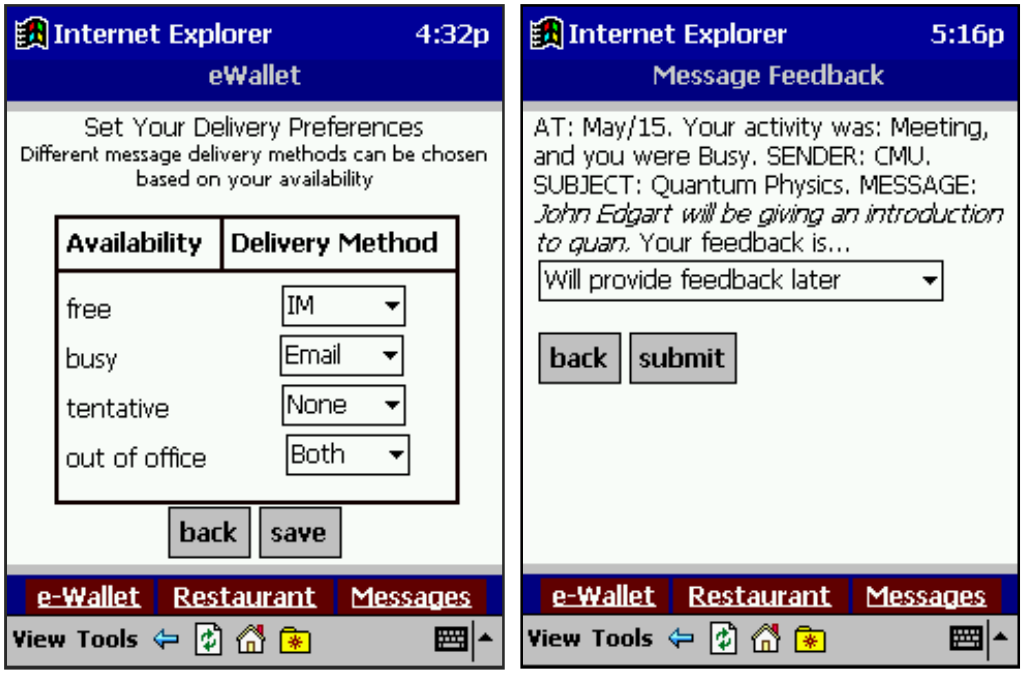


Figure 113. Deux écrans de l'interface du Messager.

6.3.3 Gérer des posters virtuels

A la suite de la première version de *my*Campus et de son évaluation. Nous avons mené un projet parallèle avec des étudiants en IHM pour identifier et évaluer un service qui, avec notre plate-forme déployée sur le campus de Carnegie Mellon, serait à la fois utile et utilisable. Les étudiants se sont focalisés sur un cas particulier du Messager : l'annonce d'évènements / manifestations / *etc.*

Traditionnellement le système le plus efficace et le plus ancien pour ce genre d'annonces est le poster. Comme l'illustre la Figure 114, les zones de posters sont un moyen simple de réaliser une communication de masse ou de personne à personne, le graphisme et l'accrochage aux tableaux permet un balayage et un filtrage rapide des annonces, et le choix du lieu où le poster est accroché est important (bâtiment des arts, résidences, *etc.*) en d'autres termes, les posters sont des messages situés. En contrepartie, les posters sont dans des lieux de passage et les passants intéressés mais pressés, risquent de les oublier rapidement. De plus ils sont contraignants dans leur production (mise en page, graphisme, impression), dans leur diffusion (afficher dans les lieux où passent des gens susceptibles d'être intéressés) et dans leur maintenance.

Figure 114. Une zone de posters à Carnegie Mellon.

Nous avons donc développé et testé un environnement de posters virtuels. Nous avons en particulier étudié le lien entre le profil d'un utilisateur (ses centres d'intérêts) et son parcours typique maximal dans une journée de travail sur le campus. La Figure 115 illustre cette idée que les utilisateurs sont, dans une certaine mesure, décrits par les lieux qu'ils fréquentent : le parcours en trait plein est celui d'un étudiant du département *design* alors que le parcours en pointillés est celui d'un étudiant du département *computer science*.

L'idée a donc été d'introduire le parcours de l'utilisateur dans les mécanismes de filtrage des messages.

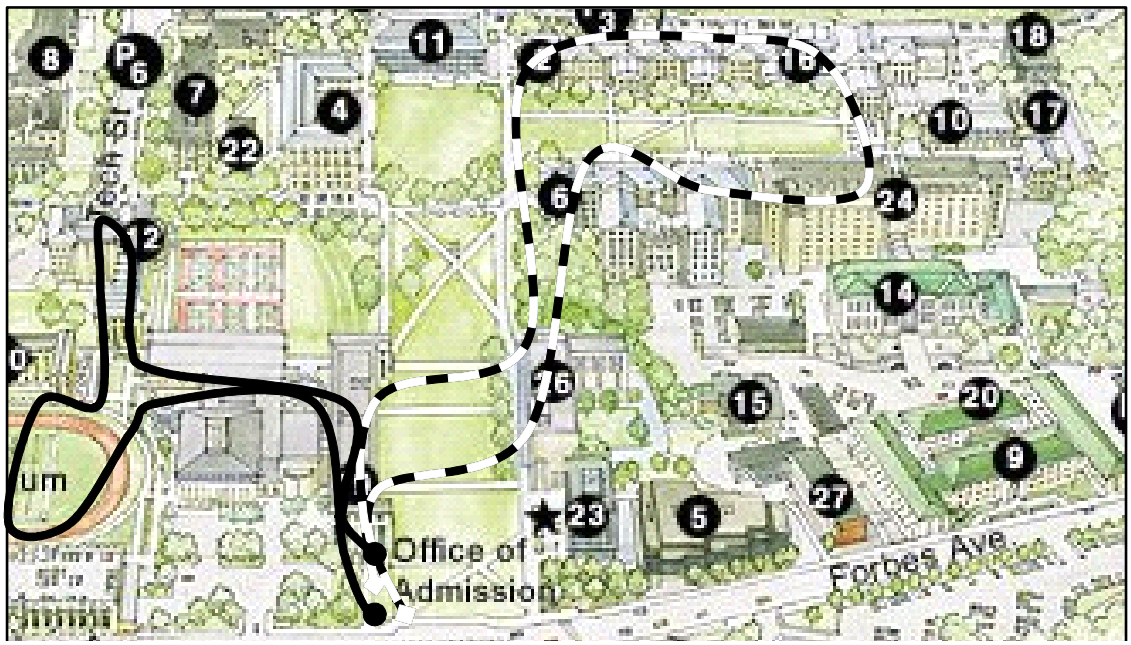


Figure 115. Parcours quotidiens typiques d'un étudiant en design (à gauche) et d'un étudiant en informatique (à droite).

Le résultat fut le prototype InfoBridge [175] dont deux écrans sont montrés en Figure 116. Comme le montre la partie gauche l'utilisateur peut chercher des posters, consulter les posters qu'il a collectés avec son PDA lors de ses déplacements sur le campus, composer un poster (détails dans la partie droite) et éditer ses préférences. Si un utilisateur affirme avoir un intérêt (resp. un désintérêt) pour un sujet, tous les posters portant sur ce sujet lui seront envoyés (resp. cachés) quels que soient ses déplacements sur le campus. Si l'utilisateur n'a pas donné de préférences sur un sujet, son parcours détermine s'il collecte les posters sur ce sujet.

Figure 116. Deux écrans de l'interface InfoBridge.

Un scénario type résume ce service : "Delphine sait qu'un groupe d'étudiants organise un débat sur les logiciels libres à la cafétéria à 16H. Elle est actuellement à la cafétéria et utilise son PDA, pour y déposer un poster virtuel contenant des informations sur l'événement. Adil, qui est un fervent défenseur de Linux, reçoit immédiatement le poster même s'il ne se trouve pas à la cafétéria. Plus tard, Renaud passe par la cafétéria et n'a pas stipulé d'intérêt ou de désintérêt pour le sujet. Puisqu'il est près de la cafétéria, son PDA sonne/vibre ; il peut consulter immédiatement le poster ou l'ignorer pour l'instant et le regarder plus tard avec les autres posters qu'il aura collectés dans la journée au cours de ses déplacements sur le campus. Quand il regarde le poster, il peut décider d'ajouter l'événement à son agenda s'il est intéressé, l'ignorer s'il n'est pas intéressé ou même stipuler qu'il ne veut plus recevoir de posters sur ce sujet. Une fois la date du débat passée, le poster est invalidé, et automatiquement supprimé du système."

6.3.4 Cinéma et météorologie

Ces deux services ainsi que les services suivants ont été développés dans la version 2 de *my*Campus pour tester et démontrer les potentialités du *e-Wallet*. Comme le montre la Figure 117, ces deux services proposent respectivement une liste des films projetés dans les environs, et les prévisions météorologiques. Aucune interface n'est nécessaire pour invoquer ces services : ils s'adressent directement aux *e-Wallets* pour obtenir les informations dont ils ont besoin (localisation). Ils illustrent l'aspect interface unifiée que le *e-Wallet* propose pour l'accès aux connaissances personnelles ou contextuelles.

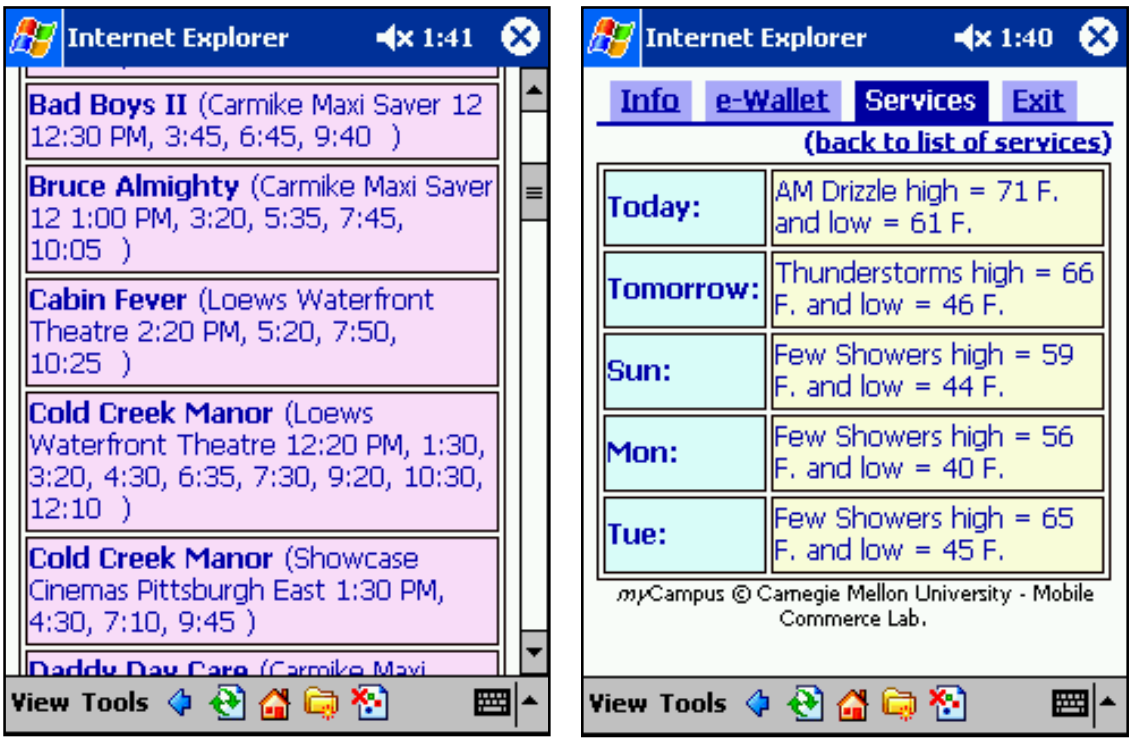


Figure 117. Les services de Cinéma et Météorologie.

6.3.5 Cartographie

L'agent de cartographie n'a pas non plus d'interface et renvoie instantanément une carte des environs à l'utilisateur. Cet agent nous est particulièrement utile pour démontrer l'importance et le fonctionnement des règles de révision du *e-Wallet*.

Considérez la Figure 118 : la partie de gauche montre le résultat obtenu par Marie qui n'a pas de règle de révision et donne directement le code postal du lieu où elle se trouve ; la partie de droite montre, pour le même service, le résultat obtenu par Jim qui ne souhaite pas donner plus de précision que la ville la plus proche.

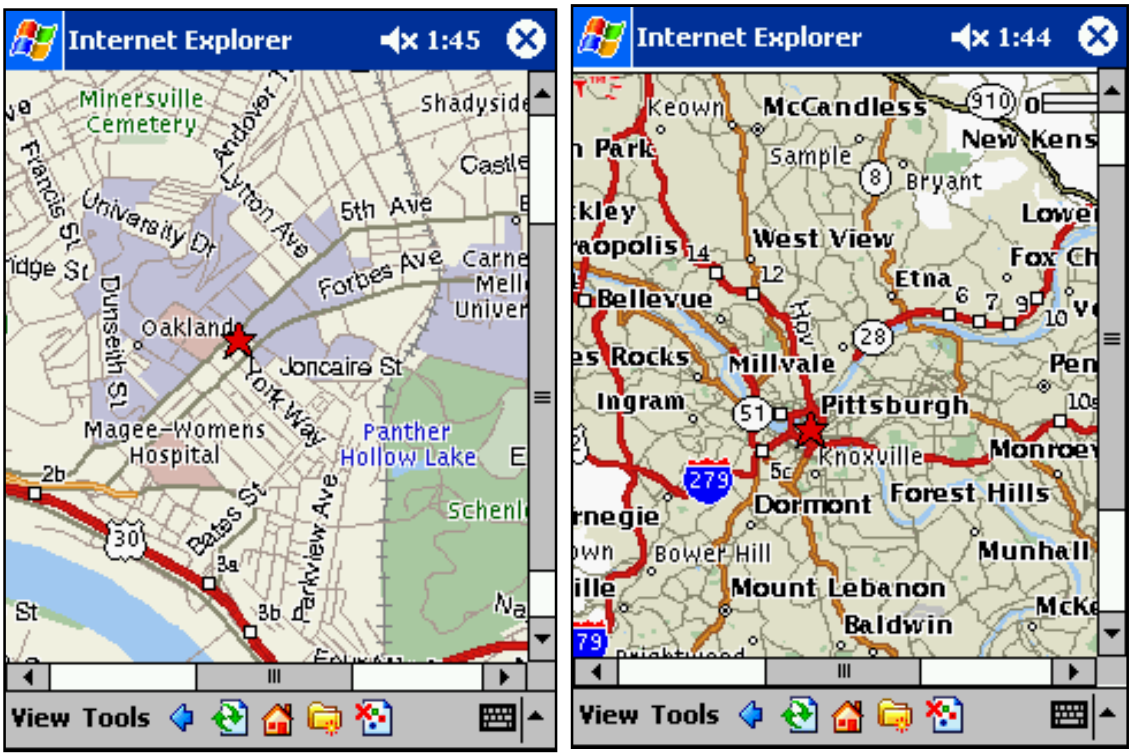


Figure 118. Cartographie et niveaux de confidentialité

6.3.6 Organiser une réunion

L'agent "Assistant de réunion" permet à un utilisateur de trouver un créneau libre dans son agenda et celui d'une autre personne afin de proposer une réunion.

La partie gauche de la Figure 119 montre l'interface nécessaire pour donner les contraintes portant sur la réunion ; le reste des informations sur les utilisateurs (ex : disponibilité) est obtenu à travers leurs *e-Wallets* qui contactent les agendas respectifs des utilisateurs. Ce service donne un exemple de scénario impliquant plusieurs *e-Wallets*.

A titre indicatif un extrait du diagramme d'interaction entre agents est donné en partie droite de la Figure 119.

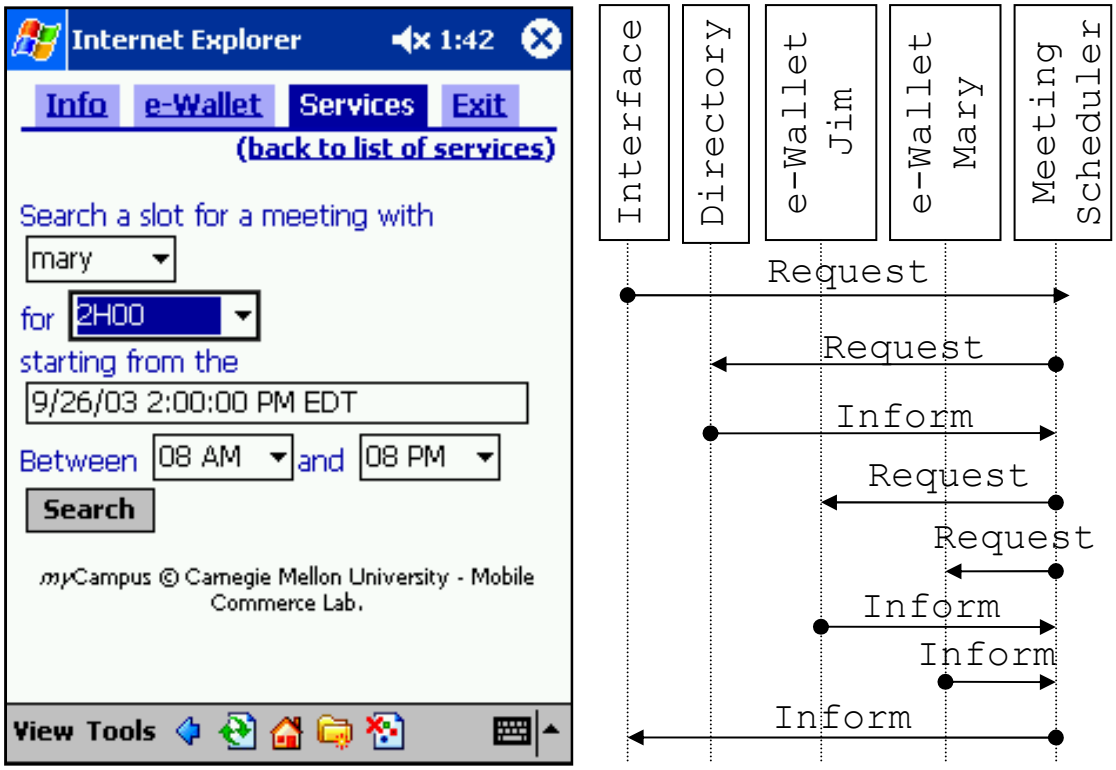


Figure 119. Organiser une réunion avec 2 e-Wallets

6.3.7 Projeter une présentation

Le dernier service est notre première expérience en matière d'informatique ambiante. Comme le montre la Figure 120, ce service permet à un utilisateur de choisir une présentation PowerPoint parmi celles annotées dans son *e-Wallet* et de la projeter sur le vidéoprojecteur le plus proche en prenant le contrôle à travers une interface/télécommande sur son PDA.

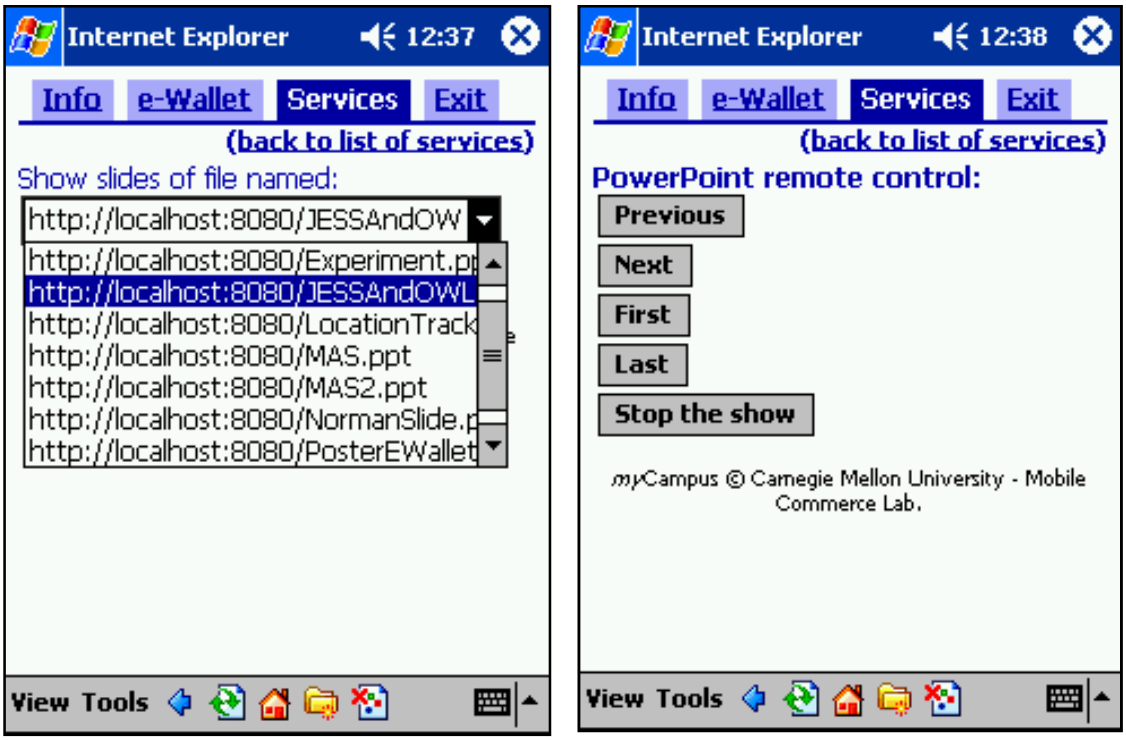


Figure 120. Projeter une présentation

6.3.8 Tests d'utilisation de la première version

La première version du système a été validée sur le campus de Carnegie Mellon en 2003 où l'environnement était accessible à travers des PDA connectés et localisés à travers le réseau sans-fil de l'université. En utilisant ce premier prototype, nous avons entrepris une première expérience pour évaluer la faisabilité, l'utilisabilité et l'utilité. L'expérience de 3 jours a impliqué 11 utilisateurs choisis pour couvrir un large spectre de profils. Ils étaient libres de se connecter depuis n'importe où sur le campus et d'utiliser le Concierge (recommandation de restaurants) et le Messager (filtrage et routage des messages). Ils devaient maintenir leur agenda à jour de façon à assurer une bonne connaissance du contexte tout au long de l'expérience.

L'expérience a totalisé 44 envois de messages filtrés par le Messager et chacune de ses décisions était évaluée par l'utilisateur concerné donnant lieu à une base de 484 retours des utilisateurs. Les utilisateurs ont également demandé 28 recommandations de restaurant au Concierge. Les actions, les résultats, le contexte et les incidents ont été enregistrés dans une base d'historique pour chacun des agents. En outre, chaque utilisateur a dû remplir un questionnaire et passer un entretien final en tête-à-tête avec un membre du projet pour faire un compte-rendu de son expérience personnelle et nous permettre de comprendre certains des résultats observés dans la base d'historique.

Nous ne nous étendrons pas sur les problèmes dus au matériel. Cependant la majeure partie des incidents étaient dus soit aux aléas de la connexion Wifi, soit aux PDA. Notons aussi que les utilisateurs ont trouvé les PDA trop larges pour être transportés comme un téléphone cellulaire et trop petits pour remplacer les ordinateurs portables dont sont équipés presque tous les acteurs du campus. De plus l'autonomie d'un PDA connecté au Wifi est très mauvaise (parfois moins d'une heure). Nous avions alors ajouté aux perspectives la possibilité de migrer le système vers des téléphones capables d'utiliser le Wifi et/ou le Bluetooth.

Pour le Concierge, les résultats ont prouvé que 12,5% des recommandations étaient acceptables pour l'utilisateur uniquement parce que la connaissance du contexte avait permis de faire le bon choix ; en d'autres termes le restaurant choisi par l'utilisateur n'aurait pas été recommandé si le système avait seulement utilisé le profil statique des utilisateurs ignorant leur localisation et le temps. Ceci représente déjà une amélioration de 14,29% par rapport à un système n'intégrant pas cette connaissance du contexte, mais nous attendions plus. Le compte rendu des entretiens a mis à jour plusieurs problèmes, en particulier :

- La plupart des utilisateurs mécontents des recommandations avaient été sur le campus assez longtemps pour connaître tous les restaurants et pour avoir développé des habitudes ; ils n'auraient pas consulté un concierge en temps normal.
- La logique de l'agent était trop naïve, ex : nous basons la recommandation sur le lieu où l'utilisateur se trouve actuellement, il est tout aussi intéressant de

recommander des restaurants proches de l'endroit où il doit se rendre dans l'heure suivante ; cette connaissance est disponible dans l'agenda.

- Des connaissances et des fonctionnalités étaient absentes : régimes spéciaux, plats du jour, longueur de la file d'attente, fonctionnalité pour organiser un repas collectif, *etc*.

Pour le Messager, les 44 messages envoyés à fréquence croissante sur 3 jours ont donné lieu à 484 évaluations : pour chacun des messages l'utilisateur devait donner la décision que le messager aurait dû prendre idéalement (ignorer le message, l'envoyer immédiatement, l'envoyer durant le prochain créneau libre dans l'agenda, l'envoyer en fin de journée après toutes les activités). La connaissance du contexte a montré une amélioration systématique des résultats par rapport à des profils utilisateurs statiques. Dans 70% des cas la meilleure décision de routage n'aurait pas pu être prise sans connaissance du contexte.

6.3.9 Analyse des traces d'utilisation

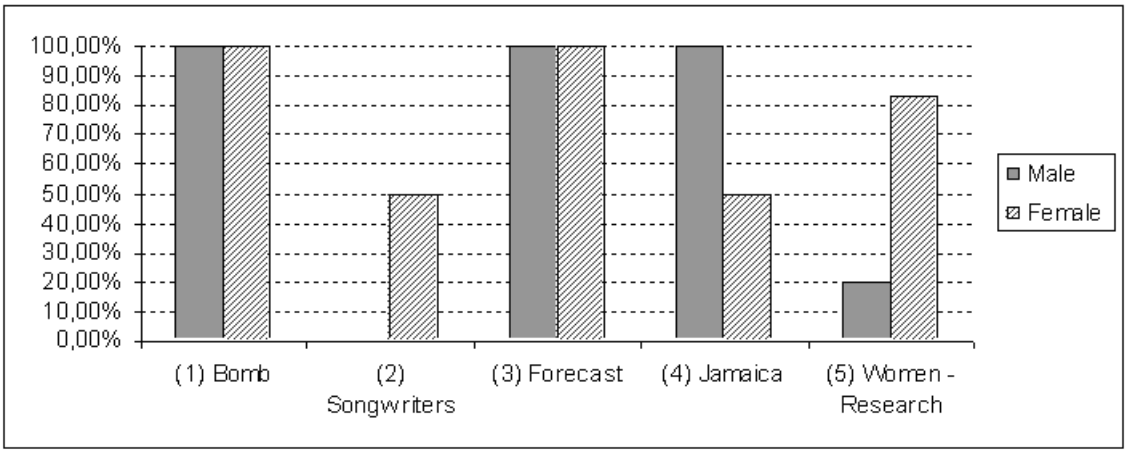
Pour illustrer la richesse et la complexité des résultats enregistrés, nous allons commenter quelques extraits de la base d'historique du Messager (Figure 121). Dans l'ensemble, un filtrage basé sur un profil statique a été jugé trop lâche ; il y a un besoin de filtrage plus intelligent et nous verrons qu'il y a des occasions pour où la connaissance du contexte apporte des améliorations. La Figure 121a montre l'intérêt porté à cinq des messages en fonction du sexe des utilisateurs, le sexe n'ayant pas été pris en compte dans l'algorithme de filtrage. Le message (1) était une information au sujet d'une attaque par bombe et le message (3) donnait les prévisions météorologiques ; tous les deux sont clairement indépendants du sexe. Le message (5) est une annonce pour une conférence au sujet de la place des femmes dans la recherche ; ici le sexe est sensiblement distinctif et c'est compréhensible. Cependant les messages (2) et (4), annonçant respectivement la réunion d'un cercle de compositeurs et un prix bon marché pour des billets vers la Jamaïque, montrent également des différences significatives entre les deux sexes sans qu'il n'y ait aucune raison évidente. Si l'on veut envisager un apprentissage symbolique des préférences cela pose un problème et demande une connaissance plus fine des utilisateurs et des critères de filtrage plus précis.

La Figure 121b montre l'intérêt porté à cinq des messages en fonction du centre d'intérêt principal des utilisateurs. Comme prévu, l'intérêt pour des informations générales, telles que le message (2) au sujet d'une attaque par bombe, est clairement indépendant du centre d'intérêt. De même le message (1) annonçant une conférence technologique et le message (3) annonçant une conférence en biologie, voient leur intérêt varier en fonction du centre d'intérêt principal de l'utilisateur. Par contre, le message (4) est pour le moins étonnant car il montre un intérêt relativement bas des étudiants en sciences humaines pour l'annonce d'une pièce de théâtre alors que l'on attendrait le contraire. La vraie raison est que la plupart des personnes intéressées par des sciences humaines avaient déjà reçu cette annonce par d'autres voies d'information ; ceci montre encore un

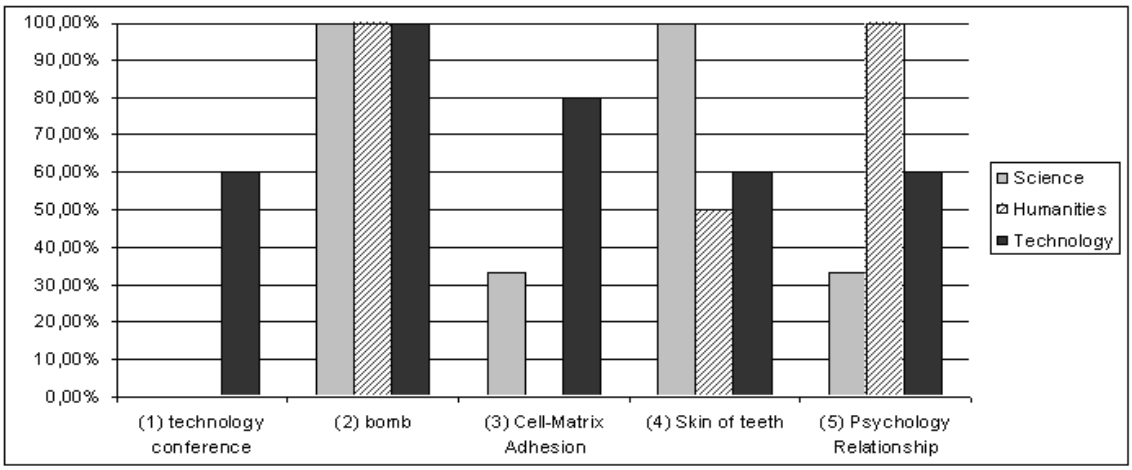
problème qu'un algorithme d'apprentissage risquerait de mal interpréter et aussi les risques d'un raisonnement en vase clos.

Enfin, la Figure 121c montre les préférences de routage des utilisateurs pour 8 des messages. Pour chacun des messages, les utilisateurs pouvaient indiquer : que le message était d'aucun intérêt pour eux (*No interest*) ; que le message était tellement intéressant qu'ils auraient voulu le recevoir immédiatement (*Instantly*) c.-à-d. ils étaient disposés à être dérangés dans leur activité pour un tel message ; que le message était intéressant mais qu'il aurait pu attendre qu'ils soient disponibles (*Available*) c.-à-d. ils ne souhaitaient pas être dérangés pour un tel message s'ils étaient occupés ; et enfin que le message était d'intérêt général et devrait être envoyé plus tard dans la journée, après le travail ou même être vu uniquement à la demande (*Any time*). La barre *Undecided* montre le pourcentage des utilisateurs qui n'ont pas voulu fournir de retour pour un message.

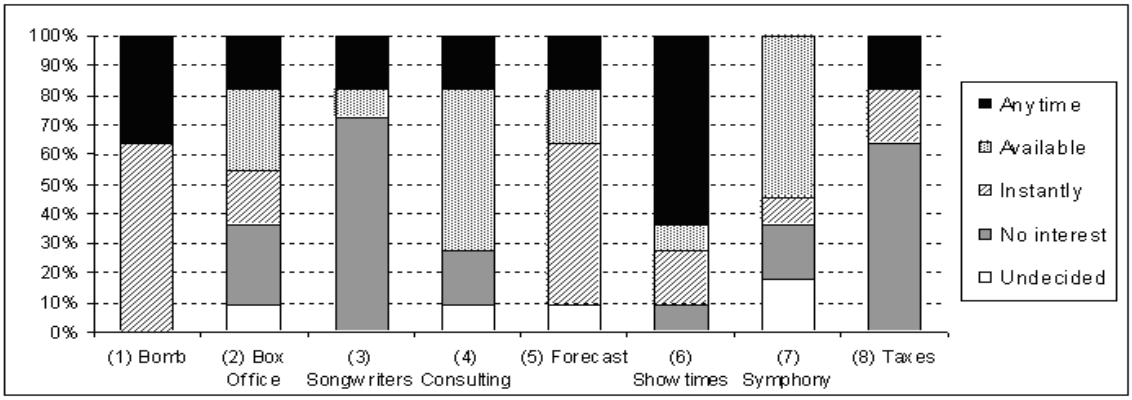
Le message (1) au sujet d'une attaque par bombe, montre typiquement les deux types de comportements pour des informations : certains des utilisateurs veulent savoir immédiatement tandis que d'autres sont du style à attendre le journal de 20H. Le message (6) au sujet du programme des Cinémas peut clairement être envoyé n'importe quand ou consulté sur demande. Nous avons déjà commenté le message (3) annonçant une réunion de compositeurs et il pourrait sembler que, de la même façon, le message (8) au sujet d'informations sur les impôts réclame un filtrage basé sur les centres d'intérêt ; cependant la raison du faible intérêt était qu'aucun des utilisateurs n'avait travaillé l'année précédente et donc qu'aucun d'eux n'était concerné cette année là par ce type de messages. Un tel filtrage est beaucoup plus complexe que le filtrage par centres d'intérêt et exige une connaissance approfondie de l'utilisateur et du domaine. Le message (5) apparaît comme un type de messages qui devrait être envoyé immédiatement, mais en fait cela est dû à un effet secondaire : ce message donne les prévisions météorologiques, mais parce qu'il a été envoyé tôt le matin, le fait qu'il soit montré immédiatement a été bien perçu, alors qu'en fait ce type de message réclame la capacité de lier aux messages/sujets des intervalles de temps de sorte que le système se rende compte de la pertinence d'un message donné à un moment donné. Les messages (4) et (7) étaient au sujet d'événements ayant lieu le jour même (une présentation sur les carrières dans la consultance et un concert en soirée) et un grand nombre de personnes ont souhaité que ces messages leur soient envoyés dès qu'elles sont disponibles dans la journée. Par conséquent les messages (4), (5) et (7) montrent bien l'intérêt d'un filtrage et d'un routage conscients du contexte : en vérifiant l'activité et la disponibilité des utilisateurs dans leurs agendas et en les combinant avec des contraintes sur l'heure de livraison des messages, un système peut trouver les meilleures fenêtres de temps pour envoyer des messages à un utilisateur de façon moins intrusive et plus efficace.



(a) extrait : nombre d'utilisateurs intéressés / sexe



(b) extrait : nombre d'utilisateurs intéressés / centre d'intérêt



(c) extrait des retours sur le filtrage et le routage

Figure 121. Extraits des statistiques sur l'historique

6.3.10 Evaluation du service InfoBridge

Les résultats de cette première expérience ont motivé une deuxième expérience (InfoBridge) en parallèle avec le développement de la deuxième génération de l'architecture. Une équipe d'IHM a commencé par étudier les habitudes et les pratiques [175] pour annoncer et être informés des évènements prévus, ainsi que la façon dont nous organisons notre vie. Ils ont appliqué la méthode du "cheminement cognitif" sur les différentes versions des interfaces du Concierge et d'InfoBridge afin d'identifier un maximum des problèmes d'interactions. Des exemples de problèmes identifiés et résolus [175] sont : la nécessité sur un PDA, qui peut être utilisé par exemple en marchant, de donner des indices visuels évidents du fonctionnement et de l'activité du système (une fois cliqués les boutons restent enfoncés jusqu'à ce que le système traite l'évènement, de façon à montrer visuellement et rapidement que le clic a été enregistré et est en cours de traitement) ; l'utilisation de groupement par couleurs pour les fonctionnalités du système afin de bien montrer l'étape courante ; la mise en place de contrôles graphiques simplifiant la saisie notoirement difficile sur un PDA ; l'introduction d'une carte du campus pour déposer des posters à distance ; *etc.*

La dernière version d'InfoBridge a été testée sur le terrain avec deux personnes du projet suivant personnellement chacun des 4 bêta-testeurs dans leurs déplacements et leur utilisation du système. Après une introduction au système il était demandé aux testeurs de suivre le(s) chemin(s) qu'ils empruntent dans une journée très chargée. Une séance de tests durait environ une heure, comportait la soumission d'au moins 3 posters et était suivie d'un entretien pour recueillir les retours d'utilisation. Les utilisateurs étaient définitivement séduits par le système mais ont soulevé un certain nombre de points [175]: la logique et les critères de sélection d'un poster demandent à être expliqués pour justifier la liste des posters collectés ; le lieu de collage virtuel des posters doit être clairement dissocié du lieu où l'événement aura lieu ; des écrans de confirmations des actions ont été rajoutés pour conforter des utilisateurs dans leurs actions ; la dimension graphique des posters papiers n'est pas compatible avec les limitations d'un PDA et le couplage avec d'autres systèmes semble nécessaire, par exemple un URL vers une page web ; les accidents tels qu'un double clic involontaire sont fréquent sur un système mobile et l'interface doit les prendre en compte ; *etc.*

La deuxième version du système *my*Campus a été terminée et démontrée fin 2003 sur des scénarios illustrant l'intérêt du *e-Wallet*.

6.3.11 Discussion

*m*yCampus est un environnement ouvert basé sur les technologies des systèmes multi-agents et du web sémantique, une plate-forme d'accès mobile à des services en ligne utilisant des informations contextuelles et personnelles tout en respectant la confidentialité des utilisateurs. L'architecture est ouverte en reposant sur l'intégration dynamique des ressources contextuelles représentées par des services web et des services représentés par des agents. Les techniques de modélisations utilisées sont elles aussi ouvertes : en reposant sur les ontologies et les langages du web sémantique nous assurons l'extensibilité des modèles. Le *e-Wallet* fournit une interface sémantique unifiée et sécurisée pour l'accès aux ressources personnelles d'un utilisateur et permet ainsi aux services de les mobiliser. Il est persistent, indépendant des scénarios choisis et exploite la flexibilité des modèles orientés ontologie.

Les résultats de nos tests réclament plus d'intelligence et de connaissances sur l'utilisateur et son contexte. L'analyse des interfaces a montré l'importance de réduire la charge cognitive des utilisateurs lors d'accès mobiles. L'ajout d'indices visuels de l'état du système fut une réponse immédiate, mais à plus long terme l'augmentation de la conscience du contexte permettra de réduire considérablement les besoins d'interactions et de les rendre les moins intrusives possible. Toutes les interfaces n'ont pas été étudiées en détail car l'évaluation et la re-conception sont extrêmement coûteuses. Seul le service des posters a fait l'objet d'une étude approfondie et complète pour nous donner une idée des contraintes existantes. Les autres interfaces sont plus ou moins dédiées : certaines permettent une manipulation générique des structures conceptuelles sous-jacentes (ex : interface du *e-Wallet*), elles sont très flexibles et utiles aux experts du système mais souvent trop complexes ; d'autres interfaces sont dédiées à un scénario (ex : les posters), elles sont beaucoup plus faciles d'accès et parlantes pour les utilisateurs, mais ne permettent que des actions anticipées et limitées au scénario et à son domaine.

Ainsi, comme pour la section précédente, un défi demeure, héritage de toutes les approches utilisant des représentations riches : réconcilier l'expressivité des langages de représentation avec les exigences ergonomiques des utilisateurs finaux.

6.4 Substituts d'information

Le web sémantique est une vision d'un web enrichi par des connaissances formalisées notamment pour l'annoter. Actuellement, il y a un grand décalage entre les structures conceptuelles qui sous-tendent le web sémantique et le rendu d'une interface permettant à un utilisateur final de consulter ou d'agir sur une partie de celui-ci.

En 2005 nous proposions une approche [159] que nous avons expérimentée pour automatiser une partie du processus de génération de représentations des concepts mobilisés dans le web sémantique. Nous réutilisons alors la notion de substitut (*surrogate*) comme définie en recherche d'information [176] [177] et nous montrions que les structures de ces substituts ont tendance à être proches de la structure de la condition d'identité utilisée en ingénierie ontologique. Partant de cette observation nous proposions et discutions un mécanisme pour générer des modèles de substituts à partir de structures trouvées dans les règles et les ontologies.

Pour interagir avec le web sémantique et ses applications nous avons besoin d'interfaces qui le rendent intelligible pour les utilisateurs finaux. Le problème de l'intelligibilité est différent de celui de l'interopérabilité. L'intelligibilité n'est pas assurée par le fait de travailler au niveau sémantique alors que l'interopérabilité peut souvent y trouver une solution. Dans nos systèmes d'inférences des éléments de connaissances sont manipulés et combinés, et même si les éléments de départ étaient intelligibles, l'intelligibilité des résultats, elle, n'est pas préservée par ces transformations [178].

Actuellement, la plupart du temps, c'est aux concepteurs d'interfaces utilisateurs de mettre en œuvre, souvent par des moyens *ad hoc*, la transformation des structures de données internes en représentations d'interface. Dans le pire des cas, et encore trop souvent, il ne s'agit même pas d'un concepteur d'interface mais d'un programmeur n'ayant pas de formation en ergonomie ou conception d'interactions et, si ce dernier point est une plaie de l'informatique en général, elle en est d'autant plus accentuée quand les structures de représentations internes se complexifient.

Quoi qu'il en soit, ces transformations *ad hoc*, ne sont de toute façon plus possibles lorsque les structures de données, leurs schémas, leurs transformations, etc. se modifient et se propagent par le biais de réseaux. En d'autres termes, ce n'est plus possible dans le contexte du web sémantique tel que nous l'imaginons à son plein potentiel. Les interfaces devront être, au moins en partie, générées dynamiquement et rendues pour chaque structure devant rentrer en contact avec les utilisateurs.

Cette section décrit une expérience, où nous avons regardé comment automatiser une partie du processus de génération des représentations pour les concepts mobilisés dans le web sémantique. L'idée est d'utiliser les conditions d'identité pour initialiser des modèles de visualisation appelés substituts (*surrogates*) et utiliser ensuite une validation humaine pour faire la distinction entre les "bons" et les "mauvais" modèles de substituts.

A titre d'exemple simple, la Figure 122 montre une requête recherchant des documents (`?d` dans le triplet de la ligne 1) avec au moins un auteur (`?a` en ligne 2), une personne (ligne 3) dont le nom (`?n` ligne 4) contient la chaine "aiman" (ligne 5). La Figure 123 montre un extrait de la réponse rendue par une feuille de style XSLT transformant la syntaxe du résultat en XHTML. La bibliothèque de modèles génériques XSLT que nous avons développée utilise le format RDF/XML des fichiers des ontologies et les résultats des requêtes pour afficher les classes et les propriétés en utilisant des étiquettes ontologiques (*rdfs:label*), ce qui nous permet également de gérer des interfaces multilingues comme dans le projet CoMMA.

```
1. ?d rdf:type  ex:Document
2. ?d ex:author   ?a
3. ?a rdf:type  ex:Person
4. ?a ex:name     ?n
5. FILTER( regex( ?n, ".*aiman.*"))
```

Figure 122. Un exemple de requête sur des annotations de documents.

- Novel (http://isbn.nu/0380789035)
  - **author** Man (http://www.neilgaiman.com/)
    - **name**: Gaiman
- Article (http://www.asee.org/jee/papers/content.cfm?name=STEPHEN-209.pdf)
  - **author** Woman (http://www.mgt.ncsu.edu/faculty/busmgt/laiman-smith.html)
    - **name**: Aiman-Smith

Figure 123. Un exemple de formatage du résultat.

Ce type d'interaction simple, à base de requêtes-réponses, est omniprésent dans tous les projets auxquels nous avons participé jusqu'à présent. Si nous regardons le résultat de la Figure 123, la réponse est parfaitement correcte d'un point de vue ontologique. Toutefois, si elle est présentée telle quelle aux utilisateurs, la seule connaissance qu'ils peuvent obtenir de cette réponse est qu'il existe un roman écrit par un homme du nom de "Gaiman" et un article écrit par une femme appelée "Aiman-Smith"; la plupart des utilisateurs auraient probablement apprécié d'avoir le titre du roman, le prénom de l'auteur, etc. s'ils étaient disponibles. On pourrait répondre à cela que les utilisateurs devraient avoir inclus ces demandes de précision dans la requête, mais l'attente de ces utilisateurs est si naturelle que ces besoins auraient dû être automatiquement inclus dans le couple requête-réponse. *En fait, nous estimons que l'utilité de ces propriétés est une connaissance ontologique*.

En recherche d'information, indexer des ressources d'information consiste à parcourir l'ensemble de ces ressources et à construire pour chacune d'entre elles un substitut la représentant (par exemple un vecteur de termes). Les substituts pour l'indexation peuvent être aussi simples qu'une sélection de certains mots d'un document ou aussi complexes que le résultat d'un traitement de la langue naturelle sur son contenu.

Les substituts ne se limitent pas à représenter le contenu des ressources, mais peuvent également inclure des métadonnées (par exemple, l'éditeur, l'auteur, l'ISBN, des dates, etc.) et plus généralement des propriétés externes de la ressource (par exemple, le nombre d'hyperliens pointant vers celle-ci, son statut dans un workflow, etc.). Le choix du substitut influence dans une large mesure l'ensemble du système d'information auquel il participe et motive beaucoup de travaux dans le domaine de la recherche d'informations [176] [177]. *Ainsi, dans les systèmes d'information, la première utilisation des substituts de ressources d'information est de fournir une structure très synthétique et représentative des caractéristiques de la ressource qui sont pertinentes pour les traitements effectués par le système* ; l'archétype étant le vecteur de termes.

Outre l'efficacité de l'algorithme de recherche, un facteur important dans l'acceptation d'un système par les utilisateurs est l'interface au travers de laquelle ils interagissent avec ce système. Les résultats d'une recherche dans un moteur de recherche classique prennent généralement la forme d'une liste de pages avec pour chaque résultat un texte extrait de la page sélectionnée justifiant la sélection. En outre, d'autres informations peuvent être données (par exemple l'URL, la date de l'indexation, une vignette, etc.) et les résultats sont classés en fonction de leur pertinence estimée à la requête. Pour représenter cet ensemble de ressources, le système utilise un deuxième type de substituts [176] [177] : *la deuxième utilisation des substituts de ressources d'information est de fournir une structure très synthétique et représentative des caractéristiques pertinentes pour les utilisateurs afin qu'ils identifient les ressources et leur position dans l'ensemble des résultats* ; l'archétype de ces substituts et celui des moteurs de recherche du web comme dans la Figure 124.

L'importance de ce deuxième type de substituts est connue des acteurs du domaine et ils cherchent activement à les améliorer. La dernière initiative en date étant le nouveau moteur de Yahoo appelé « SearchMonkey » qui rend les résultats de recherches de Yahoo ouverts à la personnalisation. Il permet à des tierces parties (développeurs, webmasters ou même tout internaute) d'enrichir la fiche de leur site ou de leur blog sur la page de résultats du moteur de recherche de Yahoo. Grâce à une série d'API, ils peuvent ainsi rajouter des photos, une brève présentation, des liens, un prix ou encore un numéro de téléphone. Les internautes utilisateurs du moteur ne sont pas en reste puisqu'ils peuvent de façon duale choisir les informations qu'ils souhaitent voir figurer dans les résultats (Figure 125).

Figure 124. Quatre exemples de substituts dans les résultats d'un moteur de recherche

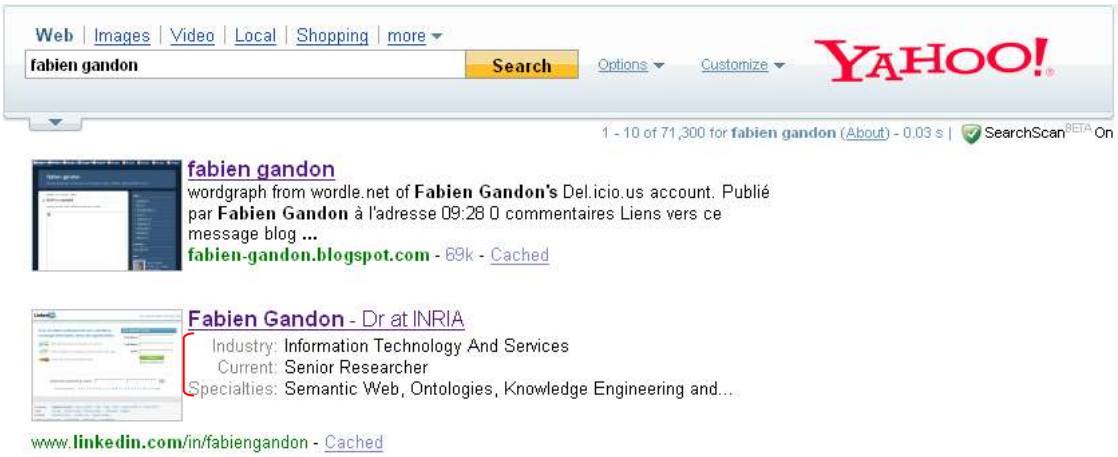


Figure 125. Amélioration automatique des substituts (2ième résultat) dans SearchMonkey

Un identificateur est toujours présent dans les deux types de substituts, mais il n'est en général pas utile aux utilisateurs car il s'agit habituellement d'un identifiant système comme une clé primaire de base de données ou une URI (par exemple `http://st5.com/ReportV278.htm#C12`) qui fournit très peu d'information intelligible sur la ressource et n'est généralement utilisée que dans des traitements techniques tels que les jointures. Ainsi, le deuxième type de substituts nécessite des informations telles que : un titre, un extrait focalisé, un aperçu, etc. Du choix de ce substitut dépend la capacité du système à proposer des vues qui organisent les résultats de manière efficace.

Dans le cas d'un moteur de recherche sémantique (un moteur de recherche exploitant des représentations formelles de connaissances) où les requêtes et les résultats sont limités par les ontologies disponibles, le problème de trouver un mécanisme générique pour construire ces substituts est ouvert. *L'une des difficultés est que la pertinence des éléments utilisés pour construire une substitution est dépendante du domaine considéré*.

Les ontologies fournissent la sémantique référentielle pour une communication et de ce fait, sont à la frontière entre la conceptualisation du système et celle des utilisateurs. Ainsi, les ontologies doivent être compréhensibles à la fois pour les humains et les machines, sinon elles ne peuvent plus jouer ce rôle central et elles ne sont plus utilisables, maintenables, etc. De plus, ces ensembles de structures conceptuelles internes ne doivent normalement pas être montrés aux utilisateurs finaux ; pas seulement parce que leurs inscriptions logiques sont absconses, mais aussi parce que nous, les humains, nous ne mobilisons pas l'ensemble de nos conceptualisations à chaque fois que nous communiquons, pensons, agissons, etc. : nous nous focalisons. Par conséquent, un système ne doit pas imposer aux utilisateurs de gérer un ensemble d'ontologies à chaque fois qu'ils ont une interaction : nous nous focalisons et les interfaces doivent se focaliser avec nous.

L'interface utilisateur a le rôle peu enviable de combler l'écart entre les conceptualisations explicites capturées dans les ontologies et l'utilisation quotidienne de signes pour désigner des concepts avec d'inévitables ambiguïtés et flous. Le simple fait de choisir les termes (labels, étiquettes) dans une ontologie introduit celle-ci dans le domaine de la sémiotique ; les problèmes d'interface utilisateur et les problèmes de représentation ontologique doivent être abordés en parallèle.

Ce dernier point nous ramène au problème du choix de substituts pour la visualisation. Cet aspect a été le plus souvent négligé dans la littérature alors qu'il est essentiel pour assister les mécanismes d'interprétation associés à nos modèles, nos inférences et de leurs résultats. Par exemple, pour représenter une personne, il est logique de construire un substitut contenant le prénom et le nom de la personne, mais l'âge, la taille et l'adresse peuvent ne pas être utiles sauf s'ils sont expressément requis par un cas d'utilisation. *Distinguer les caractéristiques principales et des caractéristiques non principales pour un substitut d'interface est une tâche dépendant des scénarios d'usage*.

La notion d'identifiant unique (par exemple, URI, ISBN) est artificielle pour nous. Nous, en tant qu'êtres humains, avons souvent besoin de plusieurs attributs essentiels pour identifier une instance, et parfois même courons le risque de confusion lorsque l'unicité n'est pas assurée à 100% par les caractéristiques de ces substituts. Entre les différents systèmes, en intégration de données ou dans des scénarios d'annotation avec différents points de vue, la preuve d'équivalence d'instances et la fusion d'instances utilisent généralement ces attributs d'identité. Ce problème est bien connu, par exemple, dans les systèmes essayant de détecter dans des réseaux d'accointances que l'auteur d'un document est la même personne que l'auteur d'un autre document ; les principaux attributs de l'identité sont généralement choisis, pondérés et combinés pour décider, si le résultat dépasse un seuil de confiance, de fusionner deux entités.

En fait, le problème de la représentation d'instances dans une interface est, d'une façon, étroitement lié à la notion d'identité *i.e.* le problème ontologique de distinguer une instance précise d'une certaine catégorie d'une autre instance de cette même catégorie, par le biais d'une propriété caractéristique, qui est unique pour elle. Guarino et Welty ont suggéré [179] que l'identité repose sur l'existence d'une propriété rigide c'est-à-dire une propriété que possède nécessairement toutes les instances d'un concept. Ils donnent l'exemple d' «être une personne » qui est rigide puisqu'une instance *x* étant une personne ne peut cesser d'être une personne à moins que l'instance elle-même cesse d'exister. Par contre, "être un étudiant" est anti-rigide puisque toutes les instances d'Etudiant ont la possibilité de cesser d'être un étudiant sans cesser d'être (heureusement d'ailleurs). Une condition d'identité pour une propriété ϕ est définie [179] comme une relation satisfaisant la condition (1) et un concept α est sémantiquement rigide si il satisfait la condition (2), l'opérateur *nec* étant l'opérateur de nécessité modale.

$$\phi(x) \wedge \phi(y) \rightarrow (\rho(x,y) \leftrightarrow x = y) \quad (1)$$

$$\forall x \quad (x \in \alpha \supset nec \ (x \in \alpha)) \quad (2)$$

Plus qu'une seule propriété caractéristique, la condition d'identité ρ peut être un groupe de propriétés complexes et de comparaisons qui établissent l'identité, par exemple, le nom de famille et le prénom ainsi que la date et le lieu de naissance. Cela donne à penser qu'il existe un chevauchement intéressant entre l'ensemble des propriétés utilisées dans la condition d'identité d'un concept et les propriétés utilisées dans son substitut.

Dans [179], les auteurs font également une distinction entre fournir une condition d'identité et simplement porter une condition d'identité : les propriétés non-rigides ne peuvent que porter une condition d'identité, héritée de celles fournies par les propriétés rigides les subsumant. Tout comme les conditions d'identité sont héritées le long de la hiérarchie, les propriétés principales des substituts sont héritées le long de la hiérarchie par exemple : si le prénom est pertinent pour le substitut d'une personne, il l'est également pour le substitut d'un enfant.

Etendons maintenant la notion de condition d'identité avec la notion de condition minimale d'identité : une condition d'identité est minimale si la condition d'identité ρ satisfaisant (1) repose sur un ensemble de comparaisons qui ne comprennent pas un

sous ensemble formant les comparaisons définissant une autre condition d'identité. Ceci est résumé en (3), les t_n sont des tests unitaires sur les propriétés de x et y.

$$Minimal(\rho) \leftrightarrow [\rho(x,y) \leftrightarrow \wedge_i\, t_i(x,y)] \\ \wedge [\, \neg\exists\, \rho' \,;\, IC\,(\rho') \wedge [\rho'(x,y) \leftrightarrow \wedge_j\, t_j(x,y)] \wedge \{t_j\} \subset \{t_i\}] \tag{3}$$

Cette restriction vise à éviter de considérer, par exemple, une condition d'identité, utilisant le nom, prénom, sexe, date et lieu de naissance, alors que l'on connaît une condition d'identité identique mais n'utilisant pas le sexe. Sur cette base, nous avons fait l'hypothèse (4) que l'ensemble des propriétés utilisées par une condition d'identité minimale d'identité peut être utilisé pour construire un candidat de substitut.

$$Minimal(\rho) \wedge [\, [\rho(x,y) \leftrightarrow \wedge_i\, t_i(x,y)] \rightarrow \\ \exists\, S;\; Surrogate\,(S) \wedge S \equiv \{property\; p_j \,;\, p_j \; used\; in\; test\; t_i(x,y)\}\,] \tag{4}$$

Bien sûr, à ce stade, le problème est de trouver une source pour ces conditions d'identité et les propriétés qu'elles utilisent. Une possibilité est d'utiliser les structures générées par un outil d'aide à ces méta-modélisations. Toutefois, nous voulions trouver ces propriétés dans les structures que nous avons à portée de main dans nos projets, l'une d'entre elles est la base de règles.

CORESE intègre un moteur d'inférence sur le principe des règles de production appliquées en chaînage avant. Par exemple, la règle en Figure 126 indique que si une personne `?m` est le chef de l'équipe `?t` qui a un membre `?p` (ligne 1), alors la personne `?m` dirige la personne `?p` (ligne 2). Les règles sont appliquées une fois les annotations chargées afin de compléter les annotations avant la résolution de requêtes.

```
01    IF  [Person: ?m]-(head)-[Team: ?t]-(hasMember)-[Person: ?p]
02    THEN[Person: ?m]-(manage)-[Person: ?p]
```

Figure 126. Exemple de règle de production

OWL comprend des primitives permettant d'indiquer l'équivalence de deux ressources. Par conséquent, en intégration de données, des règles de production peuvent être utilisées pour automatiser la détection des équivalences entre les ressources et produire des déclarations d'équivalence en OWL. La Figure 127 montre une règle donnant un exemple d'utilisation du nom (lignes 1 et 4), du prénom (lignes 2 et 5) et de la date de naissance (lignes 3 et 6) de deux instances de la classe personne pour affirmer des équivalences entre différentes instances représentant les mêmes personnes. Ces règles encodent des scripts pour détecter l'équivalence d'identité : leurs conclusions sont l'affirmation d'une équivalence et leurs prémisses expriment souvent des comparaisons dérivées d'une condition d'identité telle que définie précédemment. Il est important de souligner que la seule raison pour laquelle nous analysons les règles pour suggérer des substituts est qu'elles sont déjà disponibles dans des systèmes d'intégration de données ; d'autres formalismes, en fonction de leur expressivité, pourraient aussi être utilisés.

```
01   IF   [Person: ?p1]->(name)->?n
02                     ->(firstname)->?f
03                     ->(birthdate)->?d
04   AND  [Person: ?p2]->(name)->?n
05                     ->(firstname)->?f
06                     ->(birthdate)->?d
07   THEN [Person:?p1]->(equivalent)->[Person: ?p2]
```

Figure 127. Exemple de règle d'équivalence

Par conséquent, notre idée fut d'utiliser la prémisse de telles règles pour obtenir un substitut pour les instances du type pour lequel elles génèrent des équivalences.

S'il existe une règle de la forme (5)

$\wedge_i\{ t_i(p_j(x),p_j(y)), t_i \text{ étant un test unitaire}\} \wedge type_t(x) \wedge type_t(y) \Rightarrow x \equiv y$

Alors un substitut possible pour x quand $type_t(x)$ est vrai est :

$\cup_j\{\text{propriété } p_j(x)\}$

Par exemple, de la règle en Figure 127 et de la formule (5) on peut conclure que les propriétés nom, prénom et date de naissance peuvent être utilisées pour fournir un modèle pour des substituts d'instances de type personne c'est-à-dire qu'il est utile d'utiliser ces propriétés et leur valeur pour désigner une instance du type de personne.

Un deuxième type de règles intéressantes pour la construction de ces substituts candidats sont les règles codant des conditions suffisantes pour la définition d'un concept (d'autres constructions OWL comme les équivalences et les restrictions pourraient être utilisées ici). Un cas particulier est celui de la dépendance externe telle que définie dans [179] : une propriété φ est dépendante extérieurement d'une propriété ψ si, pour toutes ses instances x, nécessairement une instance de ψ doit exister, qui n'est ni une partie ni un constituant de x. Les auteurs de [179] donnent l'exemple de la propriété d'être parent comme étant extérieurement dépendante de la présence d'enfants (on ne peut pas être un parent sans avoir un enfant), mais la propriété d'être une personne n'est pas extérieurement dépendante d'avoir un cœur ni un corps (parce que toute personne possède un cœur et est constituée d'un corps). Considérons l'axiome (6) donnant la définition de la notion de président. L'implication correspondant à la condition suffisante de cette définition peut être représentée comme la règle dans la Figure 128.

$$president(p) \Leftrightarrow person(p) \wedge \exists c\ country(c) \wedge govern(p,c) \quad (6)$$

```
01  IF   [Person:   ?p]-(govern)-[Country: ?c]
02  THEN [President:?p]
```

Figure 128. Règle donnant une condition suffisante pour être président

On peut voir ici qu'il fait sens de considérer les propriétés d'une condition suffisante pour représenter une instance du type considéré ; ici il fait sens de donner le pays dirigé lorsque l'on parle d'un président.

S'il existe une règle de la forme (7)

$$\wedge_i\{t_i(p_j(x)),\ t_i \text{ being a test }\} \wedge type_{t1}(x) \Rightarrow type_{t2}(x)$$

Alors un substitut possible pour *x* quand $type_{t2}(x)$ est vrai est :

$$\cup_j \{\text{property } p_j(x)\}$$

Puisque différentes règles génèrent différents modèles de substituts et que ces modèles sont hérités le long de la hiérarchie des classes, nous avons besoin de les combiner. Comme le montre l'heuristique (8) nous proposons par défaut de faire l'union de tous les candidats substituts qui s'appliquent à un type et à ses super-types pour à construire un candidat maximum.

$$Max\ Surrogate\ (type_t) = \cup j\ \{Surrogate_j(type_t')\ ;\ type_t \preceq type_t'\} \quad (8)$$

En appliquant l'heuristique (8) aux règles et axiomes de cette section, nous obtiendrions que pour un président, un candidat substitut maximal serait composé, du nom, prénom, date de naissance et pays gouverné.

Au lieu de présenter directement les résultats de cette approche, nous allons les comparer avec différentes approches utilisées pour améliorer la lisibilité des résultats dans nos projets.

Les tous premiers projets dans lesquels nous avons utilisé le moteur de recherche CORESE demandaient aux utilisateurs de compléter leurs requêtes avec des parties optionnelles qu'ils voulaient voir dans la réponse *i.e.* la requête n'échouerait pas si ces parties ne pouvaient être récupérées, mais, si elles étaient disponibles, elles seraient affichées.

Le problème, nous l'avons vu en introduction, est qu'une charge supplémentaire était donnée aux utilisateurs qui ne savaient par ailleurs pas forcément ce qui était disponible comme informations dans la base. De plus ceci n'était possible que pour les requêtes écrites et soumises par un homme et pas pour les requêtes générées automatiquement par le système. En outre, les évaluations ont montré que les utilisateurs attendaient que le système propose de lui-même ce complément d'information lorsqu'il était disponible. Par conséquent, nous avons commencé à étudier les moyens d'automatiser la génération de la liste des propriétés à ajouter aux requêtes dans des parties optionnelles.

Notre première tentative fut d'ajouter systématiquement toutes les propriétés que l'on pouvait trouver. La Figure 129 montre ce que la requête initiale de la Figure 122 devient dans ce cas. Les deux lignes supplémentaires demandent respectivement toutes les propriétés disponibles sur le document (ligne 06) et sur l'auteur (ligne 07). La propriété `cos:Property` subsume toutes les propriétés et est automatiquement créée par CORESE, le type exact des propriétés est bien sûr donné dans le résultat.

```
01    ?d  rdf:type    ex:Document
02    ?d  ex:author   ?a
03    ?a  rdf:type    ex:Person
04    ?a  ex:name     ?n
05    FILTER(regex( ?n, ".*aiman.*"))
06    OPTIONAL { ?d ?p ?p1 }
07    OPTIONAL { ?a ?q  ?p2 }
```

Figure 129. Ajout de toutes les propriétés disponibles

Le premier problème rencontré a été que cette approche peut ne pas être suffisante : parfois la requête revient avec des propriétés et les URI des ressources sur lesquelles elles pointent et ceci n'est pas suffisant pour un affichage compréhensible. Ce premier point a été résolu en utilisant à l'époque des techniques d'holophrastage dans l'interface permettant aux utilisateurs de demander des informations supplémentaires sur une URI à l'extrémité de la réponse en cliquant sur une petite icône. Le deuxième et le plus important des problèmes que nous avons rencontrés a été que cette approche peut évidement rapporter trop de propriétés. En fait, elle génère beaucoup de bruit dans la réponse. Ainsi, une personne peut être affichée avec tous les documents qu'elle a écrits le résultat devenant alors terriblement long et peu facile à utiliser.

Etant donné que le principal inconvénient de cette première tentative était qu'elle générait trop de bruit et n'était pas efficace, nous avons décidé d'essayer de déterminer les propriétés pertinentes directement dans l'ontologie. Pour ce faire, nous avons introduit une propriété de haut niveau `surrogate_property` utilisée pour subsumer dans l'ontologie les propriétés de domaine utiles pour les substituts. Par exemple, la propriété désignation était une sous propriété de `surrogate_property` et la mère des propriétés titre, nom, prénom, etc. La Figure 130 montre ce que la requête initiale de la Figure 122 devient en utilisant cette approche. Les deux lignes supplémentaires demandent respectivement toutes les sous-propriétés de `surrogate_property` pour le document (ligne 06) et l'auteur (ligne 07).

```
01    ?d  rdf:type    ex:Document
02    ?d  ex:author   ?a
03    ?a  rdf:type    ex:Person
04    ?a  ex:name     ?n
05    FILTER( regex( ?n, ".*aiman.*"))
06    OPTIONAL { ?d  ex:surrogate_property  ?p1 }
07    OPTIONAL { ?a ex:surrogate_property   ?p2 }
```

Figure 130. Ajout de toutes les propriétés disponibles

Le premier problème rencontré dans cette nouvelle approche est que certaines propriétés (par exemple, une date) sont intéressantes pour les substituts d'un certain type de concept (par exemple, un évènement) alors qu'elles ne sont pas pertinentes pour les substituts de certains autres types de concept (par exemple, une entreprise) même si elles sont disponibles.

Le deuxième problème est que cette approche nécessite une charge supplémentaire dans l'ingénierie et la maintenance de l'ontologie. Pourtant, dans certaines applications, nous utilisons encore cette technique, car elle est rapide et les résultats sont acceptables. Nous verrons aussi dans les perspectives de ce chapitre comment Fresnel [180] peut permettre d'étendre cette approche.

Nous arrivons donc aux essais faits à partir de substituts générés. De la Figure 131 à la Figure 133 se trouvent des exemples de règles affirmant des équivalences de deux instances, ils ont été choisis pour leur pertinence par rapport à la requête donnée en Figure 122.

La règle en Figure 131 indique que, si (ligne 2) deux personnes (lignes 3 et 8) ont le même nom (lignes 4 et 9), le même prénom (lignes 5 et 10) et la même date de naissance (lignes 6 et 11), alors (ligne 14) elles sont la même personne (lignes 15-16).

La règle en Figure 132 indique que si deux personnes (lignes 3 et 6) ont le même e-mail (lignes 4 et 7) alors elles sont la même personne (ligne 11-12).

La règle en Figure 133 indique que si deux documents (lignes 3 et 8) ont le même titre (lignes 4 et 9), et sont du même auteur (lignes 5 et 10) avec la même date (lignes 6 et 11) alors ils sont un même document (lignes 15-16).

En utilisant une feuille de style XSLT, nous transformons automatiquement des règles affirmant une équivalence en des modèles de substituts. Comme le montre la Figure 134 les règles ont été transformées en trois modèles de substituts : deux modèles pour la classe Personne (lignes 1-5 et 7-9) et un modèle pour la classe document (lignes 11-15).

CORESE a la capacité de charger des requêtes dans sa base de connaissances comme des annotations RDF et d'interroger ensuite cette base de requêtes pour extraire des requêtes prédéfinies, correspondant par exemple à un type donné.

```
<cos:rule>
  <cos:if>
    <ex:Person rdf:about="?person1">
      <ex:name>?name</ex:name>
      <ex:firstname>?firstname</ex:firstname>
      <ex:birthdate>?birthdate</ex:birthdate>
    </ex:Person>
    <ex:Person rdf:about="?person2">
      <ex:name>?name</ex:name>
      <ex:firstname>?firstname</ex:firstname>
      <ex:birthdate>?birthdate</ex:birthdate>
    </ex:Person>
  </cos:if>
  <cos:then>
    <ex:Person rdf:about="?person1">
      <owl:sameAs rdf:resource="?person2" />
    </ex:Person>
  </cos:then>
</cos:rule>
```

Figure 131. Règles pour détecter l'équivalence entre personnes (nom & naissance).

```
<cos:rule>
  <cos:if>
    <ex:Person rdf:about="?person1">
      <ex:email>?email</ex:email>
    </ex:Person>
    <ex:Person rdf:about="?person2">
      <ex:email>?email</ex:email>
    </ex:Person>
  </cos:if>
  <cos:then>
    <ex:Person rdf:about="?person1">
      <owl:sameAs rdf:resource="?person2" />
    </ex:Person>
  </cos:then>
</cos:rule>
```

Figure 132. Règle pour détecter l'équivalence entre personnes (adresse mél).

```
<cos:rule>
  <cos:if>
    <ex:Document rdf:about="?doc1">
      <ex:title>?title</ex:title>
      <ex:author rdf:resource="?author" />
      <ex:date>?date</ex:date>
    </ex:Document>
    <ex:Document rdf:about="?doc2">
      <ex:title>?title</ex:title>
      <ex:author rdf:resource="?author" />
      <ex:date>?date</ex:date>
    </ex:Document>
  </cos:if>
  <cos:then>
    <ex:Document rdf:about="?doc1">
      <owl:sameAs rdf:resource="?doc2" />
    </ex:Document>
  </cos:then>
</cos:rule>
```

Figure 133. Règle pour détecter l'équivalence entre documents

```
<ex:Person>
 <ex:name>?name</ex:name>
 <ex:firstname>?firstname</ex:firstname>
 <ex:birthdate>?birthdate</ex:birthdate>
</ex:Person>

<ex:Person>
 <ex:email>?email</ex:email>
</ex:Person>

<ex:Document>
 <ex:title>?title</ex:title>
 <ex:author rdf:resource="?author"/>
 <ex:date>?date</ex:date>
</ex:Document>
```

Figure 134. Substituts générés

Dans notre cas, on peut charger tous les modèles de substituts et récupérer ensuite pour un type de concept la liste des modèles de substituts concernant ce type et ses supertypes et les fusionner comme vu précédemment.

Pour en revenir à notre premier exemple de la Figure 122, on peut maintenant étendre cette requête avec en option des propriétés de substituts comme le montre la Figure 135, pour la classe Document (lignes 6-8) et pour la classe Personne (lignes 9-11).

```
01          ?d rdf:type      ex:Document
02          ?d ex:author   ?a
03          ?a rdf:type      ex:Person
04          ?a ex:name      ?n
05          FILTER( regex( ?n, ".*aiman.*"))
06          OPTIONAL { ?d  ex:title   ?p1 }
07          OPTIONAL { ?d  ex:date    ?p2 }
08          OPTIONAL { ?d  ex:author    ?p3 }
09          OPTIONAL { ?a  ex:firstname ?p4 }
10          OPTIONAL { ?a  ex:birthdate ?p5 }
11          OPTIONAL { ?a  ex:email ?p6 }
```

Figure 135. Augmentation d'une requête par la technique des substituts

Une feuille de style transforme en une page XHTML le résultat comprenant maintenant les substituts ; l'affichage est amélioré comme le montre la Figure 136.

On peut remarquer que dans cet exemple, la date de naissance n'est ni disponible ni utile et pour "Neil Gaiman" et l'e-mail n'était pas disponible. Mais comme les propriétés ajoutées par les substituts sont facultatives, elles n'entravent pas la résolution de la requête ou l'affichage du résultat. En ce qui concerne les utilisateurs ou les services émetteurs d'une requête, ils soumettent la même requête qu'initialement, mais obtiennent des résultats avec des informations supplémentaires qui pourraient s'avérer utiles lorsque tout ou partie de la structure conceptuelle de ce résultat doit être affichée, prononcée, ou rendue de toute autre façon.

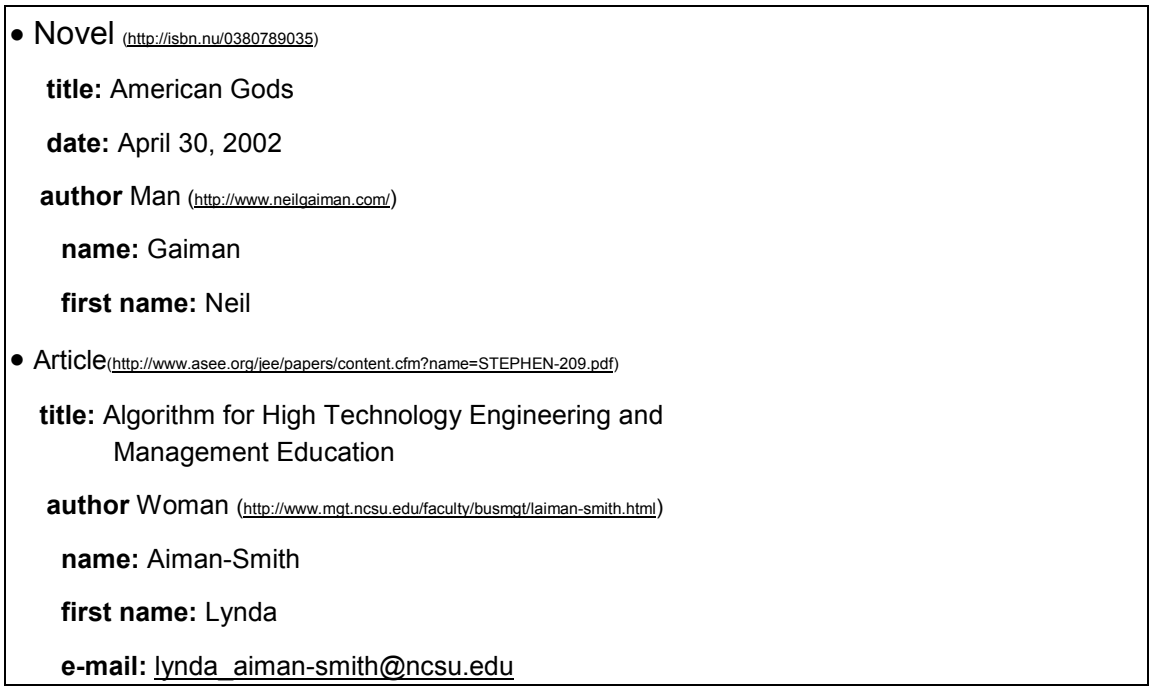


Figure 136. Une réponse augmentée par les résultats des substituts.

Dans cette section, nous nous sommes concentrés sur un problème que nous avons rencontré dans nos projets : la génération de représentations sémiotiques pour des structures conceptuelles telles que les annotations et les résultats de requêtes sur le web sémantique.

Nous ne considérons pas que la génération automatique est la seule alternative, mais selon notre expérience, nous avons constaté qu'il n'est pas raisonnable de s'attendre à ce que les utilisateurs indiquent chaque attribut pertinent pour chaque modèle de substitut d'une classe présente dans l'ontologie de l'application qu'ils utilisent. C'est pourquoi nous nous sommes penchés sur les moyens d'automatiser la production de substituts.

En nous appuyant sur le parallèle entre la structure de ces substituts et la notion de condition d'identité, nous avons proposé et expliqué un mécanisme d'exploitation des règles pour automatiser la génération de candidats substituts. Nous avons montré comment ces modèles pouvaient améliorer une représentation, par exemple lors de la visualisation des résultats d'une requête.

L'approche s'est focalisée sur la génération de modèles fournissant les propriétés à inclure dans un substitut, quelle que soit la façon dont il sera rendu (texte, graphiques, discours, etc.). Nous avons suivi une approche opportuniste où le candidat final est obtenu comme une union de candidats détectés. Une différenciation plus poussée exige une intervention humaine. Cependant, comme les substituts, ces règles d'intégration sont dépendantes du domaine et du scénario considéré : dans un système où les noms, prénoms, départements sont suffisantes pour conclure une équivalence de personnes, ces propriétés peuvent également être un bon substitut.

La qualité des modèles peut se détériorer quand des règles complexes d'équivalence sont mises en place. Nous pensons que la technique est à intégrer à une approche semi-automatique, où le système propose un certain nombre de modèles qui sont ensuite examinés par un ontologue ou raffinés à travers des techniques de sélection et d'apprentissage utilisant les retours des utilisateurs.

On peut aussi remarquer que souvent des sous-types nécessitent moins de propriétés pour leurs substituts que leurs supertypes (par exemple, l'économiste John Adam Smith est assez peu ambigu de par son type) et que cela est en conflit avec notre description. Tout d'abord, les propriétés supplémentaires introduites par un sous-type peuvent être d'excellentes conditions d'identité mais de très pauvres propriétés de substituts par exemple le numéro ISBN pour un livre ou un document est excellent pour définir une condition d'identité, mais il ne change rien au fait que le titre et les auteurs restent le meilleur moyen de décrire le livre, comme un document en général. Deuxièmement, nous ne disons pas que toutes les propriétés choisies sont nécessaires, un tel choix dépend du scénario, du contexte, des profils des utilisateurs et des cas d'usage. Notre objectif était de détecter un maximum de ces propriétés qui étaient potentiellement intéressantes, puis de voir comment ajuster leur sélection.

En outre, notre approche et sa mise en œuvre est basée sur des règles parce que la plate-forme CORESE est basée sur les graphes conceptuels et des règles de graphes. Dans d'autres plates-formes offrant d'autres formalismes, d'autres sources que les règles pourraient être exploitées pour en déduire des substituts.

Quand un substitut contient des relations avec d'autres ressources, il est tentant d'appliquer la même technique à ces ressources. Toutefois, cette récursivité est un processus qui pourrait aboutir à la récupération de toute la base de connaissances. Dans une de nos précédentes tentatives, nous avons utilisé la technique de l'holophrastage pour résoudre ce problème par l'introduction de widgets qui permettent aux utilisateurs de demander des informations supplémentaires sur une feuille de résultat. L'amélioration de la visualisation ne signifie pas que nous devions assurer l'identification à 100% par les substituts ; l'holophrastage peut permettre aux utilisateurs de demander un complément d'information sur une ressource et un début de dialogue de désambiguïsation avec le système, comme nous le faisons lorsque nous parlons entre humains. Ceci est préférable à l'agrégation de trop de détails dans les réponses.

Une autre heuristique prometteuse consiste à appliquer l'intégralité des modèles de substitution pour chaque nœud de la requête et ensuite seulement appliquer les propriétés de type littérales des modèles de substituts aux ressources liées à ces nœuds, en ignorant les propriétés type objet. Cette heuristique joint à l'holophrastage semble un bon compromis.

Enfin, nos expériences ont souligné la nécessité d'un modèle plus riche des liens entre les structures conceptuelles du web sémantique et le niveau sémiotique du web classique pour l'homme. Dans une déclaration d'intérêt [178], l'auteur appelle à la participation des sémioticiens dans les modélisations et les mécanismes d'inférence qui sous-tendent le web sémantique, mentionnant un travail sur l'algèbre sémiotique [181]. Cela devient indispensable pour un web omniprésent au travers de dispositifs multimodaux et multimédias et de plus en plus mobile tous les jours. La question du lien entre les ontologies et les systèmes sémiotiques [182] avec lesquels elles interagissent doit être explorée de manière approfondie. Pour aller encore plus loin, il est nécessaire de modéliser la combinaison de ressources du web sémantique et du web pragmatique pour produire des ressources du web sémiotique [183].

La génération des interfaces pour le web sémantique sera dynamique et utilisera : les profils utilisateurs, le contexte et l'historique des interactions, des primitives de modélisation sémiotique ajoutées à nos méta-modèles, des signes liés aux primitives de nos ontologies, des logiques de la sémiotique et la génération de substitut, en plus des structures conceptuelles à communiquer aux utilisateurs.

6.5 Wiki sémantique : SweetWiki

Le web ouvert et les intrawebs fonctionnent souvent par fertilisation croisée et de nouveaux usages du web rencontrent actuellement une attente sur les intrawebs et les sites communautaires : des wikis internes sont utilisés pour faciliter l'édition et l'organisation des documents d'entreprise, des blogs sont utilisés pour les news et la veille technologique, le web devient mobile, les utilisateurs sont de plus en plus nomades et les communautés ont des limites virtuelles.

Ces nouvelles évolutions offrent des perspectives d'extension de la notion de web sémantique communautaire et posent de nouveaux problèmes de recherche en particulier:

- Peut-on concilier des interfaces d'édition collaborative simples (ex : wiki) et l'approche web sémantique pour offrir des portails ergonomiques et performants ?
- Comment assister le cycle de vie des ressources d'un web sémantique ? Comment utiliser ces formalismes et des architectures telles que les services web pour opérationnaliser dynamiquement ces workflows ?
- Comment intégrer le contexte d'un utilisateur dans les inférences et les interactions ? Quelles connaissances décrivent le contexte ?
- Quelles sont les nouvelles contraintes de sécurité et de confidentialité ? S'expriment-elles aussi avec des ontologies ? Quelles nouvelles inférences seront alors nécessaires ?

Nous participons à deux projets s'intéressant à certains de ces problèmes : SeWeSe [75] et SweetWiki [33]. SeWeSe est un serveur web sémantique, une plate-forme de génération de portails reposant sur les technologies du web sémantique et une extension des langages du web classique pour intégrer ces nouvelles capacités de navigation et requêtes aux cœurs des technologies classiques du web. SeWeSe met l'accent sur les capacités déclaratives et dynamiques dans la construction des applications web en permettant de piloter un maximum d'opérations à partir des modèles OWL lite sous-jacents.

SweetWiki est l'une des applications web développées au-dessus de SeWeSe et propose de reconcevoir un wiki en reposant sur les technologies du web sémantique pour implanter et exploiter des fonctionnalités de "*social tagging*" au cœur d'un wiki entièrement décrit et structuré en OWL lite. SweetWiki est aussi un scénario pour des actions de standardisation comme GRDDL ou RDFa. C'est sur SweetWiki que nous allons axer cette section.

Les wikis ont été conçus au milieu des années 90 en exploitant les technologies web de l'époque : HTML, HTTP, CGIs et URL. Pour compenser le manque d'outils simples d'édition et de stockage, les Wikis ont développé des variantes de langages de marqueurs ou « WikiML » mais il n'existe aucun WikiML standard même si plusieurs efforts ont été entrepris.

Une autre caractéristique des wikis réside dans l'utilisation des WikiMots[18] pour indiquer des liens hypertexte et dans la mise à disposition de mécanismes simples pour la gestion de version ou l'édition concourante. L'idée de SweetWiki est de revisiter la logique de conception des Wikis, en tenant compte des nouveaux standards disponibles pour le web onze ans après, afin de palier à certaines des imperfections identifiées par les retours d'expérience. Après l'évaluation de plusieurs moteurs wiki nous avons décidé d'écrire un nouveau moteur parce que notre vision du wiki du futur n'était pas compatible avec ce que nous avons trouvé dans des wikis existants. Nous voulions :

- *reposer sur les standards du web* : standards pour le format des pages wiki (XHTML), pour les macros incluses dans les pages (JSPX/XML), etc. ;
- *inclure un environnement pour structurer et interroger le wiki dans les pages elles-mêmes* : le wiki est articulé autour d'un moteur web sémantique qui nous permet d'utiliser des langages de description du web sémantique (RDFa[19], RDF, RDFS, OWL) et le langage de requêtes SPARQL[20] pour respectivement décrire et interroger le modèle du Wiki, les ontologies de domaine et les métadonnées indexant les pages et leur contenu ;
- *permettre de taguer collectivement*, et d'inclure des ontologies externes ;
- *abandonner les dialectes de WikiML* employés et modifiés par la plupart des wikis ;
- *améliorer l'accès à l'information* : navigation par facettes et des outils de recherche ;
- *permettre l'édition de métadonnées* : en fusionnant dans une seule interface l'édition du contenu d'une page et de ses métadonnées et dans une approche WYSIWYG *i.e.* avec des éditeurs le plus naturels possibles ;
- *rendre les métadonnées publiques* : en permettant aux métadonnées d'être extraites et exploitées par d'autres applications.

Cette section reprend un article écrit avec Michel Buffa [33]. Dans la partie suivante nous proposons un état de l'art sur les wikis sémantiques. Puis nous expliquons le positionnement original de SweetWiki et comment il accepte de multiples ontologies, y compris celle de son propre modèle. Ensuite, nous expliquons comment il est possible de créer dans SweetWiki des « pages d'applications » contenant des requêtes. Enfin nous montrons comment une ontologie légère obtenue à partir d'une « folksonomie[21] » et composée des mot-clés (des « tags ») que les utilisateurs utilisent pour annoter les pages, images, vidéos, peut être utilisée pour améliorer la navigation, la recherche, les fonctionnalités de notification et de veille.

[18] WikiMot : une chaine composée d'au moins deux mots capitalisés et qui devient un lien hypertexte.

[19] http://www.w3.org/2001/sw/BestPractices/HTML/2006-01-24-rdfa-primer

[20] http://www.w3.org/TR/rdf-sparql-query/

[21] La folksonomie représente « le vocabulaire de la communauté des utilisateurs ».

6.5.1 Wikis et Wikis sémantiques

Beaucoup de wikis sémantiques sont actuellement développés [33] et nous pouvons distinguer les approches considérant « l'utilisation des wikis pour des ontologies » des approches considérant « l'utilisation des ontologies pour des wikis » ; très peu de moteurs fusionnent actuellement les deux approches. La plupart des projets en cours sur des wikis sémantiques entrent dans la première catégorie c.-à-d. qu'ils considèrent chacune des pages du wiki comme des concepts et les liens typés (dans le contenu des pages) comme des propriétés. Dans ce modèle, appelé « Wikitology » [184], le Wiki devient l'interface d'un système de maintenance d'une ontologie.

Un des premiers wikis à entrer dans cette catégorie est Platypus [185] qui impose d'éditer séparément les métadonnées pour chaque page du wiki dans une deuxième page dédiée. Il permet l'édition de base d'une ontologie mais sans contrôler la cohérence avec les annotations. Il ne permet pas de raisonnements et ne permet que des requêtes simples. Les métadonnées sont employées pour l'amélioration de la navigation mais les utilisateurs doivent osciller entre l'édition de la page et l'édition des annotations sémantiques puisque ces activités reposent sur deux éditeurs distincts. SHAWN [186] offre des options semblables.

Rise [184] entre également dans la première catégorie : l'ontologie utilisée par la communauté est éditée par l'intermédiaire du Wiki lui-même et un ensemble de conventions de nommage est employé pour construire automatiquement l'ontologie à partir du contenu du Wiki. Un langage propriétaire est utilisé pour décrire les métadonnées et l'exportation en RDF est possible. L'information sémantique est utilisée pour la navigation et le contrôle de la cohérence. L'ontologie est reconstruite chaque nuit pour tenir compte des évolutions.

Rhizome [187] utilise une version modifiée de WikiML (ZML) qui repose sur des conventions de formatage pour rendre les propriétés sémantiques explicites directement dans le contenu de page. Les pages sont sauvées en RDF et un autre éditeur peut être employé pour modifier le RDF directement. Ce dispositif met la structure du wiki à risque et Rhizome fournit donc des mécanismes d'autorisation précis et des outils de validation. Il n'est pas clairement dit comment les métadonnées améliorent le comportement du wiki ; il n'y a pas de recherche avancée et pas d'aide à la navigation. RDF-Wiki[22] est semblable à Rhizome en ce qu'il permet l'export des annotations RDF pour le traitement externe.

Semantic MediaWiki [188] est basé sur MediaWiki. Ici, contrairement à Rise [184], typer des liens peut également être employé pour indiquer des attributs de la page. Par exemple, le texte `San Diego est une [[est un::ville]] située dans le sud-ouest de [[est localisé dans::La Californie]]` établit les faits « San Diego est une ville » et « San Diego est situé en Californie ». Tandis que le texte `ses coordonnées sont [[coordonnées:=32°42'54"N,117°09'45"W]]` définit un attribut « coordonnées ». Ces

[22] http://infomesh.net/2001/05/sw/#rdfwiki

données sont employées pour la navigation par facettes. Semantic MediaWiki traduit ces métadonnées en RDF et le moteur de raisonnement KAON2 peut être utilisé. D'autres extensions sémantiques de MediaWiki sont développées [189].

Makna [190] est basé sur JSPWiki et fournit des extensions sémantiques basées sur des liens typés. Il repose sur JENA pour permettre des requêtes complexes. Son éditeur textuel propose des formulaires avec auto-complétion pour questionner le moteur JENA et rechercher des concepts/propriétés/relations ; ceci est utile en particulier lorsque l'ontologie est grande.

WikSar [186] [191] permet à des utilisateurs d'écrire des annotations sémantiques dans l'éditeur textuel du wiki en utilisant des WikiMots. Par exemple : si une page appelée « `PrinceHamlet` » contient une ligne « `PersonnageDe : WilliamShakespeare` », ceci sera traduit par un triplet en RDF. En combinant tous ces triplets, une ontologie peut être extraite à partir du contenu du Wiki. L'éditeur est uniquement textuel et propose ni aide ni contrôle de cohérence. Lorsque les pages sont sauvées, les métadonnées sont utilisées pour proposer une navigation par facettes. WikSar permet des requêtes complexes, et celles-ci peuvent être incluses dans les pages du wiki ou dans leurs modèles. Un outil de navigation graphique supplémentaire peut être utilisé pour explorer le wiki par son réseau des métadonnées.

Les liens typés sont créés rapidement mais l'utilisateur doit se rappeler chaque concept, relation, et propriété qu'il veut utiliser et ceci n'est pas évident à l'usage. Avec AceWiki[23] on peut ajouter et modifier des phrases écrites en utilisant le langage ACE[24] (un anglais contrôlé), à l'aide d'un éditeur interactif. Cet éditeur prend en compte l'ontologie et fournit des conseils à l'utilisateur en proposant des liens valides lors de la frappe. L'éditeur peut aussi être utilisé pour étendre l'ontologie en créant de nouveaux concepts, rôles et individus.

La deuxième famille des approches se concentre sur « l'utilisation des ontologies pour les wikis ». IkeWiki [192] combine WikiML et édition WYSIWYG du contenu et des métadonnées des pages. L'éditeur fournit certains dispositifs dynamiques comme l'auto-complétion sur les métadonnées. Il suppose qu'une ontologie existante a été chargée. Il fournit aussi une aide à l'édition de l'ontologie. Il emploie JENA comme un moteur de raisonnement, et les métadonnées sont utilisées pour la navigation et le rendu des pages. Des annotations peuvent être visualisées dans un cartouche à côté de la page et IkeWiki propose une interface utilisateur très agréable à l'usage.

SweetWiki, notre prototype, entre également dans cette deuxième catégorie. Il ne suit pas le modèle Wikitology mais nous avons fait le nécessaire pour permettre une telle évolution dans le futur. Nous permettons de taguer des pages (social tagging), les tags générant une folksonomie ; nous permettons aussi l'utilisation d'ontologies externes. SweetWiki est proche de WikSar puisque ces deux moteurs wiki partagent beaucoup de

[23] http://gopubmed.biotec.tu-dresden.de/AceWiki/

[24] Attempt to Controlled English (ACE) www.ifi.unizh.ch/attempto/

dispositifs comme la construction d'une ontologie par l'usage, l'inclusion de requêtes dans les pages, l'édition des métadonnées et du contenu de la page dans la même interface. SweetWiki ajoute un moteur de raisonnement et un éditeur WYSIWYG extensible pour le contenu et les métadonnées, (comme IkeWiki ou Makna). Nous avons écarté l'utilisation de WikiML car nous avions précédemment rencontré beaucoup de problèmes pendant la traduction entre le WikiML et XHTML. L'éditeur de SweetWiki utilise AJAX et l'annotation des pages est motivée auprès des utilisateurs par le fait que : (a) ils peuvent voir un affichage instantané des liens / facettes que l'annotation ajoutera à leur page ; (b) un mécanisme d'auto-complétion suggère des concepts existants de l'ontologie, les catégories relatives et le nombre de pages utilisant cette annotation comme une incitation à réutiliser les tags existants. En outre, SweetWiki vient avec des outils de gestion et d'édition des ontologies pour les tâches de maintenance. Cependant, SweetWiki n'est pas dédié à la gestion collaborative d'une ontologie comme peut l'être par exemple OntoWiki [193].

6.5.2 Le principe de SweetWiki

Les Wikis sont des sites web où des pages sont organisées autour de WikiMots et parfois d'autres concepts tels que les WikiWebs. Pour aller au delà de cette structure informelle d'hyperliens, nous proposons d'utiliser l'annotation sémantique et des fonctionnalités de restructuration. Pour rendre explicite, manipuler et exploiter une telle structure nous avons conçu SweetWiki autour d'une « ontologie de wiki » qui décrit les concepts des wiki eux-mêmes. SweetWiki est également capable de profiter des ontologies externes et des folksonomies développées par ses utilisateurs à travers l'annotation qu'ils font des pages [194], [195], [196].

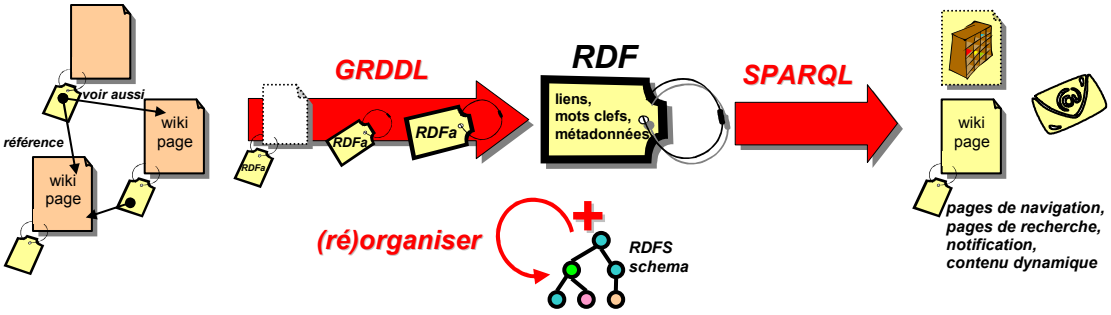


Figure 137. Schéma du principe de SweetWiki

SweetWiki se base sur ce que nous avons appelé « le modèle objet d'un Wiki », en d'autres termes : une ontologie de la structure d'un wiki. Dans beaucoup de moteurs de wiki, y compris plusieurs « wikis sémantiques », les concepts du wiki sont implicites à l'implantation. Poursuivant l'objectif des ontologies informatiques [197] nous les avons rendus explicites. On peut se demander en quoi ceci relève de l'ingénierie des connaissances. Avant tout, le domaine de modélisation n'est pas forcément le domaine d'application. En rendant le modèle de SweetWiki explicite et déclaratif, nous ne sommes pas en train de « reprogrammer » le logiciel. Le langage de programmation utilisé pour la programmation de SweetWiki n'est pas le langage de description ; par contre ce dernier permet de raisonner sur le logiciel.

Dans notre cas nous avons défini en OWL Lite tous les concepts, les propriétés et les relations que nous employons dans le wiki lui-même. Des primitives comme « document », « page », « tag », « lien en avant », « lien en arrière », « contributeur », « version », « fichier joint », « image jointe », etc. sont décrites dans cette ontologie. Les métadonnées générées pour chaque page avec ces primitives, sont incluses dans les pages wiki elles-mêmes.

Il y a plusieurs avantages à employer un modèle déclaratif du wiki : nous pouvons raisonner sur ce modèle (ex : règles de production pour compléter les données, déclaration de relations inverses pour maintenir dualité entre lien en avant et lien en arrière, etc.), nous pouvons l'étendre et le réorganiser (ex : restructuration du wiki), nous pouvons construire au-dessus (ex : interopérabilité entre plusieurs wikis), nous pouvons concevoir des éléments d'interface pour aider la navigation (ex : liste des pages connexes), etc.

En reposant sur RDF, le langage SPARQL peut être employé pour interroger toutes ces métadonnées localement ou à distance. L'ontologie de la structure des wiki est maintenue par les concepteurs du wiki. Considérons par exemple, la requête :

```
PREFIX wiki <http://www.essi.fr/sweetTwiki/wiki.rdfs#>
select distinct ?doc ?type ?page group ?doc
where { ?page wiki:includeDocument ?doc . ?doc rdf:type ?type }
```

Le résultat est l'ensemble des éléments inclus dans une page, ainsi que leur type. Dans l'ontologie du wiki, nous avons défini qu'une page pouvait inclure des documents attachés et qu'une image, un fichier ou une vidéo pouvaient être des documents attachés. La requête citée en exemple affiche donc tous les fichiers, images, vidéos inclus dans des pages. On voit ici l'intérêt d'un tel modèle déclaratif et compatible avec un langage de requête. Il va de soi qu'il serait souhaitable que tous les wikis sémantiques reposent sur des modèles communs. Des efforts tels que le framework SIOC (Semantically-interlinked Online Communities)[25] pour l'aspect communautaire ou WikiOnt [198] pour la structure du wiki ou FOAF[26] pour la description des personnes (et utilisé ici) sont intéressants à ce titre.

SweetWiki permet de taguer collectivement les pages et ressources du wiki. Les pages et les documents joints (images, fichiers attachés) peuvent être annotés dans l'éditeur. Les tags sont des mots-clefs entrés par l'utilisateur et collectés pour former une folksonomie. Afin d'améliorer la navigation tout en maintenant la simplicité native des wikis, nous avons mis en application ce mécanisme d'étiquette/tags/mot-clé avec une ontologie de domaine commune à tout le wiki. En rendant cette ontologie explicite (en RDFS) nous pouvons de nouveau raisonner dessus (ex : trouver des tags proches), nous pouvons faire des requêtes complexes (ex : grouper les pages en fonction de leurs tags ou des profils de leurs auteurs), nous pouvons modifier le wiki (ex : réorganiser

[25] SIOC: http://sioc-project.org/

[26] http://www.foaf-project.org/

l'ontologie, en fusionnant des concepts équivalents, en déclarant des liens hiérarchiques, en important des ontologies, etc.). L'ontologie de domaine est enrichie directement par les utilisateurs et peut être restructurée par des volontaires pour améliorer la navigation et les requêtes. SweetWiki vient avec un éditeur d'ontologies légères qui peut être employé par n'importe quel utilisateur afin d'organiser les tags.

D'autres ontologies peuvent être ajoutées pendant l'exécution et sont alors immédiatement accessibles aux utilisateurs pour taguer et faire des requêtes. Si le wiki est employé dans un domaine pour lequel des ontologies sont déjà disponibles, ces ontologies peuvent être chargées dans le serveur web sémantique de Sewese de SweetWiki. C'est également une excellente manière d'amorcer la collection des tags et cela facilite l'interopérabilité.

L'exécution de SweetWiki repose sur le moteur de recherche sémantique CORESE pour les requêtes et les inférences [72] et sur SEWESE [75], une extension associée fournissant un serveur web, une API et des balises JSP pour construire des interfaces exploitant des ontologies et fournissant aussi des outils génériques (gestion de la sécurité, éditeur d'ontologie, cycle de vie, etc.). Le serveur se base sur une architecture standard d'application web : des filtres contrôlent la session (ex : autorisation, profils d'utilisateur, etc.) et la mise en page (en-têtes, bas de page) ; les pages sont directement disponibles en XHTML ou JSPX pour une lecture rapide ; un servlet gère les pages sauvées ; un ensemble de balises JSP fournissent des fonctionnalités (ex : soumettre une requête SPARQL) ; des bibliothèques javascript implantent l'éditeur WYSIWYG.

SweetWiki intègre un éditeur WYSIWYG de XHTML basé sur Kupu. Cet éditeur a été considérablement étendu afin de permettre:

- l'édition des métadonnées en parallèle avec l'édition du contenu pour décrire les pages et les objets qu'elles contiennent ;
- la validation et l'inclusion de requêtes dans une page éditée ;
- la création simple de liens vers d'autres pages.

6.5.3 Un wiki comme plate-forme d'application

Des logiciel wikis tels que TWiki, JotSpot, Confluence ou encore XWiki sont appelés "*application wikis*" dans la mesure où le langage qu'ils proposent pour formater les pages, inclut de puissantes macros. Ces dernières permettent d'écrire, en éditant des pages du wiki, des « appliquettes » *i.e.* de petites applications très simples mais créées par les utilisateurs eux-mêmes lorsqu'ils détectent un besoin. Par exemple on peut définir un formulaire pour saisir des adresses et numéros de téléphone, afficher la liste des données saisies dans un tableau, faire des recherches, etc. La plupart du temps ces applications sont semblables à celles que l'on peut développer avec Excel, leur intérêt principal étant qu'elles sont en ligne, qu'elles ne nécessitent pas de base de données et qu'un utilisateur peut copier/coller une application écrite par un autre et en faire facilement une version personnalisée dans une autre page, etc.

Aucun des logiciels wikis cités n'utilise d'éditeur WYSIWYG lorsqu'il s'agit d'utiliser des macros dans les pages, ce qui rend l'édition relativement inconfortable. En effet, une étude [199] a montré que seules quelques personnes développent ce type d'applications, néanmoins la somme de ces petites applications finit par transformer le wiki en un outil aux possibilités nombreuses, mieux adapté aux besoins de ses utilisateurs que de gros logiciels génériques.

Dans SweetWiki il est possible d'inclure des requêtes dans des pages, sans quitter l'éditeur WYSIWYG. Ces requêtes peuvent « interroger » n'importe quelle « partie » du wiki en se basant sur le modèle du Wiki. On peut facilement afficher toutes les pages qui contiennent une vidéo insérée par l'utilisateur John Smith au cours de la semaine passée, ou encore afficher tous les types de vidéos acceptés. Les données sont interrogeables aussi bien que le modèle lui-même. Un utilisateur de SweetWiki peut ainsi créer des « pages d'application » en incluant des requêtes complexes dans des pages. L'éditeur WYSIWYG propose même de tester les requêtes sans quitter le mode d'édition.

Regardons comme exemple d'application la gestion des profils utilisateurs. Lorsqu'un utilisateur s'enregistre dans le wiki pour la première fois, une page personnelle est automatiquement créée. Les métadonnées de cette page sont légèrement différentes de celles des autres pages : elles décrivent le profil de l'utilisateur. Dans la version actuelle, un utilisateur peut indiquer (en éditant cette page) une liste de tags qui correspond aux sujets qui l'intéressent. Par exemple, un utilisateur qui indique qu'il est intéressé par les « wikis » se verra proposer une liste de liens vers des pages ou des images concernant ce sujet. Cette liste est affichée dans sa page personnelle mais peut lui être envoyée par mail ou être exportée via un flux RSS. Elle est le résultat d'une requête qui est insérée automatiquement dans la page. Il est intéressant de remarquer que ces requêtes tiennent compte du polymorphisme et des relations d'héritage qui peuvent exister entre les classes des métadonnées manipulées (ici les tags) ; dans l'exemple du wiki organisant des cours Java (Figure 139) si des pages ou des images sont tagguées avec "abstract class" elles seront également proposées comme résultat d'une recherche sur "class", et comme "interface" est une notion sœur, des recherches sur ce tag seront suggérées.

6.5.4 Du web sémantique pour implémenter une folksonomie

SweetWiki permet le "tagging social". Les utilisateurs peuvent taguer des pages, des images, des vidéos, etc.

Comme Bird [200] et Olsen [201] nous proposons une approche mixte afin "d'organiser les tags" : nous relions les tags entre eux au sein d'une folksonomie (vocabulaire émergeant d'une communauté) décrite à l'aide des langages du web sémantique [194] [195]. Les tags sont ainsi organisés dans une hiérarchie et reliés les uns aux autres à l'aide de relations telles `subClassOf`, `seeAlso`, etc. Nous pensons que le tagging social minimise l'effort et maximise la participation des utilisateurs. Des sites comme del.icio.us, flicker, youtube remportent un grand succès et utilisent

massivement les tags ; SweetWiki propose également de taguer pages, images… cependant nous proposons une amélioration : nous avons implémenté un système qui permet à n'importe quel utilisateur d'organiser ces tags. Ce sont les utilisateurs qui sont responsables de la création et de l'organisation de cette « ontologie-utilisateur ». Nous avons montré que cette organisation permet d'améliorer la recherche et la navigation [202] [203].

Même si un très faible pourcentage d'utilisateurs utilise cette possibilité de réorganisation, l'ensemble des utilisateurs bénéficie de chaque amélioration. Des expériences ont également été menées chez EDF [195] et ont confirmé l'intérêt d'une telle approche (même si dans cette expérience, seuls des administrateurs pouvaient mettre à jour la folksonomie). L'utilisation des technologies et des langages standards du web sémantique pour organiser et maintenir la folksonomie permet une grande souplesse dans son exploitation et sa mise à disposition pour d'autres applications.

Comme l'illustre la Figure 138, lors de l'édition d'un document, un utilisateur peut librement saisir des mots-clés dans un champ textuel. Pendant la saisie, un mécanisme d'auto-complétion propose des tags existants en envoyant des requêtes SPARQL au moteur de recherche sémantique (nous utilisons pour l'interface la technologie AJAX). Ces requêtes permettent d'identifier les concepts existants dans la folksonomie dont les labels sont compatibles avec les caractères saisis. Pour chaque suggestion, la catégorie mère (si elle existe) est mentionnée (ceci permet éventuellement de lever des ambigüités d'homonymie) et le nombre de ressources déjà tagguées est indiqué, incitant l'utilisateur à rejoindre un tag déjà présent dans la folksonomie.

Les nouveaux mots-clés sont collectés comme labels (termes) de nouvelles classes attachées par défaut à la catégorie « nouveaux concepts ». Ces classes viennent enrichir la folksonomie « au kilomètre » et plus tard, éventuellement, des utilisateurs peuvent les repositionner dans l'ontologie, les éditer, ajouter des labels dans d'autres langues, créer des relations de synonymie, fusionner des classes, etc. Il suffit qu'une personne fasse une modification pour que toute la communauté en bénéficie. Dans tous les cas le feedback provenant de ces tags est important pour découvrir le vocabulaire de la communauté et son organisation incrémentale, par les utilisateurs, garantit une bonne adéquation avec leurs besoins de classification.

Lors de la visualisation d'une page (Figure 139), le moteur CORESE est utilisé pour générer des widgets de navigation par facette *i.e.* une navigation où l'on propose à chaque étape différents points de vue pour continuer la navigation. La sémantique des tags de chaque page est utilisée pour suggérer des pages « voisines », des catégories « voisines », pour prévenir les utilisateurs intéressés par le sujet dont traite la page, etc. A chaque fois il s'agit de requêtes SPARQL insérées dans les pages du wiki. Ces inférences ne s'appliquent évidement pas aux classes catégorisées sous « nouveaux concepts ».

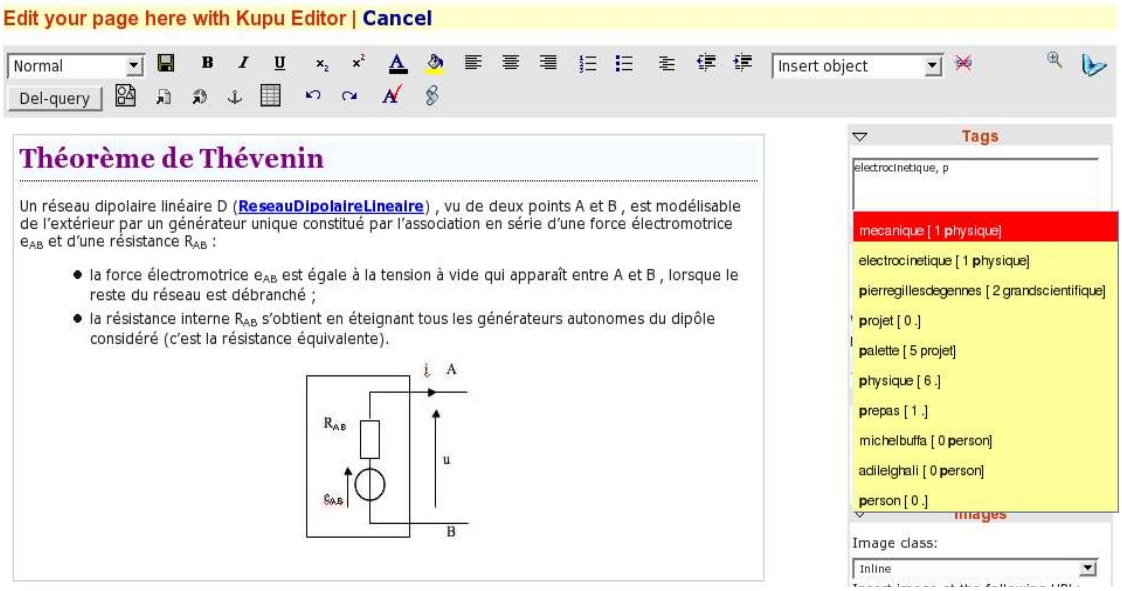


Figure 138. Editeur WYSIWYG et tags pour la page – Communauté ePre du projet IST Palette

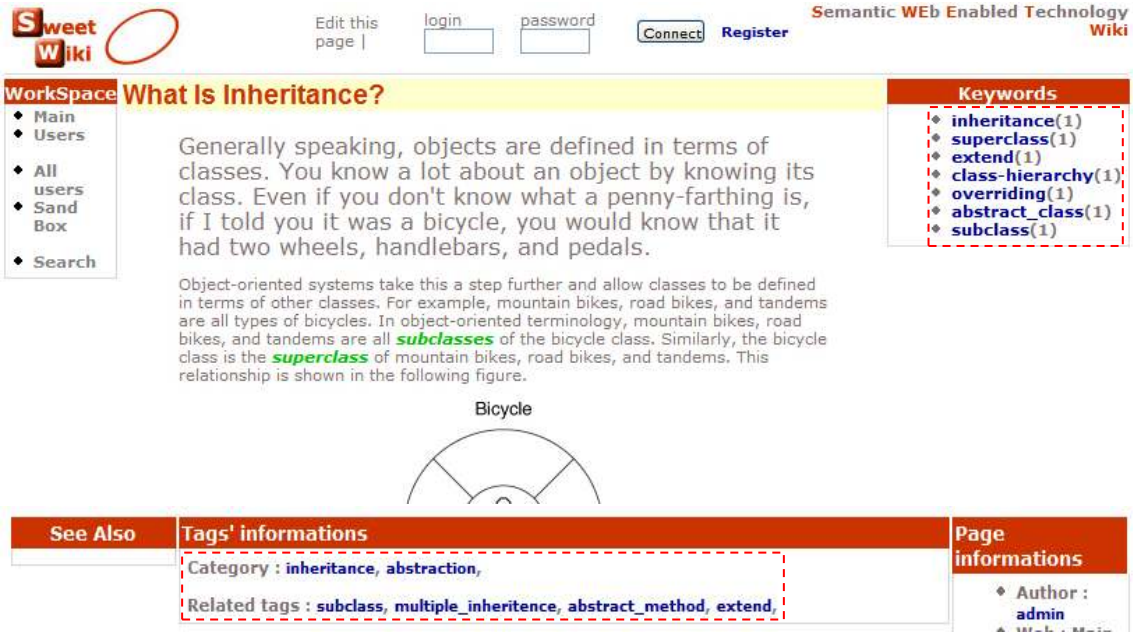


Figure 139. Visualisation de la page, de ses tags et suggestions

6.5.5 Discussion

Avec SweetWiki, nous avons proposé une approche permettant aux utilisateurs d'éditer des pages du wiki et de les tagger tout en utilisant en coulisse les représentations du web sémantique. En outre, des membres de la communauté, des "volontaires du wiki", peuvent consulter le modèle ainsi construit, examiner les tags proposés par les utilisateurs et les réorganiser. Lors de chacune des interventions de ce type, les annotations des utilisateurs demeurent inchangées mais la navigation par facette et la recherche toutes deux basées sur des requêtes sémantiques se retrouvent améliorées en proposant de nouveaux liens.

SweetWiki est disponible en ligne et il est utilisé par plusieurs projets nécessitant une plate-forme de partage d'information entre participants. Ces communautés l'utilisent pour coordonner leur travail et forment un ensemble très hétérogène quant au niveau technique de chacun (on retrouve des chercheurs, des enseignants du secondaire, des secrétaires, etc.) :

- Palette : Il s'agit d'un projet Européen ayant pour sujet l'étude d'outils et de méthodes pour assister l'émergence de communautés d'intérêt et de pratiques ; par exemple le site WikiPrepas[27] est utilisé par la communauté ePrep[28] regroupant des enseignants en CPGE ayant pour but de construire un dictionnaire de mathématiques, physique et chimie pour les classes préparatoires aux grandes écoles scientifiques.
- e-Wok[29] : projet national concernant les techniques de capture du CO_2, il implique la collaboration entre des instituts de géologie et des instituts pétroliers ;
- Tests informels : nous conduisons plusieurs expérimentations informelles avec des élèves d'écoles primaires et des instituteurs, des étudiants du supérieur, des enseignants, des secrétaires, qui l'utilisent pour des tâches quotidiennes.

Des utilisateurs occasionnels, de même que des experts peuvent utiliser le même outil pour écrire de petites applications ou pour améliorer la folksonomie en utilisant l'éditeur d'ontologie embarqué. L'intégration d'ontologies externes ainsi que la possibilité d'insérer du contenu dynamique directement dans les pages, par exemple des requêtes SPARQL, font de SweetWiki une « plate-forme de développement d'application web sémantiques ». Même des concepts au cœur du wiki tels que les pages, liens, images, auteurs, etc. peuvent être étendus et sont accessibles par requêtes.

Nous avons récemment ajouté de nouvelles fonctionnalités comme la possibilité d'insérer des vidéos dans les pages ou l'ajout de contexte sur les tags (qui taggue, où, quand, comment ?) et il a suffi de modifier quelques fichiers XML de manière

[27] http://argentera.inria.fr/wikiprepas

[28] http://www.eprep.org

[29] http://argentera.inria.fr/

déclarative. Nous n'avons pas eu besoin de modifier le code du logiciel, aussitôt les nouveaux concepts sont devenus accessibles via des requêtes (quelles sont les vidéos tagguées avec « wiki »).

Une extension envisagée porte sur le profil des utilisateurs. Nous l'avons vu, les utilisateurs peuvent indiquer leurs centres d'intérêt dans leur page personnelle et tagger les pages librement. En traçant l'historique des tags et leur contexte d'utilisation (qui a taggué une ressource, quelles ressources ont été tagguées, quand un tag a-t-il été ajouté ou supprimé, etc.) il est possible de former un réseau d'accointance entre les utilisateurs. Ainsi, nous avons implémenté de nouveaux outils de recherche et de navigation permettant :

- de trouver les personnes les plus actives sur un tag : cette facette de navigation permet de trouver les personnes qui ont le plus contribué à un sujet.
- d'établir des similarités de comportement entre les utilisateurs : cette fonctionnalité permet de trouver les auteurs qui ont des activités similaires et des centres d'intérêts communs.
- d'établir les relations construites entre les tags au cours de l'évolution du wiki : ce critère de navigation permet de retrouver rapidement les tags qui sont souvent associés et ainsi mieux se documenter sur le contexte relié à un sujet.

Un certain nombre d'autres évolutions sont actuellement envisagées:

- Traitement de la langue naturelle pour taguer automatiquement : plusieurs wikis commencent à analyser le texte des pages pour suggérer des tags potentiels, notamment en regardant si les termes associés aux concepts dans les ontologies connues par le système sont présents dans le texte de la page. La suggestion de métadonnées pourrait utiliser des traitements de la langue naturelle (semi)automatiques et même des approches légères (*shallow parsing*) peuvent améliorer l'utilisation.
- Gestion de versions plus complète : actuellement nous permettons la gestion de versions des pages, mais pas des ontologies sous-jacentes.
- Maintenance collaborative de la folksonomie : fournir des outils wiki pour aider les utilisateurs à gérer le cycle de vie des ontologies et proposer des pages de discussion relatives à ces modifications, garder trace de tout ce qui a été modifié sur les ontologies, etc.

Ce dernier point nous ramène à la vision à double face des wikis pour des ontologies et des ontologies pour les wikis. Cette division des approches actuelles est symptomatique de l'émergence des wikis sémantiques ; il s'agit en effet d'un concept récent. A plus ou moins long terme, il est probable que les deux approches vont fusionner ; plusieurs projets ont déjà commencé à travailler dans ce sens ; l'objectif étant de transformer cette vision à deux faces en un cercle vertueux où les utilisateurs maintiendront le wiki et les ontologies sous-jacentes en même temps, sans aucune distinction. Nous avons conçu SweetWiki comme une plate-forme d'expérimentation pour tester cette vision.

Actuellement, parmi les tests que nous menons, différentes expériences sont suivies : dans certains cas, il nous est demandé de verrouiller une partie de la folksonomie une fois que la classification est satisfaisante (cas de WikiPrepas, les enseignants contributeurs ayant la crainte que la folksonomie ne « dérive »), dans un autre cas (celui d'une entreprise de robotique utilisant SweetWiki pour son intranet et pour son site public communautaire) nous avons décidé de ne pas signaler la présence d'un éditeur de folksonomie. Dans quelques temps nous regarderons le vocabulaire qui aura émergé et commencerons avec l'aide de quelques volontaires à l'organiser. Nous observerons alors la réaction des utilisateurs face à la réorganisation induite dans le wiki (ex : nouveaux liens suggérés). Dans tous les cas, les folksonomies qui sont construites sont des « ontologies légères » ou des représentations intermédiaires utilisables comme point de départ pour générer des ontologies plus formelles.

Enfin, pour conclure cette section rappelons que cette application est aussi une démonstration des standards GRDDL [38] et RDFa [37] auxquels nous avons participé en particulier en rendant public le profile GRDDL de RDFa [35] utilisé pour SweetWiki et en réalisant ainsi l'un des scénarios motivants [36]du W3C.

6.6 Web sémantique et dimension sociale

Cette section prend un peu de recul par rapport aux applications présentées précédemment et reprend un argumentaire développé dans l'article [204] afin de montrer que rien dans les approches du web sémantique n'empêche leur utilisation dans des applications du web social ou du web 2.0

Pour ancrer notre discours nous commençons par une relecture des objectifs et définitions du web sémantique *i.e.* une ontologie de cette notion au sens où nous nous intéressons au web sémantique en lui-même et non tel qu'il peut nous apparaître au travers de travaux singuliers. Le web sémantique est en effet perçu très différemment par de très nombreuses personnes [205]. Non seulement, chacun le voit à sa porte mais en plus chacun s'intéresse à une partie du web sémantique, à l'image du conte indien des six aveugles qui se représentent un éléphant différemment en fonction de la partie que chacun a touchée (Figure 140 issue de [205]).

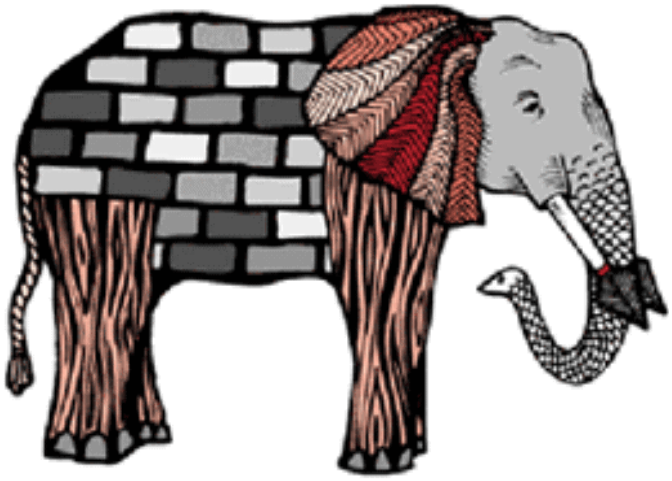


Figure 140. L'éléphant du web sémantique [205]

Le web sémantique est une extension du web actuel dans laquelle l'information se voit associée à un sens bien défini améliorant la capacité des ordinateurs et des hommes à travailler en coopération [4]. Dans cette définition la plus citée dans les travaux du web sémantique, l'ambition de faciliter la coopération entre les machines, entre les hommes et entre les hommes et les machines, est posée comme *l'objectif définitionnel* du web sémantique. Par conséquent le scénario motivant du web sémantique utilisé dans sa définition même, le place au cœur d'une problématique sociale : l'assistance à la coopération en utilisant le médium du web et potentiellement à l'échelle du monde.

Le web sémantique est une évolution et non une révolution du web. Cette caractérisation, toute aussi omniprésente dans les introductions au web sémantique [205] et les documents officiels du W3C [206], a pour corollaire que tout ce que nous pouvons imaginer de faire avec le web classique, y compris les applications sociales, nous pouvons aussi envisager de le faire avec le web sémantique, de la même façon ou même mieux.

Dans l'article [204], nous avons montré que ce qui était perçu comme conflictuel ou opposé dans les articles [207] [208] [209] [210] ne l'était pas : le web sémantique ne s'oppose pas aux dimensions sociales, sémiotiques ou pragmatiques du web ; dans le pire des cas ce sont des usages particuliers qui peuvent être incriminés, pas les standards ni les objectifs. Un point important de la relecture faite dans cet article, est qu'à notre avis la faute n'est pas technique *i.e.* elle n'est pas intrinsèque aux formalismes du web sémantique, mais elle est plutôt due à une difformité du paysage de recherche : beaucoup de chercheurs issus des langages de représentation des connaissances formelles ont vu l'intérêt et ont fait (et font encore) l'effort de porter leurs résultats en étendant les formalismes du web sémantique. D'autres aspects attendent encore que des groupes de travail se forment pour se développer. Ce manque n'est nullement dû aux standards du web sémantique (ils proposent beaucoup de mécanismes d'extensions qui ne demandent qu'à être exploités) ni à leur organisme de tutelle (tous les groupes de travail et leurs listes sont ouverts et toute communauté peut proposer un nouveau groupe).

Nous considérons que les articles [207] [208] [209] [210] ont raison de défendre l'importance de la dimension sociale dans la construction d'un cycle de vie du web sémantique. Cependant, comme le rappelle Tim Berners-Lee [211], le web sémantique est avant tout une vision. Cette vision est celle de l'intégration de données à l'échelle du web, celle de la connexion des données à leur définition et leur contexte, pour inférer, pour permettre une collaboration effective et une réutilisation à différentes échelles (personne, groupe, organisation, communauté, etc.), pour réduire les coûts technologique et social de l'intégration effective des données en réseau, pour ajouter des données manipulables par les machines là où n'y avait que des données manipulables par les humains, le tout en reposant sur des standards de représentation ouverts assurant la flexibilité des inscriptions [205]. Cette vision ne doit pas être confondue avec les formalismes qu'elle mobilise et encore moins avec une famille particulière de formalismes. Cette vision n'est pas non plus figée et s'enrichit régulièrement de nouvelles perspectives (ex : politiques sociales, règles, services, [205]).

Nous pensons aussi que, ironie chronique dans des communautés pluridisciplinaires, les débats sont biaisés par une ambiguïté des termes qui laisse lire aux lecteurs ce que l'auteur n'aurait pas voulu écrire. En particulier, dans le contexte d'une publication sur des aspects formels du web, des termes comme "vérité", "inférence", "confiance", "sémantique" ou "interopérabilité" prennent un sens parfois très différent de celui qui leur serait associé, par exemple, dans une publication sur les aspects cognitifs, sociaux ou ergonomiques. Dans un contexte pluridisciplinaire, il nous semble particulièrement important de prendre ce point en compte car, nombre de fois, le désaccord est terminologique et pourrait se résoudre si l'on accordait plus systématiquement le bénéfice du doute aux mots lors de leur interprétation. Si nous pensons, comme l'un de nos critiques, que le débat plonge parfois ses racines dans des questions plus profondes et moins contemporaines que les choix de conception du web sémantique, nous pensons aussi qu'un certain nombre de désaccords sont moins profonds qu'il n'y parait et relèvent plus de l'alignement de jargons que de l'alignement conceptuel.

Nous ne remettons en aucun cas en cause l'intérêt d'une prise en compte des aspects sociaux pour le développement d'extensions complètes et viables des outils actuels du web ; nous pensons bien au contraire que, plus généralement, l'utilité et l'utilisabilité d'une solution est fonction de l'implication des intéressés dans sa genèse et donc, qu'en particulier, les aspects humains doivent être pris en compte dès les spécifications initiales.

Nous ne remettons pas non plus en doute la prééminence actuelle de travaux sur les représentations formelles dans les contributions au web sémantique. Nous proposons simplement de voir cette hypertrophie comme un départ et une croissance asynchrones plus que comme un amalgame entre le web sémantique et un formalisme particulier de la représentation des connaissances. Le web sémantique est avant tout une vision et ne doit pas être confondu avec les formalismes qu'il mobilise et encore moins une famille de formalismes particulière. On ne peut nier que la majorité des contributions faites jusqu'à présent soient issues d'un (trop) petit nombre de domaines. Cependant, à notre humble avis, il ne faut pas sacrifier cette vision parce qu'elle est particulièrement utilisée dans une communauté. A notre sens, cette difformité de naissance ne justifie pas d'écarter le fond. De plus, pour nous, il ne faut pas non plus confondre les formalismes du web sémantique et leur opérationnalisation pour une application particulière. Nous pensons que le risque majeur de ces amalgames est de mettre en danger l'opportunité qui est donnée à la communauté de l'acquisition et l'ingénierie des connaissances d'intégrer ses différentes contributions au sein d'une même famille de langages de représentation avec la possibilité d'une diffusion et d'une utilisation à l'échelle mondiale.

Enfin l'article [204] ne remet pas en doute le fait qu'il manque encore beaucoup de travail pour atteindre la vision du web sémantique. Si RDF/S et OWL RDF/S ont atteint une visibilité qu'aucun langage de représentation n'avait atteint avant, ils n'ont pas pour autant résolu tous les problèmes qui se posaient déjà avec les autres langages de représentation des connaissances et nous ne les élevons pas en panacée. Ainsi, et à titre d'exemple, le niveau sémiotique manquant au dessus des ontologies est une de nos préoccupations actuelles [159] qui demande un travail pluridisciplinaire de fond pour remplacer les primitives sémiotiques atrophiées actuelles par des représentations plus complètes. Cependant, nous pensons aussi qu'il y a déjà beaucoup de constructeurs dans les formalismes du web sémantique et qu'il est important de poursuivre leur réutilisation systématique, une tâche qui, nous l'admettons volontiers, est rendue fastidieuse par la complexité et le nombre des recommandations du W3C.

Contrairement à l'interprétation des relecteurs de l'article [204], nous ne pensons pas que réaliser une application du web sémantique signifie implanter une solution uniquement avec les outils du web sémantique, bien au contraire : une application propriétaire qui gère un agenda électronique est une application du web sémantique dès lors qu'elle permet simplement d'exporter et d'importer ses données dans un des langages du web sémantique. Il ne lui est pas demandé d'avoir un moteur de règles, des algorithmes de tableaux ou un opérateur de projection implantant SPARQL. Le seul effort qui lui est demandé est celui de faire le seul pas vers l'interopérabilité qui ne peut

être fait à sa place : rendre explicites ses structures de données et la conceptualisation sur laquelle elle se base. C'est le défi des ontologies [197] mais à l'échelle du web.

Dans la continuité de la question de l'application et pour répondre à une autre critique, la socialité du web sémantique est pour nous un sujet trop vaste pour être simplement qualifié. En particulier, la socialité, telle que nous la comprenons, pourrait se décliner différemment pour différentes opérationnalisations envisagées. Or, s'il est une chose que le W3C se garde d'imposer dans ses recommandations c'est l'opérationnalisation du web sémantique : les formalismes proposés sont des outils de représentation où rien ne prescrit les représentations auxquelles ils seront appliqués (indépendance au domaine), ni *si* ou *comment* ils doivent être traités (indépendance à l'opérationnalisation), ni le statut de ce qu'ils représentent (éphémère *vs* immuable, en cours d'élaboration *vs* stable, public *vs* privé, informel *vs* officiel, etc.)

Toujours dans la continuation de cette indépendance à l'opérationnalisation, nous pensons que des scénarios complexes et des solutions complètes pourront articuler des phases reposant sur différentes opérationnalisations de ces mêmes langages et ce au sein d'une même application:

- Un forum ouvert pour les discussions sur un index de ressources d'informations ;
- Des phases d'exportation faisant appel à des outils de formalisation pour entretenir l'index ;
- L'appel à des outils du traitement de la langue pilotés par des formalisations pour indexer des ressources ;
- L'appel à des outils de recherche pour naviguer dans les ressources indexées ;
- L'appel à des outils de veille pour détecter de nouveaux concepts et informer le forum ; etc.

Nous pensons corollairement que dissocier les socialités comme des étapes est possible, mais que les dissocier trop est dangereux et va à l'encontre du besoin de pluridisciplinarité posé par ces scénarios. Pour nous, le fait que chaque formalité requiert des technologies et des méthodologies différentes ne justifie pas le rejet des formalismes de base du web sémantique : ils ont été conçus dans une optique d'indépendance au domaine et à l'opérationnalisation qui mérite au contraire que l'on s'intéresse à la réutilisation de ces formalismes dans différentes opérationnalisations.

Rappelons l'exemple de SKOS : nous pourrions dire que RDF/S et OWL sont inutilisables pour représenter des ressources linguistiques puisque tournés vers la modélisation de classes mais ce faisant on ferait l'erreur de croire que RDF/S et OWL prescrivent de représenter les termes dans l'ontologie, or si l'on considère que notre domaine est celui des terminologues alors l'ontologie représente les notions de "terme", "label", "hyponymie", etc.

Il en va de même pour l'interopérabilité : l'interopérabilité recherchée dans une tâche ou un scénario n'est pas forcément au même niveau que l'interopérabilité recherchée dans une autre tâche ou un autre scénario. De plus dans un même scénario l'interopérabilité (computationnelle) peut être à un certain niveau ou sur un certain domaine (ex : actes du langage) et l'intercompréhension (ou interopérabilité entre utilisateurs) peut porter sur un autre domaine (ex : arguments de conception échangés et organisés par ces actes du langage). Là encore, les multiples interprétations d'un mot comme "interopérabilité" peuvent rendre la formulation des conclusions d'un domaine de recherche ahurissante aux yeux d'un autre domaine.

Pour conclure nous voudrions redire que nous ne devrions pas condamner les formalismes du web sémantique pour n'avoir pas encore réalisé toutes les extensions nécessaires et imaginables ; d'autant plus que, contrairement à leurs ancêtres en représentation de connaissances, ils ont mis en place les moyens techniques (mécanismes d'extension) et sociaux (groupes de travail ouverts) pour les accommoder et n'attendent que l'investissement des communautés concernées (ex : Semantic Web Best Practice Working Group [212]).

Le web sémantique est une formidable opportunité de construire le web pragmatique et la conscience du contexte : il offre une brique de base pour l’assistance que nous aimerions développer à ce niveau. Il serait fort dommage de sacrifier cette initiative sur l'autel des chapelles de recherche.

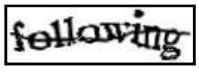
6.7 Vers des ontologies à l'état sauvage

Avant de conclure, nous aimerions revenir sur un point abordé dans la section 3.4 : les ontologies et les folksonomies. Alors que précédemment nous soulignions l'intérêt d'un point de vue représentation de concilier, notamment, ces deux approches, nous voudrions revenir ici sur les aspects « interaction » de ce rapprochement comme nous l'avons présenté dans un article [213].

En ingénierie des connaissances, l'acquisition des connaissances a longtemps été un goulot d'étranglement notamment parce que les connaissances formalisées, et en particulier les ontologies, ne se reproduisaient qu'en captivité. Les ingénieurs de la connaissance ont maintenant un choix d'outils et de méthodes pour analyser des corpus, concevoir et vérifier les modèles de connaissances, formaliser et échanger leurs modèles, mais ces approches font encore des ontologies une chasse gardée d'un petit nombre d'experts. Même si un grand nombre de méthodes et d'applications mettent l'utilisateur dans la boucle, les questions du degré de formalisation auquel il peut être exposé, du surplus de travail et de la surcharge cognitive engendrés, sont encore au cœur de débats sur la conception et l'utilisabilité d'applications à base d'ontologies. Même les approches socio-sémantiques [214] ou reposant sur des collecticiels [215] restent focalisées sur des scénarios d'usage précis (ex. aider à structurer l'argumentation entre experts) et génèrent de nouvelles tâches que l'utilisateur doit réaliser en plus de ses tâches habituelles. La construction des ontologies à l'état sauvage doit se faire au fil de l'eau, au fil des tâches habituelles de l'utilisateur.

Dans ces nouvelles approches un point important est que l'utilisateur n'est plus simplement le commanditaire d'un service pour lequel il fournit des entrées et attend des sorties, mais devient une ressource computationnelle de l'architecture logicielle ; une ressource computationnelle qui échappe à la conception informatique [216]. Ou, dit autrement, la solution pensée est une solution globale incluant des éléments de réponse humains, sociaux, aux côtés d'artefacts techniques.

Ceci demande de repenser la phase de conception trop souvent cantonnée à une ingénierie technologique et de passer à une ingénierie anthropotechnique. Un exemple symptomatique de cette limitation actuelle est la mise à mal des Captchas. Un Captcha (*Completely Automated Public Turing test to Tell Computers and Humans Apart*) est une forme de test de Turing inversé permettant à un ordinateur de différencier un utilisateur humain d'un autre ordinateur [217]. Ce test est utilisé dans les formulaires web pour se prémunir contre les soumissions automatisées et intensives réalisées par des robots malveillants.



L'hypothèse est que comme l'état actuel des algorithmes de reconnaissance optique de caractères ne permet pas de traiter ces cas, seul un humain peut utiliser ce formulaire. Il y a plusieurs approches pour mettre en échec un Captcha mais l'une des plus efficaces est de chercher une solution mixte et pas forcément technique comme implicitement supposé dans l'hypothèse de ce test : une variation de la technique dite du relais consiste

à utiliser un deuxième site très populaire pour y insérer les Captchas du premier site que l'on souhaite attaquer et utiliser ainsi l'ensemble des utilisateurs de ce deuxième site pour contourner en masse la protection du premier.

Parmi les exemples de sites exploitant l'humain comme une ressource pour acquérir des connaissances citons :

- le jeu ESP [218] permettant d'annoter des collections d'images avec des termes décrivant leur contenu (voir aussi *Google image labeler*[30] et GWAP.com) ; les utilisateurs sont en compétition pour retrouver le terme le plus souvent associé à une image par d'autres utilisateurs.
- openmind.org ou le jeu des 20 questions de l'IA[31] capturant des connaissances de sens commun ; dans ce jeu les internautes jouent contre l'ordinateur qui doit deviner l'objet auquel ils pensent et ce faisant nourrissent l'arbre des *differentia* que l'ordinateur maintient entre les définitions des objets qu'il connait.
- facebook.com, linkedin.com et autres sites permettant de capturer des profils d'utilisateurs et leurs réseaux d'accointances ; les utilisateurs sont amenés à maintenir un descriptif à jour de leurs activités, goûts, échanges, avis, appartenances, etc.
- galaxyzoo.org qui a collecté des millions de catégorisations de galaxies ; les utilisateurs sont amenés par exemple à classifier un maximum de galaxies en spirales vs. ovales et traitent ainsi des milliers d'images astronomiques.

Les pratiques du web 2.0 ont surpris par leur efficacité. Le *social tagging*, notamment, produit des folksonomies dans des proportions et avec une rapidité que l'on aimerait pouvoir transposer à l'acquisition des connaissances.

Nous avons donc deux « artefacts cognitifs » qui se constituent actuellement dans les applications web : la folksonomie et l'ontologie. Les contributions d'un utilisateur à une folksonomie restent légères et découlent de son usage : les tags employés par les utilisateurs sont collectés et analysés en tâche de fond. Les contributions aux ontologies sont très souvent directes et demandent des actions dédiées : les concepts doivent êtres validés, organisés, définis etc.

Cependant le tag n'est qu'un terme utilisé pour marquer une ressource et cela n'en fait pas un concept ou une classe dans une ontologie. Il se rapproche plus d'un nouveau type de candidat terme dont il nous faut exploiter les spécificités si l'on veut retrouver une conceptualisation [219] [220] : le contexte et la nature des cooccurrences ou les profils et les relations des co-utilisateurs.

Les ontologies se définissent par le type de leur contenu. Les folksonomies se définissent par leur moyen d'obtention. Ne peut-on obtenir des ontologies par le biais des folksonomies notamment pour des ontologies de domaines, et obtenir ainsi des folks-ontologies [221] ? Dans cette nouvelle optique, l'ontologie n'est plus la

[30] http://images.google.com/imagelabeler/
[31] http://www.20q.net/

responsabilité unique d'un ontologue mais celui-ci devient l'animateur d'une communauté qui se fédère autour de l'utilisation des applications de cette ontologie. La communauté, par son activité, nourrit le cycle de vie de cette ontologie. Dans le meilleur des cas, l'évolution de l'ontologie devient un effet secondaire de l'activité normale de la communauté ; la vie des ontologies rêvée par les ergonomes.

Du point de vue de l'ingénierie ontologique, il s'agit non pas de faire rentrer l'utilisateur dans une boucle de revue ou d'orchestration mais d'assigner des tâches élémentaires à une masse d'utilisateurs de façon à traiter sur une grande échelle des problèmes notoirement difficiles comme la détection d'un concept dans une ressource multimédia, l'organisation des concepts ou la désambiguïsation. A titre d'exemple pionnier, nous citerons le jeu OntoGame [222].

Faire intervenir une communauté dans la conception d'une ontologie est déjà une approche connue. Outiller la communauté pour que, par son activité normale, elle fasse naturellement vivre les ontologies qui la concernent reste une perspective. Les cycles de vie des folksonomies et des ontologies sont différents et la question de la gestion de leur symbiose nous paraît une perspective prometteuse.

Nous voyons au moins deux approches à combiner :

1) exploiter les folksonomies pour faire émerger des ontologies (analyse statistique des réseaux de tags, utilisation de ressources linguistiques ou d'ontologies existantes et techniques d'alignement, analyse linguistique, analyse des usages),
2) outiller les usages et faire inter-opérer les applications (ex : annoter des images en désambiguïsant avec Wikipedia) pour capturer le plus souvent possible la connaissance au moment où elle est explicite.

Reste cependant la question de la motivation ; l'utilisateur n'est processeur que s'il est motivé. Dans les exemples actuels du web 2.0, les moteurs sont : l'aspect ludique (ex : jeu ESP), l'égocentrisme (ex : Facebook), une rémunération (ex : des bons de réduction), l'esprit de compétition (ex : battre un score), une utilité immédiate (ex : marque-pages triés et accessibles sur toute machine), etc.

La motivation est d'autant plus importante qu'elle détermine une participation en masse. Or le résultat de cet appel à une cognition de masse dépend souvent de l'obtention d'une masse critique. C'est notamment le cas lorsque le traitement utilise des résultats statistiques pour s'effectuer (ex : catégoriser correctement une image ou une galaxie).

Créer et maintenir la participation devient donc un problème à résoudre à la conception d'une de ces applications du deuxième type. A titre d'exemple, même la consigne du jeu ESP est importante : « trouvez le plus rapidement possible le même terme que votre partenaire pour décrire cette image ».

En sciences cognitives, nous avons assisté au passage d'une vue de la cognition comme un processus/système interne (mental) mis en œuvre par un individu, à une vue de la cognition comme un processus/système distribué, à la fois entre un individu et son environnement physique, et entre cet individu et les autres individus avec qui il interagit [216] [223]. De sorte que le système cognitif n'est plus considéré comme un système centralisé « dans la tête » d'un individu, mais comme un système plus large, un système distribué. "*An important aspect of the larger unit is that it contains computational elements (persons) who cannot be described entirely in computational terms.*" [216]

Dans le domaine de l'ingénierie des connaissances, ou ingénierie cognitive, nous sommes en train d'assister à un processus du même ordre : nous passons d'une ingénierie cognitive centralisée (cantonnée au travail d'un ontologue ou d'ingénieurs de la connaissance) à une ingénierie cognitive distribuée.

Le système cognitif à construire n'est alors plus ici le seul système informatique, mais un système plus large composé d'artefacts et de personnes. C'est l'ensemble qu'il s'agit de faire fonctionner. Ce sont des solutions complètes qu'il nous faut construire, et les échanges de l'ingénierie des connaissances avec ce nouveau web participatif ou web 2.0 vont accentuer cette tendance.

6.8 Conclusion

Dans ce chapitre, nous avons commencé par détailler un exemple relativement simple d'utilisation des représentations et raisonnements sur les connaissances pour des interfaces. L'exemple de l'exploitation de l'ontologie comme un espace métrique pour des représentations dans un système angulaire a introduit l'idée de s'intéresser à l'utilisation des systèmes à base d'ontologies dans les interactions avec les utilisateurs.

Nous avons ensuite identifié deux aspects du projet *myCampus* liés à l'utilisation de connaissance dans les interactions :

- le contrôle de la confidentialité et le respect de la vie privée à travers une architecture à base de règles ;
- l'utilisation des connaissances de contexte pour la simplification des interactions avec des services et notamment pour réduire les informations demandées à l'utilisateur.

Nous nous sommes aussi arrêtés sur un problème clef des interactions entre système d'information et utilisateurs : la représentation (notamment visuelle) d'une connaissance sur une ressource. Nous avons réintroduit la notion de substitut qui, en recherche d'information, a deux sens pour rendre compte de deux besoins : la représentation de nos résultats en machine ; la présentation de nos résultats à l'utilisateur. Nous avons montré comment il était possible de générer et représenter les substituts à partir de connaissances.

Enfin, dans les trois dernières parties, nous avons abordé la dimension sociale des interactions à travers une expérience de sémantisation d'un wiki et deux prises de position sur l'intérêt et les enjeux d'utiliser la représentation des connaissances dans des applications du web social.

6.9 Perspectives

Concilier la complexité des systèmes à base de connaissances et l'ergonomie des interactions, mais surtout les allier, est un problème ouvert qui contient encore de nombreux défis. Parmi les directions que nous prenons actuellement citons :

- **Fresnel : des substituts aux lentilles.** Le langage Fresnel [180] propose maintenant un moyen d'identifier les parties d'un graphe RDF qui devraient êtres présentées à l'utilisateur et la façon dont ces informations devraient être présentées. Ce langage est indépendant d'un outil et permet donc d'échanger ces informations sur le web ; des connaissances sur la sémiotique des structures conceptuelles peuvent enfin être échangées. Fresnel a deux concepts fondamentaux qui sont les lentilles et les formats. Les lentilles permettent de définir les propriétés d'une ressource RDF, ou d'un groupe de ressources, à afficher et comment ces propriétés sont triées. Les formats déterminent comment les ressources et les propriétés sont rendues et fournissent des ancrages pour des langages de style comme CSS. Fresnel est décrit en RDF avec un langage d'accès particulier (FSL). Charger, manipuler et utiliser efficacement ces descriptions est un problème ouvert qui demande la coopération de logiciens, de sémioticiens et d'ergonomes. Nous commençons actuellement à défricher ce terrain avec le stage de Master de Gaoussou Camara.
- **Analyse des représentations sémantiques de réseaux sociaux.** Avec le projet ANR ISICIL et la thèse de Guillaume Erétéo qui s'inscrit dans ce projet, nous pensons que la dimension sociale des applications du web pourrait passer dans une nouvelle ère en exploitant les représentations formelles des caractéristiques des ressources du web, de leurs utilisateurs, de leurs communautés et de leurs interactions. De nombreux vocabulaires sont actuellement proposés : FOAF pour décrire les personnes [224], RELATIONSHIP [225] qui spécifie la relation *knows* de FOAF, SIOC [226] qui modélise les concepts d'applications en ligne comme les blogs, etc. Ces nouvelles connaissances disponibles en ligne sont un défi aux méthodes d'analyse de réseaux sociaux classiques et ouvrent la voie à des méthodes d'analyse sémantiques qui intègreraient dans les algorithmes de graphes classiques de l'analyse de réseaux, les notions de représentation à base d'ontologie et notamment du typage des liens.

Enfin, nous allons nous intéresser dans les années à venir à la notion d'intégration d'interactions. En effet plus les systèmes d'interactions se dotent de modèles formels et/ou explicites, plus ils génèrent des traces réutilisables et combinables pour d'autres tâches. A titre d'exemple, un site wiki, des logs d'IRC ou les archives d'une mailing-list sont des traces d'interactions sur le web pour lesquelles il existe maintenant des vocabulaires RDF. Une tâche comme la recherche d'information peut maintenant intégrer ces traces et fournir de nouveaux services d'accès et de veille transversaux aux canaux d'interaction utilisés.

Chapitre 6 : Récapitulatif de mes contributions personnelles.

A partir des spécifications d'experts du domaine, j'ai conçu et formalisé les mécanismes de transformation des représentations formelles de KmP et les visualisations intelligibles [27].

J'ai dirigé le projet myCampus pendant un an. J'ai conçu, formalisé et implanté le système de représentation et de gestion des connaissances privées à base de règles qui constitue le eWallet [13]. J'ai conçu l'architecture agents et services ainsi que plusieurs services et les fonctionnalités de localisation, et j'ai mené deux campagnes de tests sur le campus de Carnegie Mellon [16].

J'ai imaginé de réintégrer formellement la notion de substitut (*surrogate*) dans les modèles de connaissances et j'ai proposé plusieurs solutions à leur représentation et leur génération [159].

Je suis co-concepteur du logiciel SweetWiki qui intègre une approche web social et une approche web sémantique [33]. J'ai conçu et implanté l'algorithme de la transformation GRDDL RDFa au cœur de ce logiciel qui a été intégré aux standards du W3C [36] [35]. J'encadre aussi l'ingénieure Priscille Durville en charge du développement de SeWeSe et SemTag [75].

Enfin j'ai montré qu'un certain nombre de travaux opposaient à tort le web sémantique et les approches sociales, par l'exemple [33], l'analyse des représentations [204] et la suggestion de perspectives [213].

Je suis co-encadrant de la thèse de Guillaume Erétéo qui commence sur l'analyse des réseaux sémantiques en partenariat avec Orange Labs.

J'encadre le stage de master de Gaoussou Camara sur Fresnel.

7. Conclusion

Sous l'appellation de mémoires collectives, je me suis intéressé à la gestion des connaissances d'une communauté pour en assurer l'acquisition, la diffusion et l'évolution tout au long de l'activité de cette communauté.

Deux axes de recherche se posent systématiquement dans ce contexte :

- Comment matérialiser une mémoire organisationnelle en reposant sur des systèmes symboliques ayant une interprétation qui leur permet de capturer le sens des structures de la mémoire ?
- Quelles sont les inférences qui nous aident dans l'exploitation d'une mémoire et l'animation de sa communauté ? Comment les simuler par des opérations sur des systèmes symboliques et quelles architectures utiliser pour leur implantation ?

En suivant l'approche du web sémantique, c'est-à-dire l'augmentation du web classique avec des connaissances formalisées, et en l'appliquant aux intranets puis aux sites communautaires, j'ai analysé la conception d'un web sémantique où les annotations sémantiques des ressources d'informations guident des mécanismes d'exploitation de la mémoire.

Mes recherches en ingénierie des connaissances m'ont amené à développer une approche exploitant des ontologies *i.e.* des théories logiques qui rendent compte partiellement mais explicitement des notions mobilisées par la description d'une réalité, et des règles contraignant la structure de ces descriptions. Ces notions sont utilisées pour annoter les ressources de la mémoire et les acteurs de la communauté. Au-dessus d'un tel modèle, j'ai conçu des inférences assistant la gestion et l'exploitation de la mémoire. Cette approche soulève les questions suivantes :

- Quelles méthodes et quels outils utiliser pour modéliser et pour assister la mise en place et l'évolution de l'ontologie ?
- Comment prendre en compte différents points de vue et différents profils utilisateurs ?
- Comment améliorer les formalismes de représentation pour nos scénarios ?
- Quelles inférences peut-on concevoir au-dessus de ces formalismes ? Quelle est la fidélité de telles simulations ?

J'ai aussi fait appel à l'intelligence artificielle distribuée où la collaboration intelligente entre des agents logiciels permet de mettre en place des sociétés artificielles qui prennent en charge la capitalisation globale des connaissances distribuées dans l'organisation ou la communauté.

J'ai par la suite étendu cette approche à la gestion de services web dans le cadre d'accès mobiles au réseau ou dans le cadre de l'intégration d'applications d'entreprises. Les problèmes de recherche étaient alors :

- Comment passer des inférences des systèmes à base de connaissances centralisée, à des inférences distribuées et obtenir, au niveau social, les fonctionnalités demandées par la communauté et sa mémoire ?
- Comment intégrer le contexte et le profil du système et des utilisateurs et la confidentialité aux inférences?

Dans ce mémoire, j'ai caractérisé les scénarios qui m'ont amené à concevoir des représentations et des raisonnements pour des mémoires individualisées (rattachées à un utilisateur donné), organisationnelles (rattachées à une organisation, notamment une entreprise) ou plus généralement communautaires (rattachée à une communauté d'intérêt ou de pratique). J'ai résumé comment la représentation des connaissances, notamment à base d'ontologies permettait de matérialiser et gérer de telles mémoires.

Pour cette matérialisation, j'ai défendu la thèse d'une adéquation des formalismes à base de graphes pour offrir des représentations avec un degré variable de formalisation en fonction des besoins identifiés dans les scénarios d'application et des traitements à effectuer. J'ai aussi montré que certains formalismes du web ont par définition cette structure de graphes et qu'ils sont utilisables dans nos travaux et offrent des opportunités d'extensions. L'existence de plusieurs formalismes à base de graphes justifie aussi l'initiative Griwes qui, comme détaillé, cherche à factoriser la définition des structures mathématiques partagées par ces formalismes et réutiliser l'algorithmique des traitements communs à ces structures.

J'ai aussi montré que les graphes ne sont pas une simple alternative, ils offrent un support à d'autres types de raisonnement que la dérivation logique. Par exemple, la hiérarchie de notions contenue dans une ontologie peut être vue comme un espace métrique permettant de définir des distances pour comparer la proximité sémantique de deux notions. Nous avons mis en œuvre cette idée dans plusieurs scénarios comme l'allocation distribuée d'annotations, la recherche approchée ou le clustering.

J'ai détaillé diverses utilisations des distances sémantiques utilisées dans un échantillon représentatif de projets. Ceci m'a permis de démontrer l'intérêt et le potentiel des distances, et aussi de souligner l'importance du travail restant à faire pour identifier et caractériser les familles de distances existantes et leur adéquation respective aux tâches pour lesquelles les utilisateurs souhaitent être assistés.

Ces structures de graphes sont distribuées par les formalismes du web sémantique et posent donc le problème de la gestion de la distribution des ressources (données et applications). J'ai exposé un certain nombre d'approches pour tenir compte et gérer la distribution des ressources dans les webs sémantiques que nous concevons. En particulier, j'ai introduit des méthodes et formalisé des structures d'indexation d'annotations RDF pour gérer leur archivage et utiliser leur sémantique pour router les décompositions de requêtes lors de la résolution distribuée dans des architectures multi-agents ou multi-services.

J'ai ensuite résumé deux autres travaux sur la gestion de la distribution : (1) une approche pour intégrer des sources extérieures à un web interne et les annoter automatiquement ; (2) une approche pour annoter les services distribués pour, par la suite, les identifier, les retrouver, les invoquer et les composer entre eux et avec la mémoire.

Enfin j'ai voulu insister sur l'idée d'utiliser des représentations et raisonnements sur les connaissances pour gérer les interactions avec les utilisateurs. J'ai expliqué plusieurs résultats pour concevoir des interfaces, contrôler des accès, contextualiser des requêtes, visualiser des résultats et intégrer la dimension sociale des interactions.

Si l'on reprend les différents domaines de recherches qui sont mobilisés dans ces travaux, ils sont clairement unis par une très forte complémentarité :

- Les ontologies sont utiles aux logiciels distribués : la communication et les inférences dans des applications distribuées peuvent reposer sur des ontologies partagées, la description et la découverte de composants logiciels peut aussi employer des ontologies permettant l'identification et la composition automatique.
- Les logiciels distribués sont utiles aux ontologies : les architectures distribuées sont utiles pour la réalisation de collecticiels de gestion des ontologiques et de leurs sources multiples.
- Les ontologies sont utiles au web sémantique car elles fournissent le vocabulaire des annotations et permettent les inférences pour la recherche dans les bases d'annotations exploitant les propriétés de l'ontologie.
- Le web sémantique est utile aux ontologies : les langages standardisés du web sémantique sont utilisés pour l'échange et l'extension d'ontologies et les collecticiels du web offrent des solutions pour la gestion du consensus ontologique.

- Le web sémantique est utile aux applications distribuées : les langages standardisés du web sémantique sont utilisés pour les échanges entre logiciels. Ils permettent l'interopérabilité et la pérennité des formats de stockage.
- Les applications distribuées sont utiles au web sémantique : elles apportent des architectures et des protocoles d'interaction appropriés et elles offrent des paradigmes pour l'encapsulation de ressources légataires.

A la croisée de ces domaines, mes contributions concernent plus particulièrement :

- Les collecticiels pour la conception et la maintenance de systèmes à base de connaissances utilisant des ontologies et ce tout au long de leur cycle de vie ; Nous explorons actuellement l'intégration des ontologies et folksonomies pour une nouvelle génération de ces systèmes alliant les capacités d'automatisation données par les représentations formelles et la flexibilité et la dimension sociale des approches telles que le social tagging.
- L'élargissement des problèmes de mémoire organisationnelle et d'interaction aux communautés virtuelles.
- L'utilisation des profils utilisateurs pour améliorer les actions d'annotation et de recherche, faire émerger des communautés d'intérêt et connecter les individus ; une application s'intéresse actuellement à améliorer la veille et la notification a priori dans les wikis et cette problématique sera approfondie dans le projet ISICIL.
- La gestion de la jonction entre le web ouvert et l'intraweb ou comment utiliser la mémoire pour offrir un portail dynamique de l'intérieur de l'organisation vers l'extérieur.
- La mise en place, l'utilisation et la composition de services web et de connaissances ; je m'intéresse en particulier à l'intégration d'applications d'entreprises ou de communautés de pratique et à l'automatisation des processus de travail dans les communautés en exploitant les formalismes du web sémantique et les services web.
- L'intégration des contraintes de sécurité et de confidentialité au même niveau d'expression que les connaissances *i.e.* en reposant sur des ontologies ; la plate-forme SeWeSe dont j'étais responsable intégrait des mécanismes d'accès reposant sur des raisonnements sur les ontologies et les annotations connues du système.
- La conception de nouvelles inférences en particulier en exploitant les ontologies comme des espaces métriques et la conception ou l'apprentissage automatique des métriques à partir d'informations externes à l'ontologie.

Parmi les axes de recherche que j'ai mobilisés pour ce mémoire, un certain nombre de problèmes m'intéressent plus particulièrement et organisent mes travaux actuels et mes perspectives. Sans revenir en détail sur les perspectives identifiées dans chaque section j'aimerais conclure en généralisant ces points de vue qui structurent ma vision du futur :

- Afin de mettre en place et de maintenir le consensus ontologique à la base de toute approche web sémantique, les collecticiels (CSCW) peuvent offrir des méthodes et des environnements permettant d'instituer le dialogue et l'argumentation nécessaires à la conception et la maintenance d'ontologies au sein d'une communauté. Les fonctionnalités génériques à étudier porteraient sur la discussion, la conception, la mise en place et la maintenance d'ontologies dans une organisation ou une communauté. L'existence de collecticiels sur le web en particulier pour la discussion de définitions (ex : wikipedia) est un élément et une nouvelle pratique à prendre en compte ainsi que les nouveaux usages tels que le "social tagging" qui percolent dans plusieurs outils en ligne. La représentation des structures sous-jacentes aux différentes approches (ontologies, folksonomies) et les différents cycles de vie de ces structures (ingénierie de connaissances, *social working*) ouvrent des perspectives de recherche intéressantes. Notons en particulier la mise en symbiose de ces objets et l'utilisation de synergies entre leurs cycles de vie pour produire des systèmes hybrides plus efficaces et s'intégrant aux tâches quotidiennes des utilisateurs pour faire de leur évolution un effet secondaire de leurs activités. La combinaison des représentations sémantiques et des activités sociales à l'échelle du web recèle un potentiel énorme.
- La distribution des acteurs et/ou des ressources pose la question de l'architecture distribuée ou centralisée. Dans nos travaux, nous avons expérimenté les deux types (système multi-agents ou multi-services vs applications client-serveur web). La multiplication des architectures (composants, agents, services web, pair à pair, grilles, client-serveur) semble maintenant évoluer vers des architectures hybrides (services web sémantique et agents, grilles sémantiques, pair à pair sémantique, agents et grilles, serveurs fixes dans du pair à pair). Dans une perspective où les acteurs participent à plusieurs communautés et où la distribution des ressources qu'ils mobilisent est inévitable (documents, annotations, ontologies, bases de données, services et logiciels, etc.), l'intégration des architectures et l'interopérabilité des logiciels est un problème majeur lorsque l'on veut proposer des solutions qui soient à la fois : complètes, raisonnablement complexes et capables de passer à l'échelle.
- La mémoire est un objet vivant et il en est de même des ontologies. Ces deux objets suivent un cycle de vie : (1) détection des besoins (2) construction (3) diffusion (4) utilisation (5) évaluation (6) évolution $\rightarrow$ (3), avec une septième activité de gestion du cycle. Chaque étape de ce cycle de vie doit être assistée si l'on veut que le cycle tende vers un cercle vertueux où les acteurs participent à la gestion de la mémoire parce qu'ils sont conscients que la mémoire participe à leurs tâches quotidiennes. Cela appelle une exploration à long terme où le défi

est de trouver des modèles et des fonctionnalités qui aboutiront à des outils suffisamment génériques pour être réutilisables et suffisamment configurables pour être utilisables dans des scénarios d'application concrets. L'apparition de nouveaux formalismes (ex, SKOS [51]) et de nouvelles structures (ex, folksonomies) pose aussi le problème de la généralisation de ces outils.

- Il existe plusieurs profils d'utilisateurs d'une même mémoire collective : chacun a ses centres d'intérêt, ses rôles dans l'organisation, ses activités et leur contexte. Comment exploiter de manière efficace ces particularités dans les mécanismes de structuration et de diffusion des mémoires. Dans un premier temps il serait intéressant de regarder comment les profils individuels peuvent être utilisés pour améliorer les actions d'annotation et de recherche ainsi que l'ergonomie des interfaces à travers des fonctionnalités de filtrage, de tri, etc. D'autre part, cette population est aussi dispersée et les individus ne sont pas forcément conscients de l'existence d'autres utilisateurs ayant telle fonction ou telles compétences. Ils ignorent parfois l'existence, l'emplacement et la disponibilité de gens avec lesquels ils partagent des centres d'intérêts. L'utilisation de leurs profils peut permettre de connecter les individus et de faire émerger des communautés d'intérêt et de pratique. Des fonctionnalités génériques telles que la recherche de recoupements, le regroupement de profils, et l'automatisation du 'bouche à oreille' peuvent, de ce point de vue, améliorer la diffusion des connaissances. A plus long terme, un deuxième problème est celui du paramétrage des modèles, des inférences et des interfaces par les profils des utilisateurs et les contextes et points de vue d'utilisation.
- Si l'on prend plus de recul par rapport aux problèmes rencontrés dans les interfaces, on peut penser qu'ils sont symptomatiques d'un manque plus général : la gestion des liens entre la sémiotique et la sémantique. Deux grandes familles de problèmes naissent de cette brèche : (1) Le problème de l'automatisation de l'acquisition : ce problème a déjà fait l'objet d'un certain nombre de contributions (outils d'analyse linguistique, analyse de ressources multimédia, etc.). Cependant les scénarios demandent maintenant d'autres familles de connaissances que celles que recèlent les documents. La connaissance du contexte, par exemple, pose d'autres problèmes d'acquisition et demande une capacité à reconnaître et exploiter l'affordance des ressources informatiques et plus généralement de l'environnement d'utilisation. L'annotation sémantique d'applications est, d'une certaine façon, un pas dans ce sens. (2) Le problème beaucoup moins étudié de la génération automatique de représentations pour les structures conceptuelles qui doivent être présentées à un utilisateur (ex : la réponse d'un moteur de recherche sémantique). Pour répondre aux usages, ces interfaces se multiplient et utilisent des médias et des modalités de plus en plus variés (interfaces graphiques, textuelles, multimédia, en langue naturelle, systèmes de questions-réponses, synthèse vocale, interface 3D et réalité virtuelle). Le problème de l'ergonomie d'une interface reposant sur un système à base d'ontologies reste grand ouvert et les structures conceptuelles sous-jacentes étant de plus en plus

complexes les interfaces ont la tâche de plus en plus difficile de combler l'écart entre les structures génériques internes et les représentations focalisées pour une tâche. Cette situation appelle le développement de logiques de présentation ou de ce que la littérature appelle des algèbres sémiotiques ; Il s'agirait, par exemple, de pouvoir raisonner sur l'utilisation de différentes expressions d'un concept et de capturer la logique du choix d'une représentation. La future génération de serveurs web sémantique sera au cœur de ce problème.

- Le web est en train de subir trois grands changements particulièrement importants à mon avis : (1) l'augmentation du nombre d'applications en ligne change peu à peu la face du web qui, d'une nature de masse de documents et de données, évolue vers une nature plus dynamique d'espace d'activité et de rencontres, (2) les réseaux mobiles et ubiquitaires changent la connectivité du web et sa présence dans nos activités quotidiennes (3) l'augmentation du nombre de services et de types de données structurées disponibles transforme le web en une machine virtuelle mondiale fournissant un nouveau paradigme de développement à une échelle jusqu'alors inconnue. Ces trois aspects ont des répercussions directes dans les organisations : (1) les workflows peuvent reposer sur l'utilisation et la composition de services web et la communauté émergente des services web sémantiques a très bien identifié l'intérêt de combiner les deux domaines sous-jacents. De plus en plus souvent, nous rencontrons des scénarios où l'utilisateur ne veut plus uniquement une interface unifiée d'accès à l'information, mais aussi une interface unifiée d'accès aux applications qu'il utilise et compose avec ces informations. Les scénarios de gestion des connaissances fusionnent naturellement avec ceux de l'intégration d'applications d'entreprise en particulier lorsqu'ils reposent sur les mêmes modélisations. De plus le typage des documents (statut, genre, utilité, etc.) et la gestion des cycles de vie et flux documentaires forment une famille particulière de workflows, les *document flows* i.e. les processus et protocoles concernant les cycle de vie des documents (2) les communautés ne sont plus ni rigides ni tangibles, l'accès mobile aux ressources organisationnelles est une évolution évidente des systèmes d'information actuels, posant les questions de sécurité, de conscience de l'environnement et du contexte d'accès et d'ergonomie contextuelle. Les utilisateurs sont de plus en plus souvent membres de plusieurs communautés virtuelles qui maintiennent aussi différentes mémoires et interagissent dans des structures intangibles se construisant dans des applications du web dit social (ex : wikis, forums, blogs, etc.). Les règles et le fonctionnement de ces communautés et de leurs mémoires sont différentes (politique ouverte, responsabilités distribuées, structures plates, autorité émergente, etc.) et appellent de nouvelles méthodes et outils pour assister leurs cycles de vie et pour les concilier avec ceux des organisations auxquelles elles sont transversales.

- Ces évolutions mettent aussi une nouvelle emphase sur des problèmes de sécurité et de filtrage de l'accès aux connaissances. L'aspect accès - sécurité - confidentialité a une double relation avec les approches à base d'ontologies : la nouvelle expressivité des formalismes d'échanges demande de nouveaux langages d'expression des règles de sécurité ; et les nouvelles politiques de sécurité peuvent reposer sur ces nouveaux formalismes pour leur expression. Il nous faut étudier comment les ontologies peuvent fournir un vocabulaire pour définir plus finement les droits d'accès dans des systèmes où l'organisation des groupes est éphémère et pas toujours hiérarchique et où l'accès et la précision de l'information dépendent du contexte de la demande.

Dernier point, je soulignerais qu'un certain nombre de résultats de recherche du domaine sont maintenant établis [1] [74] [21] et, pour certains, standardisés [45] [37] [47] [36] [48] [206]. Ces résultats posent les fondements d'une ingénierie et dont les ingénieurs réclament une formation dans laquelle nous avons déjà commencé à nous investir mais qui demandera encore beaucoup d'efforts de diffusion et de vulgarisation dans les années à venir.

Ayant fermé ce dernier chapitre de mon mémoire, je souhaite ajouter un paragraphe qui me tient à cœur.

J'ai eu le privilège et le plaisir de passer dix ans aux côtés de Rose Dieng-Kuntz ; c'est-à-dire toute ma jeune carrière en recherche. Je lui dois entièrement ces dix ans. J'ai appris tellement de choses entre un sourire, un tintement de bracelets, un rire aux éclats et un morceau de chocolat.

Je dois mon éducation scientifique à une femme de symboles qui en parallèle m'a aussi enseigné par l'exemple...

... que l'on pouvait diriger avec conviction tout en étant d'une infinie gentillesse,

... que l'humanisme et la générosité forment le meilleur ciment d'une équipe,

... que la volonté n'a pas de limites autres que celles qu'on lui imagine,

... que la passion et la joie de vivre sont des énergies renouvelables,

... qu'un sourire peut-être inné et une réponse à tant de questions,

... que la petite voix douce et posée d'une enfant de Dakar peut traverser des murailles,

... qu'il existait au moins une personne dont la capacité de travail était illimitée,

... que l'on peut être d'une intelligence reconnue mondialement et le cacher derrière une insondable humilité,

... que l'on peut toujours ajouter quelque chose dans une valise pleine, mais ce dernier point reste entre elle et moi.

Dix ans de tes enseignements, Rose, c'était trop court ; pour la première fois de tout mon parcours, j'aurais vraiment voulu redoubler.

Fabien L. Gandon
17 aout 2008

8. Références

[1] Rose Dieng, Olivier Corby, Fabien Gandon, Alain Giboin, Joanna Golebiowska, Nada Matta, and Myriam Ribière, *Knowledge Management: Méthodes et outils pour la gestion des connaissances,* 3rd ed.: Dunod, 2005.

[2] John F Sowa, *Conceptual Structures: Information Processing in Mind and Machine*.: Addison–Wesley, 1984.

[3] A. Rabarijaona, R. Dieng, and O. Corby, "Exploitation of XML for Corporate Knowledge Management," in *Knowledge Acquisition, Modeling, and Management,11th European Workshop, EKAW*, 1999.

[4] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web ," *Scientific American*, pp. 35-43, May 2001.

[5] Olivier Corby, Rose Dieng, and Cédric Hébert, "A Conceptual Graph Model for W3C Resource Description Framework," in *8th International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues (ICCS)*, Darmstadt, 2000.

[6] Fabien Gandon. (2006) Ontologies informatiques, Interstices. [Online]. http://interstices.info/display.jsp?id=c_17672

[7] J McCarthy, "Circumscription — A Form of Nonmonotonic Reasoning," *Artificial Intelligence*, vol. 13, p. 27–39, 1980.

[8] Thomas Gruber, "A Translation Approach to Portable Ontologies," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.

[9] Jacob Lorhard, "Ogdoas scholastica," 1606.

[10] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, *The Description Logic Handbook, Theory, Implementation and Applications*.: Cambridge University Press, 2003.

[11] Norman Sadeh, *m-Commerce: Technologies, Services and Business Models*.: Wiley, 2002.

[12] Fabien L. Gandon and Norman M. Sadeh, "Gestion de connaissances personnelles et contextuelles, et respect de la vie privée," in *15ièmes Journées Francophones de l'Ingénierie des Connaissances IC*, Lyon, France, 2004, pp. 5-16.

[13] Fabien L. Gandon and Norman M. Sadeh, "Semantic Web Technologies to Reconcile Privacy and Context Awareness," *Web Semantics Journal*, vol. 1, no. 3, 2004.

[14] Fabien L. Gandon and Norman M. Sadeh, "A Semantic e-Wallet to Reconcile Privacy and Context Awareness," in *2nd International Semantic Web Conference (ISWC)*, Sanibel Island, Forida, USA, 2003, pp. 385-401.

[15] Fabien L. Gandon and Norman M. Sadeh, "Semantic Web Technologies to Reconcile Privacy and Context Awareness," Carnegie Mellon University, Computer Science Department, School of Computer Science, Pittsburgh, Technical report CMU-CS-03-211, 2003.

[16] Fabien L. Gandon and Norman M. Sadeh, "Connaissance du Contexte, Confidentialité et Accès Mobiles : une Approche Web Sémantique et Multi-agents," in *Journées Francophones: Mobilité et Ubiquité,* Nice, Sophia-Antipolis, 2004, pp. 123-130.

[17] FIPA. (2002) Specifications. [Online]. http://www.fipa.org/repository/fipa2000.html

[18] Khaled Khelif, *Web sémantique et mémoire d'expériences pour l'analyse du transcriptome*.: Thèse de l'université de Nice Sophia Antipolis, 2006.

[19] Sylvain Dehors, *Exploiting Semantic Web and Knowledge Management Technologies for E-learning*.: PhD Thesis, University of Nice-Sophia Antipolis INRIA, France, 2007.

[20] R. Dieng-Kuntz, D. Minier, M. Ruzicka, F. Corby, O. Corby, and L. Alamarguy, "Building and Using a Medical Ontology for Knowledge Management and Cooperative Work in a Health Care Network,"

Computers in Biology and Medicine, vol. 36, no. 7-8, pp. 871-892, July-August 2006, Special Issue on Medical Ontologies.

[21] Fabien Gandon, *Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web*. Nice: INRIA and University of Nice - Sophia Antipolis, Doctoral School of Sciences and Technologies of Information and Communication, 2002, Scientific Philosopher Doctorate Thesis In Informatics, Defended Thursday the 7th of November 2002.

[22] Tuan-Dung Cao and Fabien Gandon, "Integrating external sources in a corporate semantic web managed by a multi-agent system," in *AMKM, AAAI Spring Symposium on Agent-Mediated Knowledge Management*, Stanford University, 2003.

[23] Tuan-Dung Cao, *Exploitation du Web sémantique pour la veille technologique*.: PhD Thesis, University of Nice-Sophia Antipolis INRIA, 2006.

[24] Fabien Gandon, Moussa Lo, and Cheikh Niang, "Un modèle d'index pour la résolution distribuée de requêtes sur un nombre restreint de bases d'annotations RDF," in *IC, 19iemes Journées Francophones d'Ingénierie des Connaissances*, Nancy, 2008.

[25] Hacène Cherfi, Olivier Corby, Catherine Faron-Zucker, and Khaled Khelif, "Semantic Annotation of Texts with RDF Graph Contexts.," in *16th International Conference on Conceptual Structures (ICCS)*, Toulouse, 2008.

[26] Hacène Cherfi, Olivier Corby, and Cyril Masia-Tissot, "RDF(S) and SPARQL Expressiveness in Engineering Design Patterms," in *IEEE/WIC/ACM International Conference on Web Intelligence*, Silicon Valley, USA, 2007.

[27] Fabien Gandon, Olivier Corby, Alain Giboin, Nicolas Gronnier, and Cecile Guigard, "Graph-based inferences in a Semantic Web Server for the Cartography of Competencies in a Telecom Valley," in *ISWC*, Galway, 2005, pp. 247-261.

[28] Yamine Ait Ameur, Nabil Belaid, Mohammed Bennis, Olivier Corby, Rose Dieng-Kuntz, Jérémie Doucy, Priscille Durville, Chimène Fankam, Fabien Gandon, Allain Giboin, Patrick Giroux, Sandrine Grataloup, Bruno Grilheres, Florian Husson, Stéphane Jean, Joel Langlois, Phuc Hiep Luong, Laura Mastella, Olivier Morel, Michel Perrin, Guy Pierra, Jean-François Rainaud, Idir Ait-Sadoune, Eric Sardet, Francois Tertre, and Joao Francisco Valiati, "Semantic Hubs for Geological Projects," in *Semantic Metadata Management and Applications SeMMA, Workshop at ESWC*,

Tenerife, 2008.

[29] Alain Giboin, Priscille Durville, and Fabien Gandon, "Ingénierie ontologique participative : essai de mise en oeuvre avec l'éditeur collaboratif ECCO," in *Atelier IC 2.0, joint aux 19èmes Journées Francophones d'Ingénierie des Connaissances*, Nancy, 2008.

[30] A. Tifous, A. El Ghali, A. Giboin, and R. Dieng-Kuntz, "O'CoP, an Ontology Dedicated to Communities of Practice," in *I-KNOW, The 7th International Conference on Knowledge Management*, Graz, 2007.

[31] A. El Ghali, A. Giboin, and C. Vanoirbeek, "Bridging the Gap Between Technical and Pedagogical Project Partners' Perspectives on the Modelling of Communities of Practice," in *6th International Conference on Networked Learning NLC*, 2008.

[32] Bassem Makni, Khaled Khelif, Rose Dieng-Kuntz, and Hacene Cherfi, "Semi-automatic Creation of an Ontology and of Semantic Annotations from a Discussion Forum of a Community of Practice," in *16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns, EKAW* , Acitrezza, 2008.

[33] Michel Buffa, Fabien Gandon, Guillaume Ereteo, Peter Sander, and Catherine Faron, "SweetWiki: A semantic wiki," *Special Issue of the Journal of Web Semantics on Semantic Web and Web 2.0*, vol. 6, no. 1, pp. 84-97, 2008.

[34] Michael Schroeder, Albert Burger, Patty Kostkova, Robert Stevens, Bianca Habermann, and Rose Dieng-Kuntz, "Sealife: A Semantic Grid Browser for the Life Sciences Applied to the Study of Infectious Diseases," in *HealthGrid* , 2006.

[35] Fabien Gandon. (2008, August) Profile for latest GRDDL transformation for RDFa. [Online]. http://ns.inria.fr/grddl/rdfa/

[36] Fabien Gandon. (2007, April) W3C Working Group Note. [Online]. http://www.w3.org/TR/grddl-scenarios/

[37] B. Adida, M. Birbeck, S. McCarron, and S. Pemberton. (2008, June) RDFa in XHTML: Syntax and Processing, A collection of attributes and processing rules for extending XHTML to support RDF, W3C Candidate Recommendation. [Online]. http://www.w3.org/TR/rdfa-syntax

[38] D. Connolly. (2007, September) Gleaning Resource Descriptions from Dialects of Languages (GRDDL), W3C Recommendation. [Online]. http://www.w3.org/TR/grddl/

[39] Khaled Khelif, Fabien Gandon, Olivier Corby, and Rose Dieng-Kuntz,

"Using the Intension of Classes and Properties definition in Ontologies for Word Sense Disambiguation," in *EKAW 2008 - 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns*, Acitrezza, Catania, Italy, 2008.

[40] G. Klyne and J. Carroll. (2004, February) Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation. [Online]. http://www.w3.org/TR/rdf-concepts/

[41] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau. (2006, August) Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation. [Online]. http://www.w3.org/TR/2006/REC-xml-20060816/

[42] Douglas R. Hofstadter, *Godel, Escher, Bach: An Eternal Golden Braid*, 20th ed.: Basic Books, 1999.

[43] Dave Beckett. (2004, February) RDF/XML Syntax Specification (Revised), W3C Recommendation. [Online]. http://www.w3.org/TR/rdf-syntax-grammar/

[44] E. Prud'hommeaux and A. Seaborne. (2008, January) SPARQL Query Language for RDF, W3C Recommendation. [Online]. http://www.w3.org/TR/rdf-sparql-query/

[45] D. Beckett and J. Broekstra. (2008, January) SPARQL Query Results XML Format, W3C Recommendation. [Online]. http://www.w3.org/TR/rdf-sparql-XMLres/

[46] K. G. Clark, L. Feigenbaum, and E. Torres. (2008, January) W3C Recommendation. [Online]. http://www.w3.org/TR/rdf-sparql-protocol/

[47] D. Brickley and R.V. Guha. (2004, February) RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation. [Online]. http://www.w3.org/TR/rdf-schema/

[48] D.L. McGuinness and F. van Harmelen. (2004, February) OWL Web Ontology Language, Overview, W3C Recommendation. [Online]. http://www.w3.org/TR/owl-features/

[49] B. Parsia and P.F. Patel-Schneider. (2008, April) OWL 2 Web Ontology Language: Primer, W3C Working Draft. [Online]. http://www.w3.org/TR/owl2-primer/

[50] Stasinos Konstantopoulos and Phil Archer. (2008, Jullet) Protocol for Web Description Resources (POWDER): Formal Semantics, W3C. [Online]. http://www.w3.org/TR/2008/WD-powder-formal-20080709/

[51] Alistair Miles and Sean Bechhofer. (2008, Juin) SKOS Simple Knowledge Organization System, W3C. [Online]. http://www.w3.org/TR/skos-reference/

[52] Steve Pepper and Graham Moore. (2001) XML Topic Maps (XTM) 1.0, TopicMaps.Org. [Online]. http://www.topicmaps.org/xtm/

[53] Hiroshi Uchida. (2008, Mars) Common Web Language, Incubator Group Report, W3C. [Online]. http://www.w3.org/2005/Incubator/cwl/XGR-cwl-20080331/

[54] Ashok Malhotra. (2008, mars) RDB2RDF Incubator Group Charter, W3C. [Online]. http://www.w3.org/2005/Incubator/rdb2rdf/charter-20080304

[55] J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," in *1st International Semantic Web Conference, ISWC* , Sardinia, Italy, 2002, pp. 54-68.

[56] L. Miller, A. Seaborne, and A. Reggiori, "Three Implementations of SquishQL, a Simple RDF Query Language," in *1st International Semantic Web Conference, ISWC*, Sardinia, Italy, 2002, pp. 423-435.

[57] HP Labs Semantic Web Programme. Jena – A Semantic Web Framework for Java. [Online]. http://jena.sourceforge.net/

[58] M. Sintek and S. Decker, "Triple: A Query, Inference and Transformation Language for the Semantic Web," in *1st International Semantic Web Conference, ISWC*, Sardinia, Italy, 2002, pp. 364-378.

[59] Joseph Kopena and William Regli, "DAMLJessKB: A Tool for Reasoning with the Semantic Web," *IEEE Intelligent Systems*, vol. 8, no. 3, pp. 74-77, May/June 2003.

[60] BBN and DARPA. (2006, Jan.) DARPA Agent Markup Language (DAML). [Online]. http://www.daml.org/

[61] Friedman-Hill, *Jess in Action: Java Rule-based Systems*.: Manning Publications Company, 2003, http://herzberg.ca.sandia.gov/jess/.

[62] Paul V. Biron and Ashok Malhotra. (2004, October) XML Schema Part 2: Datatypes Second Edition, W3C Recommendation. [Online]. http://www.w3.org/TR/xmlschema-2/

[63] S. Decker, M., Fensel, D. Erdmann, and R. Studer, "OntoBroker: Ontology based Access to Distributed and Semi-structured Information," in *Semantic Issue in Multimedia Systems*. Boston: Kluwer Academic Publisher, 1999.

[64] Ph. Martin, "How WebKB could contribute to PORT," in *2nd PORT workshop at ICCS*, 2002.

[65] P. Martin and P. Eklund., "Knowledge Retrieval and the World Wide Web," *IEEE Intelligent Systems*, vol. 15, no. 3, pp. 18-25, 2000.

[66] N. Guarino, C. Masolo, and G. Vetere, "Ontoseek: Content-based access to the Web," *IEEE Intelligent Systems* , vol. 14, no. 3, pp. 70-80, 1999.

[67] Dmitry Tsarkov and Ian Horrocks, "FaCT++ Description Logic Reasoner: System Description," in *LNCS Volume 4130.*, 2006.

[68] Motik Sattler and Ulrike Sattler, "KAON2, A Comparison of Reasoning Techniques for Querying Large Description Logic Aboxes," in *Boris Book: Logic for Programming Artificial Intelligence, and Reasoning.*, 2006.

[69] Racer Systems GmbH & Co. Racer. [Online]. http://www.racer-systems.com/

[70] Clark and Parsia. Pellet. [Online]. http://pellet.owldl.com/

[71] Jean-François Baget and Yannic Tognetti, "Backtracking through Biconnected Components of a Constraint Graph," in *17th International Joint Conference on Artificial Intelligence, IJCAI*, 2001, pp. 291-296.

[72] Olivier Corby, "Web, Graphs & Semantics," in *ICCS, 16th International Conference on Conceptual Structures*, Toulouse, 2008, pp. 43-61.

[73] Olivier Corby and Catherine Faron-Zucker, "RDF/SPARQL Design Pattern for Contextual Metadata," in *WI, International Conference on Web Intelligence*, Silicon Valley, 2007.

[74] Olivier Corby, Rose Dieng-Kuntz, Catherine Faron-Zucker, and Fabien Gandon, "Searching the Semantic Web: Approximate Query Processing based on Ontologies," *IEEE Intelligent Systems Journa* , vol. 1, no. 21, 2006.

[75] P. Durville and F. Gandon, "SeWeSe : Semantic Web Server," in *WWW2007 Developers track*, Banff, 2007, pp. http://dannyayers.com/www2007/dev-track-resources.

[76] J-F Baget and M-L Mugnier, "Extensions of Simple Conceptual Graphs: the Complexity of Rules and Constraints," *Journal of Artificial Intelligence Research (JAIR)*, vol. 16, pp. 425-465, 2002.

[77] Jean-François Baget and Marie-Laure Mugnier, "The SG Family: Extensions of Simple Conceptual Graphs," in *17th International Joint*

Conference on Artificial Intelligence, IJCAI , 2001, pp. 205-210.

[78] J.-F. Baget, "RDF Entailment as a Graph Homomorphism," in *4th International Semantic Web Conference, ISWC* , Galway, 2005, pp. 82-96.

[79] Jean-François Baget, "Simple conceptual graphs revisited: hypergraphs and conjunctive types for efficient projection algorithms," in *11th International Conference on Conceptual Sructures, ICCS*, 2003, pp. 195-208.

[80] Marie-Laure Mugnier and Michel Chein, "Characterization and Algorithmic Recognition of Canonical Conceptual Graphs," in *International Conference on Conceptual Structures, ICCS*, 1993, pp. 294-311.

[81] C.J. Rijsbergen, "A new theoretical framework for information retrieval," in *ACM Conference on Research and Development in Information Retrieval*, Pisa, 1986, pp. 194-200.

[82] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker, "Querying the Semantic Web with the Corese Search Engine," in *European Conference on Artificial Intelligence, ECAI*, 2004, p. 705–709.

[83] Jean-François Baget, "Homomorphismes d'hypergraphes pour la subsomption en RDF/RDFS," *RSTI - L'objet*, vol. 2-3, no. 10, pp. 203-216, 2004.

[84] O. Corby and C. Faron Zucker, "Corese : A Corporate Semantic Web Engine," in *International Workshop on Real World RDF and Semantic Web Applications, 11th International World Wide Web Conference*, Hawai, USA, 2002.

[85] Jeremy Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler, "Named Graphs, Provenance and Trust," in *WWW*, Chiba, 2005.

[86] H. Stuckenschmidt, R. Vdovjak, G.J. Houben, and J. Broekstra, "Index Structures and Algorithms for Querying Distributed RDF Repositories," in *WWW* , New York, 2004.

[87] Fabien Gandon, Virginie Bottollier, Olivier Corby, and Priscille Durville, "RDF/XML Source Declaration," in *IADIS International Conference WWW/Internet*, Vila Real, 2007, http://www-sop.inria.fr/edelweiss/people/Fabien.Gandon/docs/WWW_Internet_IADIS_2007_Fabien_Gandon.pdf.

[88] Fabien Gandon, Virginie Bottollier, Olivier Corby, and Priscille Durville. (2007, Septembre) W3C Member Submission. [Online]. http://www.w3.org/Submission/rdfsource/

[89] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. (1998, August) Uniform Resource Identifiers (URI): Generic Syntax. [Online]. http://www.ietf.org/rfc/rfc2396.txt

[90] Jean-François Baget, Olivier Corby, Rose Dieng-Kuntz, Catherine Faron-Zucker, Fabien Gandon, Alain Giboin, Alain Gutierrez, Michel Leclère, Marie-Laure Mugnier, and Rallou Thomopoulos, "Griwes: Generic Model and Preliminary Specifications for a Graph-Based Knowledge Representation Toolkit," in *ICCS*, Toulouse, 2008, pp. 297-310, http://www-sop.inria.fr/edelweiss/people/Fabien.Gandon/docs/iccs_2008/Griwes_ICCS2008.pdf.

[91] David Genest and Eric Salvat, "A Platform Allowing Typed Nested Graphs: How CoGITo Became CoGITaNT," in *6th International Conference on Conceptual Structures*, Montpellier, France, 1998, pp. 154-161.

[92] Christian De Sainte Marie. (2008) RIF Production Rule Dialect. [Online]. http://www.w3.org/TR/2008/WD-rif-prd-20080730/

[93] Harold Boley and Michael Kifer. (2008) RIF Basic Logic Dialect, W3C Working Draft 30 July 2008. [Online]. http://www.w3.org/TR/2008/WD-rif-bld-20080730/

[94] Jos De Bruijn. (2008, Juillet) RIF RDF and OWL Compatibility, W3C Working Draft. [Online]. http://www.w3.org/TR/2008/WD-rif-rdf-owl-20080730/

[95] Jean-François Baget, "Improving the Forward Chaining Algorithm for Conceptual Graphs Rules," in *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2004, pp. 407-414.

[96] J-F Baget and M-L Mugnier, "Extensions of Simple Conceptual Graphs: the Complexity of Rules and Constraints," *Journal of Artificial Intelligence Research (JAIR)*, vol. 16, pp. 425-465, 2002.

[97] Eric Salvat, "Theorem Proving Using Graph Operations in the Conceptual Graph Formalism," in *13th European Conference on Artificial Intelligence (ECAI)*, Brighton, UK, 1998.

[98] Steve Pepper, Fabio Vitali, Lars, Marius Garshol, Nicola Gessa, and Valentina Presutti. (2006, February) A Survey of RDF/Topic Maps Interoperability Proposals, W3C Working Group Note. [Online]. http://www.w3.org/TR/rdftm-survey/

[99] Freddy Limpens, Fabien Gandon, and Michel Buffa, "Rapprocher les ontologies et les folksonomies pour la gestion des connaissances partagées : un état de l'art," in *19èmes Journées Francophones d'Ingénierie des Connaissances, IC*, Nancy, 2008.

[100] N. Guarino and P. Giaretta, "Ontologies and Knowledge Bases: Towards a Terminological Clarification.," in *Towards Very Large Knowledge Bases*.: IOS Press, 1995.

[101] M. Quillian, "Semantic Memory," in *Semantic Information Processing, Readings in Cognitive Science*, M. Minsky, Ed.: MIT Press reprinted in Collins & Smith, 1968, ch. section 2.1, pp. 227-270.

[102] A. Collins and E. Loftus., "A spreading activation theory of semantic processing," *Psycho-logical Review*, no. 82, pp. 407-428, 1975.

[103] Roy Rada, Hafedh, Bicknell, Ellen Mili, and Maria Blettner, "Development and application of a metric on semantic nets," *IEEE Transaction on Systems, Man, and Cybernetics,* , vol. 1, no. 19, pp. 17-30, February 1989.

[104] Z. Wu and M. Palmer, "Verb Semantics and Lexical Selection," in *32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico., 1994.

[105] P. Resnik, "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Applications to Problems of Ambiguity in Natural Language," *In Journal of Artificial Intelligence Research*, no. 11, pp. 95-130, 1995.

[106] J. Jiang and D. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," in *International Conference on Research in Computational Linguistics*, Taiwan, 1997.

[107] A.L. Ralescu and A. Fadlalla, "The Issue of Semantic Distance in Knowledge Representation with Conceptual Graphs," in *AWOCS*, 1990, pp. 141-142.

[108] J. Zhong, H. Zhu, J. Li, and Y. Yu, "Conceptual Graph Matching for Semantic Search," in *10th International Conference on Conceptual Structures, ICCS* , Borovets, Bulgaria, 2002, pp. 92-106.

[109] C. Leacock and M. Chodorow, "Filling in a sparse training space for word sense identification," ms 1994.

[110] H. Zargayouna and S. Salotti, "Mesure de similarité dans une ontologie pour l'indexation sémantique de documents XML," in *Ingénierie des Connaissances*, 2004.

[111] G. Hirst and D. St-Onge, "Lexical Chains as representation of context for the detection and correction malapropisms," in *WordNet: An electronic lexical database and some of its applications*, C. Fellbaum, Ed. Cambrige, MA, USA: The MIT Press, 1997.

[112] A Tversky, "Features of similarity ," *Psychological Review*, no. 84, pp. 327-352, 1977.

[113] P. Resnik, *Selection and Information: A Class-Based Approach to Lexical Relationships*.: Ph.D. thesis, University of Pennsylvania, 1993.

[114] P. Resnik, "Semantic classes and syntactic ambiguity," in *ARPA Human Language Technology Workshop*, 1993.

[115] D. Lin, "An information-theoretic definition of similarity," in *International Conference on Machine Learning*, 1998.

[116] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 3, no. 31, pp. 264-323, 1999.

[117] Fabien Gandon, Olivier Corby, Ibrahima Diop, and Moussa Lo, "Distances sémantiques dans des applications de gestion d'information utilisant le web sémantique," in *Semantic similarity workshop at EGC*, Sophia Antipolis, 2008,.

[118] M Boutet, A Canto, and E Roux, "Plateforme d'étude et de comparaison de distances conceptuelles," Ecole Supérieure En Sciences Informatiques, Nice - Sophia Antipolis, Rapport de Master 2005.

[119] Mustapha Baziz, *Indexation Conceptuelle Guidée par Ontologie pour la Recherche d'Information*. Toulouse: Université Paul Sabatier, IRIT, 2005.

[120] Yann Vigile Hoareau, Fabien Gandon, Alain Giboin, Guy Denhière, Sandra Jhean-Larose, Wolfgang Lenhard, and Herbert Baier, "Similarity measurement applied to information research and indexing.," in *European Workshop on Latent Semantic Analysis in Technology Enhanced Learning,* Heerlen , 2007, pp. 15-16.

[121] T.K. Landauer and S.T Dumais, "A solution to Plato's problem: The Latent Se-mantic Analysis theory of the acquisition, induction, and representation of knowledge," *Psychological Review*, no. 104, pp. 211-240, 1997.

[122] K. Sparck Jones, "A statistical interpretation of term specificity and its application," *Journal of Documentation*, vol. 28, pp. 11-21, 1972.

[123] William James, *Principles of psychology*.: New York: Holt, 1890.

[124] B. Bachimont, "Engagement sémantique et engagement ontologique: conception et réalisation d'ontologies en ingénierie des connaissances," in *Ingénierie des connaissances Evolutions récentes et nouveaux défis*, J. Charlet, M. Zacklad, G. Kassel, and D. Bourigault, Eds.: Eyrolles, 2000.

[125] Emmanuel Blanchard, Mounira Harzallah, and Pascale Kuntz, "A generic framework for comparing semantic similarities on a subsumption hierarchy," in *ECAI*, 2008.

[126] Bastian Quilitz and Ulf Leser, "Querying Distributed RDF Data Sources with SPARQL," in *ESWC*, 2008, p. 524–538.

[127] M. Cai and M. Frank, "RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network," in *International World Wide Web Conference.*, 2004.

[128] G. Kokkinidis, L. Sidirourgos, and V. Christophides, "Book chapter," in *Semantic Web and Peer-to-Peer*, S. Staab and H. Stuckenschmidt, Eds.: Springer-Verlag, 2006.

[129] A. Harth and S. Decker, "Optimized index structures for querying rdf from the web," in *LA-WEB the 3rd Latin American Web Congress*, 2005.

[130] V Christophides, D Plexousakisa, M Scholl, and S Tourtounis, "On labeling schemes for the semantic web," in *13th World Wide Web Conference*, 2003, p. 544–555.

[131] Steels L., "Corporate Knowledge Management ," in *ISMICK*, Compiègne, 1993, pp. 9-30.

[132] M. Wooldridge, N. Jennings, and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design," in *Autonomous Agents and Multi-Agent Systems, AAMAS*, 2000, pp. 285-312.

[133] J. Ferber and O. Gutnecht, "Aalaadin : a meta-model for the analysis and design of organizations in multi-agent systems, ," LIRMM , Montpellier, Rapport de Recherche 97189, 1997.

[134] R. Davis and R. G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving.," *Artificial Intelligence*, vol. 1, no. 20, pp. 63-100, 1983.

[135] E. Bertino, "An indexing technique for object-oriented databases," in *International Conference on Data Engineering, IEEE Computer Society*, 1991, pp. 160-170.

[136] E. Bertino and P. Foscoli, "Index organizations for object-oriented database systems," *TKDE*, vol. 7, no. 2, pp. 193-209, 1995.

[137] B. Shidlovsky and E. Bertino, "A graph-theoretic approach to indexing in object-oriented databases," in *Twelfth International Conference on Data Engineering, IEEE Computer Society*, 1996, pp. 230-237.

[138] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "The TSIMMIS Project: Integration of Heterogeneous Information Sources ," in *IPSJ*, 1994.

[139] I. Muslea, S. Minton, and C. Knoblock, "A Hierarchical Approach to Wrapper Induction," in *Third Annual Conference on Autonomous Agents*, Seattle, WA USA, 1999, pp. 190-197.

[140] L. Liu, C. Pu, and W. Han, "XWRAP : An XML Enabled Wrapper Construction System for Web Information Sources. ," in *International Conference on Data Engineering (ICDE)*, San Diego, 2000.

[141] A. Sahuguet and F. Azavant, "Building Light Weight Wrapper Construction System for Legacy Web Data Sources Using W4F," in *International Conference on Very Large Databases (VLDB)*, Edinburgh, Scotland, 1999.

[142] R. Akkiraju, J. Farrell, J.A. Miller, M. Nagarajan, M.-T. Schmidt, A. Sheth, and K. Verma, "Web Service Semantics WSDL-S," Technical Note 2005.

[143] OWL-S Coalition. (2004) OWL-S Specification. [Online]. http://www.daml.org/services/owl-s/1.1/

[144] WSMO working group. (2004) Web Service modeling Language. [Online]. http://www.wsmo.org/2004/d2/

[145] M. Lo and F. Gandon, "Semantic web services in corporate memories," in *ICIW International Conference on Internet and Web Applications and Services*, Mauritius, 2007.

[146] D. S. Linthicum, *Enterprise Application Integration*.: Addison-Wesley Information Technology Series, 1999.

[147] W3C. (2002) Web Service Activity. [Online]. http://www.w3.org/2002/ws/

[148] IBM. (2002) BEPL4WS, Business Process Execution Language for Web Services. [Online]. http://www-130.ibm.com/developerworks/webservices/

[149] BPMI. (2004) BPML, Business Process Modeling Language. [Online]. http://www.bpmi.org/

[150] W3C. (2002) WSCI, Web Service Choreography Interface. [Online]. http://www.w3.org/TR/wsci/

[151] Microsoft. (2001) XLANG, Web Services for Business Process Design. [Online]. http://www.ebpml.org/xlang.htm

[152] W3C. (2002) WSCL, Web Services Conversation Language. [Online]. http://www.w3.org/TR/2002/NOTE-wscl10-20020314/

[153] IBM. (2001) WSFL, Web Service Flow Language. [Online]. http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[154] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara, "Bringing Semantics to Web Services : the OWL-S Approach," in *SWSWPC*, 2004.

[155] C. Bussler, D. Fensel, and A. Maedche, "A Conceptual Architecture for Semantic Web Enabled Web Services, ," ACM , 2002.

[156] Semantic Web Enabled Web Services Project. [Online]. http://swws.semanticweb.org

[157] J. Farrell and H. Lause. (2007, August) Semantic Annotations for WSDL and XML Schema, W3C Recommendation. [Online]. http://www.w3.org/TR/sawsdl/

[158] Moussa Lo and Fabien Gandon, "Integrating dynamic resources in corporate semantic web: an approach to enterprise application integration using semantic web services," INRIA, Sophia Antipolis, Research Report RR-5663, 2005.

[159] Fabien Gandon, "Generating Surrogates to Make the Semantic Web Intelligible to End-Users," in *IEEE/WIC/ACM International Conference on Web Intelligence*, Compiegne University of Technology, 2005.

[160] Fabien Gandon, "Combining reactive & deliberative agents for complete ecosystems in infospheres," in *IEEE/WIC International Conference on Intelligent Agent Technology (IAT)*, Halifax, Canada, 2003.

[161] L. Floridi, "Information Ethics: On the Theoretical Foundations of Computer Ethics," *Ethics and Information Technology*, vol. 1, no. 1, pp. 37-56, 1999.

[162] M. Klusch, "Information Agent Technology for the Internet: A Survey," *Journal on Data and Knowledge Engineering*, vol. 36, no. 3, 2001, Special Issue on Intelligent Information Integration.

[163] H. V. Parunak, S. Brueckner, and J. Sauter, "ERIM's Approach to Fine-Grained Agents," in *NASA Workshop on Radical Agent Concepts*, Greenbelt, MD, USA, 2002.

[164] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence*.: Oxford University Press, 1999.

[165] O. Babaoglu, H. Meling, and A. Montresor, "Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems," in *22nd International Conference on Distributed Computing Systems (ICDCS)*, Vienna, 2002.

[166] K. P. Sycara, "Multiagent Systems," *AI Magazine*, vol. 19, no. 2, pp. 79-92, 1998.

[167] P. Horn. (2003, July) Autonomic Computing: IBM's Perspective on the State of Information Technology. [Online]. http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf

[168] P. Grassé, "La Reconstruction du nid et les Coordinations Inter-Individuelles chez Bellicositermes Natalensis et Cubitermes sp. La théorie de la Stigmergie: Essai d'interprétation du Comportement des Termites Constructeurs," *Insectes Sociaux*, vol. 6, pp. 41-84, 1959.

[169] A. Newell, "The knowledge level," *Artificial Intelligence*, no. 18 , p. 87–127, 1982.

[170] James Clark. (1999, November) XSL Transformations (XSLT), W3C Recommendation. [Online]. http://www.w3.org/TR/xslt

[171] G. J. Bex, S. Maneth, and F. Neven, "A formal model for an expressive fragment of XSLT," in *International Conference on Computational Logic*, 2000, pp. 1137-1151.

[172] G. Tummarello, C. Morbidoni, P. Puliti, and F. Piazza, "Signing individual fragments of an RDF graph," in *Special interest tracks and posters of the World Wide Web Conference*, 2005, pp. 10-14.

[173] Boley, Grosof, Tabet, and Wagner. (2003) RuleML DTDs, The Rule Markup Initiative RuleML. [Online]. http://www.dfki.uni-kl.de/ruleml/indtd0.8.html

[174] OASIS. Universal Description, Discovery and Integration standard. [Online]. http://www.uddi.org/

[175] Chang, Sherrill, Tolle, and Weber, "InfoBridge: Peer-to-peer location-aware virtual posters, ," Carnegie Mellon University, project report 2003.

[176] E. Greengrass, "Information Retrieval: A Survey," , 2000, http://www.csee.umbc.edu/cadip/readings/IR.report.120600.book.pdf.

[177] R. Korfhage, *Information Storage and Retrieval*.: Wiley & Sons, 1997.

[178] J. Euzenat, "Towards formal knowledge intelligibility at the semiotic level ," in *Workshop Applied Semiotics: Control Problems at ECAI*, 2000, p. 59–61.

[179] N. Guarino and C. Welty, "Towards a methodology for ontology-based model engineering," in *Workshop on Model Engineering, ECOOP*, 2000.

[180] Chris Bizer, Ryan Lee, and Emmanuel Pietriga. (2005, June) Fresnel - Display Vocabulary for RDF, W3C User Manual. [Online]. http://www.w3.org/2005/04/fresnel-info/manual/

[181] J Gogen, "An introduction to algebraic semiotics with applications to user interface designn," , 1999.

[182] J. Sowa, "Ontology, Metadata, and Semiotics, Conceptual Structures: Logical, Linguistic, and Computational Issues," , 2000, p. 55–81.

[183] A Pietarinen, "The semantic + pragmatic web = the semiotic web," in *IADIS International Conference WWW/Internet*, 2003, p. 981–984.

[184] B. Decker, E. Ras, J. Rech, B. Klein, and C. Hoecht, "Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis," in *Workshop on Semantic Web Enabled Software Engineering (SWESE), ISWC*, Galway, 2005.

[185] S.E. Campanini, P. Castagna, and R. Tazzoli, "Platypus Wiki: a Semantic Wiki Wiki Web. Semantic Web Applications and Perspectives," in *Semantic Web Workshop* , 2004.

[186] D. Aumueller, "SHAWN: Structure Helps a Wiki Navigate," in *Proc. BTW-Workshop*, 2005, http://dbs.uni-leipzig.de/~david/2005/aumueller05shawn.pdf.

[187] A. Souzis, "Building a Semantic Wiki," *IEEE Intelligent Systems*, vol. 5, no. 20, pp. 87-91, September/October 2005.

[188] M. Krötsch, D. Vrandečić, and M. Völke, "Wikipedia and the Semantic Web - The Missing Links ," in *WikiMania*, 2005.

[189] H. Muljadi and H. Takeda, "Semantic Wiki as an Integrated Content and Metadata Management System," in *ISWC*, Galway, Ireland, 2005.

[190] K. Dello, R. Tolksdorf, and E. Paslaru. Makna. [Online]. http://www.apps.ag-nbi.de/makna/wiki/About

[191] D. Aumueller and S. Auer, "Towards a Semantic Wiki Experience – Desktop Integration and Interactivity in WikSAR," in *Workshop Semantic Desktop at ISWC*, Galway, 2005.

[192] S. Schaffert, A. Gruber, and R. Westenthaler, "A Semantic Wiki for Collaborative Knowledge Formation.," in *Semantics*, Vienna, Austria, 2005.

[193] M. Hepp, D. Bachlechner, and K. Siorpaes, "OntoWiki: Community-driven Ontology Engineering and Ontology Usage based on Wikis," in *International Symposium on Wikis*, 2005.

[194] Thomas Gruber, "Ontology of Folksonomy: A Mash-up of Apples and Oranges.," in *First on-Line conference on Metadata and Semantics Research, MTSR*, 2005, p. Invited keynote.

[195] A. Passant, "Using Ontologies to Strengthen Folksonomies and Enrich Information Retrieval in Weblogs: Theoretical background and corporate use-case," in *ICWSM 2007*, Boulder, 2007.

[196] R. Newman. (2005, March) Tag ontology design. [Online]. http://www.holygoat.co.uk/blog/entry/2005-03-23-2

[197] R. Mizoguchi, M. Ikeda, and K. Sinitsa, "Roles of Shared Ontology ," in *AI-ED Research, Intelligence, Conceptualization, Standardization, and Reusability, AIED* , 1997, pp. 537-544.

[198] A. Harth and H. Gassert, "WikiOnt Vocabulary Specification," in *WikiMania*, 2005.

[199] M. Buffa, "Intranet Wikis. ," in *Intraweb workshop, WWW Conference*, Edinburgh, 2006.

[200] F. Bird., "Some ideas to improve tags use in social software, flat hierarchy versus categories in social software.," 2005.

[201] Olsen. H., "Navigation blindness, how to deal with the fact that people tend to ignore navigation tools," The Interaction Designer's Coffee Break. Issue 13 2005.

[202] S. Powers. (2005) Cheap Eats at the Semantic Web Café. [Online]. http://weblog.burningbird.net/archives/2005/01/27/cheap-eats-at-the-semantic-web-cafe/

[203] T. Hammond, T. Hannay, B. Lund, and J. Scott, "Social Bookmarking Tools, a General Review ," *D-Lib Magazine*, vol. 11, no. 4, April 2005.

[204] Fabien Gandon, "Le Web sémantique n'est pas antisocial," in *Ingénierie*

des Connaissances, Semaine de la Connaissance, Nantes, 2006, pp. 131-140.

[205] E. Miller, "The Semantic Web is Here. ," in *Semantics Technology Conference* , W3C Semantic Web Activity Lead, 2005.

[206] W3C. (2001) Semantic Web Activity. [Online]. http://www.w3.org/2001/sw/

[207] J-P. Cahier, M. Zacklad, and A. Monceaux, "Une application du Web socio sémantique à la définition d'un annuaire métier en ingénierie," in *Journées Ingénierie des Connaissances IC*, Lyon, 2004, pp. 29-40.

[208] J Caussanel, J-P Cahier, M Zacklad, and J Charlet, "Les Topic Maps sont-ils un bon candidat pour l'ingénierie du Web Sémantique," in *Ingénierie des Connaissances, IC*, Rouen, 2002, pp. 3-14.

[209] M. Zacklad, J.P. Cahier, and X. Pétard, "Du Web Cognitivement Sémantique au Web Socio Sémantique," in *Journée Web Sémantique et SHS*, 2003, http://www.lalic.paris4.sorbonne.fr/stic/mai2003/Trans.

[210] M. Zacklad, "Introduction aux ontologies sémiotiques dans le Web socio sémantique," in *Conférence Ingénierie des Connaissances*, Nice, 2005, pp. 241-252.

[211] T. Berners-Lee. (2005) Putting the Web back in Semantic Web. [Online]. http://www.w3.org/2005/Talks/1110-iswc-tbl/

[212] W3C. (2001) Semantic Web Best Practice Working Group. [Online]. http://www.w3.org/2001/sw/BestPractices/

[213] Fabien Gandon and Alain Giboin, "Vers des ontologies à l'état sauvage," in *Atelier IC 2.0, joint aux 19èmes Journées Francophones d'Ingénierie des Connaissances*, Nancy, 2008.

[214] M. Zacklad, J-P. Cahier, A. Benel, L. Zaher, C. Lejeune, and C. Zhou, "Hypertopic : une métasémiotique et un protocole pour le Web socio-sémantique," in *Journées Francophones d'Ingénierie des Connaissances*, 2007.

[215] G. Mark, V. Gonzalez, M. Sarini, and C. Simone, "Reconciling Different Perspectives: An Experiment on Technology Support for Articulation," in *COOP*, 2002, pp. 23-37.

[216] E. Hutchins, *Cognition in the Wild*, MIT Press ed., 1995.

[217] L. Von Ahn, M. Blum, and J. John Langford, "Telling Humans and Computers Apart Automatically," *Communications of the ACM*, vol. 2, no. 47, p. 57, February 2004.

[218] L. Von Ahn and L. Dabbish, "Labeling Images with a Computer Game," in *CHI*, 2004.

[219] H. Halpin, V. Robu, and H. Shepherd, "The Complex Dynamics of Collaborative Tagging.," in *WWW*, 2007.

[220] P. Mika, "Ontologies are Us: a Unified Model of Social Networks and Semantics," in *ISWC*, 2005, pp. 522-536.

[221] C. Van Damme, M. Hepp, and K. Siorpaes, "Folksontology: An integrated approach for turning folksonomies into ontologies," in *Bridging the Gap between Semantic Web and Web 2.0 (SemNet 2007)*, 2007, pp. 57-70.

[222] K. Siorpaes and M. Hepp, "Games with a Purpose for the Semantic Web," *IEEE Intelligent Systems*, vol. 23, no. 3, pp. 50-60, 2008.

[223] J. Zhang and D. A. Norman, "Representations in Distributed Cognitive Tasks," *Cognitive Science* , vol. 18, no. 1, pp. 87-122, 1994.

[224] Libby Miller and Dan Brickley. (2000) The Friend of a Friend (FOAF) project. [Online]. http://www.foaf-project.org/

[225] Ian Davis and Eric Vitiello. (2006) RELATIONSHIP: A vocabulary for describing relationships between people. [Online]. http://vocab.org/relationship/

[226] J. Breslin, S. Decker, and U. Bojars. (2008) SIOC initiative : Semantically-Interlinked Online Communities. [Online]. http://sioc-project.org/

9. Table of figures

Figure 1. Positionnement sur les axes de recherche de l'équipe Edelweiss 12

Figure 2. Un exemple schématique d'ontologie : l'incontournable exemple des cubes..... 21

Figure 3. Trois types de signes en sémiotique. 22

Figure 4. Une micro-ontologie pour notre scénario de recherche de livres..... 24

Figure 5. Une annotation utilisant le vocabulaire conceptuel de notre ontologie pour décrire un livre en donnant son titre et son auteur. 24

Figure 6. Une requête utilisant le vocabulaire conceptuel de notre ontologie pour rechercher des livres en donnant leur auteur. 24

Figure 7. Exemple d'ontologie en chimie : composition de molécules ; une partonomie / méronymie en chimie organique est une connaissance ontologique..... 25

Figure 8. Le cycle de vie d'une ontologie. 28

Figure 9. Un environnement ouvert pour l'accès mobile aux services en ligne. 34

Figure 10. Interfaces d'accès aux connaissances et règles d'accès du *e-Wallet*..... 35

Figure 11. MeatAnnot utilise la plate-forme de traitement de la langue naturelle GATE et des ontologies pour permettre l'annotation automatique d'articles en biologie. [18]..... 36

Figure 12. Cours entré sous Microsoft Word (en haut) en utilisant des styles prédéfinis et (en bas) le même cours automatiquement réorganisé pour des séances de TP ou TD..... 38

Figure 13. Application au suivi médical et à la coopération autour du dossier d'un patient : Structurer le dossier patient pour aider la coopération médicale entre acteurs d'un réseau de soins 39

Figure 14. Premières interfaces de requêtes pour Aprobatiom en 2001 43

Figure 15. Dans le système CoMMA, négociation entre agents logiciels gérant la distribution des connaissances (ici, des enchères pour décider de la meilleure base d'archivage pour une nouvelle annotation). 44

Figure 16. Un exemple d'extraction d'annotation de PubMed [22]. 45

Figure 17. Visualisation 3D enrichie par des requêtes SPARQL dans SevenPro pour annoter des scènes explicatives sur le montage d'un produit 46

Figure 18. Dans le portail KmP, visualisation de l'analyse des échanges entre les membres de la Telecom Valley. 48

Figure 19. Visualisation des groupes de compétences sur la Telecom Valley. 49

Figure 20. Résultats de recherche approximée pour une requête n'ayant pas de résultat exact dans la base : « qui conçoit des produits wifi sur Sophia ? » 49

Figure 21. Quelques interfaces d'ECCO illustrant les aspects collaboratifs de cet éditeur. 51

Figure 22. Traitement d'un corpus de mails pour en extraire des annotations, les organiser et naviguer comme dans une FAQ [32] 52

Figure 23. Utilisation d'un wiki sémantique dans une communauté de pédagogues [31] [30]; voir aussi la section 6.5 sur SweetWiki [33]. 53

Figure 24. Navigation hyperbolique dans les annotations et l'ontologie de Sealife avec suggestion de mots-clefs par interrogation de Corese durant la frappe et gestion de l'historique. Ici l'utilisateur navigue autour du concept « fever » 54

Figure 25. Une fois un nœud sélectionné dans la navigation hyperbolique, les documents annotés par le concept sélectionné sont listés, triés et documentés par leurs métadonnées. 55

Figure 26. Piles des activités de standardisation au W3C reposant sur les structures d'arbre de XML et de graphe de RDF (source : www.w3.org). 60

Figure 27. Le triplet est la plus petite division de connaissances en RDF 61

Figure 28. Triplets RDF comme les arcs d'un graphe orienté étiqueté distribué sur le web. 61

Figure 29. Exemple de syntaxe XML pour RDF. 61

Figure 30. Exemple de syntaxe RDFa pour du RDF dans une page web. 62

Figure 31. Principe général du moteur de recherche sémantique Corese 69

Figure 32. Les couches du web sémantique (www.w3.org 2007) 71

Figure 33. Requête SPARQL utilisant la source des triplets ou graphes nommés. 76

Figure 34. Un exemple de graphe RDF classique 77

Figure 35. Un exemple de sérialisation du graphe RDF en Figure 34 77

Figure 36. Triplets correspondants au graphe RDF en Figure 34..... 77

Figure 37. Un exemple de sérialisation avec des déclarations de source 78

Figure 38. Les quadruplets obtenus à partir de la sérialisation en Figure 37 78

Figure 39. Graphe obtenu à partir de la sérialisation en Figure 37..... 78

Figure 40. Un exemple déclaration de source sur la propriété d'un nœud anonyme 79

Figure 41. Graphe RDF sérialisé en Figure 40 79

Figure 42. Quadruplets générés par le RDF sérialisé en Figure 40 79

Figure 43. Spécification d'une source extérieure pour la déclaration d'une propriété..... 80

Figure 44. Annotation du graphe RDF de l'ontologie en identifiant sa source..... 80

Figure 45. Architecture de Griwes..... 83

Figure 46. Exemple de graphe bipartite représentant un ERGraph 84

Figure 47. Sous-ERGraph induit par $E_{G'}=\{e_1, e_2, e_3,e_4,e_7\}$ 85

Figure 48. Hiérarchie des *mappings* définis dans Griwes 86

Figure 49. Un homomorphisme de *H* dans *G* 86

Figure 50. Requête SPARQL avec une clause FILTER à traduire en système de contraintes 87

Figure 51. Tableau de définition de RDF dans le modèle de Griwes..... 90

Figure 52. Transformation effectuée par $dist_{MH}$ 109

Figure 53. Transformation effectuée par $dist_{CH}$..... 110

Figure 54. Exemple pour la vérification de l'ordre de regroupement 112

Figure 55. Vue en Radar sur 180° des regroupements de compétences. 113

Figure 56. Applet de l'exercice de placement..... 114

Figure 57. Distances moyenne entre Camion et d'autres véhicules sur un échantillon de trente personnes. 115

Figure 58. Nuages de termes obtenus avec la distance en fonction d'un concept sélectionné par l'utilisateur 117

Figure 59. Suggestion dans l'interface en utilisant la distance pour ordonner les options alors qu'un utilisateur construit une requête simple..... 117

Figure 60. Précision de la désambiguïsation entre subsomption et signature 120

Figure 61. Première ontologie pour la comparaison LSA vs. distance ontologique 122

Figure 62. Corrélations LSA distance - ontologique pour la première ontologie 122

Figure 63. Deuxième ontologie pour la comparaison LSA vs. distance ontologique 123

Figure 64. Corrélations LSA distance - ontologique pour la deuxième ontologie 123

Figure 65. Performance à l'attribution de mots-clés pour 10 rapports de recherche de l'INRIA en fonction des différents espaces métriques : LSA classique, LSA sur un corpus lemmatisé et LSA augmenté par des répétitions de termes basées sur une ontologie. L'ontologie utilisée est basée sur le thésaurus ACM98. 124

Figure 66. Schéma général du système CoMMA [21] 135

Figure 67. Société hiérarchique 138

Figure 68. Sous-partie d'un diagramme d'interaction pour l'allocation d'une annotation.. 141

Figure 69. Sous-partie d'un diagramme d'interaction pour la résolution d'une requête 141

Figure 70. Architecture globale d'un système à base de *hubs*. 143

Figure 71. Exemple d'annotation représentant un chemin d'index de longueur 3. 145

Figure 72. Exemples de chemins et d'étoiles d'index générés à partir d'une annotation. .. 146

Figure 73. Exemple de corps de requête. 148

Figure 74. Exemple de sous requête générée pour un chemin de la Figure 73. 148

Figure 75. Exemple de sous requête générée pour une étoile de la Figure 73. 149

Figure 76. Copie d'écran de l'interface de visualisation des index dans *SevenPro*. 149

Figure 77. Copie d'écran de l'interface de suivi de la résolution distribuée. 150

Figure 78. Visualisation 3D enrichie par des requêtes SPARQL dans *SevenPro*. 150

Figure 79. Processus d'extraction d'annotations par des feuilles XSLT 153

Figure 80. Template de haut niveau réutilisable pour l'extraction de listes. 155

Figure 81. Interface d'édition de feuilles d'extraction d'annotation. 156

Figure 82. Exemple d'extraction d'annotation sur un catalogue de rapports de recherche 158

Figure 83. Triptyque de l'architecture de service [17] 162

Figure 84. Partie implantée de la pile des standards 163

Figure 85. Corese utilisé comme un registre UDDI sémantique 164

Figure 86. Extrait de l'interface de recherche d'un service 165

Figure 87. Découverte et invocation d'un SWS d'entreprise 166

Figure 88. Exemple de composition interactive de services 168

Figure 89. Règle pour la détection de services composables 169

Figure 90. Séquence de services pour l'achat d'un livre 170

Figure 91. Complétion automatique des entrées possibles 171

Figure 92. Anatomie d'un agent XSLT réactif 179

Figure 93. Règle minimale de recopie de l'information rencontrée 179

Figure 94. Un agent XSLT propageant un changement d'URI 180

Figure 95. Phéromone laissée par un agent de bee2bee 182

Figure 96. Nombre de rapports vs. Recouvrements trouvés entre les auteurs 182

Figure 97. Clusters comme obtenus en section 4.4 189

Figure 98. Représentation des clusters de compétences sur la Telecom Valley de Sophia Antipolis telle que souhaitée par les utilisateurs de KmP 190

Figure 99. Analyse de répartition projetée sur trois niveaux de détails différents dans KmP 191

Figure 100. Méthodes ascendantes et descendantes d'attribution d'un angle à une classe de l'ontologie 192

Figure 101. Division descendante de l'ontologie de KmP appliquée sur 360° 193

Figure 102. Placement des clusters selon leur angle ontologique et leur cardinal 194

Figure 103. Détail du cluster de compétences central à la Telecom Valley de Sophia Antipolis 194

Figure 104. Etapes de traitement dans le *e-Wallet* 198

Figure 105. Architecture en couches du *e-Wallet* 199

Figure 106. Architecture du *e-Wallet* pour la traduction et traitement. 199

Figure 107. Déclarer en CLIPS la symétrie des relations et sa sémantique en OWL 200

Figure 108. Règle d'invocation d'un service pour accéder à l'agenda de l'utilisateur. 201

Figure 109. Règle de confidentialité limitant la localisation à la présence sur le campus. 202

Figure 110. Vue d'ensemble du système *my*Campus. 206

Figure 111. Ecran d'accueil et liste des services disponibles. 206

Figure 112. Deux écrans de l'interface du Concierge. 207

Figure 113. Deux écrans de l'interface du Messager. 208

Figure 114. Une zone de posters à Carnegie Mellon. 209

Figure 115. Parcours quotidiens typiques d'un étudiant en design (à gauche) et d'un étudiant en informatique (à droite). 210

Figure 116. Deux écrans de l'interface InfoBridge. 210

Figure 117. Les services de Cinéma et Météorologie. 211

Figure 118. Cartographie et niveaux de confidentialité 212

Figure 119. Organiser une réunion avec 2 e-Wallets 213
Figure 120. Projeter une présentation 214
Figure 121. Extraits des statistiques sur l'historique 218
Figure 122. Un exemple de requête sur des annotations de documents. 222
Figure 123. Un exemple de formatage du résultat. 222
Figure 124. Quatre exemples de substituts dans les résultats d'un moteur de recherche 224
Figure 125. Amélioration automatique des substituts (2ième résultat) dans SearchMonkey 224
Figure 126. Exemple de règle de production 227
Figure 127. Exemple de règle d'équivalence 228
Figure 128. Règle donnant une condition suffisante pour être président 228
Figure 129. Ajout de toutes les propriétés disponibles 230
Figure 130. Ajout de toutes les propriétés disponibles 230
Figure 131. Règles pour détecter l'équivalence entre personnes (nom & naissance) 232
Figure 132. Règle pour détecter l'équivalence entre personnes (adresse mél). 232
Figure 133. Règle pour détecter l'équivalence entre documents 233
Figure 134. Substituts générés 233
Figure 135. Augmentation d'une requête par la technique des substituts 234
Figure 136. Une réponse augmentée par les résultats des substituts. 235
Figure 137. Schéma du principe de SweetWiki 242
Figure 138. Editeur WYSIWYG et tags pour la page – Communauté ePre du projet IST Palette 247
Figure 139. Visualisation de la page, de ses tags et suggestions 247
Figure 140. L'éléphant du web sémantique [205] 251