Découverte de correspondances sémantiques entre ressources hétérogènes dans un environnement coopératif

Catarina Ferreira da Silva

▶ **To cite this version:**

Catarina Ferreira da Silva. Découverte de correspondances sémantiques entre ressources hétérogènes dans un environnement coopératif. Autre [cs.OH]. Université Claude Bernard - Lyon I, 2007. Français. NNT : . tel-00776673

HAL Id: tel-00776673

https://theses.hal.science/tel-00776673v1

Submitted on 16 Jan 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre 281-2007 Année 2007

THESE

présentée

devant l'UNIVERSITE CLAUDE BERNARD - LYON 1

pour l'obtention

du DIPLOME DE DOCTORAT EN INFORMATIQUE

(arrêté du 7 août 2006)

présentée et soutenue publiquement le 7 décembre 2007

par

DOMINGUES ALVES FERREIRA DA SILVA Catarina Eufémia

TITRE : Découverte de correspondances sémantiques entre ressources hétérogènes dans un environnement coopératif

JURY :

Présidente : Mme Chantal REYNAUD, Professeur, Université de Paris XI

Rapporteurs :

Mme Corine CAUVET, Professeur, Université Paul Cézanne Aix-Marseille 3

M. Aris M. OUKSEL, Professeur, Université d'Illinois, Chicago, États-Unis

Examinateurs :

M. Paulo RUPINO DA CUNHA, Professeur, Université de Coimbra, Portugal

M. Djamal BENSLIMANE, Professeur, Université Claude Bernard Lyon 1

Directeurs :

Mme Parisa GHODOUS, Professeur, Université Claude Bernard Lyon 1

M. Celson LIMA, Chercheur, Centre Scientifique et Technique du Bâtiment

M. Lionel MÉDINI, Maître de Conférences, Université Claude Bernard Lyon 1

À Erwan, Isabel et Armando

Remerciements

Je tiens à remercier l'Association Nationale de Recherche Scientifique et le Centre Scientifique et Technique du Bâtiment (CSTB) pour leur soutien matériel sans lequel cette thèse n'aurait pas abouti.

Je remercie chaleureusement Madame la Professeur **Parisa GHODOUS** qui, en sa qualité de directrice de thèse, a été la garante des grandes orientations de ce travail. J'ai été particulièrement touché par sa disponibilité lorsque j'en ai eu besoin. Je remercie Monsieur **Lionel MÉDINI**, Maître de Conférences à l'Université Claude Bernard Lyon 1 et co-directeur de thèse, et Monsieur **Celson LIMA**, Docteur en informatique, chercheur au CSTB et responsable industriel de la convention CIFRE, qui m'ont guidée sur le chemin de cette thèse grâce à leurs conseils avisés et leurs remarques constructives. Je suis reconnaissante à tous les trois de m'avoir fait confiance.

Je remercie Monsieur **Lionel MÉDINI** pour sa relecture attentive. Ses suggestions ont grandement contribué à la lisibilité du document. Je remercie aussi Monsieur **Eric TOSAN**, Professeur à l'Université Claude Bernard Lyon 1, et Monsieur **Nicolas LUMINEAU**, Maître de Conférences à l'Université Claude Bernard Lyon 1, pour leurs relectures supplémentaires.

Je voudrais témoigner ma reconnaissance à Monsieur **Patrick MORAND**, directeur du site de Sophia-Antipolis du CSTB et à Monsieur **Alain ZARLI**, chef du service Ingénierie de l'Innovation et des Services du CSTB pour m'avoir accueilli dans l'équipe.

Je voudrais également témoigner ma reconnaissance à Monsieur le professeur **Bernard PÉROCHE**, directeur du Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS), pour m'avoir accueilli dans ce laboratoire.

J'adresse mes remerciements à Madame **Corine CAUVET**, Professeur à l'Université Paul Cézanne Aix-Marseille 3 ainsi qu'à Monsieur **Aris OUKSEL**, Professeur à l'Université d'Illinois Chicago aux États-Unis, qui m'ont fait l'honneur d'être rapporteurs de ma thèse et de faire partie de mon jury.

Que Madame **Chantal REYNAUD**, Professeur à l'université de Paris-Sud, Orsay, soit remerciée pour avoir accepté de présider mon jury.

Je remercie aussi Monsieur **Paulo RUPINO DA CUNHA**, Professeur à l'Université de Coimbra au Portugal et Monsieur **Djamal BENSLIMANE**, Professeur à l'Université Claude Bernard Lyon 1 de l'intérêt qu'ils ont porté à mon travail et de leur participation au jury.

J'adresse aussi mes remerciements aux membres de l'équipe Systèmes d'Information Collaboratifs du LIRIS avec lesquels j'ai pu partager des moments agréables de recherche mais aussi de détente.

Ce mémoire est la trace de quatre années de travail mais aussi de quatre années de vie. Je tiens à remercier tous ces qui m'ont accompagnée et encouragée pendant ce temps, en particulier mes parents, ma sœur et mes amis : **Jocelyne**, **Bernard**, **Tangi**, **Gireg**, **Gilbert**, **Barbara**, **Cláudia**, **Catarina**, **Teresa**, **Luísa**, **Hugues**, **Manuela**, **Luís**, **Isabelle**, **Mee Ja**, **Albert**, **Moisés**, **Samer** et **Patrick**.

Mon dernier remerciement, mais pas le moindre, ira à **Erwan** pour son amour et pour partager sa vie avec moi.

Table des matières

Introduction 17

Organisation de la thèse 20

Partie I - État de l'art sur les ressources sémantiques, les méthodes et les outils 21

Chapitre 1 – Hétérogénéité, Interopérabilité et Alignement 23

1.1 Syntaxe et sémantique 23

1.2 Types de ressources sémantiques 27

1.2.1 Dictionnaires, taxonomies, thésaurus et ontologies 27
1.2.2 Analyse des liens entre les types de ressources sémantiques 31
1.3 Étude sur les relations 31

1.4 Types d'hétérogénéités 34

1.5 Interopérabilité des ressources sémantiques 40

1.6 Stratégies de résolution des problèmes d'interopérabilité sémantique 44

1.7 Méthodes d'alignement 45

1.7.1 Méthodes terminologiques 45
1.7.1.1 Méthodes lexicales 46
1.7.1.2 Méthodes linguistiques 46
1.7.2 Méthodes structurelles 47
1.7.2.1 Méthodes internes 47
1.7.2.2 Méthodes externes 48
1.7.3 Méthodes extensionnelles 49
1.7.4 Méthodes sémantiques 49
1.7.4.1 Méthodes fondées sur la satisfiabilité propositionnelle 49
1.7.4.2 Méthodes fondées sur les logiques de description 50
1.7.5 Méthodes fondées sur le langage de représentation 50
1.7.6 Synthèse sur les méthodes d'alignement 50
1.8 Étude des systèmes d'alignement 51

1.9 Synthèse 62

Chapitre 2 - Méthodes, techniques et outils 65

2.1 Langages de représentation de l'information 65

2.1.1 Le langage EXPRESS 65
2.1.1.1 Application au domaine de la construction 68
2.1.2 Langages dédiés au *World Wide Web* 69
2.2 Inférence et moteurs de raisonnement 82

2.2.1 Les types d'inférences 82
2.2.2 L'algorithme des tableaux sémantiques pour les logiques de description 84
2.2.3 Les moteurs d'inférences 87
2.3 Interfaces de programmation pour les langages du Web Sémantique 89

2.3.1 L'interface de programmation Jena 90

2.3.2 L'interface de programmation OWL API 90
2.4 La logique floue 91
2.5 Synthèse 92
Partie II - Découverte de correspondances sémantiques, application et mise en œuvre 95
Chapitre 3 - Méthodologie de découverte de correspondances sémantiques 97
3.1 Problématique de thèse 97
3.2 Méthodologie proposée 98
3.2.1 Analyse et sélection des ressources sémantiques 99
3.2.2 Homogénéisation syntaxique 100
3.2.2.1 Conversion EXPRESS vers OWL-DL 102
3.2.3 Découverte de correspondances sémantiques 108
3.2.3.1 Paramètres à prendre en compte 108
3.2.3.2 Processus de découverte de correspondances sémantiques 111
3.2.3.3 Algorithme pour la découverte de correspondances sémantiques 112
3.2.3.4 Exemples de résultats de correspondances sémantiques 114
3.2.4 Processus d'unification de propriétés 116
3.2.4.1 Algorithme d'unification de propriétés 116
3.2.5 Algorithme d'inférence d'instances 117
3.2.6 Exemples de résultats pour l'unifications de propriétés 118
3.2.7 Modélisation pour l'évaluation des correspondances sémantiques 120
3.3 Synthèse 126
Chapitre 4 – Application et mise en œuvre 129
4.1 Contexte de l'application 129
4.2 Ressources sémantiques sélectionnées 130
4.2.1 L'ontologie bcBuildingDefinitions 130
4.2.2 L'ontologie e-Cognos 132
4.2.3 Le standard ISO 12006-3 135
4.2.4 Le standard IFC 138
4.2.5 Comparaison des quatre ressources sémantiques 140
4.3 Scénario d'utilisation : l'infrastructure OSIECS 140
4.4 Prototypes réalisés 142
4.4.1 Le prototype FUNONDIL 142
4.4.1.1 Chargement des ressources sémantiques 143
4.4.1.2 Visualisation des ressources sémantiques 145
4.4.1.3 Processus de découverte des correspondances sémantiques 147
4.4.1.4 Visualisation des correspondances sémantiques 149
4.4.2 Le prototype UNIREL 150
4.5 Synthèse 153
Chapitre 5 – Évaluation et discussion 155
5.1 Découverte de correspondances entre les méta-schémas des ressources sémantiques 155

5.1.1 Découverte de correspondances entre les méta-schémas de bcBuildingDefinitions et e-Cognos 155
5.1.2 Découverte de correspondances entre les méta-schémas de bcBuildingDefinitions et ISO 12006-3 157
5.1.3 Découverte de correspondances entre les méta-schémas de bcBuildingDefinitions et IFC-kernel 158
5.1.4 Découverte de correspondances entre les méta-schémas e-Cognos et ISO 12006-3 159
5.1.5 Découverte de correspondances entre les méta-schémas e-Cognos et IFC-kernel 160
5.1.6 Découverte de correspondances entre les méta-schémas IFC-kernel et ISO 12006-3 161
5.1.7 Analyse des correspondances produites au niveau des méta-schémas 162
5.2 Découverte de correspondances entre schémas des ressources sémantiques 163
5.2.1 Expérimentation menée 163
5.2.2 Résultats produits avec les schémas 163
5.2.3 Analyse des correspondances produites au niveau des schémas 165
5.3 Unification de propriétés 165
5.3.1 Influence de l'unification des propriétés sur les correspondances découvertes entre les méta-schémas IFC-kernel et ISO 12006-3 166
5.3.2 Spécificités des ressources *ad hoc* utilisées 166
5.3.3 Adjonction d'axiome pour l'unification de propriétés 169
5.3.4 Résultats de la découverte de correspondances après l'unification de propriétés 169
5.3.5 Analyse des résultats de la découverte de correspondances après l'unification de propriétés 172
5.4 Inférence d'instances 173
5.5 Analyse globale des résultats 178
5.5.1 Tests avec le prototype FUNONDIL 178
5.5.2 Tests avec le prototype UNIREL 179
5.6 Synthèse 180
Conclusions et perspectives 181
Nos contributions 181
Perspectives 183
Références bibliographiques 185
Annexes 197
Annexe 1 – Langages de représentation XML, RDF et RDFS 199
Annexe 2 – Comparaison d'un ensemble de ressources sémantiques européennes 203
Annexe 3 – Implantation Java 207
Annexe 4 – Représentation OWL-DL des correspondances sémantiques entre les méta-schémas de bcBuildingDefinitions, d'e-Cognos, d'IFC-kernel et d'ISO 12006-3 217
Annexe 5 – Représentation OWL-DL des correspondances sémantiques entre le schéma de bcBuildingDefinitions et celui d'eCognos 239

Annexe 6 – Sous-partie d'e-Cognos après l'unification de propriétés 247
Annexe 7 – Sous-parties d'IFC-kernel après l'unification de propriétés 251

Table des figures

Figure 1 - Les disciplines telles que le TALN et l'IC utilisent la sémantique computationnelle en faisant le pont avec la sémantique cognitive par un processus de formalisation. 25
Figure 2 – Exemple d'un graphe qui représente trois prédicats unaires, les concepts usinage, fraisage et trou et deux prédicats binaires, les relations sous-classe et évaser. 29
Figure 3 – Relations entre langage, ontologie, conceptualisation et modèles visés. Adaptée à partir de (Guarino 1998) et de (Ferrario 2006). 30
Figure 4 - Emboîtement des définitions génériques de ressources sémantiques. 31
Figure 5 – Correspondances entre les classifications des hétérogénéités ontologiques de (Visser *et al.* 1998) et (Klein 2001). 36
Figure 6 - Deux structurations différentes d'un même domaine de connaissances. 42
Figure 7 - Représentation des niveaux d'interopérabilité par des couches successives qui s'appuient sur les couches inférieures. 43
Figure 8 – Représentation des dimensions horizontale et verticale de l'interopérabilité entre les ressources sémantiques. 63
Figure 9 – Architecture en couches d'un ensemble de technologies du Web Sémantique, Copyright ©[2006] *World Wide Web Consortium, All Rights Reserved.* 69
Figure 10 – Les langages, les modèles et les types d'interopérabilité 70
Figure 11 – Exemple de représentation d'instances des concepts *Actor* et *Project* 74
Figure 12– Composants d'un système d'inférence floue (Guillaume 2005). 92
Figure 13 – Méthodologie proposée dans ces travaux de thèse. 98
Figure 14 – Processus de conversion syntaxique 101
Figure 15 – Intersection A des concepts A_1 et A_2 ($A_1 \sqcap A_2$) 103
Figure 16 – Types de correspondances : a) Équivalence ; b) Subsomption ; c) Transitivité et d) Conjonction. 110
Figure 17 – Processus de l'algorithme de découverte de correspondances sémantiques. 112
Figure 18 – Fonction d'appartenance pour la variable d'entrée E_1 124
Figure 19 – Fonction d'appartenance pour la variable d'entrée E_2 124
Figure 20 – Fonction d'appartenance pour la variable d'entrée E_3 125
Figure 21 – Représentation UML du méta-schéma XTD, version Lite (Tolman *et al.* 2001). 132
Figure 22 – a) Taxonomie de concepts. b) Taxonomie de relations. 133

Figure 23 – Représentation graphique UML du méta-schéma de l'ontologie e-Cognos (Wetherill *et al.* 2002).....134
Figure 24 – Ensemble de concepts qui spécialisent la classe eCognosConcept (Wetherill *et al.* 2002).....135
Figure 25 – Entités de base du méta-schéma d'ISO 12006-3, représentées dans le langage graphique EXPRESS-G (ISO/PAS 12006-3 version 2, 2002).....136
Figure 26 – Détails de l'entité xtdRelationship du méta-schéma d'ISO 12006-3, représenté en EXPRESS-G (ISO/PAS 12006-3 version 2, 2002).....137
Figure 27 – Représentation en EXPRESS-G du méta-schéma racine de kernel IFC version 2x2 qui a été utilisé pour ce travail. La version la plus récente, IFC2x3, ajoute l'entité abstraite *IfcObjectDefinition*.....139
Figure 28 – Scénario d'application dans le projet FUNSIEC (Lima *et al.* 2005a).....141
Figure 29 – Copie d'écran de la page web du prototype FUNONDIL. Explication sur les types de correspondances sémantiques.....144
Figure 30 – Sous-menus des formats de représentation permis pour chargement des ressources sémantiques.....145
Figure 31 – Étape d'assignement d'identifiant à une ressource sémantique, après le choix du format de représentation et avant son chargement.....145
Figure 32 – Navigation dans le système de fichiers de la machine pour sélection du fichier à charger.....146
Figure 33 – Sous-menus des options de visualisation.....147
Figure 34 – Visualisation graphique d'une partie de la ressource ISO 12006-3 à travers l'outil Treebolic.....148
Figure 35 – Choix des noms des concepts des ressources dans la liste des concepts pour établir des axiomes d'amorçage.....148
Figure 36 – Choix des noms des ressources pour lancer le processus de découverte de correspondances sémantiques.....149
Figure 37 – Visualisation du modèle OSIECS.....150
Figure 38 – Copie d'écran du prototype UNIREL : visualisation de la ressource IFC-Kernel en OWL-DL.....152
Figure 39 – Sélection des propriétés à rendre équivalentes.....152
Figure 40 – Sélection d'une classe d'une des deux ressources pour lancer l'inférence d'instances.....153

Figure 41 – Correspondances découvertes entre les méta-schémas de bcBuildingDefinitions et e-Cognos par FUNONDIL. 156
Figure 42 – Correspondances découvertes entre les méta-schémas de bcBuildingDefinitions et ISO 12006-3 par FUNONDIL....157
Figure 43 – Correspondances découvertes entre les méta-schémas de bcBuildingDefinitions et IFC-kernel par FUNONDIL. 158
Figure 44 – Correspondances découvertes entre les méta-schémas e-Cognos et ISO 12006-3 par FUNONDIL. 159
Figure 45 – Correspondances découvertes entre les méta-schémas e-Cognos et IFC-kernel par FUNONDIL. 161
Figure 46 – Correspondances découvertes entre les méta-schémas ISO 12006-3 et IFC-kernel par FUNONDIL. 162
Figure 47 – Correspondances découvertes entre les ontologies e-Cognos et bcBuildingDefinitions par FUNONDIL. 164
Figure 48 – D'autres correspondances découvertes entre les ontologies e-Cognos et bcBuildingDefinitions par FUNONDIL. 164
Figure 49 – Hiérarchie de concepts dede la sous-partie e-Cognos. 167
Figure 50 – Hiérarchie de concepts de la sous-partie d'IFC-kernel. 167
Figure 51 – Correspondances entre les sous-parties d'eCognos et d'IFC-kernel (cas 5, tableau 17) après l'unification de relations, produites par le prototype UNIREL. 171
Figure 52 – Correspondances entre les sous-parties d'eCognos et IFC (cas 4, tableau 17) après l'unification de propriétés, produites par le prototype UNIREL. Dans ce cas, la première ressource ne contient pas la restriction globale sur la propriété `e-Cognos:isInvolvedIn`. 172
Figure 53 – Inférence d'instances des concepts de la sous-partie d'IFC-kernel de l'exemple 1 (cas 5, tableau 18), produites par le prototype UNIREL. 173
Figure 54 – Inférence d'instances pour l'exemple 2 (cas 4, tableau 18), produites par UNIREL. 177

Table des tableaux

Tableau 1 – Comparaison des outils d'alignement 53
Tableau 2 – Constructeurs du langage OWL DL pour décrire des classes, des types de données, des individus et des valeurs de données (Horrocks, Patel-Schneider et Van Harmelen 2003). 80
Tableau 3 – Constructeurs du langage OWL DL pour décrire des axiomes et des faits (Horrocks, Patel-Schneider et Van Harmelen 2003) 81
Tableau 4 – Réduction des problèmes de raisonnement d'une TBox à des problèmes de subsomption (Baader et Nutt 2003). 83
Tableau 5 – Réduction des problèmes de raisonnement d'une TBox à des problèmes de satisfiabilité (Baader et Nutt 2003) 83
Tableau 6 – Règles d'expansion des tableaux sémantiques pour la LD $\mathcal{ALC}$. 86
Tableau 7 – Correspondances entre les constructeurs des langages EXPRESS, DL et OWL-DL. 104
Tableau 8 – Résumé de l'attribution des plages de valeurs spécifiées pour les termes linguistiques aux variables d'entrée définies (*n* est un entier) 122
Tableau 9 – Similarités entre les entités des quatre méta-schémas. 140
Tableau 10 – Axiomes d'amorçage entre les méta-schémas de bcBuildingDefinitions et e-Cognos. 156
Tableau 11 – Axiomes d'amorçage entre les méta-schémas de bcBuildingDefinitions et ISO 12006-3. 157
Tableau 12 – Axiomes d'amorçage entre les méta-schémas bcBuildingDefinitions et IFC-kernel. 158
Tableau 13 – Axiomes d'amorçage entre les méta-schémas e-Cognos et ISO 12006-3. 159
Tableau 14 – Axiomes d'amorçage entre les méta-schémas e-Cognos et IFC-kernel 160
Tableau 15 – Axiomes d'amorçage entre les méta-schémas IFC-kernel et ISO 12006-3 161
Tableau 16 – Les neuf cas de combinaisons de restrictions étudiés. 169
Tableau 17 – Types de restrictions et correspondances découvertes. La colonne intitulé Ressources sémantiques représente les sous-parties des ressources mentionnées. 170
Tableau 18 – Types de restrictions et instances inférées. 174
Tableau 19 – Comparaison d'un ensemble de ressources sémantiques européennes, données issues du projet FUNSIEC (Barresi *et al.* 2004). 203

Introduction

Le travail de recherche présenté dans ce mémoire traite de la problématique de l'interopérabilité sémantique entre ressources hétérogènes appliquée aux environnements coopératifs. Dans de tels environnements, différentes approches de représentation de l'expertise et des connaissances métier existent. Ces approches varient en fonction des cultures d'entreprise et des finalités recherchées et sont regroupées sous le terme générique de **ressources sémantiques**[1]. Ces dernières comprennent les dictionnaires, les systèmes de classification, les taxonomies, les thésaurus et les ontologies.

Chaque expert dans un domaine d'application utilise fréquemment une ou plusieurs ressources sémantiques. Ces ressources représentent les concepts du domaine de travail. Les différents partenaires d'un projet peuvent également avoir accès à d'autres ressources sémantiques, mises à disposition par leurs partenaires. Ces partenaires devraient pouvoir collaborer à travers une infrastructure de communication, d'échange et de partage, interopérable. Autrement dit, une infrastructure qui permettrait la communication de façon non ambiguë entre les systèmes logiciels, que ces systèmes soient similaires ou différents.

Toutefois l'interopérabilité n'est pas acquise. Les hétérogénéités entre les ressources sémantiques constituent la richesse de la représentation des divers points de vue sur le domaine. Mais, elles apportent autant de difficultés pour l'interopérabilité syntaxique, structurelle et surtout sémantique.

Notre travail porte sur les insuffisances d'interopérabilité sémantique entre ressources hétérogènes et sur une manière d'amoindrir cette insuffisance. Pour cela, nous étudions tout d'abord plusieurs stratégies de résolution des déficiences d'interopérabilité sémantique telles que l'alignement, la fusion et l'intégration d'ontologies. Ainsi, notre état de l'art catégorise et étudie plusieurs types de méthodes d'alignement : terminologiques, structurelles, extensionnelles, sémantiques et les méthodes fondées sur les langages de représentation. Nous analysons aussi plusieurs systèmes d'alignement d'ontologies et d'arbres par la génération de correspondances entre entités des ressources.

---

[1] L'expression *ressources sémantiques* a été employée depuis le projet européen SPICE, *SPecifications for Integrated Construction E-standards*, http://spice.bouw.tno.nl/

En ce qui concerne les méthodes sémantiques, nous constatons l'absence d'une approche intégrée, adaptée à l'alignement de ressources hétérogènes. D'autre part, il nous semble possible d'approfondir l'étude des méthodes sémantiques, peu traitées jusqu'ici.

Pour aider à atténuer l'insuffisance d'interopérabilité sémantique entre les ressources, nous proposons une méthodologie visant, pour l'essentiel, à découvrir des correspondances sémantiques entre des concepts de ressources différentes. Pour atteindre cet objectif, notre méthodologie est constituée de cinq phases.

La première vise à identifier un ensemble de critères d'analyse afin de sélectionner un groupe de ressources sémantiques du domaine d'application (ici, la construction en Europe).

La deuxième phase est dédiée à la conversion de formats. En effet, les ressources sémantiques sont hétérogènes à différents niveaux, y compris au niveau syntaxique. Convertir les différents formats d'origine en un format commun (OWL-DL) rend plus facile la tâche suivante, puisque celle-ci peut alors se focaliser sur les aspects sémantiques.

La troisième étape a pour but de comparer les définitions des ressources sémantiques, représentées en logique formelle, de manière à produire des correspondances sémantiques entre les concepts des ressources. Cette étape s'appuie sur les services d'un moteur d'inférences et sur les logiques de description (LD).

La quatrième phase consiste à enrichir les données existantes en identifiant des équivalences entre relations inter-concepts d'une ressource à une autre. L'objectif est de générer de nouvelles correspondances entre concepts de ces ressources. De plus, elle permet d'inférer les instances de concepts d'une ressource comme instances de concepts d'autre ressource.

La dernière étape consiste à modéliser les connaissances du domaine pour quantifier les correspondances sémantiques non équivalentes produites. Le modèle obtenu utilise la logique floue et pourra assister les experts dans leur tâche d'évaluation des équivalences découvertes.

Notre méthodologie est en partie mise en œuvre dans deux prototypes, qui ont été développés pour assister les experts dans les tâches de découverte de correspondances sémantiques. L'applicabilité de nos propositions a été vérifiée, en particulier dans le projet européen FUNSIEC[2] (*Feasibility study for a UNified Semantic Infrastructure in the European Construction sector*), en utilisant un ensemble de quatre ressources sémantiques du secteur de la construction.

[2] http://www.funsiec.org/

C'est dans le cadre de ce projet que le prototype FUNONDIL a été implanté. Ce prototype permet aux experts :

- de convertir les ressources sémantiques représentées dans les formats EXPRESS, XMI, XSD et XML vers la syntaxe OWL-DL et la syntaxe classique des LD.
- de visualiser ces ressources sémantiques en syntaxe OWL-DL, en syntaxe LD et en représentation graphique avec Treebolic[3].
- de visualiser et de prendre connaissance des types de correspondances sémantiques que l'outil peut découvrir.
- d'ajouter des axiomes d'amorçage.
- de sélectionner les ressources sémantiques et d'appeler les services d'un moteur d'inférences pour détecter des correspondances sémantiques.
- de visualiser les correspondances sémantiques découvertes entre les quatre ressources sémantiques sélectionnées dans le cadre du projet.
- d'appeler un ensemble de services d'inférences de base, tels que tester la satisfiabilité ou interroger sur la subsomption et l'équivalence entre deux concepts de deux ressources sémantiques.

La méthodologie proposée est aussi mise en œuvre, partiellement, dans le prototype UNIREL qui étend FUNONDIL. UNIREL implante le travail sur les relations et permet de réaliser la découverte de correspondances sémantiques ainsi que l'inférence d'instances entre concepts des ressources sémantiques.

Ce travail de thèse a été mené dans le cadre d'une Convention Industrielle de Formation par la Recherche (CIFRE) au sein de l'équipe Systèmes d'Information Collaboratifs (SICo) du Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS) de Lyon. Le partenaire industriel de la convention CIFRE est le Centre Scientifique et Technique du Bâtiment (CSTB), et en particulier le département Technologies de l'Information et Diffusion du Savoir (TIDS).

[3] http://treebolic.sourceforge.net/en/home.htm

Organisation de la thèse

Ce mémoire de thèse est composé de cinq chapitres répartis en deux parties. La première partie présente l'état de l'art sur les ressources sémantiques, les méthodes et les outils. Cette partie est composée de deux chapitres :

- Dans le premier chapitre nous étudions les différents types de ressources sémantiques et les hétérogénéités qui leurs sont liés. Nous analysons les approches qui visent atténuer les hétérogénéités, et en particulier l'alignement d'ontologies. Ce chapitre étudie aussi un ensemble de systèmes qui implantent ces approches.
- Dans le deuxième chapitre nous analysons un ensemble de méthodes, de techniques et d'outils, que nous avons utilisés dans cette thèse. Ce sont les langages de représentation des connaissances, tels que les langages EXPRESS et OWL, les mécanismes d'inférences, les moteurs de raisonnement, deux interfaces de programmation adaptées aux langages du Web Sémantique et la logique floue.

La deuxième partie du mémoire décrit notre apport et présente la méthodologie proposée pour la découverte de correspondances sémantiques. Cette partie présente aussi l'application et les résultats de mise en œuvre. Cette partie est constituée de trois chapitres :

- Le troisième chapitre est le noyau de la thèse et présente nos travaux : la méthodologie de découverte de correspondances sémantiques. Nous détaillons les phases de cette méthodologie et proposons des algorithmes pour la découverte de correspondances sémantiques et pour l'unification de propriétés. Des exemples accompagnent l'explication des algorithmes.
- Le quatrième chapitre concerne l'application et la mise en œuvre. Nous présentons le contexte de l'application, les ressources sémantiques sélectionnées, les prototypes FUNONDIL et UNIREL développés et expliquons leur fonctionnement.
- Le cinquième chapitre présente le protocole expérimentale des testes effectués, les résultats, ainsi que l'analyse de ces derniers.

Nous terminons le mémoire avec les conclusions, nos contributions principales et des perspectives de développement pour cette étude.

Partie I - État de l'art sur les ressources sémantiques, les méthodes et les outils

Chapitre 1 – Hétérogénéité, Interopérabilité et Alignement

L'objectif de ce chapitre est de présenter les définitions des concepts importants pour l'interopérabilité des ressources sémantiques. Nous décrivons différents types de ressources sémantiques et examinons le problème de l'interopérabilité du point de vue de l'hétérogénéité des ressources sémantiques. Nous décrivons un ensemble de méthodes d'alignement qui visent à faciliter l'interopérabilité sémantique. Nous comparons plusieurs systèmes d'alignement.

L'utilisation progressive du Web pour le travail collaboratif et distribué, associée à la standardisation des langages et outils liés au Web Sémantique ont conduit à la disponibilité de nombreuses *ressources sémantiques* dans les domaines professionnels qui utilisent le Web. Nous adoptons l'expression *ressources sémantiques* pour nous référer à l'ensemble de structures qui organisent les informations sur des domaines de connaissances selon certaines règles. Des types de ressources sémantiques sont les dictionnaires, les systèmes de classification, les taxonomies, les thésaurus, les ontologies ou les standards. Ces ressources peuvent traiter un même sujet d'un domaine de connaissances en suivant des points de vue différents et en ayant des objectifs divers. Une ressource sémantique peut représenter des informations en langage naturel. Cependant il est possible de se servir d'un langage formel pour représenter son contenu. Ceci est d'autant plus souhaitable que l'objectif est de permettre des traitements informatiques sur le contenu de la ressource. Pour définir l'interopérabilité des ressources sémantiques, plus loin dans ce chapitre, nous avons besoin d'approfondir les notions de syntaxe et de sémantique qui sont fondamentales dans le domaine de représentation des informations et des connaissances (Liu 2000, Cardoso et Sheth 2006).

1.1 Syntaxe et sémantique

Syntaxe. La syntaxe s'occupe de la combinaison formelle et structurelle des signes ou symboles porteurs d'un sens et de la production d'autres signes (Vaillant 1999, Cardoso et Sheth 2006). Autrement dit, la syntaxe décrit les règles de combinaison de symboles prédéfinis de manière à ce que leur ensemble soit bien formé. Par exemple, en langage naturel, l'ensemble de mots "le bâtiment est habitable" est une phrase bien formée, mais "est habitable bâtiment le" ne l'est pas. Cependant, une syntaxe bien formée est insuffisante pour exprimer de la sémantique. Par

exemple, en informatique, un compilateur commence par valider la syntaxe. Ceci n'assure pas que le programme est sémantiquement correct, *i.e.* qu'il retourne le résultat attendu. Un analyseur syntaxique ou parseur commence par vérifier que le document est bien formé. La formalité syntaxique est l'élément de base et sa force repose sur la standardisation.

Sémantique. D'une manière générale, la sémantique est l'étude du sens des unités linguistiques et de leurs combinaisons (Willerval, Boisseau et Serres-Cousiné 1989). Plus précisément, nous devons distinguer sémantique cognitive et sémantique computationnelle (Zacklad 2005).

La sémantique cognitive se rapporte à l'étude des méthodes pour acquérir, intégrer, conserver et transmettre des connaissances. Dans cette thématique la linguistique cognitive prend une place importante et étudie le langage naturel et la signification des composants dans les textes, tels que les mots, les expressions et les phrases (Djioua 2000). La sémantique cognitive étudie aussi les relations entre différentes expressions linguistiques tels que l'homonymie, la synonymie, l'antonymie, la polysémie, l'hyperonymie, l'hyponymie, la métonymie, etc.

La sémantique computationnelle vise à maîtriser les outils de la sémantique formelle (au sens des formalismes logiques et des théories des modèles), ainsi que les langages et problèmes liés à l'implémentation des modèles sémantiques (Amsili 2007). Cette sémantique se rapporte à l'interprétation d'une grammaire formelle et non ambiguë et à l'étude de propositions d'une théorie déductive. Des déductions sont formalisées avec des règles suffisamment précises pour enchaîner un raisonnement capable de décider quelles sont les conditions de vérité d'une proposition.

Plusieurs disciplines de l'intelligence artificielle font le pont entre la sémantique cognitive et la sémantique computationnelle, notamment la linguistique computationnelle et la représentation des connaissances. En particulier, la linguistique computationnelle étudie les méthodes et outils informatiques visant à analyser la sémantique du langage naturel et retrouve son champ d'application dans le traitement automatique des langues naturelles (TALN). La représentation des connaissances est un processus de formalisation qui permet d'expliciter des processus cognitifs en ingénierie des connaissances (IC). Nous mettons ces notions en relation dans la Figure 1.

(Sheth, Ramakrishnan et Thomas 2005) catégorisent trois types de sémantique : implicite, formelle et "puissante" (de l'anglais *powerful semantics*).

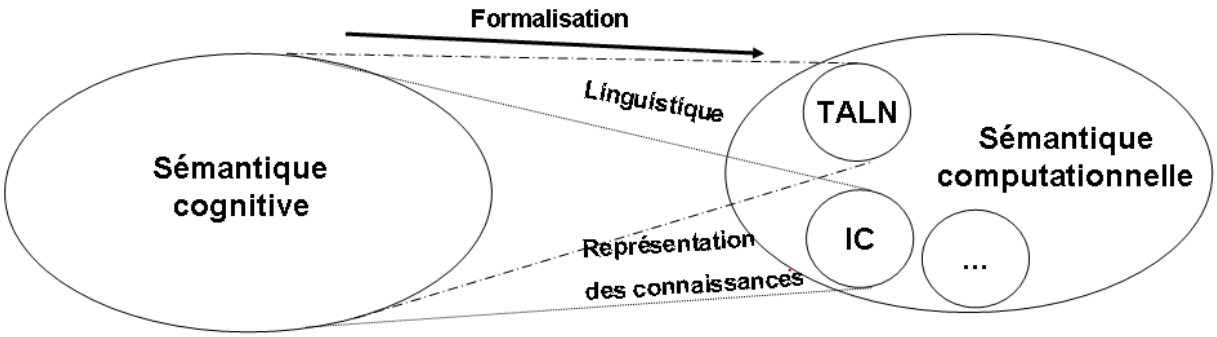

Figure 1 - Les disciplines telles que le TALN et l'IC utilisent la sémantique computationnelle en faisant le pont avec la sémantique cognitive par un processus de formalisation.

La sémantique **implicite** se réfère à celle qui est cachée dans les données et qui n'est pas représentée ouvertement, *i.e.* n'est pas formulée clairement, en utilisant une quelconque syntaxe exploitable par la machine. Parmi les exemples : la classification automatique de documents permettant de déterminer sur quoi les documents portent par rapport à une taxonomie de données ; les techniques d'apprentissage automatique, ainsi que les techniques de traitement automatique de la langue et celles de fouille de données exploitent habituellement ce type de sémantique.

En fait, les informations doivent être exprimées de manière à être traitables par les logiciels. Du point de vue cognitif, le langage naturel peut être ambigu. Les cas d'ambiguïté, par exemple la métaphore, sont difficiles à traiter par la sémantique computationnelle.

La sémantique **formelle** fixe la signification des symboles élémentaires et une méthode de calcul pour la composition des significations (Amsili 2007). En d'autres termes, une sémantique formelle signifie une sémantique traitable par la machine, une sémantique dont le sens des expressions est non ambiguë et exprimé en utilisant la syntaxe d'un langage formel. Un sous-ensemble très limité du langage naturel est donc rendu disponible pour le traitement par logiciel. Des exemples de cette sémantique sont :

- La sémantique de la relation de subsomption dans les logiques de description. Elle reflète le penchant humain à catégoriser au moyen de spécialisations et de généralisations.
- La sémantique de la relation de *partonomie* pour représenter les objets qui font partie d'autres, *i.e.* une collection d'objets éventuellement constituée de types d'objets différents.

La sémantique formelle peut s'exprimer à l'aide de différentes syntaxes. Celles-ci varient selon le langage de représentation de l'information qui les utilise. En effet, les langages de

représentation formels empruntent les techniques de la logique symbolique formelle pour représenter les relations d'implication sémantique[4] A ⊨ B entre deux propositions et les règles de raisonnement. Donner une sémantique à un langage formel consiste à définir une fonction qui soit capable d'associer à toute formule bien formée un sens. Le sens d'une formule sera simplement une valeur de vérité (vrai ou faux). Les sujets des langages de représentation de l'information et du raisonnement sont abordés plus en détail aux sections 2.1 et 2.2.

La sémantique **puissante** : une partie des travaux sur la sémantique formelle (Baader et. al. 2003) a engendré une limitation de l'expressivité des langages de représentation d'informations, de manière à permettre des caractéristiques de calculabilité acceptables, telles que la décidabilité et la limitation du temps de calcul. Puisque la plupart des formalismes de représentation d'informations ainsi que le modèle relationnel de données sont fondés sur la théorie des ensembles, la capacité à représenter des informations incertaines, imprécises, vraies en partie ou de manière approximative fait défaut dans les formalismes standards. Avec ces formalismes de représentation fondés sur les logiques formelles dites classiques, il est possible d'attribuer aux assertions une valeur de vérité vraie ou fausse, et en aucun cas les deux ou une valeur entre les deux.

D'autres logiques, dites non classiques, rejettent certains principes des logiques classiques. Par exemple le principe du tiers exclu, qui établit que toute proposition ne peut être que vraie ou fausse et qui a pour conséquence le principe de non contradiction, est refusé par la logique floue. Néanmoins celle-ci définit comme valeurs acceptables l'ensemble des réels compris entre zéro et un. Effectivement, certaines connaissances du domaine détenues par les experts humains sont intrinsèquement complexes et peuvent nécessiter des représentations plus expressives et des techniques de calcul adaptées. Plusieurs chercheurs (Cardoso et Sheth 2006, Laskey, Laskey et Costa 2006) jugent décisif pour la réussite du web sémantique de pouvoir représenter et utiliser des types d'information dits plus puissants (*soft computing*) (Cardoso et Sheth 2006, Sheth, Ramakrishnan et Thomas 2005). Par exemple, la logique floue tient compte de la représentation de degrés d'appartenance et de degrés de certitude qui sont plus adaptés à la représentation

[4] Nous utilisons l'expression 'implication sémantique' en référence au terme anglais *entailment*, pour signifier dans la théorie des modèles de Tarski (Feferman 2006) que l'ensemble de phrases A implique sémantiquement l'ensemble de phrases B si et seulement si – *i.e.* de manière nécessaire et suffisante – dans tous les modèles où A est vrai, B est aussi vrai.

intrinsèquement complexe de la connaissance des experts humains. La logique floue est approfondie dans la section 2.4.

Les aspects syntaxique et sémantique présentés dans cette section contribuent à des degrés divers à l'organisation des connaissances. Différentes manières d'agencer les connaissances génèrent différents types de ressources sémantiques qui sont analysés dans la section suivante.

1.2 Types de ressources sémantiques

De nombreux systèmes classiques de classification sont connus dans les domaines des sciences de la nature et des sciences bibliothécaires. Citons par exemple le système de classification des espèces issu de la classification du vivant établie par Linné[5] pour le premier domaine et la classification décimale de Dewey[6] pour le deuxième. D'autres domaines, tel que celui de la construction, ont développé plusieurs systèmes de classification. Dans cette sous-section nous présentons quelques définitions de ces ressources sémantiques.

1.2.1 Dictionnaires, taxonomies, thésaurus et ontologies

Dictionnaire. Un dictionnaire est un recueil de mots rangés par ordre alphabétique et suivis de leur définition ou de leur traduction dans une autre langue (Willerval, Boisseau et Serres-Cousiné 1989). Un **lexique** peut être considéré comme un type spécifique de dictionnaire, *i.e.* un dictionnaire spécialisé regroupant les termes utilisés par une science ou une technique. Par exemple, le Dicobat[7] est un lexique spécialisé dans le domaine du bâtiment.

Taxonomie. Une taxonomie ou taxinomie vise à structurer les informations de manière classificatoire pour ensuite les représenter sous forme d'un arbre qui privilégie la relation hiérarchique. Cette ressource classifie les informations d'un domaine selon une relation généralisation → spécialisation, de manière à ce que les concepts plus génériques contiennent les concepts plus spécialisés et que ces derniers soient les plus détaillés. À l'origine, ce type de ressource a été utilisé dans les domaines des sciences naturelles pour classifier les organismes en groupes suivant leurs similarités de structure ou de nature.

---

[5] http://www.fundp.ac.be/sciences/biologie/bio2001/bioscope/1735_linne/linne.html

[6] http://www.oclc.org/ca/fr/dewey/

[7] Voir http://boutique.cstb.fr/dyn/cstb/fiche_produit.asp?pf_id=314&dept_id=90

Thésaurus. Selon la norme française NF Z47-100 (NF Z47-100 : 1981) un thésaurus est une *liste d'autorité* organisée de descripteurs[8] et de non-descripteurs obéissant à des règles terminologiques propres et reliés entre eux par des relations sémantiques : hiérarchiques, associatives ou d'équivalence. D'après la norme, cette liste sert à traduire en un langage artificiel dépourvu d'ambiguïté des notions exprimées en langage naturel. Cette définition est en concordance technique avec celle de la norme ISO 2788 (ISO 2788:1986) qui définit un thésaurus comme étant le vocabulaire contrôlé d'un langage d'indexation, organisé formellement de façon à expliciter les relations *a priori* entre les notions.

Ainsi, un thésaurus est constitué par un ensemble de termes (descripteurs et non-descripteurs) et de relations qui précisent leur environnement sémantique. Un thésaurus ne fournit qu'accessoirement des définitions, les relations sont mises en avant. Le réseau des relations d'un descripteur avec les autres termes (descripteurs et non-descripteurs) fournit ainsi une sorte de définition et contribue à la diminution des risques d'ambiguïté en situant le descripteur dans un contexte qui en précise le sens.

Ontologie. Ce type de ressource vise une représentation plus complexe et univoque que celles présentées précédemment. Elle s'applique à la représentation d'une sémantique computationnelle non ambiguë. Le terme ontologie a été emprunté à la philosophie, où il signifie l'étude de l'existant et des catégories principales de ce dernier (Hofweber 2005). En informatique et en particulier en intelligence artificielle, le sens est différent. J. Sowa (Sowa 2000) précise, dans le domaine des bases de données et des bases de connaissances, qu'une ontologie établit les catégories de choses qui existent ou peuvent exister dans un domaine d'application $\mathcal{D}$ du point de vue d'une personne qui utilise un langage $\mathcal{L}$ pour représenter une partie de $\mathcal{D}$. La définition d'ontologie de Gruber (Gruber 1995) est évoquée fréquemment : *une ontologie est une spécification explicite d'une conceptualisation*. Une spécification exprime de manière précise les caractéristiques essentielles des artefacts visés. Une conceptualisation est un système de catégories qui établit un point de vue sur un univers du discours, ou plus spécifiquement qui est formé par un ensemble de relations pertinentes pour un domaine d'application. En effet, une conceptualisation a été définie (Genesereth et Nilsson 1987) par une structure <D, R>, où D est un domaine et R est un ensemble de relations pertinentes dans D. Nous pouvons alors dire qu'une conceptualisation est un ensemble de relations conceptuelles définies dans un espace du domaine.

[8] Un *descripteur* est un mot ou un groupe de mots retenus dans un thésaurus et choisis parmi un ensemble de termes équivalents pour représenter sans ambiguïté une notion contenue dans un document ou dans une demande de recherche documentaire.

N. Guarino (Guarino 1998) raffine la définition d'ontologie en écrivant qu'elle est une théorie logique qui tient compte du sens délibéré d'un vocabulaire, *i.e.* un engagement ontologique (ou interprétation intentionnelle) envers une conceptualisation spécifique du domaine d'application (Figure 3). En d'autres termes, une ontologie est un objet d'ingénierie constitué d'une part d'un vocabulaire partagé pour décrire une réalité donnée et d'autre part d'un ensemble d'axiomes explicites concernant le sens visé pour les mots du vocabulaire.

Un **axiome** est un énoncé dont la vérité est reconnue sans démonstration. Plus particulièrement en logique, c'est une proposition posée à la base d'un système hypothético-déductif. Les axiomes contraignent le sens et expriment d'autres relations sémantiques entre les concepts en plus de la relation de subsumption. En général, le vocabulaire est composé de concepts et de relations représentés sous la forme de noms de prédicats unaires ou binaires respectivement (Figure 2).

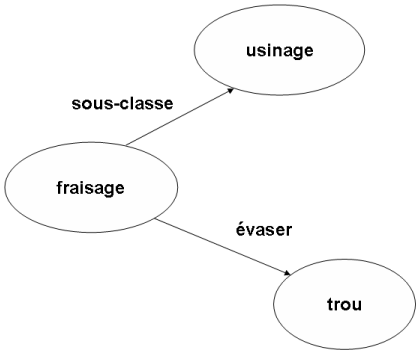

Figure 2 – Exemple d'un graphe qui représente trois prédicats unaires, les concepts usinage, fraisage et trou et deux prédicats binaires, les relations sous-classe et évaser.

Selon Guarino, l'ensemble $I_k(L)$ de tous les modèles possibles représentables avec le langage L et suivant l'engagement ontologique K (Figure 3) est appelé l'ensemble des modèles visés (*intended models*, selon la version originale).

Alors qu'une conceptualisation est indépendante du langage, une ontologie est contrainte par le langage choisi pour la représenter. C'est-à-dire qu'une ontologie est assujettie au pouvoir de représentation du langage de représentation adopté.

D'une manière générale, la tâche principale d'une ontologie est de rendre explicite une structure conceptuelle sous-jacente à un certain domaine d'application et de formaliser une terminologie utilisée pour le décrire. Pour cela, il est usuel de distinguer les concepts, les relations les unissant, les individus qui peuplent le domaine et d'exprimer des axiomes qui relèvent les propriétés principales des prédicats.

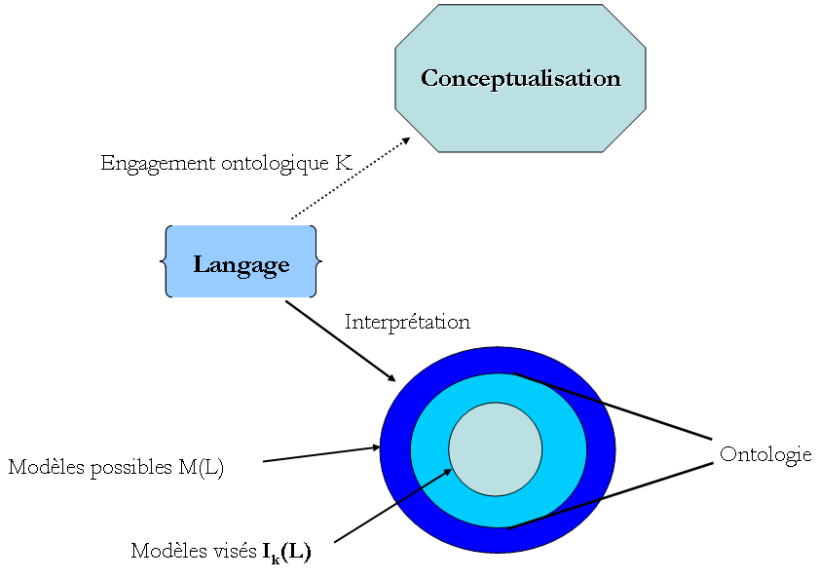

Figure 3 – Relations entre langage, ontologie, conceptualisation et modèles visés. Adaptée à partir de (Guarino 1998) et de (Ferrario 2006).

R. Ferrario (Ferrario 2006) met l'accent sur l'importance de se fonder sur une méthode d'axiomatisation logique et rigoureuse pour représenter une ontologie afin d'éliminer les ambiguïtés terminologiques et conceptuelles de manière à ce que l'ontologie soit un instrument précis et avéré dans un processus d'échange d'information.

Parmi les ontologies formelles nous pouvons distinguer les ontologies dites *spécifiques à un domaine* et les ontologies dites *de haut niveau*. Les premières sont des objets d'ingénierie spécifiquement conçus pour des applications de domaine et utilisées par des communautés bien établies. Leur sens est plus ou moins partagé au sein de la communauté concernée. Les secondes[9] décrivent les concepts de manière générale, c'est-à-dire indépendamment d'un domaine ou problème particulier, comme les concepts *espace*, *temps*, *matière*, *événement*, *action*. DOLCE[10] et OpenCyc[11] sont des exemples d'ontologies de haut niveau. Cependant, ce dernier type d'ontologies, malgré son ambition universelle, ne peut pas non plus faire abstraction d'un point de vue et peut, de ce fait, être objet de critiques (Thomasson 2004).

En définitif, les ontologies doivent contenir les caractéristiques essentielles, *i.e.* les invariants, du domaine d'application à représenter. Elles fournissent une représentation partielle des modèles à décrire.

---

[9] (Guarino 1998) précise d'autres types d'ontologies, tels qu'ontologies de domaine, de tâches, d'application, etc.

[10] DOLCE signifie *Descriptive Ontology for Linguistic and Cognitive Engineering*. Plus de détails sont disponibles à http://www.loa-cnr.it/DOLCE.html

[11] Voir http://www.opencyc.org/

1.2.2 Analyse des liens entre les types de ressources sémantiques

La section précédente présente les différentes types de ressources sémantiques, il nous est apparu utile de les classer les unes par rapport au autres. Pour cela nous suggérons de catégoriser les définitions des ressources sémantiques de manière hiérarchiquement imbriquée. Cette imbrication (Figure 4) exprime l'ordre croissant d'éléments de représentation pris en compte qui va de pair avec la complexité de la représentation. Ainsi, dans cette classification, un dictionnaire se trouve au début et est imbriqué dans des formes d'organisation des informations de plus en plus complexes mais aussi plus complètes. Les taxonomies et les thésaurus lient leurs concepts propres via les relations spécifiques à chacune, mais n'expriment pas de contraintes pour imposer un sens unique et formel pour un domaine d'application donné. L'ontologie englobe cette organisation imbriquée, permettant des relations de types divers et joignant des contraintes qui explicitent la sémantique computationnelle. Nous pouvons dire qu'une ontologie est une ressource plus formelle que celles qu'elle contient. Pour cela elle peut être utilisée pour réaliser des traitements plus complexes tels que l'inférence. Du point de vue structurel, l'emboîtement des ressources sémantiques représenté dans la Figure 4 évolue d'un arbre taxonomique vers un graphe ontologique.

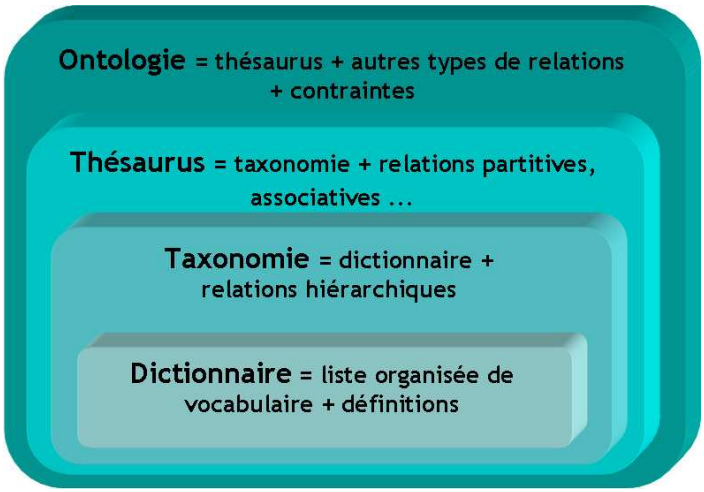

Figure 4 - Emboîtement des définitions génériques de ressources sémantiques.

1.3 Étude sur les relations

Dans l'analyse des types des ressources sémantiques, nous avons jugé important d'analyser plus en détail les caractéristiques des types de relations des ressources sémantiques. De manière informelle, une *relation* dans un ensemble est une proposition qui lie un certain nombre d'éléments. Une relation indique l'existence d'un rapport entre, au moins, deux ensembles. Les relations peuvent se trouver partout et les définitions peuvent varier selon le domaine de

connaissances. La relation est définie en première approche comme un ensemble de couples, c'est-à-dire que si deux éléments sont reliés entre eux, alors le couple est un élément de l'ensemble relation. En algèbre générale, la notion de relation, ou de *correspondance*, est une abstraction de notions telles que l'égalité, l'ordre alphabétique ou la comparaison. Dans un ensemble, une *relation d'ordre* est une relation binaire, réflexive, antisymétrique et transitive, qui permet de comparer ses éléments entre eux de manière cohérente. Dans un ensemble, une *relation d'équivalence* est toute relation binaire réflexive, symétrique et transitive. Dans la théorie des ensembles, l'appartenance d'un élément x à un ensemble A, noté $x \in A$, indique une *relation d'appartenance*.

Dans le domaine des bases de données, et en particulier en modélisation relationnelle, une relation a un sens plus général. Elle est représentée par un ensemble de *tuples* et est souvent assimilée à une table. L'algèbre relationnelle définit un ensemble d'opérateurs ensemblistes qui permettent de générer des relations (ou tables) à partir d'opérations élémentaires sur d'autres tables, par exemple l'union, l'intersection, la différence, le produit cartésien.

La norme française relative à l'établissement des thésaurus monolingues (NF Z47-100 1981) détaille les types de relations qu'elle préconise : équivalence, hiérarchique et associative. Dans le domaine linguistique d'autres types de relations sont définis, tels que l'homonymie, la synonymie, la polysémie, hyponymie – hyperonymie, la méronymie – holonymie et la métonymie.

La relation d'**hyponymie** est l'expression linguistique de la relation logique d'inclusion d'une classe dans une autre et établit un arbre de concepts allant du pus général aux plus spécifiques. À l'opposé, la relation d**'hyperonymie** définit un arbre de concepts de plus en plus généraux.

La **méronymie**[12] est une relation partitive hiérarchique existant entre deux concepts ou deux signes linguistiques, dans laquelle le premier est une partie d'un tout que constitue le second. Elle est aussi nommée relation partie-tout, partitive ou méronymique. Différents types de relations méronymiques existent, comme celles unissant un composant et l'objet dont il est un élément, un objet et la collection dont il fait partie, une portion et la masse dont elle provient, un lieu et la région où il se trouve, etc. Par exemple, *doigt* et *main* entretiennent une relation de méronymie, de même que *joueur* et *équipe*, *Bretagne* et *France*. À l'inverse, la relation d'**holonymie**

[12] Du grand dictionnaire terminologique québécois, disponible à http://granddictionnaire.com

caractérise le lien entre deux concepts dont le premier est le tout qui contient le second. Par exemple, le *bâtiment* est un holonyme de *toit*.

En logique des prédicats, une relation est une fonction qui attribue à ses arguments une valeur vraie ou fausse. Sowa (Sowa 2000) présente l'exemple du prédicat *strictement plus petit*, symbolisé par <.

Une relation relie des concepts. D'autres termes sont utilisés pour nommer une relation, tels que *propriété, attribut* et *rôle* (Baader *et al.* 2003). Certains langages de représentation différencient entre une relation et un attribut, comme par exemple *Unified Modeling Language* (UML). Alors que d'autres, comme par exemple *Web Ontology Language* (OWL) (McGuinness et Van Harmelen 2004), ne le font pas explicitement. Dans les logiques de description (Baader *et al.* 2003), la famille de langages sur lesquels se sont inspirés les auteurs d'OWL, un concept possède un définition formelle qui se construit à l'aide d'un ensemble de constructeurs introduisant les rôles associés au concept et les restrictions attachées à ces rôles. Les constructeurs admis dans la description sont définis par le langage de description adopté. Les restrictions portent généralement sur :

- le *domaine* du rôle, qui définit le concept source du rôle.
- le *range*, aussi nommé co-domaine, ou image du rôle, qui est le concept avec lequel le rôle établit une relation, *i.e.* sa portée ou cible du rôle. Le domaine et le range contraignent le type d'arguments de la relation, *i.e.* définissent la signature de la relation.
- le *type* de rôle, ce qui détermine ces propriétés. Par exemple un rôle peut être transitif, inverse ou fonctionnel (Baader *et al.* 2003).
- la *cardinalité* du rôle, qui fixe le nombre minimal et maximal de valeurs élémentaires que peut prendre le rôle. Les valeurs élémentaires sont en général des instances de concepts ou bien des valeurs qui relèvent des types de base comme entier, réel, et chaîne de caractères.

Dans cette sous-section nous avons brièvement analysé les types et caractéristiques des relations utilisées, d'une manière générale, dans la construction d'une ressource sémantique et les restrictions imposées aux relations par certains langages de représentation, comme OWL. De plus, nous analysons les types d'hétérogénéités qu'il est possible de rencontrer entre diverses ressources sémantiques.

1.4 Types d'hétérogénéités

Les hétérogénéités entre les ressources sémantiques sont des désaccords qui entravent l'utilisation conjointe de ressources qui ont été développées séparément sans coordination (Naiman et Ouksel 1995, Costa et Cunha 2006). Certains travaux analysent les différents types d'hétérogénéité. Nous adoptons la classification de (Visser *et al.* 1998) qui distingue quatre types d'hétérogénéités entre des systèmes :

- hétérogénéités entre paradigmes, apparaissant quand deux systèmes expriment leurs connaissances utilisant différents paradigmes de modélisation. Les auteurs citent l'exemple suivant : entre les bases de données relationnelles et les bases de données orientées objets. Nous incluons dans cette catégorie ce que (Klein 2001) nomme par désaccords de représentation logique et par désaccords de sémantique des primitives. Un désaccord de représentation logique concerne les différences dans la représentation de notions logiques, ce qui est distinct de la représentation de concepts. Par exemple, un langage de représentation déclare que les classes A et B sont disjointes (`disjoint A B`), alors qu'un autre langage utilise la négation dans la déclaration de sous-classes (`A subclass-of (NOT B)`, `B subclass-of (NOT A)`). Ces déclarations sont équivalentes en termes logiques, la différence est dans le choix des constructeurs du langage pour exprimer les axiomes. Ce type de désaccord est relativement facile à résoudre en définissant des règles de traduction entre les différentes représentations logiques. Un désaccord de sémantique des primitives, se rapporte à la sémantique des constructeurs d'un langage. Parfois un même nom de constructeur est utilisé avec des significations différentes dans deux langages. Klein signale l'exemple des interprétations diverses que deux langages peuvent faire de (`A equalTo B`). On pourrait nommer ce type de désaccord, ce que Klein ne spécifie pas, comme une "polysémie des constructeurs".

  Nous nommons *hétérogénéité de noms de constructeurs* un désaccord dont les constructeurs des deux langages ont des noms différents mais dont la sémantique est la même. Ici l'interprétation sémantique d'un constructeur d'un langage peut alors correspondre à celle d'un ensemble organisé de constructeurs dans un autre langage.
- hétérogénéités entre langages, apparaissant quand deux systèmes expriment leurs connaissances en utilisant différents langages de représentation. Dans cette thèse (*cf.* chapitre 3) nous rencontrons ce type d'hétérogénéité entre OWL et le langage EXPRESS (Schenck et Wilson 1994). (Klein 2001) répertorie deux catégories de désaccords qui s'insèrent ici : les désaccord de syntaxe et ceux d'expressivité des langages. Un désaccord

des syntaxes vient du fait que différents langages de représentation utilisent habituellement des syntaxes différentes. Ceci est probablement le type le plus simple de désaccord. En général, quand le désaccord est seulement du type syntaxique, un mécanisme de réécriture suffit pour le résoudre. Néanmoins ce désaccord ne vient le plus souvent pas seul, mais plutôt joint à d'autres désaccords au niveau du langage. Un désaccord d'expressivité des langages, est considéré par Klein comme celui qui a le plus d'impact sur l'hétérogénéité. Différentes expressivités des langages laisse supposer que certains langages peuvent exprimer certaines informations qui ne sont pas exprimables dans d'autres langages. Klein donne l'exemple des constructeurs pour exprimer la négation, définir des listes, des ensembles ou des valeurs par défaut, de certains langages que d'autres n'ont pas.

- hétérogénéités ontologiques, apparaissant quand deux systèmes font des suppositions ontologiques différentes sur leurs connaissances du domaine d'application. Par exemple, une ressource assume que le concept *maison* est défini par les éléments qui le composent tels que les *portes* et *fenêtres* et une autre définit le concept de même nom par la composition de pièces tels que *chambres*, *salles de bain*, *cuisines*.
- hétérogénéités entre contenus, apparaissant quand deux systèmes expriment simplement des connaissances différentes.

(Visser *et al.* 1998) signale que les deux derniers types, *i.e.* hétérogénéités ontologiques et de contenus, sont des instances d'hétérogénéités sémantiques alors que les deux premiers types sont des instances d'hétérogénéités non sémantiques. Il nous semble que cette dichotomie est reprise par (Klein 2001), sans qu'il l'ait exprimé explicitement. Cependant, il utilise d'autres termes pour identifier les hétérogénéités. En fait, Klein reconnaît des hétérogénéités de niveau du langage ou de méta-modèle, *i.e.* le niveau des primitives du langage qui sont utilisés pour spécifier une ontologie. Dans ce niveau, les désaccords concernent les mécanismes utilisées pour définir les classes et relations pour représenter les connaissances du domaine d'application. Ceci est proche de ce que (Visser *et al.* 1998) identifie par des hétérogénéités non sémantiques. Le deuxième niveau d'hétérogénéités est celui de l'ontologie ou du modèle. Dans ce niveau, un désaccord concerne les différences dans la manière de modéliser le domaine d'application, ce qui se rapproche de la catégorie d'hétérogénéités sémantiques mentionné ci-dessus.

Du point de la représentation des connaissances, nous nous intéressons aux hétérogénéités ontologiques. (Visser *et al.* 1998) proposent aussi une classification de désaccords qui peuvent exister entre différentes ontologies. Dans cette classification, les auteurs suivent la définition

d'ontologie de Gruber pour diviser deux grands groupes de désaccords : les désaccords de *conceptualisation d'un domaine* et les désaccords *d'explication d'une conceptualisation*. Les premières sont des désaccords entre deux conceptualisations, ou plus, d'un domaine d'application. Ces conceptualisations divergent dans les concepts ou dans la manière dont ces concepts sont liés. Les deuxièmes concernent les manières dont les conceptualisations sont spécifiées. Par exemple, il a un désaccord d'explication quand deux ontologies ont deux définitions différentes d'un même concept.

Dans la catégorie de désaccords de *conceptualisation*, (Visser *et al.* 1998) classifie les désaccords de *classe* et les désaccords de *relation*. Les premiers sont ensuite spécialisés en désaccords de *catégorisation* et désaccords *d'agrégation*. Les deuxièmes sont séparés en désaccords de *structure*, désaccords *d'assignation d'attribut* et désaccords de *type d'attribut*. La Figure 5 illustre cette catégorisation des désaccords. Les désaccords de *classe* concernent naturellement les classes distinguées entres les différentes conceptualisations et leurs sous-classes.

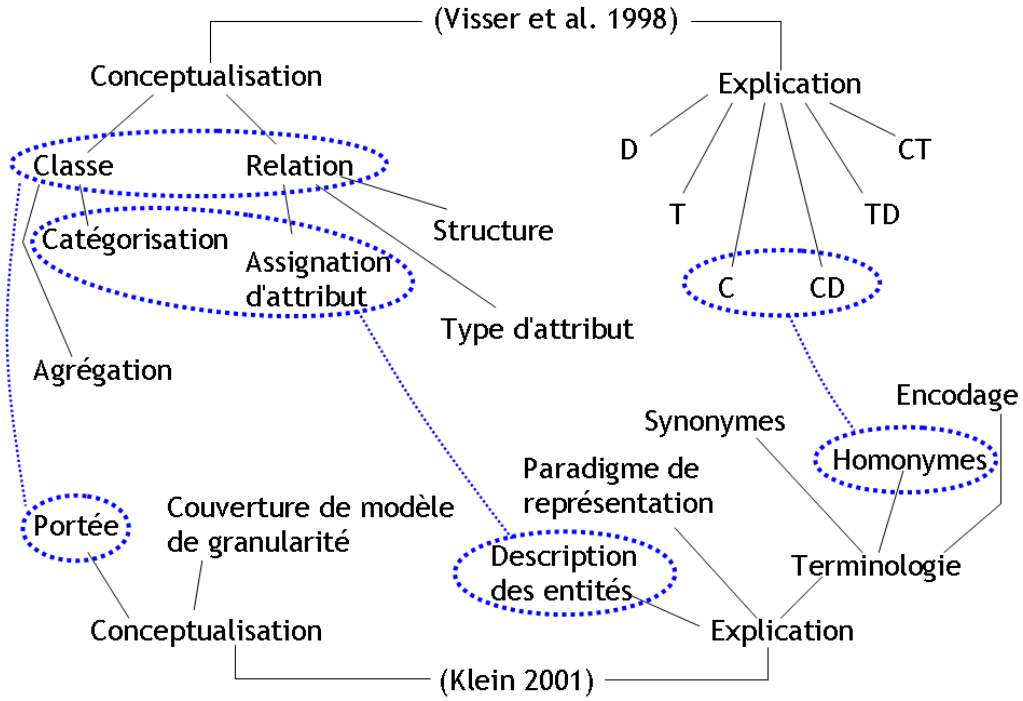

Figure 5 – Correspondances entre les classifications des hétérogénéités ontologiques de (Visser *et al.* 1998) et (Klein 2001).

Un désaccord de *catégorisation* se présente quand deux conceptualisations distinguent la même classe mais spécialisent cette classe en des sous-classes différentes. Un désaccord *d'agrégation* se produit lorsque deux conceptualisations définissent des classes à différents niveaux d'abstraction.

Les désaccords de *relation* sont des désaccords de conceptualisation concernant les relations distinguées dans les deux conceptualisations. (Visser *et al.* 1998) fournissent comme exemples les relations hiérarchiques entre les classes et l'assignation d'attributs aux classes.

Un désaccord de *structure* a lieu quand deux conceptualisations discernent le même ensemble de classes mais différent quant à la manière dont ces classes sont structurées à travers les relations (*cf.* section 1.3).

Un désaccord *d'assignation d'attribut* se produit quand deux conceptualisations varient quant à la façon d'assigner un attribut de classe à d'autres classes. (Visser *et al.* 1998) considèrent deux conceptualisations qui distinguent toutes les deux les classes "véhicule" et "voiture". Les conceptualisations ont un désaccord d'assignation d'attribut quand l'une assigne l'attribut "couleur" à la classe "véhicule" et l'autre assigne "couleur" à la classe "voiture".

Un désaccord de *type d'attribut* se présente lorsque deux conceptualisations reconnaissent le même attribut (ou classe) mais différent dans leurs instanciations, *i.e.* en termes de type. Par exemple, pour deux conceptualisations qui définissent la même classe "longueur", l'une instancie en kilomètres et l'autre en "miles" anglais.

Pour sa part (Klein 2001) reprend en partie cette distinction entre désaccords de conceptualisation et désaccords d'explication de (Visser *et al.* 1998). En fait, Klein analyse les désaccords différemment d'une part et dégage des sous-catégories différentes de désaccords d'autre part (Figure 5). Pour Klein les désaccords de classe et de relation catégorisés par (Visser *et al.* 1998) sont des désaccords de *portée*, *i.e.* deux entités de conceptualisations différentes représentent le même concept mais n'ont pas exactement les mêmes instances, bien qu'elles s'intersectent. Dans la catégorie de désaccords de conceptualisation, Klein définit aussi la sous-catégorie de désaccords de *couverture de modèle de granularité* : c'est un désaccord sur la partie du domaine d'application qui est couverte par les ontologies ou le niveau de détail de modélisation du domaine. Par exemple, la spécification de la distance d'un bâtiment à une voie routière : une ressource exprime simplement que "le bâtiment est proche de la voie routière" alors que l'autre spécifique "le bâtiment est à 15 mètres de la voie routière".

Les désaccords *d'explication* d'une conceptualisation concernent la manière dont les conceptualisations sont spécifiées. Visser *et al.* définissent des désaccords entre les définitions des classes, les définitions des relations et les définitions d'instances. Chaque définition est un triplet $Def = <T, D, C>$, où T est un terme ou une formule atomique qui identifie l'entité, D est la définition formelle et C est le concept défini en langage naturel (CLN). Les trois composants de la définition permettent en principe huit relations différentes entre deux définitions, donc huit types de désaccords différents. Néanmoins, les auteurs font l'hypothèse que si ces trois

composants sont tous différents alors il n'a pas de désaccord, *i.e.* il n'a pas de correspondance possible. Similairement, quand les trois composants sont tous égaux entre deux conceptualisations alors il n'a pas de désaccord puisqu'il a une correspondance complète. Visser *et al.* considèrent qu'il existe un désaccord d'explication quand un ou deux des composants des définitions, *T, D* ou *C,* sont égaux. Ceci donne donc six types de désaccords différents :

- *un désaccord de C* est un désaccord de concept CLN (concept en langage naturel) : les ressources utilisent les mêmes termes T et la même définition formelle D, mais les définitions en langage naturel divergent. Ce désaccord implique aussi que T est homonyme.
- *un désaccord de T* est un désaccord de terme T : les ressources définissent le même CLN et la même définition formelle D, mais s'y réfèrent avec des termes différents (*i.e.* hétérogénéité de noms). Ce désaccord implique que les entités sont des synonymes.
- *un désaccord de D* est un désaccord de définition formelle D : les ressources définissent le même CLN et s'y réfèrent avec le même terme T, mais définissent différentes définitions formelles.
- *un désaccord de CT* est un désaccord de concept CLN et de terme T : les ressources utilisent la même définition formelle D pour une entité mais le C et T sont différentes.
- *un désaccord de CD* est un désaccord de concept CLN et de définition formelle D : les ressources utilisent le même terme T mais différent dans les définitions explicitées en langage naturel et en langage formelle. Ce désaccord implique que T est homonyme.
- *un désaccord de TD* est désaccord de terme T et de définition formelle D : les ressources définissent le même concept CLN, mais divergent dans le terme qui les identifie et dans les définitions formelles. Ce désaccord implique que les deux T sont des synonymes.

En fait, le choix de la définition à trois composants *T, D* et *C* fait par (Visser *et al.* 1998) qu'ils utilisent pour définir les désaccords d'explication, est déjà en soit un choix de paradigme de modélisation d'ontologies. Donc, ces auteurs supposent que les ontologies ont adopté le même paradigme de modélisation lors de leurs conceptions ou du moins très comparables.

Klein suppose que les désaccords d'explication résultent du style de modélisation. Dans cette catégorie de désaccords, il sous-classe les désaccords de *paradigme de représentation* et de *description des entités*. En fait, ce que Klein nomme par désaccords de paradigme de représentation semble correspondre à la catégorie des hétérogénéités entre paradigmes (voir début de cette section) définit par (Visser *et al.* 1998), *i.e.* séparée des hétérogénéités ontologiques. Le choix de Klein est compréhensible si on tient compte que la manière dont une conceptualisation est spécifiée ou expliquée, est conditionnée par le paradigme de représentation sélectionné.

Klein explique les désaccords de description des entités par rapport aux conventions de modélisation. Effectivement ceci semble correspondre à ce que (Visser *et al.* 1998) range comme des désaccords de catégorisation et d'assignation d'attributs. Klein dégage deux autres catégories de désaccords d'explication : *terminologiques* et d'*encodage*. Dans les premiers il inclut les désaccords de *synonymie* et d'*homonymie* entre les termes identifiant les concepts. Un désaccord d'homonymie correspond à ce que (Visser *et al.* 1998) nomme par désaccord de concept *C* ou *CD*. Finalement un désaccord d'encodage est un type de désaccord évident et apparaît quand des valeurs sont encodées en des formats différents.

(Visser *et al.* 1998) argument que la syntaxe est manifestement insuffisante pour détecter tous les désaccords et en particulier certains désaccords peuvent seulement être détectés en fonction du contexte dans lequel les entités sont utilisées. Ces auteurs classent aussi leurs catégories de désaccords en fonction de la difficulté à les résoudre :

- *Gérable* : désaccords de T, CT, CD et de type d'attribut ;
- *Difficile* : désaccords de catégorisation et de structure ;
- *Dépendant du contexte* : désaccords de TD, D, d'assignation d'attribut et d'agrégation.

Une autre classification est celle de (Missikoff 2002), qui classifie les désaccords menant à des déficits d'interopérabilité, provoqués par des différences dans les schémas conceptuels de deux applications qui essayent de coopérer, entre ceux sans perte et ces avec perte :

- les désaccords sans perte peuvent être résolus sans perte d'information. Les exemples sont les désaccords de terme T ou d'hétérogénéités de noms ; des désaccords de classe ; de relation, d'encodage.
- les désaccords avec perte incluent ceux pour lesquels n'importe quelle transformation (dans les deux directions ou dans une seule) provoquera une perte d'information. Les cas typiques concernent ceux pour lesquels l'information est représentée à des différents niveaux de granularité et de précision, comme les désaccords de couverture de modèle de granularité. Un autre exemple concerne les transformations entre langages qui ont des expressivités différentes. C'est le cas que nous éprouvons lors que nous voulons transformer une ressource sémantique exprimée en utilisant le langage EXPRESS vers OWL (*cf.* chapitre 3) en vue d'aboutir à une homogénéisation syntaxique.

Dans la section suivante nous traitons de l'interopérabilité des ressources sémantiques qui est fortement lié à ce que nous venons d'analyser.

1.5 Interopérabilité des ressources sémantiques

Selon l'*IEEE Standard Computer Dictionary* (IEEE SCD), l'interopérabilité "est l'aptitude que deux ou plusieurs systèmes ou composants ont à utiliser l'information qui a été échangé".

Le document *ISO/IEC 2382-01, Information Technology Vocabulary, Fundamental Terms*, définit interopérabilité par "la capacité à communiquer, exécuter des programmes, ou transférer des données entre plusieurs unités fonctionnelles d'une manière qui impose à l'utilisateur d'avoir peu ou pas de connaissances sur les caractéristiques singulières de ces unités". La notion d'interopérabilité est définie à différents niveaux. En fait, les deux définitions ci-dessous concernent les niveaux techniques de l'utilisation, communication et transfert de données et d'informations. Un déficit d'interopérabilité est détecté lorsqu'on souhaite échanger des données, informations ou connaissances et vient des hétérogénéités entre les ressources sémantiques (Guzelian et Cauvet 2005). Pour cela nous nous intéressons aux trois aspects suivants de l'interopérabilité :

- **l'interopérabilité syntaxique** est la capacité à permettre l'échange d'information entre deux ressources sémantiques au niveau syntaxique. Elle est donc liée aux langages ou formats de représentation des connaissances. Nous sommes face à des hétérogénéités dites syntaxiques quand deux ressources sémantiques sont représentées en des formats différents ou en utilisant des langages pour la représentation de l'information de nature différente. L'aspect syntaxique de l'interopérabilité est donc lié aux hétérogénéités entre langages et entre paradigmes (*cf.* section 1.4). La standardisation des langages de représentation joue un rôle important pour surmonter les hétérogénéités syntaxiques. Le problème des ressources sémantiques exprimées en des langages ou formats de représentation d'information différents est bien étudié. En général un logiciel traducteur permet d'obtenir une homogénéité syntaxique. Cependant, quand des ressources sont représentées en utilisant des langages de représentation avec différents pouvoirs d'expression ou issus de différents paradigmes, le passage d'un langage à l'autre peut être accompagné de pertes d'information.
- **l'interopérabilité structurelle** permet l'échange d'information entre ressources sémantiques au niveau structurel et est liée aux différentes manières de schématiser les ressources. L'aspect structurel de l'interopérabilité est donc lié aux désaccords de conceptualisation d'un domaine. Le problème de l'interopérabilité structurelle est bien connu depuis plusieurs années dans la communauté des systèmes de base de données (Batini, Lenzerini et Navathe 1986, Zhang 1992, Sheth et Kashyap 1992, Kashyap et Sheth 1998, Rham et Bernstein 2001, Suwanmanee *et al.* 2005) dominé par l'intégration de

schémas de base de données hétérogènes qui a généré différentes approches de médiation de données[13]. Quelques similitudes peuvent être trouvées entre les schémas de base de données et les ressources sémantiques puisque les deux fournissent un vocabulaire des termes qui décrivent le domaine d'application et contraignent la signification des termes utilisés. Par conséquent, les solutions d'alignement obtenues dans ces deux domaines de représentation peuvent être mutuellement utiles (*cf.* section 1.7.2). Le langage EXPRESS-X (Wilson 1998) est utilisé pour la traduction de données, sémantiquement similaires, entre deux modèles représentées dans le même langage EXPRESS (ISO 12303-11 1994) qui organisent leurs informations de manière structurellement différent.

Notons de plus que les problèmes d'hétérogénéités syntaxique et structurelle sont très liés entre eux. La traduction entre formats est associée à la conversion schématique quand par exemple les champs sont groupés différemment dans les deux structures. Une ressource peut avoir une structure à plat qui liste simplement les champs alors que l'autre ressource regroupe les champs apparentés.

Les approches visant à améliorer l'interopérabilité fondées sur la structure peuvent s'avérer intéressantes quand il existe une certaine homogénéité conceptuelle entre les graphes à comparer, *i.e.* quand des possibles désaccords dans la structuration des éléments des schémas ainsi que dans le vocabulaire utilisé ne rendent pas les similarités trouvées éminemment fausses. L'exemple simple de la Figure 6 montre qu'il n'est pas toujours suffisant de faire seulement une comparaison structurelle ou d'aligner des éléments atomiques de structure sans tenir compte de la sémantique qui est convoyée par chaque élément ou groupe d'éléments. Dans l'arbre de gauche, les deux concepts du premier niveau (les concepts femme et homme) après la racine correspondent dans l'arbre de droite au premier concept du premier niveau (parent). Les concepts soeur et frère, qui dans l'arbre de gauche sont des sous-concepts de concepts différents se trouvent dans l'arbre de droite, regroupés sous le concept fratrie. La problématique doit donc recouvrir aussi l'interopérabilité sémantique.

[13] Le problème de médiation de données est résumé (Solar et Doucet 2002) de la manière suivante : étant donné un ensemble de sources (niveau local) et d'applications, il s'agit de construire une infrastructure intermédiaire facilitant l'accès (expression, traitement et optimisation de requêtes) et la manipulation (mise à jour, gestion de la cohérence) des données (niveau global).

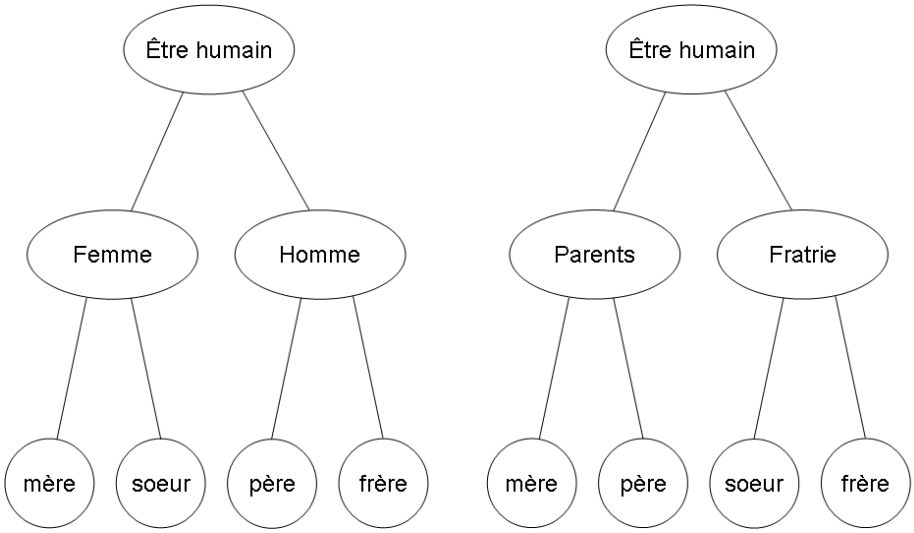

Figure 6 - Deux structurations différentes d'un même domaine de connaissances.

- **l'interopérabilité sémantique**. Plus récemment d'autres définitions tendent à mettre en valeur l'aspect sémantique du concept d'interopérabilité. L'interopérabilité sémantique est l'aptitude à échanger et traiter l'information, en s'assurant que la signification précise de l'information est comprise par n'importe quelle application qui n'avait pas été initialement développée exprès pour le faire (Hughes 2004). Ainsi, il serait possible de parvenir à l'interopérabilité sémantique si un modèle de référence pour l'échange d'information est utilisé pendant la communication d'éléments sémantiques où ceux-ci sont définis de manière précise.

  (Euzenat et Stuckenschmidt 2002) mentionnent que l'interopérabilité sémantique est le problème de comparaison et d'interprétation d'annotations (de pages web) au niveau sémantique, *i.e.* attribuer à chaque pièce de connaissances importée l'interprétation ou l'ensemble de modèles corrects. Cependant par la suite les auteurs insistent plutôt sur l'aspect syntaxique de transformation entre langages de représentation et de préservation sémantique pendant les transformations, et laissent de côté d'autres problèmes importants tels que les désaccords de représentation sémantique.

  Pour atteindre l'interopérabilité sémantique, (Heflin et Hendler 2000) considèrent que les systèmes doivent être capables d'échanger des données de manière à ce que la signification précise des données soit facilement accessible et que ces données puissent être traduites par un autre système en un format que ce système comprend. (Degoulet, Fieschi et Attali 1997) considère que l'interopérabilité sémantique "recherche une interopérabilité globale sur la signification des échanges, c'est-à-dire sur leur interprétation ou leur finalité. Cet aspect est crucial et constitue le principal facteur limitant le développement d'application intégrables dans des systèmes complexes".

Le standard STEP[14] (ISO 12303-11 1994) est composé de plusieurs parties et est un des exemples de normalisation et de convergence lente depuis plus de vingt ans vers des protocoles d'échange de données qui retrouvent encore des problèmes l'interopérabilité sémantique.

Pour conclure cette sous-section il faut ajouter que tous les aspects de l'interopérabilité sont indissociables. Nous essayons de schématiser les trois aspects fondamentaux (Figure 7) dont chaque couche supérieure s'appuie sur celle d'en dessous.

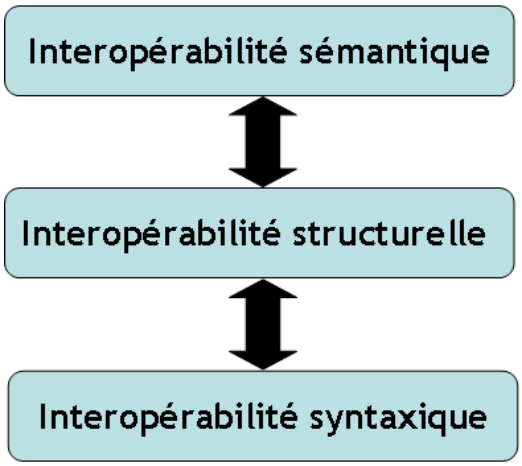

Figure 7 - Représentation des niveaux d'interopérabilité par des couches successives qui s'appuient sur les couches inférieures.

Aboutir à l'interopérabilité sémantique est une réalisation complexe. De plus en plus de ressources sémantiques existent qui sont autant de point de vue cognitifs différents. En fait, la situation idéale pour surmonter le manque d'interopérabilité semble être l'obtention d'un consensus standard concernant la représentation et spécification conceptuelle dans domaine d'application. Cependant, réaliser un standard sémantique accepté et partagé par le plus grand nombre des membres d'un domaine d'application est une tâche très lente et dont le résultat est incertain et peut ne pas convenir à tous les intervenants, *a fortiori* quand le domaine d'application regroupe des disciplines ou des points de vue différents.

[14] ***St****andard for the* ***E****xchange of* ***P****roduct Model Data*, est le standard ISO 10303 sert à représenter et échanger l'information numérique des produits industriels.

1.6 Stratégies de résolution des problèmes d'interopérabilité sémantique

Pour essayer d'améliorer l'interopérabilité (sémantique) et résoudre les désaccords d'hétérogénéités ontologiques, différentes stratégies sont mentionnées dans la littérature. Nous pensons que le choix de la stratégie à employer est en général guidé par le contexte de l'application et par le but visé.

Les termes *alignement*, *intégration*, *fusion*, *articulation*, *morphisme* (Kalfoglou et Schorlemmer 2003), sont appliqués dans une grande quantité de travaux, provenant de différentes communautés, et qui revendiquent avoir un rapport avec la mise en correspondance (*mapping*) d'ontologies. Des définitions unanimes et standards de ces notions n'existent pas.

Alignement d'ontologies. Pour (Kalfoglou et Schorlemmer 2003) l'alignement établit une collection de relations binaires entre les vocabulaires des deux ontologies. (Bouquet *et al.* 2005) définissent l'alignement comme un ensemble de correspondances entre deux ou plusieurs ontologies (dans le cas d'alignement multiple). Un alignement est caractérisé plus en détail par les notions de couverture du domaine d'application, de granularité des descriptions et de perspectives, *i.e.* de point de vue, des ontologies source. Pour (Abels, Haak et Hahn 2005) l'alignement permet de ramener deux ontologies à un accord mutuel pour les rendre compatibles et cohérentes.

Fusion d'ontologies. En général, il est accepté (Noy et Musen 1999) que la fusion d'ontologies implique une mise en commun des sources de départ, qui probablement se chevauchent (Bouquet *et al.* 2005), pour générer une ontologie cible unifié. D'après (Pinto et Martins 2001), à la suite de la fusion il est difficile d'identifier les régions de l'ontologie cible qui ont été réutilisées à partir des ontologies source. Pour (Bouquet *et al.* 2005) la fusion d'ontologies est très liée à l'intégration.

Intégration d'ontologies. Selon (Kalfoglou et Schorlemmer 2003) l'intégration d'ontologies est la composition d'ontologies pour construire d'autres, mais dont les vocabulaires respectifs ne sont pas forcement interprétés dans le même domaine de discours. (Pinto et Martins 2001) précisent que l'intégration est le processus de construction d'une ontologie en réutilisant (par combinaison, agrégation, spécialisation, etc.) d'autres ontologies. En s'appuyant sur cette définition (Abels, Haak et Hahn 2005) considèrent que l'alignement et la fusion d'ontologies sont des sous-types d'intégration d'ontologies.

Dans ce travail de thèse, puisque nous voulons que les différents points de vue, *i.e.* chacune des ressources sémantiques, restent indépendant nous adoptons une méthode d'alignement de ressources sémantiques. La méthode d'alignement vise à découvrir des correspondances sémantiques entre les ressources (*cf.* chapitre 3). L'objectif de l'interopérabilité sémantique revient alors à relier les ressources sémantiques et leurs schémas, en utilisant des outils génériques permettant la connexion entre les entités des ressources sémantiques. Plus particulièrement, il est nécessaire de relier les sémantiques en tirant partie de la richesse terminologique de chaque ressources sémantiques, sans perdre de vue les limites de la sémantique computationnelle.

1.7 Méthodes d'alignement

Deux ontologies qui modélisent un même domaine de connaissances n'auront très probablement jamais une mise en correspondance parfaite l'une avec l'autre, sauf si elles ont été dupliquées. Elles peuvent, par contre, avoir des définitions en commun, *i.e.* représenter des sous-ensembles de connaissances du domaine d'application qui se chevauchent ou dont les instances forment une intersection non nulle.

Le but principal dans l'alignement est de trouver des correspondances entre entités (classes de concepts, propriétés ou instances) ou axiomes entre deux ontologies. Dans cette section nous présentons les méthodes qui permettent de comparer un élément d'une ontologie avec un autre élément d'une autre ontologie. Cet état de l'art sur les méthodes d'alignement est fondé sur celui de (Bach 2006, Rahm et Bernstein 2001) et les travaux de (Euzenat *et al.* 2004), notamment en ce qui concerne la catégorisation des méthodes d'alignement : terminologiques, structurelles, extensionnelles (*i.e.* fondées sur les instances) et sémantiques. Nous ajoutons à cette classification une catégorie supplémentaire, pour les méthodes qui se fondent sur les langages de représentation.

1.7.1 Méthodes terminologiques

Les méthodes terminologiques comparent des chaînes de caractères, des termes ou des textes. Elles peuvent être appliquées par exemple au nom, étiquette ou commentaire des entités à comparer. Ce type de méthodes peut être divisé en deux sous-catégories : méthodes lexicales et méthodes linguistiques. Les premières sont des méthodes qui se basent exclusivement sur les éléments à comparer et les deuxièmes utilisent des connaissances supplémentaires et spécifiques au langage naturel.

1.7.1.1 Méthodes lexicales

Ces méthodes exploitent la structure de la chaîne de caractères en tant que séquence de caractères, l'ordre des caractères dans la chaîne, le cardinal d'une lettre dans la chaîne (Cohen, Ravikumar et Fienberg 2003). Mais elles n'exploitent pas la signification des termes. Elles retrouvent des correspondances entre les termes `construction` et `Construction` mais pas entre l'un de ces deux termes et `bâtiment`. Ces méthodes emploient fréquemment des techniques issues du TALN pour, par exemple, normaliser les chaînes en substituant les majuscules par des minuscules, supprimer les mots dits vides et lemmatiser. Pour la comparaison de textes, par exemple, de commentaires ou de définitions en langage naturel, le contenu de textes est découpé en morceaux (*tokens*). Des ensembles de tokens sont regroupés en vecteurs dont chaque dimension correspond à un terme et la valeur est le cardinal de l'apparition du terme dans le token. La comparaison de textes se traduit par la comparaison de ces vecteurs. Les résultats sont, en général, des mesures de distance ou de similarité entre les vecteurs qui sont d'autant plus élevées si les mêmes chaînes de caractères ou des termes avec la même racine se retrouvent dans ces vecteurs.

1.7.1.2 Méthodes linguistiques

Ces méthodes s'appuient sur les techniques bien connues du TALN et utilisent surtout les caractéristiques expressives et productives de la langue naturelle, telles que la variation d'un terme (*i.e.* un terme technique peut être exprimé en différentes manières sans que le sens en soit modifié). Les méthodes linguistiques peuvent utiliser les propriétés internes des instances, telles que les propriétés morphologiques et la syntaxique (dans le cas des méthodes linguistiques intrinsèques), ou exploiter des ressources externes, telles que des dictionnaires et lexiques de la langue ou multi-lingues (dans le cas des méthodes linguistiques extrinsèques).

Les méthodes intrinsèques exécutent la comparaison terminologique en utilisant l'analyse morphologique et syntaxique pour réaliser la normalisation des termes. Cette normalisation vise à trouver la forme canonique d'un mot (lemme) à partir de ses variantes linguistiques (lexème). La recherche du lemme peut se servir d'un dictionnaire ou utiliser un algorithme *stemmer* qui retrouve la forme radicale à partir d'une forme infléchie ou dérivée d'un mot (Porter 1980).

Les méthodes extrinsèques utilisent des ressources externes pour retrouver des liens entre des termes. Les correspondances sont calculées grâce aux relations déjà exprimées dans une ressource externe, telles que la synonymie (pour l'équivalence) et l'hyponymie/hypernymie (pour

la subsomption). WordNet (Fellbaum 1998) est une ressource externe lexicologique qui est employée pour retrouver ces types de relations. Les ressources externes, telles que WordNet, EuroWordNet[15] (Vossen 2002) et VerbNet[16], sont des ressources génériques, similaires à des thésaurus de par leurs caractéristiques, qui peuvent être utilisées pour retrouver des liens de correspondance entre des entités d'autres ressources sémantiques. Par contre ces ressources externes semblent être plus appropriées pour relier des ressources dont des contextes sont génériques, par exemple dans le web en général, et moins pertinentes pour trouver des correspondances entre des ressources d'un domaine d'application bien ciblé et dont le contexte est très spécifique.

1.7.2 Méthodes structurelles

Les méthodes structurelles d'alignement sont celles qui exploitent l'organisation des ressources sémantiques, lorsque les concepts de ces dernières sont reliés entre eux, formant une hiérarchie ou un graphe. Les méthodes qui n'exploitent que la structure interne des entités (par exemple les informations relatives aux propriétés qui prennent leurs valeurs dans un type de données particulier) sont dites méthodes structurelles internes. Celles qui tirent partie des relations entre des entités sont nommées méthodes structurelles externes. Par exemple, les méthodes qui exploitent les relations d'héritage entre les concepts sont des méthodes structurelles externes.

1.7.2.1 Méthodes internes

Les méthodes fondées sur la structure interne des entités exploitent les informations sur le domaine des propriétés, les valeurs des propriétés (*i.e.* la cible ou *range* des propriétés, en termes d'OWL), leur cardinalité et les caractéristiques de transitivité, de symétrie et de restriction des valeurs des propriétés. Elles peuvent aussi être combinées avec d'autres approches (par exemple, terminologiques) pour faire en sorte d'éliminer de l'ensemble de correspondances possibles, les propriétés qui sont clairement incompatibles. Par exemple, en excluant les propriétés dont les valeurs cible ne sont pas du même type.

En dépit des différences entre alignement de schémas de BD et alignement de graphes ontologiques (Silva 2004, Noy et Klein 2002), quelques similitudes peuvent être trouvées entre eux, puisque les deux fournissent un vocabulaire de termes qui décrivent le domaine d'application et contraignent la signification des termes utilisés. Par exemple, les travaux de (Madhavan,

[15] http://www.illc.uva.nl/EuroWordNet/#EuroWordnet

[16] http://homepages.inf.ed.ac.uk/bcrabbe/vndoc.html

Bernstein et Rahm 2001) transforment d'abord chaque schéma de BD à aligner en un arbre. Les éléments de chaque schéma deviennent des nœuds dans l'arbre respectif. Ensuite, l'algorithme Cupid cherche des correspondances entre des noms de nœuds et des types de données, en comparant les feuilles des arbres (*cf.* section 1.8). Cupid cherche la compatibilité entre les types de données des propriétés en utilisant une table de recherche. Il est attribuée une valeur de compatibilité supérieure aux propriétés dont le type de données est identique.

1.7.2.2 Méthodes externes

Ces méthodes exécutent la comparaison entre entités de deux ontologies en tenant compte de la position des entités dans leurs structures respectives, qui sont le plus souvent des structures fondées sur des relations de subsomption (*i.e.* généralisation/subsomption) ou moins souvent de méréologie (*i.e.* partie/tout). Ces méthodes estiment que si deux entités de deux ontologies sont similaires alors leurs voisins pourraient eux aussi être semblables. En analysant les relations avec d'autres entités dans les hiérarchies, ces méthodes peuvent considérer une similarité entre les entités si par exemple leurs ancêtres directs (ou tous leurs ancêtres) et/ou leurs soeurs et/ou leurs descendants directs (ou tous leurs descendants) sont déjà similaires.

(Noy et Musen 2001) proposent de détecter la similarité entre sous-graphes en tenant compte des caractéristiques structurelles (*graph matching*) : l'algorithme de l'outil Anchor-PROMPT analyse le chemin de deux sous-graphes et identifie les concepts qui apparaissent en des positions similaires ou les portions de chemins similaires (*cf.* section 1.8). Les auteurs déclarent que ces concepts représentent probablement des concepts sémantiquement similaires.

D'autres travaux (Reynaud et Safar 2007) prennent en compte la profondeur entre deux concepts de graphes, *i.e.* leur distance à la racine des graphes respectifs en nombre d'arcs et celle de leur plus petit ancêtre commun pour calculer une mesure de similarité entre ces concepts. D'après les auteurs, cette mesure n'établit pas une correspondance sémantique prouvée par un mécanisme d'inférence, mais permet d'étiqueter une nouvelle relation de proximité entre les concepts.

Les approches fondées sur la structure et l'isomorphisme de graphes peuvent s'avérer intéressantes quand il existe une certaine homogénéité conceptuelle entre les graphes à comparer. Cependant ces méthodes montrent leurs limites quand le point de vue de modélisation et la granularité des ontologies sont très différents. Un exemple de ce type de situation est donné dans la Figure 6 (*cf.* section 1.5).

La recherche de similarité entre concepts peut aussi exploiter les relations reliant ces concepts. Autrement dit, la similarité entre deux relations de deux ontologies peut être exploitée pour déduire la similarité entre les domaines et les co-domaines des relations. Par exemple, considérons que la classe A établit un lien avec la classe B à travers la relation R dans une ontologie, et que la classe A' établit un lien avec la classe B' à travers la relation R' dans une autre ontologie. Si nous avons l'information qu'il existe une similarité entre B et B' d'une part et entre R et R' d'autre part, alors nous pourrions déduire qu'il existe une similarité entre A et A'. De la même manière si A et A' sont similaires et R et R' aussi, alors une relation de similarité existe peut-être aussi entre B et B'. Ou encore, si nous savons que A et A' sont similaires et que B et B' le sont aussi, alors R et R' le sont peut-être aussi. Ce type d'information est utilisé dans le calcul de distance de similarité de (Mädche et Staab 2002).

1.7.3 Méthodes extensionnelles

Ces méthodes cherchent à aligner deux ontologies en analysant les extensions de ces dernières, *i.e.* leurs instances. Ces méthodes emploient en général des techniques de fouille de données, d'apprentissage automatique ou encore d'analyse en Concepts Formels. Les systèmes GLUE et FCA-Merge (*cf.* section 1.8) exploitent ce type d'information. Des techniques statistiques, d'analyse de fréquence d'apparition et de calcul de probabilités sont utilisées pour calculer des mesures de similarité. Puisque les données manipulées par ces méthodes sont les instances et de par les techniques utilisées, les résultats sont plus intéressants quand les instances sont nombreuses.

1.7.4 Méthodes sémantiques

La caractéristique fondamentale des méthodes sémantiques est le traitement de l'information en termes de son interprétation sémantique formelle (par exemple, *model-theoretic semantics*). Ce sont donc des méthodes déductives. Les exemples sont les techniques de satisfiabilité propositionnelle (SAT) et le raisonnement fondé sur les logiques de description (LD).

1.7.4.1 Méthodes fondées sur la satisfiabilité propositionnelle

Cette approche décompose le problème de mise en correspondance de graphes ou d'arbres en un ensemble de problèmes de mise en correspondance de nœuds (Bouquet 2003). Ensuite chaque paire de nœuds des deux structures à faire correspondre est traduite dans une formule propositionnelle et leur validité testée. Une formule propositionnelle est valide si et seulement si sa négation est insatisfiable. L'insatisfiabilité est vérifiée en utilisant des procédures de recherche de satisfiabilité (*solver*) SAT. Le système S-Match (Giunchiglia, Shvaiko et Yatskevich 2004) utilise

la satisfiabilité propositionnelle pour trouver des correspondances entre ontologies (*cf.* section 1.8). La limite de cette approche est qu'elle n'utilise que des prédicats unaires, donc des concepts.

1.7.4.2 Méthodes fondées sur les logiques de description

Ce type de méthodes est utilisé dans cette thèse. Deux ontologies, représentées en une logique de description (LD), sont testées par rapport à la subsomption pour aligner les éléments ontologiques. Le test est réalisé en utilisant un moteur d'inférences et l'algorithme des tableaux sémantiques. Cet algorithme utilise un ensemble de règles d'expansion. Le fonctionnement de cet algorithme est détaillé à la section 2.2.2. La vérification des relations sémantiques entre des entités ontologiques permet de déduire, en utilisant un moteur d'inférences (*cf.* section 2.2.3), des correspondances entre ontologies, telles que l'équivalence (la correspondance est totale), la subsomption (la correspondance est partielle) ou l'exclusion (il n'y a pas de correspondance). Le système CtxMatch 2.1 (*cf.* section 1.8) applique cette méthode.

1.7.5 Méthodes fondées sur le langage de représentation

Ce type de méthode se fonde sur l'analyse et la recherche des constructeurs du langage utilisés pour représenter les ontologies. Cette méthode est donc guidée par la syntaxe des définitions du langage de représentation. Cependant, ce n'est pas la sémantique du contenu qui est traité mais plutôt celle du langage de représentation. En général, une méthode fondée sur l'analyse des constructeurs d'un langage est utilisée après l'utilisation d'autres méthodes et prend les résultats précédents, par exemple d'une méthode terminologique, en entrée. C'est le cas du système oMAP (*cf.* section 1.8).

1.7.6 Synthèse sur les méthodes d'alignement

Le choix d'une méthode plutôt qu'une autre dépend des entités existantes dans les ontologies et de la tâche à réaliser. Une méthode terminologique compare plutôt les entités à partir des termes qui les dénotent, *i.e.* exploite le signifiant (la forme du signe linguistique) mais pas le signifié (*i.e.* le contenu sémantique du signe linguistique). Les méthodes terminologiques nous semblent plus opportunes pour un domaine d'application bien ciblé, dont le contexte est parfaitement défini et le vocabulaire du domaine est non polysémique. Les méthodes qui utilisent des thésaurus produisent de meilleurs résultats s'ils sont spécifiques au domaine d'application. Cependant ces thésaurus du domaine ne sont pas toujours disponibles. Néanmoins une approche exclusivement linguistique ne semble pas tirer parti de la conceptualisation d'une ontologie et de la manière dont les connaissances y sont organisées.

Si les instances sont peu nombreuses, une méthode fondée sur la fouille de données n'est pas un choix approprié. Par ailleurs, une approche exclusivement structurelle ne peut pas distinguer entre les graphes qui sont isomorphes mais sémantiquement différents. L'utilisation exclusive d'une méthode structurelle implique en général que les noms des concepts correspondent exactement, alors qu'ils peuvent être des homonymes. La structure des ontologies encapsule rarement suffisamment de sémantique pour utiliser une méthode structurelle comme seule méthode d'alignement. Enfin, quelques systèmes combinent plusieurs techniques.

Pour conclure, une méthode fondée sur la syntaxe du langage de représentation ne prendre pas en compte toute la richesse de la découverte de relations supplémentaires, qui peut s'effectuer en utilisant un moteur d'inférences pour les LD.

1.8 Étude des systèmes d'alignement

Nous analysons ensuite quelques systèmes de recherche de correspondances entre ressources sémantiques. Le Tableau 1 résume les caractéristiques de chaque système.

GLUE (Doan *et al.* 2002) vise à trouver des correspondances 1 à 1 entre des concepts de taxonomies en calculant des mesures de similarité probabiliste de l'intervalle [0;1] entre deux nœuds, de manière semi-automatique. Pour cela ce système utilise des techniques d'apprentissage automatique, des instances des concepts et quelques heuristiques du domaine d'application relatives aux nœuds, *i.e.* concepts, voisins dans les structures taxonomiques. La technique d'apprentissage automatique employée utilise les résultats d'application de techniques de TALN, tels que la fréquence des mots du contenu textuel d'une instance ainsi que son nom pour générer des prédictions probabilistes quant à l'appartenance de l'instance à un concept donné. Cette technique fonctionne mieux en présence de textes longs qu'en présence de valeurs courtes et numériques. Ce système se base sur la distribution des probabilités des instances des concepts pour calculer une mesure de similarité. GLUE est peu efficace si les instances sont peu nombreuses. Il ne peut pas être utilisé avec une ressource sémantique privée d'instances.

MAFRA - *Ontology MAapping FRAmework* (Maedche *et al.* 2002) est un système composé de plusieurs modules servant à gérer des correspondances semi-automatiques entre des ontologies. Un des modules vise à créer des ponts sémantiques entre des concepts et entre des relations de deux ontologies, et ainsi générer une ontologie contenant ces ponts sémantiques dans le langage DAML+OIL. Les ponts sémantiques sont générés à partir d'informations lexicales, structurelles et de tables de transformation. La recherche de similarités lexicales entre entités, *i.e.* concept et

relation des ontologies sources, utilise WordNet et une adaptation de l'algorithme de Resnik (Resnik 1999). Les ponts sémantiques sont ensuite définis en tenant compte du résultat de la recherche de similarités lexicales et d'heuristiques représentées dans des tables de transformation. Dans ce processus, l'expert du domaine peut fournir de l'information supplémentaire. MAFRA propose aussi un format de représentation des ponts sémantiques qui sert à les manipuler dans l'architecture proposée, mais qui ne semble *a priori* pas être ouvert à des usages externes.

LOM - *Lexicon-based Ontology Mapping* (Li 2004) retrouve des similarités morphologiques entre vocabulaires, qui représentent des entités de deux ontologies. Cet outil utilise quatre techniques fondées sur la similarité lexicale des vocabulaires : le nom des termes, les lettres des termes, les synonymes (en utilisant les *synsets* de WordNet) et ce que les auteurs appellent la *mise en correspondance de type*. Par exemple, deux termes qui ont le plus grand nombre de synsets en commun sont sauvegardés et présentés à l'utilisateur. Quant au à la mise en correspondance de type, LOM se sert de correspondances trouvées auparavant entre les synsets de WordNet et d'ontologies généralistes SUMO[17] et MILO[18] pour rechercher d'autres similarités lexicales. LOM ne garantit pas l'exactitude des correspondances et son traitement des symboles abstraits est limité. Les experts humains sont impliqués dans la phase de vérification et de validation des résultats.

QOM - *Quick Ontology Mapping* (Ehrig et Staab 2004) utilise un processus itératif pour rechercher des correspondances. Ce système recherche des similarités entre les entités (*i.e.* concept, relations et instances), en utilisant des techniques lexicales, surtout par comparaison de chaînes de caractères et la structure des ontologies, *i.e.* les relations d'héritage entre les entités. Les valeurs obtenues sont ensuite agrégées. Finalement les différentes valeurs obtenues sont départagées suivant différents mécanismes : utilisation d'un seuil, utilisation d'heuristiques pour rejeter les correspondances moins bien classifiées ou des critères structurels. Le résultat est une table qui contient les correspondances. Une entité peut avoir au plus une autre entité correspondante.

[17] *Suggested Upper Merged Ontology*, http://www.ontologyportal.org/

[18] *Mid-level Ontology*, http://sigmakee.cvs.sourceforge.net/*checkout*/sigmakee/KBs/Mid-level-ontology.kif

Tableau 1 – Comparaison des outils d'alignement

Caractéristiques / Systèmes	Entrée	Sortie	Techniques	Information Externe	Niveau de connaissances exploité	Interaction avec l'utilisateur	Langages de représentation	Disponibilité du logiciel
GLUE (Doan *et al.* 2002)	deux taxonomies, documents textuels	mesure de similarité probabiliste entre [0;1] entre deux noeuds	techniques d'apprentissage automatique, distribution des probabilités des instances	heuristiques du domaine d'application	instances, lexicales, spécifiques au domaine d'application	correspondances initiales définies par l'utilisateur, poids pour le programme de *machine learning*, analyse du résultat	non spécifié	non spécifié
MAFRA (Maedche *et al.* 2002)	deux ontologies, tables de transformation	correspondances dans une ontologie de ponts sémantiques	TALN : algorithme de Resnik, tables de transformation et création de ponts sémantiques, propose aussi un format de représentation des ponts sémantiques adapté à l'architecture proposée	thésaurus (WordNet)	instances, lexicales, spécifiques au domaine d'application	interface graphique	RDF(S) et DAML+OIL	GNU General Public License https://sourceforge.net/projects/mafra-toolkit/
LOM (Li 2004)	liste de vocabulaires des deux ontologies, séparés en classes, propriétés, instances	liste de correspondances entre termes avec des points de classement de similarité entre [0;1]	TALN : (similarité lexicale des vocabulaires) le nom des termes, les lettres des termes, les synonymes (synsets de WordNet) et le type de données	thésaurus (WordNet) et ontologies généralistes (SUMO et MILO)	instances, lexicales, généraliste	vérification et validation des correspondances ; interface de requête	non spécifié	non spécifié

QOM (Ehrig et Staab 2004)	deux ontologies	table de correspondances	TALN et structurelles : comparaison de chaînes de caractères et les relations d'héritage entre les entités	information non disponible	identificateurs, primitives du langage de représentation, instances, lexicales	information non disponible	RDF(S)	non spécifié
FOAM (Ehrig et Sure 2005)	deux ontologies et similarités prédéfinies, par exemple entre les identifiants des concepts	correspondances 1 à 1	techniques d'apprentissage automatique (apprenti arbre de décision)	-	concepts, instances	dans la phase de vérification des résultats	OWL-DL	http://ontoware.org/projects/map
ONION (Mitra et Wiederhold 2002)	deux ontologies	règles d'articulation liant 2 concepts et un poids de similarité linguistique	TALN : comparaison linguistique en utilisant une table de correspondances entre des termes, recherche d'isomorphisme structurel	WordNet et corpus de documents (pour établir la table de correspondances entre des termes)	concepts, lexicales	interface graphique pour vérification et validation des résultats	RDF	non spécifié
FCA-Merge (Stumme et Maedche 2001)	deux ontologies et un ensemble de documents en langage naturel	une ontologie fusionnéee	extraction d'instances à partir des documents, techniques de TALN, Analyse en Concepts Formels, algorithme Titanic pour tailler le treillis	lexiques généralistes des domaines (non spécifiés) utilisés pour l'analyse lexicale	instances	l'utilisateur intervient dans la phase finale pour sélectionner les concepts du treillis qui feront partie de l'ontologie fusionnée	logique de frames	non spécifié

Anchor-PROMPT (Noy et Musen 2001)	deux ontologies, des ancres, des paramètres tels que la longueur (nombre d'arcs) des sous graphes à comparer	liste de paires de termes correspondants, classés par score de proximité	*matching* lexical, comparaison structurelle	-	concepts, instances	l'utilisateur est appelé à identifier manuellement quelques ancres initiales, *i.e.* quelques paires de termes similaires dans les 2 ontologies	DAML (version initiale, RDF(S) et OWL (plus récemment)	Mozilla Public License http://protege.cim3.net/cgi-bin/wiki.pl?Prompt
Cupid (Madhavan, Bernstein et Rahm 2001)	schémas de BD transformés en arbres	corrrespondances en termes de similiarités obtenues par somme ponderée, entre les noeuds des deux arbres (correspondances 1:n possibles)	linguistiques (par exemple, normalisation de chaînes de caractères) etstructurelles (fondée principalement sur la comparaison des feuilles des arbres)	thèsaurus de synonymes et d'hyperonymes	lexicales, structure, instances	-	XML, XSD	non spécifié
oMAP (Straccia et Troncy 2005)	deux ontologies, de préférence avec une grande quantité d'instances	correspondances acompagnées de poids	linguistiques (par exemple, calcule de distance d'édition), extensionnelles (probabilistes) et fondées sur le langage de représentation	WordNet (pour un des classifieurs de calcul linguistique)	instances, constructeurs du langage de représentation	-	OWL	http://homepages.cwi.nl/%7Etroncy/oMAP/
S-Match (Giunchiglia, Shvaiko et Yatskevich 2004, Shvaiko 2006)	deux arbres	correspondances 1 à 1 du type $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$ entre les concepts	linguistiques (par exemple, normalisation de chaînes de caractères), transformation en logique propositionnelle et procédure décisionnelle SAT	thésaurus (WordNet)	lexicales, structure, concepts	-	logique propositionnelle	non spécifié

CtxMatch (Bouquet, Serafini et Zanobini 2003) et **CtxMatch-2.1** (Bouquet, Serafini et Zanobini 2006)	deux hiérarchies de concepts	correspondances 1 a 1 du type $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$ entre les concepts	linguistiques par utilisation de thèsaurus, logique propositionnelle (SAT) pour CtxMatch et inférence logique pour CtxMatch 2.1	thésaurus (WordNet)	conceptuel, lexical	-	logique propositionnelle pour CtxMatch et logique de description *ALCIO* pour CtxMatch-2.1	Pour CtxMatch 2.1 : GNU general public license http://dit.unitn.it/%7Ezanobini/downloads.html
Alignment API (Euzenat 2004)	deux ontologies	un alignement composé d'une liste de correspondances	comparaison entre chaînes de caractères et calcul de distance en fonction de cette comparaison, fourniture d'un format de représentation d'alignement et une interface de programmation	-	classes, propriétés, instances	-	formats de lecture : OWL/RDF ; formats de sortie : RDF, OWL, XSLT, SWRL	GNU Lesser General Public License http://alignapi.gforge.inria.fr/

FOAM - *Framework for Ontology Alignment and Mapping* (Ehrig et Sure 2005) suit la même méthodologie que QOM, mais étend ce dernier avec des techniques d'apprentissage automatique et des règles fondées sur des heuristiques. Pour lancer le processus, le système nécessite en entrée, en plus des ontologies à comparer, un ensemble de similarités prédéfinies par exemple entre des noms d'entités. Ce système utilise un programme apprenti (*machine learning*) nommé arbre de décision (Ehrig, Staab et Sure 2005). Il nécessite une quantité suffisante d'exemples d'apprentissage pour entraîner le système.

ONION - *ONtology compositION system* (Mitra et Wiederhold 2002) génère des règles d'articulation, *i.e.* des relations entre deux concepts, pour établir des ponts sémantiques entre ontologies. Les règles d'articulation sont du type : [(*MatchConcept1Concept2*), poids de similarité], pour indiquer que les concepts 1 et 2 de deux ontologies sont liés avec un poids de similarité donné, sans plus de détail sur le type de liaison. La détermination de celui-ci est laissée à l'expert humain. ONION contient un générateur d'articulations qui utilise une bibliothèque d'heuristiques de liaisons préétablies pour générer d'autres règles. Le système utilise un algorithme de comparaison linguistique des noms des termes. Il compare des paires de termes et attribue des points à chaque paire, en consultant une table de correspondances entre termes, préalablement établie à partir de thésaurus tels que WordNet ou de corpus d'environ 200 pages recherchées sur Internet avec des mots-clés du domaine. Le corpus, *i.e.* un ensemble de documents représentatif du domaine d'application, a produit de meilleurs résultats que le thésaurus (généraliste). Le poids de similarité est calculé en fonction du nombre de mots de chaque terme et de la fréquence de la présence dans la table de correspondances entre termes. Si le total des points de similarité dépasse un certain seuil, alors la comparaison correspondante est acceptée et une règle d'articulation entre la paire de concepts est générée. Un deuxième algorithme complète les règles générées par une recherche d'isomorphisme structurel entre des sous-graphes d'ontologies en partant d'une paire de concepts liés par l'algorithme linguistique, pour chercher à mettre en correspondance leurs sous-concepts ou super-concepts s'ils ne le sont pas encore.

FCA-Merge (Stumme et Maedche 2001) exploite des instances dans le but de fusionner des ontologies. D'abord les auteurs proposent d'extraire ces instances de documents textes, en utilisant des techniques de TALN (par exemple l'analyse lexicale en utilisant des lexiques pour effectuer de l'analyse morphologique des documents), puis d'appliquer l'Analyse en Concepts Formels. Ce système génère un treillis de concepts par l'analyse de la fréquence de chaque instance. Chaque nœud du treillis de concepts obtenu est associé à un ensemble de concepts des

ontologies d'entrée lorsque les instances de ces ontologies sont contenues dans les mêmes documents. L'étape finale d'analyse du treillis pour construire l'ontologie fusionnée est à la charge de l'expert concepteur d'ontologies.

Anchor-PROMPT (Noy et Musen 2001) est un algorithme qui travaille sur la structure des ontologies pour retrouver des corrélations entre des concepts de différentes ontologies. Il a été implanté comme un plugin pour l'outil d'édition d'ontologies Protégé[19]. Anchor-PROMPT prend en entrée un ensemble de paires de termes (ancres) des ontologies définies par l'utilisateur, ou déterminées par mise en correspondance lexical préalable. L'algorithme analyse les sous-graphes limités par les ancres, chaque paire d'ancres étant issue d'un des graphes à comparer, et détermine les classes qui se trouvent dans des positions similaires des sous-graphes semblables. Les auteurs considèrent que ces classes représentent probablement des concepts sémantiquement similaires. Un score de similarité est associé à chaque paire de classes retrouvée. Il est fonction de la fréquence des classes dans des positions identiques des sous-graphes. L'algorithme est fondé sur une supposition structurelle et conceptuelle, trop forte à notre avis, que les termes des ontologies sont liés de manière similaire, même si les termes sont nommés différemment. Ceci supposerait alors que les concepteurs des ontologies ont utilisé la même méthodologie de modélisation et qu'ils ont aussi les mêmes points de vue sur le domaine d'application. Nous avons constaté dans notre étude de cas que dans certaines ontologies, des concepts équivalents sont positionnés en des structures complètement différentes, et dans d'autres des concepts complètement dissemblables appartiennent à des structures isomorphes.

Cupid. Les travaux de (Madhavan, Bernstein et Rahm 2001) transforment d'abord chaque schéma de BD à aligner en un arbre dont les éléments des schémas deviennent des nœuds. Ensuite, l'algorithme Cupid cherche des correspondances entre des noms et des types de données, en comparant les feuilles des arbres. Une première phase utilise des méthodes linguistiques pour comparer les noms et les types de données, en utilisant un thésaurus de synonymes et d'hyperonymes non précisé. Dans une deuxième phase les auteurs appliquent une méthode structurelle. Celle-ci est fondée sur la comparaison des voisins des éléments (ancêtres et frères) à partir des feuilles (approche ascendante), en tenant compte des correspondances linguistiques trouvées dans la phase précédente. L'algorithme se fonde sur un calcul de similarité. La valeur de similarité calculée pour les feuilles va ensuite influencer le calcul de similarité

[19] http://protege.stanford.edu/

structurelle des nœuds non-feuilles. Les auteurs affirment que la similarité structurelle est faible entre deux nœuds dont la quantité de feuilles est très différente. En accord avec (Bouquet, Serafini et Zanobini 2003), Cupid fonctionne mieux pour la comparaison d'hiérarchies de concepts représentant des types de données que pour les hiérarchies qui n'ont pas de types de données.

oMAP (Straccia et Troncy 2005) applique les méthodes d'alignement terminologiques, extensionnelles et celles fondées sur le langage de représentation. Il propose des *classifieurs* d'alignement qui calculent le poids attribué à l'alignement. Le classifieur terminologique travaille sur le nom des entités. Si les entités ont le même nom ou la même racine (radicale) alors le poids est 1, sinon il est 0. Des mesures de similarité sont aussi calculées en utilisant, par exemple, le calcul de distance d'édition (de chaînes de caractères). Comme le système GLUE, oMAP applique des méthodes extensionnelles pour calculer des mesures de similarité ; le classifieur naïf de Bayes utilise des techniques de calcul de probabilités pour classification de texte, dont les mots sont les instances des concepts et des relations. Le troisième type de classifieur est nommé, par les auteurs, classifieur structurel. Ce classifieur exploite la syntaxe du langage de représentation. Il peut prendre en compte le résultat des classifieurs précédents et calculer d'autres poids d'alignement. Par exemple, si les termes à comparer sont déclarés comme noms de concepts, le poids de la correspondance est recalculé à partir des définitions des parents directs de ces concepts. Les auteurs attribuent des valeurs (0, 1/3, 1/4, 1) aux connecteurs logiques, aux quantificateurs et aux cardinalités minimale et maximale. Celles-ci sont utilisées dans la formule de calcul du poids d'alignement.

S-match (Giunchiglia, Shvaiko et Yatskevich 2004, Shvaiko 2006) recherche des correspondances en utilisant des méthodes sémantiques fondées sur une recherche de la satisfiabilité (SAT). Il prend en entrée deux arbres à comparer et produit en sortie une relation entre les concepts (les nœuds). S-match tient compte du sens des étiquettes des nœuds et de la position des nœuds dans les arbres. Une phase initiale d'enrichissement linguistique et de transformation en logique propositionnelle est effectuée sur chaque arbre séparément : le sens des étiquettes des nœuds est d'abord construit en appliquant des techniques linguistiques (par exemple, la normalisation) ; ensuite par l'extraction du sens à partir des synsets de WordNet ; les termes en langage naturel sont traduits dans le langage propositionnel en utilisant les connecteurs logiques. Les différents sens que WordNet attache à un terme (*i.e.* le nœud) sont représentés par leur union et deviennent l'étiquette du nœud. Ensuite, le concept de chaque nœud est représenté par la conjonction entre l'étiquette de ce nœud et celles de tous ses ancêtres sur le chemin de

l'arbre depuis la racine. Autrement dit, la représentation typique en LD d'une chaîne de subsomptions entre concepts est transformée ici en une chaîne de conjonctions. La phase suivante met en œuvre la recherche de correspondances. Le problème de mise en correspondance des nœuds est alors traduit de la manière suivante : chaque paire de nœuds des deux structures à faire correspondre est traduite dans une requête de mise en correspondance de la forme *axiome* → *rel* ($n\text{œ}ud_1$, $n\text{œ}ud_2$). La partie *axiome* est la conjonction de toutes les relations entre les concepts atomiques associés aux nœuds et *rel* correspond à la requête de relations possibles entre une paire de nœuds des deux graphes. Les relations possibles sont l'équivalence, le plus spécifique, le plus générique et la disjonction (*mismatch*), traduites en connecteurs propositionnels. La procédure de décision SAT est ensuite lancée pour tester la validité de la formule, *i.e.* pour prouver que la négation de la formule est insatisfiable. Si ceci se vérifie, alors la formule est valide et une relation de correspondance existe entre la paire de nœuds.

CtxMatch (Bouquet, Serafini et Zanobini 2003) est un algorithme fondé sur la logique propositionnelle, utilisant les connecteurs ⊔, ⊓ et ¬, qui vise à traiter le problème de la coordination de classifications hiérarchiques (CH). Ceci revient ici à trouver un ensemble de relations 1 à 1 entre des concepts de deux CH. Ainsi cet algorithme prend en entrée deux CH H et H', des parties des annuaires de Google™ et de Yahoo!™ par exemple. Pour chaque paire de concepts $k \in H$ *et* $k' \in H'$, l'algorithme renvoie une relation entre eux. Ces relations sont : k est moins général que k' ($k \subseteq k'$), k est plus général que k' ($k \supseteq k'$), k est équivalent à k' ($k \equiv k'$), k est compatible avec k' ($k * k'$) et k est incompatible avec, *i.e.* disjoint de, k' ($k \perp k'$). L'algorithme a deux phases : l'obtention sémantique et la comparaison sémantique. La première utilise une ressource externe telle que WordNet, pour construire la définition formelle de chaque concept (*i.e.* les nœuds) des CH et de cette manière construire les axiomes logiques de subsumption. La deuxième phase cherche les relations entre les concepts des deux CH en utilisant un programme pour résoudre le problème de satisfiabilité propositionnelle, non spécifié par les auteurs. La contribution de CtxMatch est de définir les correspondances en utilisant la sémantique de la théorie des modèles. L'avantage de cet algorithme repose sur l'utilisation d'une ressource externe qui permet de définir et de préciser les concepts des CH, puisque les CH initiales sont pauvres sémantiquement. L'inconvénient est le fait que cette recherche de correspondances entre les concepts de CH ne s'appuie que sur la relation structurelle de subsomption. Les CH sont des taxonomies dont les définitions de concepts n'ont pas de rôles, au sens LD.

CtxMatch-2.1 (Bouquet, Serafini et Zanobini 2006) est une extension du système précédent. Cette nouvelle version propose d'utiliser la LD $\mathcal{ALCOI}$[20] combinée dans ce que les auteurs appellent WDL (pour *WordNet Description Logic*) et un moteur d'inférences tel que RACER (*cf.* section 2.2.3) pour détecter des correspondances entre des CH. Ce système prend donc les CH en entrée, transforme ces structures en LD et les enrichit par un ajout d'éléments de ressources linguistiques externes telles qu'en utilisant WordNet. Comme pour le système S-match, tous les sens d'un terme renvoyé par WordNet sont représentés dans le nœud par leur union. Ensuite, le système compare deux formules en LD pour détecter s'il existe une relation (équivalence, subsomption) entre elles, en utilisant un moteur d'inférences.

Alignment API (Euzenat 2004) implante des algorithmes terminologiques classiques, comme le calcul de distance entre deux sous-chaînes de caractères (*edit distance* ou distance de Levenshtein), *i.e.* le calcul du nombre minimal d'opérations à effectuer pour transformer le nom d'une entité (concept, propriété, individu) en une autre. Mais l'atout de ce système est plutôt la mise à disposition d'un format de représentation d'alignements et d'une interface de programmation qui vise à manipuler des alignements. Ceux-ci peuvent être réutilisés par d'autres systèmes et étendus à d'autres méthodes d'alignement.

D'autres systèmes revendiquent leur contribution à la résolution du problème de l'alignement d'ontologies, mais parfois en le nommant différemment. Ne pouvant pas tous les analyser ici, nous avons décidé de sélectionner ceux qui nous semblent plus en rapport avec ces travaux de thèse. (Bach 2006, Euzenat *et al.* 2004, Choi, Song et Han 2006, Abels, Haak et Hahn 2005, Kalfoglou et Schorlemmer 2003) fournissent des analyses d'autres systèmes qui complètent celles présentées ici. Les algorithmes et les résultats des processus d'alignement sont parfois difficilement comparables, parce que les méthodes utilisées sont différentes. Certains systèmes appliquent des techniques qui sont plus adaptées au traitement des instances (GLUE, QOM, FOAM et FCA-Merge), tandis que d'autres exploitent le niveau conceptuel (S-Match et CtxMatch). D'autres encore tirent partie de ces deux niveaux et exploitent plutôt l'aspect structurel (Anchor-Prompt et Cupid) ou utilisent les constructeurs du langage de représentation (oMAP). Le dernier système présenté est le plus proche de la démarche que nous proposons

[20] $\mathcal{ALCOI}$ est la logique de description *Attributive Language with Complements* à laquelle sont ajoutés les contructeurs permentant de designer un ensembles d'individus (pour le *O)* et des rôles (ou relations) inverses (pour le *I*).

puisque CtxMatch 2.1 applique la logique déductive pour retrouver des correspondances sémantiques.

1.9 Synthèse

Ce chapitre commence par approfondir les notions de syntaxe et de sémantique qui sont fondamentales dans le domaine de représentation des informations et des connaissances. En particulier nous détaillons trois types de sémantiques, nommés implicite, formelle et puissante. Ensuite différentes manières d'agencer les connaissances sont présentées : dictionnaires, taxonomies, thésaurus et ontologies. Ces ressources sémantiques sont analysées les unes par rapport aux autres. Nous étudions les types de relations que nous retrouvons dans les ressources. Après différents types d'hétérogénéité sont analysés, ce qui finalement rend compte de la complexité du problème de l'interopérabilité entre des ressources sémantiques. En fait, l'hétérogénéité mène à un déficit d'interopérabilité. Toutefois l'hétérogénéité représente aussi la richesse des ressources sémantiques. Il est intéressant de rechercher les ponts sémantiques qui permettent de relier les entités des ressources.

La section 1.5 présente plusieurs définitions du concept d'interopérabilité des ressources sémantiques selon les aspects syntaxiques, structurels et sémantiques. Les deux premiers aspects ont déjà été traités maintes fois par des travaux précédents et nous en citons quelques uns. Quant à l'interopérabilité sémantique, les définitions divergent et sa réalisation s'avère une tâche complexe. De plus en plus de ressources sémantiques existent qui sont autant de points de vue cognitifs différents. Le manque d'interopérabilité laisse supposer que les ressources sémantiques n'ont pas été construites avec un objectif de standardisation par rapport aux ressources déjà existantes. En fait les conséquences du manque d'interopérabilité sont souvent sous-estimées, surtout lors de la conception des ressources sémantiques. Mais, aboutir à une ressource sémantique standard et partagée n'est pas une solution simple, parce que réaliser ce consensus est une tâche très lente et dont le résultat est incertain. Ce résultat peut ne pas convenir à tous les intervenants, *a fortiori* quand le domaine d'application regroupe des disciplines ou des points de vue différents.

Nous soutenons l'existence de deux dimensions de l'interopérabilité entre les ressources sémantiques : horizontale et verticale (Figure 8). La dimension horizontale concerne l'interopérabilité entre deux ressources du même type, par exemple entre deux ontologies ou deux thésaurus. La dimension verticale implique l'interopérabilité entre deux types différents de ressources sémantiques, par exemple entre une ontologie et une taxonomie. Manifestement, cette

dernière dimension apporte une difficulté supplémentaire puisque des hétérogénéités entre paradigmes de modélisation (*cf.* section 1.4) sont ardues à surmonter sans pertes.

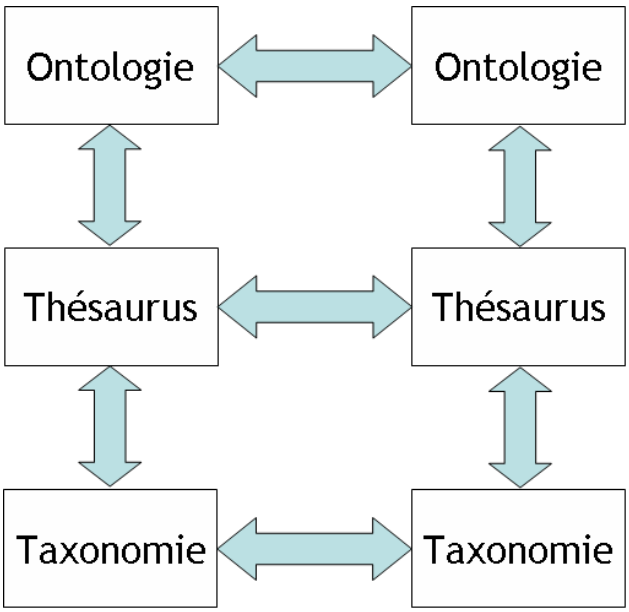

Figure 8 – Représentation des dimensions horizontale et verticale de l'interopérabilité entre les ressources sémantiques.

Dans la section 1.6 avons présenté différentes approches de résolution des problèmes d'interopérabilité sémantique. Nous avons introduit les méthodes d'alignement d'ontologies dans la section 1.7. Les méthodes d'alignement d'ontologies sont présentées et catégorisées selon cinq sortes : terminologiques, structurelles, extensionnelles, sémantiques et langages de représentation. Nous avons analysé plusieurs systèmes (*cf.* section 1.8) qui se proposent d'aligner des ontologies et des arbres par la génération de correspondances entre les entités de ressources à aligner. Certains de ces systèmes proposent des techniques issues d'une des méthodes d'alignement préalablement présentées, alors que d'autres combinent des techniques de plusieurs de ces méthodes. Le Tableau 1 résume et permet de comparer les caractéristiques des divers systèmes et outils d'alignement étudiés. Le chapitre suivant présent les méthodes, les techniques et les outils que nous utilisons dans ces travaux de thèse.

Chapitre 2 - Méthodes, techniques et outils

L'objectif de ce chapitre est de présenter les langages de représentation des connaissances qui sont utilisés au cours de ce travail. Nous présentons aussi plusieurs types d'inférences, l'algorithme des tableaux sémantiques de base pour les logiques de description et trois moteurs d'inférences que se fondent sur cet algorithme. Nous exposons deux interfaces de programmation pour les langages du Web Sémantique.

Différents systèmes nécessitent d'accéder à de multiples ressources sémantiques, de manière à rendre effective l'aptitude à échanger et traiter intelligemment l'information pendant le temps d'exécution, *i.e.* l'interopérabilité sémantique. Chercher à fournir l'interopérabilité sémantique fait rencontrer des problèmes similaires à ceux associés à la communication entre différentes communautés. La grande différence est que les acteurs ici ne sont pas des personnes capables de faire abstraction et de raisonner en contexte ambigu sur la signification des termes, mais plutôt des machines (Stuckenschmidt 2003). Donc, il est nécessaire d'expliciter le contexte de chaque système à un niveau de formalité élevé de manière à permettre à des machines de se comprendre entre elles. C'est pour cela que nous entamons cette section par l'exposition de langages conçus pour exprimer et échanger les connaissances.

2.1 Langages de représentation de l'information

La représentation de l'information, que certains auteurs (Brachman et Levesque 2004) nomment représentation des connaissances, est le domaine d'étude qui s'intéresse à l'utilisation des symboles formels pour représenter un ensemble de propositions admises comme réelles. Nous présentons ici les langages de représentation qui ont servi à représenter les ressources sémantiques et que nous utilisons dans la deuxième partie de ces travaux.

2.1.1 Le langage EXPRESS

Le langage EXPRESS (ISO 10303-11:1994) est un langage conceptuel de modélisation des données de produits et est spécifié dans la partie 11 du standard STEP. Ce langage vise la standardisation de la représentation et de la description des données de produits pour faciliter l'échange de ces données. EXPRESS est utilisé, par exemple, par l'industrie aéronautique et

l'industrie de la construction (voir par exemple l'*International Alliance for Interoperability*[21], IAI). EXPRESS emprunte certaines caractéristiques des langages orienté objets, sans être un langage de programmation. EXPRESS est un langage structuré composé d'une partie déclarative et d'une partie procédurale. Les définitions sont aussi souvent accompagnées par une partie en langage naturel. L'héritage multiple est permis. Le mot réservé `SUBTYPE OF` est utilisé dans la déclaration d'une entité (classe) pour représenter la relation hiérarchique de spécialisation. Le mot réservé `SUPERTYPE OF` représente la relation hiérarchique de généralisation. Les éléments de base du langage sont les suivants (Schenck et Wilson 1994) :

- **Entité** : elle représente les concepts du domaine d'application. Dans la définition d'une entité, sont aussi déclarés les attributs et comportements qui la caractérisent. Une entité est déclarée, dans un schéma, avec le mot réservé du langage `ENTITY` suivi de son nom identifiant, d'un point-virgule et se termine avec le mot réservé `END_ENTITY;` Exemple générique :

```
SCHEMA Nom_schéma;
ENTITY Nom_entité;
SUBTYPE OF (Nom_entité_plus_générique);
...
END_ENTITY;
END_SCHEMA;
```

- **Attribut** : c'est une propriété particulière qui identifie une caractéristique d'une entité et dont les valeurs distingueront un objet instance de l'entité d'autres objets de la même entité. Chaque attribut a une relation avec l'entité dont il fait partie. Les attributs sont représentés par des types de données simples ou par d'autres entités. Dans ce dernier cas un attribut représente alors une relation vers une autre entité. Ceci est représenté dans l'entité par le nom de la relation suivi de deux-points et du nom de l'entité avec laquelle la relation existe, *i.e.* le co-domaine ou entité cible de la relation. Exemple générique :

```
ENTITY Nom_entité;
...
Nom_relation_1 : Nom_entité_cible;
Nom_relation_2 : type_données;
...
END_ENTITY;
```

- **Contrainte** : elle sert à affiner les entités ou ses attributs ou les interactions entre les entités du modèle représenté dans son ensemble. Les contraintes peuvent être spécifiées

[21] http://www.iai-international.org/

dans la partie déclarative de la classe ou dans la partie procédurale. Les contraintes sont aussi appelées règles. Les contraintes peuvent définir des règles locales ou globales. Les règles globales sont des contraintes qui s'appliquent entre entités dans le modèle ou s'appliquent à un ensemble spécifique d'instances d'une entité particulière. Des règles globales peuvent fonctionner comme des processus de vérification des contraintes entre entités et utilisent souvent des procédures. Elles peuvent être représentées en utilisant le constructeur `RULE`, suivi de son nom (identifiant), du mot réservé `FOR`, du nom ou noms des entités auxquelles la règle s'applique, d'un point-virgule, de la règle à expliciter et terminent par `END_RULE;` Dans la partie déclarative une contrainte est, en général, une restriction locale imposée à une propriété d'une classe. Les propriétés peuvent être obligatoires ou optionnelles. Un attribut *optionnel* est exprimé avec le mot réservé `OPTIONAL` devant le nom de la classe ou les types de données cible. (Schenck et Wilson 1994) déconseillent l'utilisation de ce type d'attribut. D'autres contraintes sur les propriétés existent, telles que : 1) les contraintes *d'unicité* (mot réservé `UNIQUE`, suivi du nom de la contrainte, de deux points et du nom de l'attribut contraint) qui spécifient que la valeur de l'attribut est unique pour chaque instance de la classe, 2) les contraintes *locales* aussi nommées *règles de domaine* (la contrainte est précédée par le mot réservé `WHERE`) qui servent par exemple à imposer les valeurs qu'un attribut peut prendre, et 3) les contraintes *d'existence* (la contrainte est précédée par le mot réservé `INVERSE`) qui représentent la notion de dépendance entre les instances : l'existence d'une instance d'une entité (classe) dépend de l'existence d'une instance d'une autre classe qui lui est liée. Cette contrainte est posée sur une propriété déclarée dans l'entité. Les contraintes d'unicité et de domaine sont des règles locales dû à l'emplacement où elles sont déclarées, *i. e.* dans une entité, et non pas à cause de leur portée. Leur portée reste valide dans toute la base d'informations. Cet exemple générique englobe les quatre contraintes déclaratives :

```
ENTITY Nom_entité;
...
Nom_relation_1 : OPTIONAL Nom_entité_cible_1;
Nom_relation_2 : Nom_entité_cible_2;
Nom_relation_3 : LIST [1:?] OF Nom_entité_cible_3;
Nom_relation_4 : LIST [1:?] OF Nom_entité_cible_4;
UNIQUE
Unique_Nom_étiquette : Nom_relation_2;
WHERE
Where_Nom_étiquette : (HiIndex(SELF.Nom_relation_3) =
(HiIndex(Self.Nom_relation_4));
INVERSE
```

```
    Nom_relation_dépendante : nom_entité_liée FOR nom_relation_liée;
    END_ENTITY;
```

`LIST [1:?]` représente une collection d'instances ordonnées de la classe identifiée par `Nom_entité_cible`, dont la limite supérieure est indéfinie puisque représentée par un point d'interrogation. Dans la contrainte locale `WHERE`, la fonction `HiIndex` renvoie le nombre d'éléments de l'instance. Cette contrainte indique que les deux listes doivent avoir le même nombre d'éléments. `SELF` indique que la référence, *i.e.* `Nom_relation_3` et `Nom_relation_4` dans la contrainte, concerne l'entité déclarée.

2.1.1.1 Application au domaine de la construction

Le modèle IFC, *Industry Foundation Classes* (ISO/PAS 16739:2005), dont la version la plus récente est la 2x édition 3 (IFC2x3), est utilisé dans le domaine de la construction et du management du patrimoine. IFC est défini en utilisant le langage EXPRESS. Un exemple de représentation d'une entité en IFC (IAI) :

```
ENTITY IfcProduct
   ABSTRACT SUPERTYPE OF (IfcProxy)
   SUBTYPE OF (IfcObject);
       ObjectPlacement : OPTIONAL IfcObjectPlacement;
       Representation : OPTIONAL IfcProductRepresentation;
   INVERSE
       ReferencedBy : SET OF IfcRelAssignsToProduct FOR RelatingProduct;
   WHERE
    WR1 : (EXISTS(Representation) AND EXISTS(ObjectPlacement)) OR
(EXISTS(Representation) AND
(NOT('IFCREPRESENTATIONRESOURCE.IfcProductDefinitionShape' IN
TYPEOF(Representation)))) OR (NOT(EXISTS(Representation)));
END_ENTITY;
```

La classe `IfcProduct` permet de définir des produits de construction à incorporer dans un projet d'AEC/FM (*Architecture, Engineering and Construction/Facilities Management*). L'attribut `ObjectPlacement` concerne le placement d'un produit dans l'espace, qui peut être absolu (relatif aux système de coordonnées) ou relatif (relatif au placement d'un autre produit). Ceci est défini par les sous-classes de `IfcObjectPlacement` qui incluent des informations de placement. L'attribut `Representation` établit une relation avec la classe `IfcProductRepresentation`. Cette dernière permet de définir, à travers ses sous-classes, des représentations géométriques du

produit. L'attribut `ReferencedBy`, sur lequel est posé une contrainte d'existence, fait dépendre l'existence des instances de la présente classe de la classe liée `IfcRelAssignsToProduct`, à travers l'attribut `RelatingProduct` de cette dernière. Sachant que l'entité `IfcProductDefinitionShape`, déclarée dans le schéma `IFCREPRESENTATIONRESOURCE`, est une spécialisation de l'entité `IfcProductRepresentation` et que la fonction `TYPEOF` retourne le nom des entités qui généralisent l'argument (`Representation`), la contrainte locale `WR1` impose que si la valeur de l'attribut `Representation` est convenue alors la valeur de l'attribut `ObjectPlacement` doit aussi être fixée.

2.1.2 Langages dédiés au *World Wide Web*

La tendance croissante qui vise à permettre l'accès à des ressources sémantiques multiples via le *World Wide Web* contraint davantage l'utilisation de langages adaptés à l'échange d'informations sur le Web. Le Web Sémantique (Berners-Lee, Hendler et Lassila 2001) est une extension du web de base dont l'information porte un sens bien défini, permettant de cette manière une meilleure coopération entre ordinateurs et entre ordinateurs et personnes. Le Web Sémantique vise alors une meilleure maîtrise des contenus et non seulement de la syntaxe. Berners-Lee a proposé une architecture en couches d'un ensemble de technologies du Web Sémantique (Figure 9). Les couches RIF (*Rule Interchange Format*), *Unifying logic*, *Proof* et *Trust* ne connaissent pas encore de standards stabilisés. La Figure 10 fait le parallèle entre différents modèles et langages et le niveau d'interopérabilité qu'ils peuvent aider à appliquer. Nous allons présenter, de manière succincte, au cours de cette section des langages adoptés par le Web Sémantique et que nous utilisons dans ce travail de thèse : XML, RDF, RDF-S et OWL.

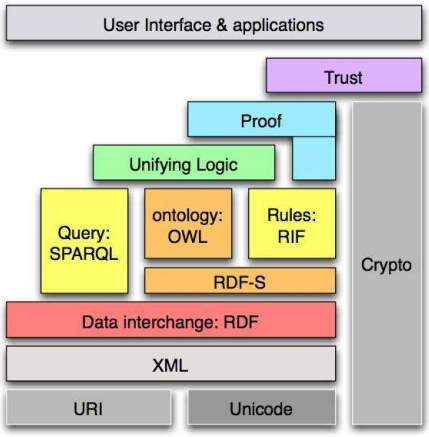

Figure 9 – Architecture en couches d'un ensemble de technologies du Web Sémantique, Copyright ©[2006] *World Wide Web Consortium, All Rights Reserved*[22].

[22] http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html#(14)

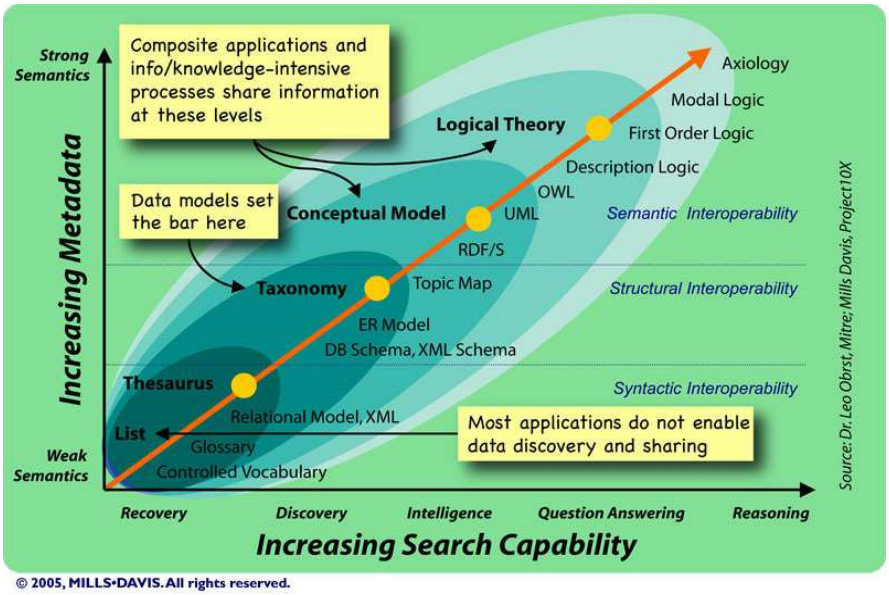

Figure 10 – Les langages, les modèles et les types d'interopérabilité[23]

XML 1.0 – *Extensible Markup Language* – (Bray *et al.* 2006a) est une recommandation du consortium W3C (http://www.w3.org) depuis le 10 février 1998 et a été remplacé par sa quatrième édition, suite à une nouvelle recommandation W3C, le 16 août 2006. Le langage XML vise à décrire l'information de manière structurée et est adapté au stockage et à l'échange de données descriptives. XML est un méta-langage à balises qui fournit une syntaxe pour les documents structurés hiérarchiquement, *i.e.* dans un arbre, mais n'impose aucune contrainte sémantique à la signification de ces documents. Des informations complémentaires et des exemples sur ce langage se trouvent à l'annexe 1.

RDF – *Resource Description Framework* – est une recommandation du W3C, dont la spécification la plus récente date du 10 février 2004 (Beckett 2004). RDF a été conçu pour fournir un langage commun pour décrire les ressources du web, de manière à que le sens représenté par celles-ci peut être traité par des applications informatiques. Les descriptions RDF ne sont pas conçues pour être présentées sur le web et lues directement par les humains. Les documents RDF sont écrits en XML. Une ressource RDF est tout élément identifiable par un URI (*Uniform Resource Identifier* - identifiant uniforme de ressource). Les ressources sont décrites plus en détail par des propriétés et les valeurs des propriétés. Les valeurs des propriétés peuvent être d'autres ressources ou des littéraux. La combinaison des trois ressources, *sujet*, *propriété* et *objet*, constitue un triplet qui forme une déclaration RDF. Ceci permet de représenter les ressources comme des

[23] http://www.semantic-conference.com/images/MD_slides/context_continuum_big.jpg

graphes, dont le sujet et l'objet sont des nœuds ; la propriété est l'arc qui relie ces nœuds, dirigé du nœud sujet vers le nœud objet. L'objet de ce triplet peut alors être un concept cible ou une valeur de la propriété ou attribut. Le concept cible peut à son tour être le sujet d'un autre triplet RDF. En annexe 1 se trouve un exemple d'une représentation en RDF.

RDF Vocabulary Description Language 1.0: RDF Schema (Brickley et Guha 2004) décrit le vocabulaire spécifique RDF à utiliser pour décrire les ressources d'une instanciation RDF. Le langage schéma RDF, *RDF-S*, fournit une structure et une sémantique du langage pour spécifier les types de classes et de propriétés. Il établit les constructeurs du langage, tel que `rdfs:domain`, `rdfs:range` et `rdfs:subClassOf`. RDF et RDF-S ne permettent pas de représenter la cardinalité d'une propriété ou de spécifier des caractéristiques des propriétés, telles que la symétrie, la transitivité, la fonctionnalité (= unicité de la valeur de la propriété), ou d'imposer des contraintes sur des classes.

OWL – *Web Ontology Language* (McGuinness et van Harmelen 2004) est un langage recommandé par le W3C depuis le 10 février 2004, spécifique à la représentation d'ontologies. OWL vise à faciliter la représentation et le traitement automatique du contenu du web en fournissant, par rapport aux langages XML, RDF et RDF-S, du vocabulaire supplémentaire et une sémantique formelle. OWL est une révision du précédent langage DAML+OIL (Connolly *et al.* 2001) et tire parti de la syntaxe XML qui permet de définir des schémas balisés, utiliser des espaces de noms et de RDF-S en ce qui concerne sa flexibilité à représenter les données. Le langage RDF-S n'est pas suffisant si nous attendons des machines qu'elles effectuent des tâches de raisonnement déductif.

Un concept en OWL peut se déclarer de plusieurs manières et se traduit par la déclaration d'une classe. Comme pour les classes RDF, chaque classe OWL est associée à un ensemble d'individus qu'on appelle *extension de classe*. Les individus dans l'extension de classe sont appelés des *instances* de la classe. Une classe a une signification intensionnelle (le concept sous-jacent) liée mais non égale à son extension de classe. Ainsi, deux classes peuvent avoir la même extension de classe et pourtant être des classes différentes (Bechhofer *et al.* 2004a). Les classes OWL sont définies par des descriptions de classe. Une description de classe définit une classe OWL en nommant la classe, ou bien en définissant l'extension de classe d'une classe anonyme. Le langage OWL distingue six types de description de classe, dont les cinq derniers décrivent une classe anonyme en exerçant des contraintes sur l'extension de classe :

- l'identificateur de classe au moyen d'un nom de classe (*i.e.* un appel d'adresse URI) ;
- l'énumération exhaustive des individus formant collectivement les instances de la classe,

- la restriction de propriété ;
- l'intersection de deux descriptions de classe ou plus ;
- l'union de deux descriptions de classe ou plus ;
- le complémentaire d'une description de classe.

OWL réutilise certains constructeurs du langage RDF-S, tels que `rdfs:domain`, `rdfs:range` et `rdfs:subClassOf`, et ajoute d'autres vocabulaires visant par exemple à représenter des combinaisons logiques entre classes (*i.e.* l'intersection, l'union, le complément et la disjonction), la cardinalité, un typage plus riche des propriétés, les caractéristiques des propriétés (par exemple, la symétrie et la transitivité) et l'énumération d'objets spécifiques. L'équivalence peut être définie entre classes ou propriétés. La disjonction peut se faire entre classes. L'égalité et l'inégalité peuvent être exprimées entre des individus, *i.e.* entre les instances.

OWL fait appelle aux logiques de description[24] (LD), en particulier en ce qui concerne le formalisme fondé sur la logique pour la représentation de connaissances et le choix des ses constructeurs. Les LD sont apparus en partie à cause du manque de sémantique formelle des premiers réseaux sémantiques et des systèmes à base de *frames* (Baader *et al.* 2003). Pour justifier les aspects formels avec une sémantique bien définie non ambiguë, une LD utilise la sémantique fondée sur la théorie des modèles, dont le but est d'expliquer la relation entre la syntaxe du langage et les modèles du domaine d'application.

Définition 1 : Un modèle en LD est constitué d'un domaine d'interprétation (symbolisé par Δ^I) et d'une fonction d'interprétation (symbolisé par $\cdot^I$). Le domaine d'interprétation est un ensemble d'objets (individus) du domaine d'application. La fonction d'interprétation est une correspondance entre noms d'individus et éléments, entre noms de classes et sous-ensembles, et entre noms de propriétés et relations binaires du domaine (Horrocks, Patel-Schneider et Van Harmelen 2003). Dit autrement, la fonction d'interprétation assigne à chaque concept atomique ou nom de concept A un ensemble tel que $A^I \subseteq \Delta^I$, et à chaque rôle atomique R une relation

[24] Les logiques de description sont une famille de langages de représentation de connaissances qui exploitent, en général, des sous-ensembles décidables de la logique de premier ordre. Pour une logique, un problème de raisonnement est décidable si une machine de Turing peut le résoudre en un nombre fini d'étapes.

binaire $R^I \subseteq \Delta^I \times \Delta^I$. Pour la LD $\mathcal{ALC}$[25] de base, la fonction d'interprétation s'étend aux descriptions de concepts comme suit :

Concept universel ou *top* : $\top^I = \Delta^I$

Concept vide ou *bottom* : $\bot^I = \varnothing$

Négation de concept : $(\neg A)^I = \Delta^I \setminus A^I$

Conjonction de concepts : $(C \sqcap D)^I = C^I \cap D^I$

Disjonction de concepts : $(C \sqcup D)^I = C^I \cup D^I$

Restriction universelle : $(\forall R.C)^I = \{a \in \Delta^I \mid \forall b.\ (a,b) \in R^I \rightarrow b \in C^I\}$

Restriction existentielle : $(\exists R.C)^I = \{a \in \Delta^I \mid \exists b.\ (a,b) \in R^I \wedge b \in C^I\}$

Définition 2 : Une collection d'ontologies, d'axiomes et de faits est cohérente par rapport aux types de données spécifiés par les références URI si et seulement si une interprétation *I* existe en ce qui concerne ces types de données, de manière que *I* satisfasse chaque ontologie, axiome et fait dans la collection (Patel-Schneider, Hayes et Horrocks 2004).

L'avantage de formaliser la sémantique du langage de cette manière est de pouvoir utiliser des techniques de raisonnement automatique pour vérifier la cohérence de l'ontologie et pour déduire de nouvelles relations.

L'exemple suivant montre une représentation graphique partielle (Figure 11) et la représentation déclarative en OWL d'un exemple sur les concepts *Actor* et *Project* et les relations associées, avec de l'information supplémentaire :

[25] *Attributive Language with Complements* est une LD qui correspond à un fragment de la logique du premier ordre obtenu par une restriction syntaxique à des formules qui contiennent deux variables. $\mathcal{ALC}$ est la LD minimale qui admet les constructeurs suivants: $\bot$ (concept le plus spécifique, qui dénote un ensemble vide), $\top$ (concept le plus général), A (nom de concept atomique), $\neg C$ (négation de concept), $C \wedge D$ (intersection de concepts composés), $C \vee D$ (union de concepts composés), $\forall R.C$ (quantification universelle complète), $\exists R.C$ (quantification existentielle complète).

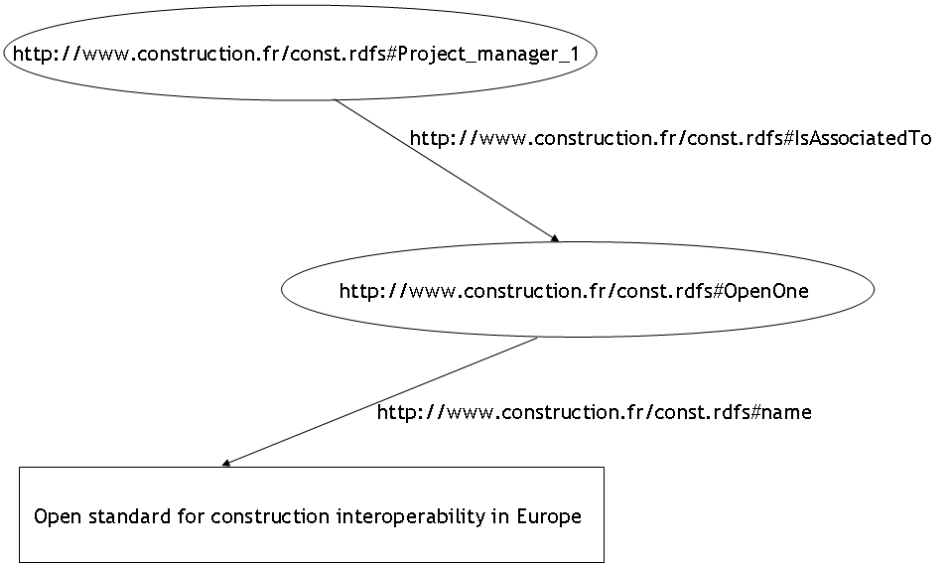

Figure 11 – Exemple de représentation d'instances des concepts *Actor* et *Project*.

```
<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY constru "http://www.construction.fr/constru#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF
     xml:base="http://www.owl-ontologies.com/const.owl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:constru="http://www.construction.fr/constru#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>
     <owl:Class rdf:about="&constru;Actor"/>
    <owl:ObjectProperty rdf:about="&constru;hasTheme"/>
    <owl:ObjectProperty rdf:about="&constru;isAssociatedTo">
        <rdfs:domain rdf:resource="&constru;Actor"/>
        <rdfs:range rdf:resource="&constru;Project"/>
    </owl:ObjectProperty>
    <owl:DatatypeProperty rdf:about="&constru;name">
        <rdfs:domain rdf:resource="&constru;Project"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:about="&constru;Project">
```

```
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:subClassOf>
         <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
                <owl:onProperty rdf:resource="&constru;hasTheme"/>
                <owl:allValuesFrom rdf:resource="&constru;Subject"/>
            </owl:Restriction>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&constru;hasTheme"/>
                <owl:someValuesFrom rdf:resource="&constru;Subject"/>
            </owl:Restriction>
       </owl:intersectionOf>
      </owl:Class>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:about="&constru;Subject"/>
    <constru:Subject rdf:ID="Construction"/>
    <constru:Project rdf:ID="OpenOne">
        <constru:name xml:lang="en"
            >Open standard for construction interoperability in
Europe</constru:name>
        <constru:hasTheme rdf:resource="#Construction"/>
    </constru:Project>
    <constru:Actor rdf:ID="Project_manager_1">
        <constru:isAssociatedTo rdf:resource="#OpenOne"/>
    </constru:Actor>
</rdf:RDF>
```

L'ontologie de l'exemple ci-dessus est cohérente parce qu'elle ne contient pas de faits contradictoires. La cohérence de cette ontologie a été validée par l'*OWL Consistency Checker* du laboratoire Mindswap (http://www.mindswap.org/2003/pellet/demo.shtml).

Dans cet exemple la partie conceptuelle et les instances sont maintenant représentées dans le même document OWL. L'ensemble de ces deux parties est, en général, appelé base de connaissances. Le constructeur `owl:ObjectProperty` sert à déclarer une relation ou propriété entre deux concepts, *i.e.* entre les individus instances de ces classes concepts. Le constructeur `owl:DatatypeProperty` sert à déclarer une relation ou attribut entre un concept, *i.e.* les individus instances de cette classe concept, et un type de données.

Dans cet exemple d'ontologie OWL, l'information supplémentaire par rapport à la représentation RDF-S spécifie des contraintes en utilisant le constructeur du langage `owl:Restriction`. Une restriction définit une classe en référençant une des propriétés des individus qui font partie des membres de la classe. La contrainte avec la restriction universelle, représentée par le constructeur `owl:allValuesFrom`, équivalente au quantificateur universel ∀ de la logique des prédicats, impose que pour tous les projets qui ont un thème, celui-ci concerne forcement un sujet. Autrement dit, pour les individus instances de la classe `Project`, s'ils ont une ou plusieurs propriétés `hasTheme`, la valeur de celle-ci est obligatoirement une instance de la classe `Subject`. Ceci peut s'écrire `Project` ⊑ ∀ `hasTheme.Subject`[26]. Cette définition n'empêche pas l'existence d'une ou plusieurs instances de `Project` qui n'ont pas la propriété `hasTheme`. En effet, une restriction universelle ne spécifie pas l'existence de la relation. Elle précise simplement que si une certaine relation existe, alors les individus cibles sont inévitablement des instances de la classe définie par la restriction (*i.e.* `Subject`). Ainsi, une définition de classe (anonyme) seulement par une restriction universelle est, en général, insuffisante puisque cette dernière ne garantit pas l'existence de la relation. Un autre mécanisme de restriction permet aussi de stipuler qu'au moins une valeur d'une propriété appartient à une certaine classe (ou type de données), *i.e.* en utilisant le constructeur `owl:someValuesFrom`, ce qui correspond au quantificateur existentiel ∃ de la logique des prédicats. Dans l'exemple, `Project` ⊑ ∃ `hasTheme.Subject` décrit les instances de `Project` qui ont au moins une propriété `hasTheme` vers une instance de la classe `Subject`. Autrement dit, cette définition stipule que tous les projets ont un thème qui est un sujet spécifique. La restriction existentielle n'empêche pas que la même propriété relie la classe `Project` et une autre classe différente de `Subject`. C'est-à-dire qu'il peut y avoir d'autres instances de la propriété qui prennent des valeurs dans d'autres classes. L'ensemble des deux définitions se traduit par l'intersection suivante : `Project` ⊑ ∀ `hasTheme.Subject` ⊓ ∃ `hasTheme.Subject`. Ce qui représente l'ensemble des projets dont au moins un des thèmes est un sujet spécifique, qui est obligatoirement un des membres de la classe `Subject`.

La propriété `hasTheme`, définie au moyen de restrictions, est dite **locale**, *i.e.* d'autres classes de concepts qui spécifient la même propriété ne sont pas contraintes par les mêmes restrictions concernant la valeur de `hasTheme` dans le cadre de l'exemple ci-dessus. Cette propriété est

[26] Nous présentons souvent les définitions en utilisant les symboles de la logique des prédicats parce que ceci génère des définitions moins verbeuses que l'équivalent en OWL. Cependant une équivalence entre les interprétations sémantiques est maintenue.

instanciée pour le projet dont l'identificateur est `OpenOne` et le sujet dont l'identificateur est `Construction`.

Cet exemple montre qu'OWL permet de définir des classes dont une propriété particulière est contrainte de manière à que toutes les valeurs de la propriété des instances de la classe doivent impérativement appartenir à une certaine classe (ou type de données). En termes de cardinalité, OWL permet aussi de stipuler une quantité exacte d'éléments dans une relation, une cardinalité maximale ou une cardinalité minimale.

OWL est divisé en trois sous-langages dont l'expressivité est croissante : **OWL Lite**, **OWL DL** et **OWL FULL**. Chacun de ces sous-langages est une extension de son prédécesseur, par rapport à ce qui peut être exprimé et ce qui peut être déduit.

OWL Lite a la plus basse complexité formelle des trois et l'expressivité minimale, *i.e.* un ensemble restreint de constructeurs du langage est disponible pour spécifier une ressource sémantique. Cependant, il est suffisant pour la représentation de classifications hiérarchiques avec des contraintes simples. Par exemple, il permet les valeurs de cardinalité 0 et 1 seulement, interdit l'utilisation des connecteurs d'union et de complément, la description d'une classe à travers l'énumération de ses individus (avec le constructeur `owl:oneOf`) ou l'énumération de types de données. OWL Lite correspond à la variante de la logique de description $\mathcal{SHIF}(\mathcal{D})$[27] (Horrocks, Patel-Schneider et Van Harmelen 2003) dont la complexité temporelle est dans le pire des cas exponentielle déterministe (ExpTIME).

OWL DL a l'expressivité maximale qui permet la complétude computationnelle (*i.e.* toutes les conclusions sont calculables) et la décidabilité (*i.e.* tous les calculs termineront dans un temps fini). OWL DL correspond à la variante de la logique de description $\mathcal{SHOIN}(\mathcal{D})$[28] (Horrocks, Patel-Schneider et Van Harmelen 2003) dont la complexité temporelle est dans le pire des cas

---

[27] Par rapport à $\mathcal{SHIF}(\mathcal{D})$, la lettre $\mathcal{S}$ désigne la logique $\mathcal{ALC}$ additionnée de la transitivité de rôles (désigné par $\mathcal{R}+$), la lettre $\mathcal{H}$ désigne l'inclusion ou hiérarchie de rôles ($R_1 \sqsubseteq R_2$), la lettre $\mathcal{I}$ les rôles inverses et $\mathcal{F}$ une contrainte de cardinalité sur les rôles (nommée fonctionnelle et représentée par ≤ 1 R, en OWL-Lite). Finalement, la lettre $\mathcal{D}$ ajoute le support des types primitifs, et par conséquent un deuxième domaine d'interprétation Δ_D^I qui représente l'ensemble des valeurs de type primitif et est disjoint de Δ^I.

[28] $\mathcal{SHOIN}(\mathcal{D})$ admet tous les constructeurs de $\mathcal{SHIF}(\mathcal{D})$ et ajoute la lettre $\mathcal{O}$ qui permet la description de concepts par l'énumération d'individus nommés (par le constructeur `owl:oneOf`) et la lettre $\mathcal{N}$ qui apporte les restrictions de cardinalité simples ($\geq$n R) et ($\leq$n R).

exponentielle non-deterministe (NExpTIME). Le langage OWL DL définit des conditions de combinaison avec RDF et impose une disjonction des classes, des propriétés, des individus et des valeurs de données. OWL DL est alors adéquat pour représenter des ontologies ayant besoin de la puissance d'expressivité tout en gardant la calculabilité.

OWL FULL est une extension syntaxique et sémantique de RDFS. Le langage OWL Full permet de mélanger arbitrairement OWL et le schéma RDF. Il n'impose pas une séparation stricte des classes, des propriétés, des individus et des valeurs de données. Donc, le pouvoir d'expression est le plus élevé, *i.e.* impose moins de contraintes, mais il n'est pas décidable. OWL FULL est approprié pour définir des ontologies qui nécessitent l'expressivité maximale et la liberté syntaxique de RDF mais sans garantie de calculabilité. Les outils de raisonnement actuels ne supportent pas toutes les caractéristiques de ce sous-langage.

OWL FULL est une extension d'OWL DL et ce dernier est une extension de OWL Lite. Toutes les ontologies qui respectent la syntaxique OWL (Lite, DL et Full) sont valides en RDF. Un document RDF est aussi un document OWL Full, mais seulement une partie restreinte des documents RDF sont des documents légaux en OWL Lite et OWL DL.

D'autres caractéristiques importantes d'OWL, qui distinguent ce dernier des LD, en le rapprochant d'un langage du Web Sémantique, sont par exemple les suivantes :

- l'utilisation de références URI pour nommer les classes et les propriétés de la même manière que dans RDF ;
- l'utilisation des propriétés d'annotation RDF ;
- l'utilisation de types de données RDF et les types de données du schéma XML pour fournir des types de données et des valeurs (par exemple, `"8"^^xsd:integer`) ;
- l'import d'une ou plusieurs ontologies dans une autre ontologie (avec le constructeur `owl:imports`). L'import est totale. Importer uniquement une partie d'une ontologie n'est pas encore possible dans cette version d'OWL.

Les langages OWL Full et OWL DL gèrent le même ensemble de structures du langage OWL. Ils se différencient par les restrictions d'utilisation sur certaines fonctionnalités et par l'emploi de fonctionnalités RDF. Par la suite nous nous focalisons sur le langage OWL DL parce que ceci est le sous-langage qui permet de spécifier une ontologie d'expressivité maximale tout en restant traitable par un moteur d'inférences.

Le Tableau 2 montre les constructeurs du langage OWL DL pour décrire des classes, des types de données, des individus et des valeurs de données. La première colonne présente la syntaxe abstraite dans le style des *frames*, la deuxième présente la syntaxe correspondante en LD et la troisième présente l'interprétation sémantique où Δ^I correspond aux individus du domaine et Δ_D^I correspond au domaine des valeurs des données (types primitifs). *C* représente le nom d'un concept et *R* représente le nom d'une relation ou rôle.

Un concept est dit *primitif* ou *partiel* ($A \sqsubseteq D$, dont A est un nom atomique) si sa description spécifie seulement une ou que des conditions *nécessaires* (par exemple, en utilisant le constructeur `owl:subClassOf` ou le constructeur `owl:disjointWith`). Si un concept A est primitif, nous pouvons conclure qu'une instance de A doit satisfaire la condition nécessaire. Mais rien ne peut être dit sur une instance quelconque, qui satisfait la condition, qu'elle est une instance de A.

Un concept est dit *défini* ou *complet* ($A \equiv D$) si sa description spécifie au moins une condition *nécessaire et suffisante* (par exemple, avec le constructeur `owl:equivalentClass`). Ainsi, si un concept A est défini en utilisant une condition nécessaire et suffisante, une instance quelconque, qui satisfait ces conditions, est une extension du concept A.

Le Tableau 3 présente les constructeurs du langage OWL DL pour définir des axiomes et des faits. Ce tableau est organisé en trois colonnes, de manière similaire au précédent tableau. L'interprétation sémantique des axiomes est donc donnée en termes de contraintes sur les modèles du domaine.

Une équivalence de relations est un axiome de propriété qui spécifie un lien d'équivalence d'une propriété à une autre. En particulier dans le paradigme OWL, la structure `owl:equivalentProperty` (Bechhoffer *et al.* 2004) déclare que deux propriétés ont la même extension de propriété. L'extension d'une propriété est l'ensemble d'instances associé à la propriété. Les instances de propriétés ne sont pas des éléments isolés mais des couples de déclarations de propriété sujet – objet. Dans le langage des bases de données relationnelles, on appellerait les instances d'une propriété les tuples d'une relation binaire (la propriété).

L'équivalence de propriété est différente de l'égalité de propriété. Les propriétés équivalentes ont les mêmes valeurs, *i.e.* la même extension de propriété, mais elles peuvent avoir des significations intensionnelles différentes, *i.e.* représenter des concepts différents. L'égalité de propriété représente autre chose et doit s'exprimer, dans le paradigme OWL, avec la structure

`owl:sameAs`. Puisque cette dernière structure impose de traiter les propriétés comme des individus, les axiomes qui utilisent `owl:sameAs` ne sont permis que dans le langage OWL Full.

Tableau 2 – Constructeurs du langage OWL DL pour décrire des classes, des types de données, des individus et des valeurs de données (Horrocks, Patel-Schneider et Van Harmelen 2003).

Abstract Syntax	DL Syntax	Semantics
Descriptions (C)		
A (URI reference)	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
`owl:Thing`	$\top$	$\texttt{owl:Thing}^{\mathcal{I}} = \Delta^{\mathcal{I}}$
`owl:Nothing`	$\bot$	$\texttt{owl:Nothing}^{\mathcal{I}} = \{\}$
`intersectionOf`(C_1 C_2 ...)	$C_1 \sqcap C_2$	$(C_1 \sqcap D_1)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap D_2^{\mathcal{I}}$
`unionOf`(C_1 C_2 ...)	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
`complementOf`(C)	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
`oneOf`(o_1 ...)	$\{o_1, \ldots\}$	$\{o_1, \ldots\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots\}$
`restriction`(R `someValuesFrom`(C))	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
`restriction`(R `allValuesFrom`(C))	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.\langle x,y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
`restriction`(R `hasValue`(o))	$R : o$	$(\forall R.o)^{\mathcal{I}} = \{x \mid \langle x, o^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\}$
`restriction`(R `minCardinality`(n))	$\geqslant n\, R$	$(\geqslant n\, R)^{\mathcal{I}} = \{x \mid \sharp(\{y.\langle x,y\rangle \in R^{\mathcal{I}}\}) \geqslant n\}$
`restriction`(R `minCardinality`(n))	$\leqslant n\, R$	$(\geqslant n\, R)^{\mathcal{I}} = \{x \mid \sharp(\{y.\langle x,y\rangle \in R^{\mathcal{I}}\}) \leqslant n\}$
`restriction`(U `someValuesFrom`(D))	$\exists U.D$	$(\exists U.D)^{\mathcal{I}} = \{x \mid \exists y.\langle x,y\rangle \in U^{\mathcal{I}} \text{ and } y \in D^{\mathbf{D}}\}$
`restriction`(U `allValuesFrom`(D))	$\forall U.D$	$(\forall U.D)^{\mathcal{I}} = \{x \mid \forall y.\langle x,y\rangle \in U^{\mathcal{I}} \rightarrow y \in D^{\mathbf{D}}\}$
`restriction`(U `hasValue`(v))	$U : v$	$(U : v)^{\mathcal{I}} = \{x \mid \langle x, v^{\mathcal{I}}\rangle \in U^{\mathcal{I}}\}$
`restriction`(U `minCardinality`(n))	$\geqslant n\, U$	$(\geqslant n\, U)^{\mathcal{I}} = \{x \mid \sharp(\{y.\langle x,y\rangle \in U^{\mathcal{I}}\}) \geqslant n\}$
`restriction`(U `maxCardinality`(n))	$\leqslant n\, U$	$(\leqslant n\, U)^{\mathcal{I}} = \{x \mid \sharp(\{y.\langle x,y\rangle \in U^{\mathcal{I}}\}) \leqslant n\}$
Data Ranges (D)		
D (URI reference)	D	$D^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}^{\mathcal{I}}$
`oneOf`(v_1 ...)	$\{v_1, \ldots\}$	$\{v_1, \ldots\}^{\mathcal{I}} = \{v_1^{\mathcal{I}}, \ldots\}$
Object Properties (R)		
R (URI reference)	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
	R^-	$(R^-)^{\mathcal{I}} = (R^{\mathcal{I}})^-$
Datatype Properties (U)		
U (URI reference)	U	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}}^{\mathcal{I}}$
Individuals (o)		
o (URI reference)	o	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
Data Values (v)		
v (RDF literal)	v	$v^{\mathcal{I}} = v^{\mathbf{D}}$

OWL ne suppose pas que les noms identifiant les entités (individus, classes et propriétés), sont uniques (au contraire de l'hypothèse de nom unique). Si deux classes ont des noms différents il reste possible d'inférer qu'elles sont les mêmes. Ceci diffère en général des applications de BD. Ainsi, dans OWL, deux entités peuvent être identiques et avoir des URI différents.

Tableau 3 – Constructeurs du langage OWL DL pour décrire des axiomes et des faits (Horrocks, Patel-Schneider et Van Harmelen 2003).

Abstract Syntax	DL Syntax	Semantics
Class(A partial C_1 ...C_n)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$	$A^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
Class(A complete C_1 ...C_n)	$A = C_1 \sqcap \ldots \sqcap C_n$	$A^{\mathcal{I}} = C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
EnumeratedClass(A o_1 ...o_n)	$A = \{o_1, \ldots, o_n\}$	$A^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
EquivalentClasses(C_1 ...C_n)	$C_1 = \ldots = C_n$	$C_1^{\mathcal{I}} = \ldots = C_n^{\mathcal{I}}$
DisjointClasses(C_1 ...C_n)	$C_i \sqcap C_j = \bot, i \neq j$	$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}}\{\}, i \neq j$
Datatype(D)		$D^{\mathcal{I}} \subseteq \Delta_{\mathbf{D}}^{\mathcal{I}}$
DatatypeProperty(U super(U_1)...super(U_n)	$U \sqsubseteq U_i$	$U^{\mathcal{I}} \subseteq U_i^{\mathcal{I}}$
domain(C_1) ...domain(C_m)	$\geqslant 1\, U \sqsubseteq C_i$	$U^{\mathcal{I}} \subseteq C_i^{\mathcal{I}} \times \Delta_{\mathbf{D}}^{\mathcal{I}}$
range(D_1) ...range(D_l)	$\top \sqsubseteq \forall U.D_i$	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times D_i^{\mathcal{I}}$
[Functional])	$\top \sqsubseteq \leqslant 1\, U$	$U^{\mathcal{I}}$ is functional
SubPropertyOf(U_1 U_2)	$U_1 \sqsubseteq U_2$	$U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$
EquivalentProperties(U_1 ...U_n)	$U_1 = \ldots = U_n$	$U_1^{\mathcal{I}} = \ldots = U_n^{\mathcal{I}}$
ObjectProperty(R super(R_1)...super(R_n)	$R \sqsubseteq R_i$	$R^{\mathcal{I}} \subseteq R_i^{\mathcal{I}}$
domain(C_1) ...domain(C_m)	$\geqslant 1\, R \sqsubseteq C_i$	$R^{\mathcal{I}} \subseteq C_i^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
range(C_1) ...range(C_l)	$\top \sqsubseteq \forall R.C_i$	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times C_i^{\mathcal{I}}$
[inverseOf(R_0]	$R = ({}^{-}R_0)$	$R^{\mathcal{I}} = (R_0^{\mathcal{I}})^{-}$
[Symmetric]	$R = ({}^{-}R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^{-}$
[Functional]	$\top \sqsubseteq \leqslant 1\, R$	$R^{\mathcal{I}}$ is functional
[InverseFunctional]	$\top \sqsubseteq \leqslant 1\, R^{-}$	$(R^{\mathcal{I}})^{-}$ is functional
[Transitive])	$Tr(R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^{+}$
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
EquivalentProperties(R_1 ...R_n)	$R_1 = \ldots = R_n$	$R_1^{\mathcal{I}} = \ldots = R_n^{\mathcal{I}}$
AnnotationProperty(S)		
Individual(o type(C_1) ...type(C_n)	$o \in C_i$	$o^{\mathcal{I}} \in C_i^{\mathcal{I}}$
value(R_1 o_1)...value(R_n o_n)	$\langle o, o_i \rangle \in R_i$	$\langle o^{\mathcal{I}}, o_i^{\mathcal{I}} \rangle \in R_i^{\mathcal{I}}$
value(U_1 v_1)...value(U_n v_n))	$\langle o, v_i \rangle \in U_i$	$\langle o^{\mathcal{I}}, v_i^{\mathcal{I}} \rangle \in U_i^{\mathcal{I}}$
SameIndividual(o_1 ...o_n)	$o_1 = \ldots = o_n$	$o_i^{\mathcal{I}} = o_j^{\mathcal{I}}$
DifferentIndividuals(o_1 ...o_n)	$o_i \neq o_j, i \neq j$	$o_i^{\mathcal{I}} \neq o_j^{\mathcal{I}}, i \neq j$

Finalement, la sémantique du langage OWL, ainsi qu'en général celle des LD, suppose un monde ouvert (*open world semantics*). Contrairement à la sémantique des BD traditionnelles, l'absence d'information représente seulement le manque d'information plutôt qu'une information négative. Ceci veut dire que ce n'est pas parce qu'une information n'est pas spécifiée explicitement qu'elle est fausse, *i.e.* l'échec de déduction d'un fait n'implique pas non plus le contraire de ce fait. Pour les LD la connaissance est admise incomplète.

2.2 Inférence et moteurs de raisonnement

Par comparaison avec les BD, l'hypothèse du monde ouvert (Baader *et al.* 2003, Fournier-Viger 2005) implique que répondre à une requête par un système bâti sur les LD nécessite d'effectuer un raisonnement logique souvent plus complexe qu'une simple recherche pour vérifier la présence d'une information, car le système doit souvent considérer plusieurs interprétations possibles. La modélisation et la consécutive représentation des connaissances d'un domaine d'application avec les LD considèrent en général deux niveaux :

- le TBox est composé de *concepts*, qui dénotent des ensembles d'individus, et de *rôles*, qui dénotent des relations binaires entre individus. Généralement, le TBox contient des axiomes terminologiques dits d'inclusion et d'égalité, respectivement de la forme $C \sqsubseteq D$ ($R \sqsubseteq S$) ou $C \equiv D$ ($R \equiv S$), dont C et D sont des concepts (et R, S sont des rôles).
- l'ABox contient un ensemble d'assertions sur les individus nommés (faits) en accord avec le TBox duquel il est instance. L'ABox contient en particulier des assertions de concepts et des assertions de rôles, représentées génériquement et respectivement par *C(a)* et *R(a,b)*. Plusieurs ABox peuvent être instanciés par le même TBox.

2.2.1 Les types d'inférences

Le raisonnement s'effectue au niveau terminologique ou factuel. Quatre principaux types d'inférences se présentent au niveau terminologique (Baader et Nutt 2003) :

- **La satisfiabilité** : un concept C d'une terminologie $\mathcal{T}$ est satisfiable si et seulement s'il existe un modèle I de $\mathcal{T}$ tel que $C^I \neq \emptyset$. On peut alors dire que I est un modèle de C.
- **La subsomption** : un concept C est subsumé par un concept D pour une terminologie $\mathcal{T}$ si et seulement si $C^I \subseteq D^I$ pour tout modèle I de $\mathcal{T}$. On peut alors écrire $C \sqsubseteq_{\mathcal{T}} D$ ou $\mathcal{T} \vDash C \sqsubseteq D$.
- **L'équivalence** : deux concepts C et D sont équivalents pour une terminologie $\mathcal{T}$ si et seulement si $C^I = D^I$ pour tout modèle I de $\mathcal{T}$. On écrit alors $C \equiv_{\mathcal{T}} D$ ou $\mathcal{T} \vDash C \equiv D$.
- **La disjonction** : deux concepts C et D sont disjoints par rapport à une terminologie $\mathcal{T}$ si et seulement si $C^I \cap D^I = \emptyset$ pour tout modèle I de $\mathcal{T}$.

Les quatre types de raisonnement ci-dessus peuvent être réduits à des problèmes de subsomption ou de satisfiabilité. Ceci entraîne que les moteurs d'inférences des LD ne nécessitent

souvent qu'un seul algorithme d'inférence pour raisonner au niveau terminologique. Ainsi les deux grandes classes d'algorithmes de raisonnement qui sont les algorithmes de subsomption de type normalisation/comparaison et les algorithmes de vérification de la satisfiabilité fondés sur les tableaux, correspondent aux manières de réduire respectivement des problèmes d'inférences à des problèmes de subsomption (Tableau 4) et de satisfiabilité (Tableau 5).

Tableau 4 – Réduction des problèmes de raisonnement d'une TBox à des problèmes de subsomption (Baader et Nutt 2003).

C est insatisfiable	*C* est subsumé par $\bot$
C et *D* sont équivalents	*C* est subsumé par *D*, et *D* par *C*
C et *D* sont disjoints	$C \sqcap D$ est subsumé par $\bot$

Tableau 5 – Réduction des problèmes de raisonnement d'une TBox à des problèmes de satisfiabilité (Baader et Nutt 2003).

C est subsumé par *D*	$C \sqcap \neg D$ est insatisfiable
C et *D* sont équivalents	$C \sqcap \neg D$ et $\neg C \sqcap D$ sont insatisfiables
C et *D* sont disjoints	$C \sqcap D$ est insatisfiable

Au niveau factuel, quatre autres principaux types d'inférences se présentent (Baader et Nutt 2003) :

- **La cohérence** : une ABox $\mathcal{A}$ est cohérente par rapport à un TBox $\mathcal{T}$, s'il existe une interprétation qui est un modèle de $\mathcal{A}$ et de $\mathcal{T}$.
- **La vérification d'instance** : une assertion *C(a)* est déduite à partir d'une ABox $\mathcal{A}$, et on écrit $\mathcal{A} \models C(a)$, si toute interprétation qui satisfait $\mathcal{A}$, *i.e.* tout modèle de $\mathcal{A}$, satisfait aussi *C(a)*.
- **La vérification de rôle** : une assertion *R(a,b)* est déduite à partir d'une ABox $\mathcal{A}$, et on écrit $\mathcal{A} \models R(a,b)$, si toute interprétation qui satisfait $\mathcal{A}$, *i.e.* tout modèle de $\mathcal{A}$, satisfait aussi *R(a,b)*.
- **La récupération d'instance** : pour une Abox $\mathcal{A}$ et un concept C d'une terminologie $\mathcal{T}$, extraire tous les individus de manière que $\mathcal{A} \models C(a)$.

2.2.2 L'algorithme des tableaux sémantiques pour les logiques de description

La majorité des systèmes de LD utilise des procédures de décision fondées sur l'algorithme des tableaux et accompagne les évolutions récentes des langages de représentation des connaissances du Web Sémantique. Ces algorithmes permettent de prouver l'existence de subsomptions entre expressions de concepts. Celles-ci sont des combinaisons de nom(s) de concept(s) et d'opérateurs ou connecteurs logiques pour former des expressions.

Les algorithmes logiques utilisent une preuve fondée sur la réfutation : C est subsumé par D s'il est possible de démontrer que l'existence d'un individu x qui appartient à l'extension du concept C ($x \in C^I$) mais non pas à celle de D ($x \notin D^I$) est incohérent logiquement. Ceci correspond au test de (in)satisfiabilité logique du concept $C \sqcap \neg D$. Autrement dit, $C \sqsubseteq D$ si et seulement si $C \sqcap \neg D$ est insatisfiable. En particulier, les algorithmes des tableaux essayent de prouver la satisfiabilité d'un concept D en construisant un *modèle*, *i.e.* une interprétation I, dans laquelle D^I n'est pas vide (Horrocks 2003). Un tableau est un graphe qui représente un tel modèle, dont les nœuds correspondent aux individus (éléments de Δ^I) et les arcs correspondent aux relations (rôles ou propriétés) entre les individus (éléments de $\Delta^I \times \Delta^I$). Un algorithme typique commencera par un individu qui satisfait D et essai de construire un tableau, en inférant l'existence d'individus additionnels ou de contraintes sur les individus. Pour cela l'algorithme (Horrocks 2003) commence par tester l'existence d'un individu qui satisfait $C \sqcap \neg D$, où C et D sont deux expressions de concepts. Il démontre ensuite qu'une tentative de construire une interprétation complète, en utilisant un ensemble de règles d'expansion (Tableau 6) mène à une contradiction logique. Si une interprétation complète et non contradictoire est retrouvée, ceci représente un contre-exemple, *i.e.* une interprétation dont un élément du domaine d'application est dans C^I mais pas dans D^I, qui réfute la relation de subsomption conjecturée. La correction et complétude de cet algorithme sont démontrées dans (Baader et Sattler 2000).

Avant le test de subsomption de la procédure des tableaux, un processus de dépliage est, en général, appliqué. Ce processus vise à éliminer les dépendances entre les définitions, par des substitutions dans les définitions de concepts. Pour cela tous les axiomes de l'ontologie doivent être des définitions acycliques et uniques. Un axiome est une définition de A s'il est de la forme $A \sqsubseteq D$ ou $A \equiv D$, où A est un nom atomique. La définition est unique si la base de connaissances (BC) ne contient pas d'autres définitions de A. La définition est acyclique si D ne fait pas référence directement ou indirectement (à travers d'autres axiomes) à A. Que ce soit pour des concepts primitifs ou définis, pendant le processus de dépliage, chaque occurrence d'un nom

de concept dans le côté droit d'une définition est substituée par la définition qu'il explicite. Ce processus d'élimination des dépendances est effectué jusqu'à ce que des noms de concepts (du côté gauche d'une définition) n'apparaissent plus aussi du côté droit d'une définition. Autrement dit, le processus récursif est effectué jusqu'à qu'il n'existe plus de cycles dans les définitions. Quand ces conditions sont satisfaites la BC est dépliée. Si ces conditions ne sont pas satisfaites, des optimisations existent (Horrocks 2003), comportant parfois des heuristiques, qui permettent d'éviter des cycles. Même si une ontologie dépliée peut augmenter exponentiellement de taille quand comparée à la taille originale, cette étape préliminaire permet de réduire le processus de raisonnement à la détermination de la satisfiabilité de concepts.

Le mécanisme d'inférence consiste en l'application d'un ensemble de règles d'expansion (Tableau 6). L'algorithme s'arrête quand le tableau est terminé, *i.e.* d'autres inférences ne sont plus possibles, ou quand des contradictions évidentes ont été révélées. En utilisant la notation de graphes étiquetés, dans un graphe orienté, chaque nœud x est étiqueté avec un ensemble de concepts ($\mathcal{L}(x) = \{C_1,..., C_n\}$), et chaque arc $\langle x, y\rangle$ est étiqueté avec un rôle ($\mathcal{L}(\langle x, y\rangle) = R$). Quand un concept C est dans l'étiquette d'un nœud x ($C \in \mathcal{L}(x)$), il représente un modèle dans lequel un individu correspondant à x est dans l'interprétation de C. Quand un arc $\langle x, y\rangle$ est étiqueté R, il représente un modèle dans lequel le couple correspondant à $\langle x, y\rangle$ est dans l'interprétation de R. Une contradiction (*clash*) est détectée quand $\{C, \neg C\} \subseteq \mathcal{L}(x)$ pour un concept C et un nœud x.

Chaque concept doit être dans sa forme normale négative (FNN), *i.e.* les négations appliquées directement aux noms de concepts seulement. Les expressions de concepts peuvent être converties en FNN en utilisant les règles de Morgan (1) et (2) avec les équivalences suivantes.

$$\neg(\exists R.C) \Leftrightarrow (\forall R.\ \neg C) \quad \textbf{(1)}$$

$$\neg(\forall R.C) \Leftrightarrow (\exists R.\ \neg C) \quad \textbf{(2)}$$

Chaque description de concept peut donc être transformée, en temps linéaire, dans sa description FNN équivalente. Ainsi, pour tester la satisfiabilité d'un concept D, un algorithme initialise un graphe (en général un arbre) contenant un nœud racine x avec $\mathcal{L}(x) = \{D\}$ et, en appliquant les règles d'expansion, développe le graphe en ajoutant des étiquettes de nœuds ou des nœuds feuilles. Un ensemble de règles d'expansion pour la LD $\mathcal{ALC}$ (Baader et Sattler 2000) est présenté dans le Tableau 6, dont C et D sont des concepts et R est un rôle. Le graphe est complètement développé quand plus aucune des règles ne peut s'appliquer. Si le graphe est

complètement développé et une contradiction n'est pas retrouvée, l'algorithme retourne *satisfiable*, sinon il retourne *insatisfiable*.

Tableau 6 – Règles d'expansion des tableaux sémantiques pour la LD $\mathcal{ALC}$.

Règle $\sqcap$	Si 1. $(C \sqcap D) \in \mathcal{L}(x)$ 2. $\{C, D\} \notin \mathcal{L}(x)$ Alors $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C, D\}$
Règle $\sqcup$	Si 1. $(C \sqcup D) \in \mathcal{L}(x)$ 2. $\{C, D\} \cap \mathcal{L}(x) = \varnothing$ Alors $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ ou $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{D\}$
Règle $\exists$	Si 1. $\exists R.C \in \mathcal{L}(x)$ 2. il n'existe pas un y de manière que $\mathcal{L}(\langle x, y\rangle) = R$ et $C \in \mathcal{L}(y)$ Alors créer un nouveau nœud y et arc $\langle x, y\rangle$ avec $\mathcal{L}(y) = \{C\}$ et $\mathcal{L}(\langle x, y\rangle) = R$
Règle $\forall$	Si 1. $\forall R.C \in \mathcal{L}(x)$ 2. il existe un y de manière que $\mathcal{L}(\langle x, y\rangle) = R$ et $C \notin \mathcal{L}(y)$ Alors $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$

Ceci est la procédure d'inférence de base. Les inférences sur les logiques plus expressives, avec des constructeurs supplémentaires et des techniques d'optimisation, ajoutent, la plus part du temps, d'autres règles d'expansion. La détection de cycles (*blocking*) est parfois nécessaire dans les pré-conditions de certaines règles d'inférence, pour pouvoir garantir la terminaison. C'est ce qui se passe dans le raisonnement avec la LD $\mathcal{SHOIQ}$.

(Horrocks et Sattler 2007) expose l'extension de l'algorithme à la LD $\mathcal{SHOIQ}$ avec des types de données simples (*i.e.* l'équivalent d'OWL-DL, avec des restrictions de cardinalité composées, dites aussi qualifiées. Autrement dit, pour le langage de représentation $\mathcal{SHOIN}$ avec $(\geq n\ R.C)$ et $(\leq n\ R.C)$). De nouvelles conditions de terminaison de l'algorithme sont proposées, relatives à la cardinalité.

Les moteurs d'inférences implantent un ensemble de techniques d'optimisation visant à améliorer le temps de calcul et à fournir des services supplémentaires, comme par exemple le

dépliage paresseux (*lazy unfolding*), la normalisation, la simplification, l'absorption d'individus, etc. (Sirin *et al.* 2007, Horrocks 2003).

2.2.3 Les moteurs d'inférences

Les moteurs d'inférences fournissent des services pour retrouver des conséquences déductives implicites à partir des connaissances explicitement représentées dans les ontologies. En général, les services standards fournis par les moteurs d'inférences sont les suivants :

- **La vérification de la cohérence** assure que l'ontologie ne contient pas des faits contradictoires. La définition formelle de la cohérence d'une ontologie (Patel-Schneider, Hayes et Horrocks 2004) considère qu'un ensemble d'ontologies en OWL, d'axiomes et de faits est cohérent par rapport à l'application de type de données D[29] s'il existe une interprétation I par rapport à D qui satisfait chaque ontologie, axiome et fait dans l'ensemble.
- **La satisfiabilité de description de concept** détermine si une classe de concept peut avoir des instances. Si une classe est insatisfiable, la définition d'une instance de cette classe provoquera l'incohérence de toute l'ontologie.
- **La classification** calcule les relations hiérarchiques entre toutes les classes de concepts nommées pour créer l'organisation hiérarchique complète de l'ontologie. Ceci revient à placer une expression de concept à la bonne place dans la taxonomie hiérarchique de concepts. Ce service s'accomplit à travers la vérification de l'existence d'une subsomption entre l'expression de concept en cours et celle de chaque concept. Chaque concept sera placé entre le concept le plus spécifique qui le subsume et le concept le plus générique qu'il subsume. La hiérarchie des classes peut être utilisée, par exemple, pour récupérer toutes les sous-classes d'une classe ou seulement les sous-classes directes.
- **La réalisation** retrouve les classes de concepts les plus spécifiques ou toutes les classes auxquelles un individu appartient, par rapport à la relation de subsomption. Autrement dit, la réalisation calcule les types de chaque individu. Ce service peut être exécuté seulement après la classification, puisque les types sont définis par rapport à une hiérarchie de classes.

[29] Une application de types de données D est une application partielle des appels d'adresses URI aux types de données, qui relie *xsd:string* et *xsd:integer* aux types de données appropriés du schéma XML.

Nous présentons très brièvement trois moteurs de raisonnement : FaCT++[30] (Tsarkov et Horrocks 2004), RacerPro[31] (Möller et Haarslev 2003) et Pellet[32] (Parsia, Sirin et Kalyanpur 2005). Plusieurs autres moteurs de raisonnement existent[33]. Nous avons décidé d'en analyser que les trois moteurs ci-dessous parce qu'ils sont parmi ceux qui assurent la décidabilité et présentent une meilleure appréciation en ce qui concerne le passage à échelle (Fournier-Viger 2005) et donc une performance plus adaptée à une utilisation industrielle.

FaCT++ (version 1.1.4) est une évolution de FaCT (Horrocks 1998), ce dernier n'étant plus maintenu. FaCT++ est un logiciel programmé en C++, distribué selon la licence publique générale GNU v2[34]. C'est un moteur pour la LD $\mathcal{SHOIQ}$ avec des types de données simples (*i.e.* l'équivalent d'OWL-DL, avec des restrictions de cardinalité composées, dites aussi qualifiées). Depuis décembre 2006, il supporte aussi le langage $\mathcal{SROIQ}$[35] (*i.e.* pour OWL 1.1[36]). Il implante une procédure de décision fondée sur l'algorithme des tableaux pour des TBox génériques (subsomption, satisfiabilité, classification) et fournit un support incomplet pour l'ABox (récupération). Il supporte l'interface de programmation (API) DIG[37] (Bechhofer 2003).

RacerPro (version 1.9) est un moteur commercial, mais des licences de recherche et d'évaluation sont disponibles. Le moteur est fondé sur Lisp pour la LD $\mathcal{SHIQ}(\mathcal{D})^-$ (*i.e.* l'équivalent d'OWL-DL avec des restrictions de cardinalité qualifiées, mais sans support pour l'énumération des individus pour définir une classe, *nominals*). Il implante aussi une procédure de décision fondée sur l'algorithme des tableaux pour des TBox génériques (subsomption, satisfiabilité, classification) et

---

[30] http://owl.man.ac.uk/factplusplus

[31] http://www.racer-systems.com/

[32] http://pellet.owldl.com/

[33] Uli Sattler maintient une liste de moteurs d'inférences très complète à http://www.cs.man.ac.uk/%7Esattler/reasoners.html

[34] http://fsffrance.org/gpl/gpl-fr.fr.html

[35] $\mathcal{SROIQ}$ représente les fondations logiques de OWL version 1.1 et admet, en plus des constructeurs de OWL-DL, les restrictions de cardinalités qualifiées ($\geq n\ R.C$ et $\leq n\ R.C$), l'inclusion de rôles complexes ($Ro\mathcal{S} \subseteq R$, $Ro\mathcal{S} \subseteq S$) et quelques autres caractéristiques additionnelles concernant les types de données.

[36] http://www.webont.org/owl1.1/

[37] L'interface de programmation DIG est un standard XML permettant à des outils clients d'interagir avec des moteurs LD. Voir http://dig.sourceforge.net/

pour des ABox (récupération et réponse aux requêtes du langage *new Racer Query Language*, nRQL). Il supporte l'API OWL[38] (Bechhofer, Lord et Volz 2003) et l'API DIG.

Pellet (version 1.5.0) est un moteur dont le code est ouvert et libre, programmé en Java, distribué selon la licence MIT[39]. Pellet a été le premier moteur natif à permettre des inférences sur des ontologies représentées en OWL-DL, y compris des inférences avec les individus énumérés dans des classes, les *nominals*. Plus récemment (Sirin *et al.* 2007), il a été étendu pour permet des inférences pour le langage DL $\mathcal{SROIQ}(\mathcal{D})$ avec des types de données simples. Autrement dit, Pellet permet des inférences pour OWL 1.1, à l'exception des types de données d'arité *n*. Il implante une procédure de décision fondée sur l'algorithme des tableaux pour des TBox génériques (subsomption, satisfiabilité, classification) et pour des ABox (récupération et réponse aux requêtes conjonctives). Il supporte les API OWL, DIG et Jena (Wilkinson *et al.* 2003).

2.3 Interfaces de programmation pour les langages du Web Sémantique

Les interfaces de programmation pour les langages du Web Sémantique sont des bibliothèques de code qui permettent de manipuler et de gérer des ontologies représentées dans ces langages. L'utilisation de ces API permet aux développeurs de s'affranchir de la programmation liée à l'analyse syntaxique (*parsing*) et à l'écriture (sérialisation) de syntaxes spécifiques. L'expérience (McBride 2001, Bechhofer, Lord et Volz 2003) montre que les développeurs d'applications peuvent interpréter les spécifications des langages de représentation de manières différentes et confondre par exemple les espaces de noms et la portée de certains constructeurs. L'utilisation d'une API facilite la tâche et permet aux développeurs de se concentrer sur les opérations sur des ontologies à un niveau supérieur d'abstraction.

Les API d'accès aux systèmes de bases de connaissances se sont traditionnellement centrés dans des protocoles, tels *que Open Knowledge Base Connectivity*[40] (OKBC) et *Generic Frame Protocol*[41] (GFP). Ces approches, fondées sur des protocoles, adoptent une architecture client-serveur pour le développement d'applications. Une autre approche, plutôt fondée sur des composants réutilisables, similaires à *Simple API for XML* (SAX), est apparue. C'est le cas des API Jena et OWL, que nous présentons brièvement ci-dessous.

---

[38] http://sourceforge.net/projects/owlapi

[39] http://www.opensource.org/licenses/mit-license.php

[40] http://www-ksl.stanford.edu/software/OKBC/

[41] http://www.ai.sri.com/~gfp/

2.3.1 L'interface de programmation Jena

Jena – a Semantic Web Framework for Java[42] (Wilkinson *et al.* 2003), fournit une API de code ouvert pour :

- traiter (créer, modifier...) les ontologies représentées selon les spécifications des langages RDF et OWL ;
- lire et écrire dans les formats RDF/XML, N3 et N-Triples ;
- gérer le stockage en mémoire et en persistance (en couplage avec un SGBD) ;
- supporter la gestion de plusieurs ontologies.

Jena propose aussi un moteur pour le langage de requêtes RDF SPARQL[43] et une configuration pour l'interfaçage avec des moteurs d'inférences fondés sur les LD. L'approche de représentation dans Jena est fondée sur la notion de triplet RDF (sujet, prédicat, objet), puisqu'elle implante la spécification syntaxique du modèle RDF (McBride 2001). L'API Jena est moins adaptée à une représentation orientée axiome, spécifié par le langage OWL 1.1, comme `SubClassOf(ClsA ClsB)`. Cette API traite les représentations OWL, mais en analysant un graphe RDF. Les constructeurs d'arité *n*, tels que l'intersection et l'union, sont cassés en plusieurs triplets dans le graphe RDF, ce qui rend plus compliqué leurs manipulation.

2.3.2 L'interface de programmation OWL API

L'API OWL est programmée en Java et est une implantation pour le langage OWL W3C. Le code de l'API est ouvert et disponible avec la licence LGPL[44]. La version initiale de cette API (Bechhofer, Lord et Volz 2003) est aussi connue par WonderWeb API et est conforme à la spécification OWL 1.0. La version 2.1.1 de cette API, plus récente, se concentre sur la spécification OWL 1.1 (Horridge, Bechhofer et Noppens 2007), ce qui englobe toujours OWL-Lite, OWL-DL et quelques éléments d'OWL-Full. Les caractéristiques de cette API sont les suivantes :

- un modèle interne de représentation reposant sur la syntaxe abstraite d'OWL (Patel-Schneider, Hayes et Horrocks 2004), contrairement à l'API Jena, qui se conforme à une représentation en graphes RDF. Ainsi, dans le modèle de représentation interne de l'API OWL, une ontologie contient des axiomes et des faits, qui à leur tour fournissent

[42] http://jena.sourceforge.net/

[43] http://www.w3.org/TR/rdf-sparql-query/

[44] http://www.gnu.org/licenses/lgpl.html

l'information concernant les classes, les propriétés et les individus dans l'ontologie. Les annotations sont aussi représentées comme des axiomes.

- la lecture, l'analyse et l'écriture des syntaxes OWL : représentation en triplets RDF, syntaxe au style *frame* et syntaxe orientée axiomes.
- l'intégration des analyseurs syntaxiques (parseurs) pour la syntaxe fonctionnelle d'OWL 1.1 (Motik, Patel-Schneider et Horrocks 2006), OWL 1.1 XML, RDF/XML et pour la syntaxe OWL de Manchester (Horridge *et al.* 2006), entre autres.
- la gestion des modifications et la possibilité de suivre la trace des changements et le stockage de méta-données associées aux modifications, comme par exemple l'identification de l'utilisateur qui a demandé une modification.
- la possibilité de détecter le sous-langage d'OWL 1.0 et les fragments d'OWL 1.1 des ontologies.
- le support de la gestion de plusieurs ontologies.
- l'interfaçage pour des moteurs d'inférences, au niveau des Tbox et aussi pour requêter sur les individus. Cette dernière fonctionnalité est particulièrement intéressante pour retrouver, par exemple, les individus qui sont liés à travers une propriété donnée, les valeurs des propriétés pour un individu, etc.

La représentation syntaxique fondée sur les axiomes, qui est d'ailleurs celle suggérée par la feuille de route des spécifications d'implantations d'OWL 1.1, est plus simple à traiter et moins verbeuse que les représentations RDF ou celles fondées sur les *frames*. Une syntaxe dans le style *frames* est par exemple `Class(CLsA partial ClsB)` qui est équivalente dans la représentation orientée axiomes à `SubClassOf(ClsA ClsB)`.

2.4 La logique floue

La logique floue (Zadeh 1965) étend la logique *nette* et duale du tout ou rien, en autorisant la modélisation combinée de plusieurs informations imprécises et incertaines. Ce type de modélisation est en général utilisé dans le contrôle et la simulation des systèmes automatiques et dans la robotique.

Dans une modélisation par la logique floue, il est nécessaire de définir en premier lieu un ensemble de variables linguistiques. Chaque variable linguistique correspond à un ensemble flou. Un ensemble flou est défini par sa fonction d'appartenance. Un élément x appartient à un ensemble A, avec un degré d'appartenance $\mu_A(x)$ compris entre 0 et 1. On nomme *fuzzification* la modélisation qui consiste à pondérer les fonctions d'appartenance des entrées aux divers sous-

ensembles. Les sous-ensembles ne sont pas exclusifs, *i.e.* ils partagent des valeurs. Les variables linguistiques, les fonctions d'appartenance et les chevauchements des formes des sous-ensembles peuvent être choisis librement. Cependant il convient de respecter la convexité de chaque sous-ensemble et le chevauchement partiel des sous-ensembles consécutifs afin d'éviter les zones indéterminées (Guillaume 2005). Ainsi, une variable linguistique peut appartenir à plusieurs sous-ensembles dans les zones de chevauchement. Son appartenance partielle à chaque sous-ensemble est alors caractérisée par un degré d'appartenance quantifié.

Les valeurs floues obtenues en sortie de la phase de fuzzification sont ensuite utilisées dans le cadre d'un processus d'inférence floue (Figure 12). Ce processus est composé de trois phases. La première, la phase de fuzzification, transforme donc les valeurs numériques en degrés d'appartenance aux différents ensembles flous de la partition. La deuxième phase consiste à faire appel à un moteur d'inférences flou, en initiant le processus à l'aide d'un ensemble de règles. Ces règles sont, en général, une combinaison de règles du type ***Si*** *j'ai un ensemble d'informations* ***Alors*** *j'en tire telle conclusion*. La sortie de ce processus est un ensemble de variables floues. Enfin, la phase de *défuzzification* permet, si nécessaire, d'inférer une valeur nette, qui quantifie les correspondances sémantiques, à partir du résultat de l'agrégation des règles.

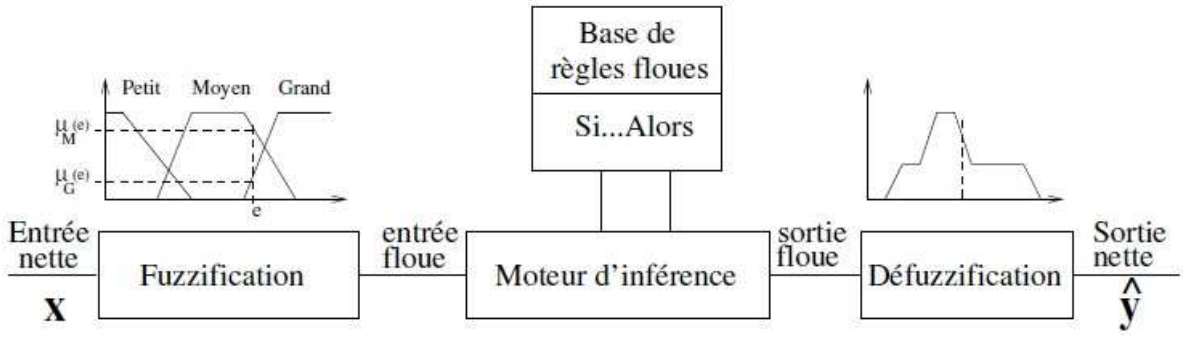

Figure 12– Composants d'un système d'inférence floue (Guillaume 2005).

Dans ces travaux de thèse nous proposons une méthode de fuzzification adaptée à notre problème, *i.e.* la phase de modélisation de l'information visant à l'évaluation de la qualité des correspondances sémantiques en utilisant la logique floue (*cf.* section 3.2.5).

2.5 Synthèse

Ce chapitre présente les langages de représentation de connaissances qui sont utilisés au cours de cette thèse : EXPRESS, XML et OWL. Le langage RDF(S) est aussi présenté parce qu'il recommande plusieurs constructeurs de base du langage OWL. Ensuite, nous analysons plusieurs types d'inférences fondés sur les logiques de description. L'algorithme des tableaux sémantiques utilisé dans le processus d'inférence décidable est présenté, ainsi que les moteurs d'inférences

FaCT++, Racer et Pellet qui implantent cet algorithme. Nous présentons deux interfaces de programmation pour les langages du Web Sémantique : l'API Jena et l'API OWL. Finalement, nous faisons une brève présentation de la logique floue. Le chapitre suivant présente la méthodologie de découverte de correspondances sémantiques proposée et mise en œuvre dans ce travail.

Partie II - Découverte de correspondances sémantiques, application et mise en œuvre

Chapitre 3 - Méthodologie de découverte de correspondances sémantiques

L'objectif de ce chapitre est de présenter la solution que nous proposons pour améliorer à l'interopérabilité sémantique des ressources de connaissances. Nous proposons une méthodologie en détaillant ses diverses phases.

Ce chapitre est le noyau de la thèse car ici nous proposons notre méthodologie de travail pour aider à répondre au manque d'interopérabilité sémantique entre des ressources de connaissances d'un domaine d'application. La première section du chapitre positionne la problématique de la thèse par rapport à l'état de l'art présenté aux chapitres précédents. Ensuite nous présentons le plan méthodologique de notre travail. Enfin, nous développons chacune des phases de la méthodologie proposée.

3.1 Problématique de thèse

Dans le premier chapitre nous avons analysé différentes ressources sémantiques afin de mettre en évidence leurs hétérogénéités à divers niveaux. Pour améliorer l'interopérabilité sémantique entre les ressources sémantiques il s'avère nécessaire de trouver des ponts entre les sens explicités par les ressources sémantiques. Pour cela nous avons présenté l'état de l'art des méthodes d'alignement qui peuvent aider à relier les éléments (concepts ou relations) d'ontologies et de taxonomies. Cependant nous constatons l'absence d'une approche intégrée, adaptée à l'alignement de ressources hétérogènes. D'autre part, il nous semble possible d'approfondir l'étude des méthodes sémantiques, en particulier quant aux propriétés, peu traitées jusqu'ici.

Dans ces travaux de thèse nous faisons l'hypothèse suivante : une méthode sémantique d'alignement peut aider à trouver des liens entre les concepts des ressources sémantiques. Les méthodes sémantiques permettent de traiter le sens associé aux ressources sémantiques, quand il est explicité en logique formelle. Tel que mis en évidence aux chapitres précédents, l'avantage d'une méthode sémantique est le traitement de la sémantique formelle, ce qui permet de lever les ambiguïtés liées à l'interprétation du contenu des ressources sémantiques, à l'aide d'un moteur d'inférences. Ceux-ci aident à trouver des liens entre les concepts des ressources sémantiques, *i.e.* des correspondances sémantiques.

3.2 Méthodologie proposée

La méthodologie proposée (Ferreira da Silva *et al.* 2006c, Ferreira da Silva *et al.* 2006b, Ferreira da Silva *et al.* 2005) contient les phases suivantes (Figure 13) :

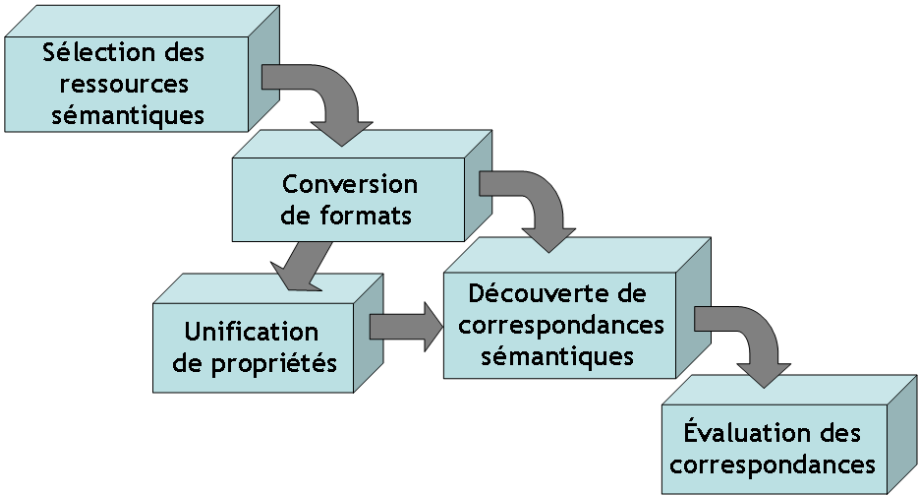

Figure 13 – Méthodologie proposée dans ces travaux de thèse.

1. **Sélection des ressources sémantiques** : la phase initiale de l'étude consiste à analyser un ensemble de ressources sémantiques du domaine d'application, *i.e.* la construction en Europe. Pour cela nous avons pris en compte un ensemble de critères d'analyse.
2. **Conversion de formats** : ces ressources sémantiques sont hétérogènes à différents niveaux, y compris au niveau syntaxique. Convertir les différents formats d'origine en un seul format commun rend plus facile la tâche suivante, puisque celle-ci peut alors se focaliser sur les aspects sémantiques. Pour cela il s'est avéré nécessaire d'opérer une homogénéisation syntaxique.
3. **Découverte de correspondances sémantiques** : cette phase consiste en un processus qui utilise les services d'un moteur d'inférences fondé sur les logiques de description (LD), pour comparer les définitions des ressources sémantiques représentées en logique formelle. De cette manière le moteur produit des correspondances sémantiques non ambiguës entre les concepts des ressources sémantiques.
4. **Unification de propriétés** : il s'agit d'une étape supplémentaire et intermédiaire entre la phase de conversion de formats et celle de découverte de correspondances sémantiques. Cette étape établit des équivalences entre propriétés de définitions de concepts différents. Cette proposition vise à générer de nouvelles correspondances sémantiques entre les concepts, et à inférer des instances entre concepts lors du deuxième passage à la phase de découverte de correspondances sémantiques.

5. **Évaluation des correspondances** : cette dernière phase a besoin de l'intervention des experts du domaine d'application, puisque ce sont eux qui peuvent vérifier que les correspondances sémantiques produites sont cohérentes avec les connaissances modélisées par les ressources sémantiques et ainsi valider les résultats. Nous proposons une méthode de modélisation de connaissances du domaine pour quantifier les correspondances sémantiques non équivalentes produites, et ceci en utilisant la logique floue, de manière à aider les experts dans la tâche d'évaluation.

Les sous-sections suivantes détaillent chacune des phases de la méthodologie proposée.

3.2.1 Analyse et sélection des ressources sémantiques

Plusieurs ressources sémantiques existent et se focalisent sur les termes d'une partie du domaine d'application étudié pour modéliser le domaine de la construction. Évidemment, chaque ressource ne conceptualise pas tout le domaine d'application, mais se focalise plutôt sur un sous domaine. Par exemple, dans le domaine de la construction, l'ontologie e-Cognos (Wetherill *et al.* 2002) conceptualise le domaine de la gestion des connaissances de la construction, alors que le schéma XML *bcBuildingSpecification* (Van Rees, Tolman et Beheshti 2002) a été conçu pour soutenir la création de catalogues de produits de la construction. Par ailleurs, les ressources sémantiques sont également hétérogènes, au niveau syntaxique, puisqu'elles sont représentées dans des formats différents. Pour cela, la sélection des ressources sémantiques est soumise à une analyse selon les critères suivants, qui ont été définis pendant le projet FUNSIEC (Barresi *et al.* 2004) :

- Langue(s) naturelle(s) utilisée(s) : français, anglais, allemand, etc.
- Langage de représentation des connaissances utilisé pour encoder la ressource : XML, RDF(S), EXPRESS, DAML+OIL, OWL, etc.
- Caractéristiques d'interopérabilité : par exemple est-ce que l'outil logiciel de gestion de la ressource sémantique fournit un ou plusieurs formats d'import/export, permettant de traiter la ressource sémantique par d'autres outils.
- Domaine(s) de connaissance représenté(s) : dans le domaine d'application la ressource sémantique peut représenter le domaine de manière général ou se dédier à un sous-domaine seulement.
- Type(s) d'application(s) connu(s) pour la ressource : contexte dans lequel la ressource sémantique est utilisée, par exemple, pour le management des connaissances du domaine,

pour la génération et publication de catalogues de produits, pour l'échange et le partage de connaissances, etc.

- Disponibilité de la ressource : la ressource sémantique est d'utilisation libre ou non.
- Modèle et méta-modèle sous-jacents à la représentation de la ressource : disponibilité des modèle et méta-modèle de la ressource sémantique qui permettent de mieux comprendre l'objectif de la représentation et les connaissances qu'y sont représentées.
- Méthode de conceptualisation des connaissances : par exemple, dans certaines ressources sémantiques une relation est une entité à part entière représentée de la même manière qu'un concept (c'est le cas des ressources sémantiques IFC ou ISO 12006-3, *cf.* section 4.2) ; dans d'autres, une relation ou propriété est un lien entre les concepts et n'existe pas de manière autonome, *i.e.* sans faire partie d'une définition de concept.

La section suivante est dédiée à la deuxième phase de la méthodologie proposée, *i.e.* traite les problèmes liés à la conversion syntaxique entre les langages que nous avons utilisé.

3.2.2 Homogénéisation syntaxique

Une fois les ressources sémantiques sélectionnées, il est nécessaire de réduire les hétérogénéités syntaxiques parce que les ressources sémantiques sont représentées en des formats inhomogènes. Pour cela nous avons choisi comme format cible commun le langage OWL. Ce dernier a été adopté en raison de son expressivité et de la sémantique explicite qu'il propose, permettant la déduction automatique. La conversion des méta-schémas et des schémas des ressources sémantiques vers un format unique facilite le traitement postérieur de l'hétérogénéité sémantique visant à détecter des correspondances entre ressources sémantiques. Ainsi, le processus de conversion entre formats produit les règles de traduction à partir de chaque format initial vers le format OWL. Cette phase a nécessité une étude de la sémantique des formats et l'identification des éléments syntaxiques utilisés par les formats originaux qui correspondent aux éléments syntaxiques d'OWL (Figure 14). Ce processus de conversion a reçu en entrée les méta-schémas et schémas des ressources sémantiques dans leurs formats initiaux. Les experts du domaine d'application ont joué un rôle stratégique dans la phase d'analyse sémantique et de conversion syntaxique. Ils ont aidé à analyser les méta-schémas et schémas pour que nous puissions en groupe créer un ensemble de règles qui ont permis de convertir chaque entité de leur format original vers la syntaxe OWL. Ces règles de traduction ont ensuite été utilisées par un

convertisseur fondé sur le compilateur JavaCC[45], qui a généré les traducteurs syntaxiques. Ceux-ci prennent en entrée les méta-schémas et schémas des ressources sémantiques pour les transformer, de manière automatique, en OWL.

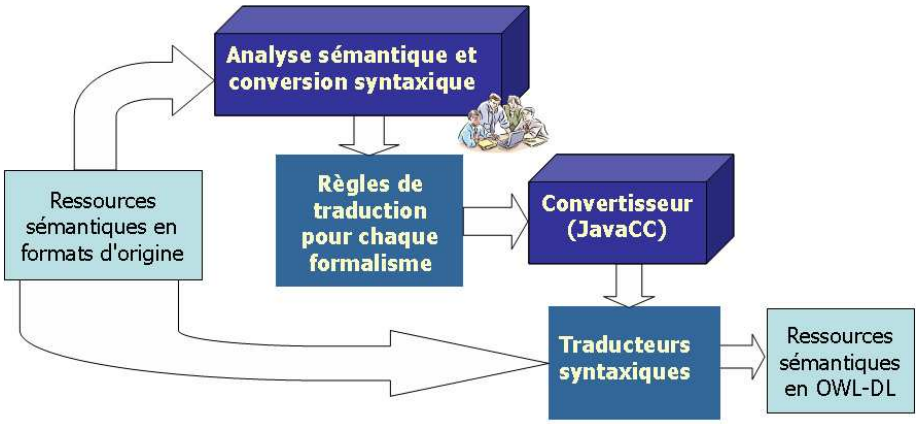

Figure 14 – Processus de conversion syntaxique.

Plus de détails sur les ressources sémantiques sélectionnées sont fournis dans le chapitre 4. Dans cette section nous nous focalisons sur leurs formats de représentation. De ce fait, les formats initiaux pris en compte sont, pour les méta-schémas des ressources sémantiques, EXPRESS et XMI[46], et pour les schémas des ressources sémantiques, XSD et DAML+OIL. Cette étape de conversion est menée de façon à minimiser la perte sémantique lors de l'obtention du contenu des ressources dans un format commun. D'autres travaux ont traité la conversion d'UML vers OWL (par exemple, Djurić 2004), de XML vers OWL (par exemple, Bohring et Auer 2005) et de XSD vers OWL (Gil, García et Delgado 2005). La conversion de DAML+OIL vers OWL est assez aisée parce que le paradigme de ces deux langages de représentation est le même. En particulier OWL-DL hérite de DAML+OIL pour beaucoup de concepts et de constructeurs. Dans (Bechhofer *et al.* 2004b) se trouve une explication des différences entre les deux langages, ainsi qu'un tableau des correspondances syntaxiques entre les constructeurs de ces langages. De ce fait, nous ne détaillons ensuite que la conversion du langage EXPRESS vers OWL-DL.

[45] https://javacc.dev.java.net/

[46] *XML Metadata Interchange* (XMI) est un standard de l'*Object Management Group* (http://www.omg.org/technology/documents/formal/xmi.htm) pour l'échange de meta-données en utilisant XML. XMI est habituellement utilisé pour l'échange de modèles au format UML.

3.2.2.1 Conversion EXPRESS vers OWL-DL

Le Tableau 7 présente dix exemples de traduction entre les constructeurs des langages EXPRESS, DL et OWL-DL. La troisième colonne, intitulée signification, explique la sémantique des constructeurs et dans certains cas les choix effectués. Le constructeur EXPRESS `ONEOF` spécifie des entités (classes) mutuellement exclusives. En OWL-DL, l'utilisation du constructeur `owl:oneOf` impose que les valeurs cibles constituent une énumération des individus instances des classes. Si une énumération de classes est utilisée à la place d'une énumération des individus instances des classes, il s'agit d'une ontologie OWL-FULL. Un moteur d'inférences détectera alors une inconsistance au niveau de la définition du concept qui contient une énumération de noms de classes. Pour ne pas perdre de la sémantique lors de la conversion et puisque les définitions en EXPRESS spécifient l'énumération de classes mutuellement exclusives, nous préférons utiliser le constructeur `owl:disjointWith`. Un exemple est présenté dans la ligne 5 (Tableau 7).

La traduction d'un attribut de l'un des deux langages à l'autre met en évidence la différence des paradigmes EXPRESS et OWL. Le langage EXPRESS spécifie qu'un attribut est défini dans le cadre d'une entité. Un attribut qui a le même nom mais qui est défini dans une autre entité est un attribut différent : il relie des entités différentes.

Par contre, OWL permet de spécifier pour les attributs/propriétés une identité plus indépendante, permettant de définir des axiomes de propriétés[47]. En particulier, les restrictions globales contraignent toute l'ontologie. Les restrictions de *domain* et *range* sont donc associées à la propriété indépendamment des descriptions de classes sur lesquelles elles sont appelées à s'appliquer. Un concept qui réutilise cette propriété dans sa définition est restreint par la source et la cible définies auparavant pour la propriété. Si le même nom de propriété est utilisé pour définir à nouveau d'autres sources ou cibles, ces multiples sources et cibles de la propriété doivent être interprétées par défaut comme l'intersection des sources (Figure 15) et l'intersection des cibles respectivement (Bechhoffer et. al. 2004a). Il s'agit alors d'une restriction aux individus appartenant à l'intersection des descriptions des classes. Cependant, si un concepteur d'ontologie veut signifier que plusieurs classes agissent comme une seule source ou une seule cible, il doit spécifier l'union et utiliser pour cela le constructeur `owl:unionOf`.

[47] Un axiome de propriété définit les caractéristiques d'une propriété. Dans sa forme minimale, un axiome de propriété définit simplement l'existence d'une propriété.

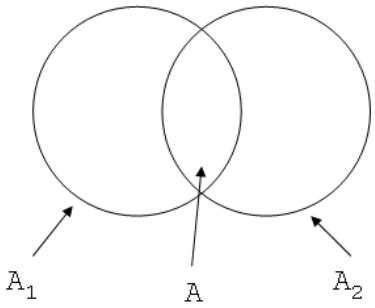

Figure 15 – Intersection A des concepts A_1 et A_2 ($A_1 \sqcap A_2$).

La sémantique d'une intersection de classes de concepts correspond à une classe anonyme qui est décrite par une sous-classe de chacune des classes qui participent à l'intersection (A). La classe d'intersection anonyme est maintenant une nouvelle description de classe de concept.

Une conversion qui utilise les quantificateurs existentiels et universels génère une contrainte de valeur qui est une contrainte locale, *i.e.* elle agit sur la propriété seulement si celle-ci s'applique à la description de classe où la contrainte est définie. Ceci veut dire qu'une contrainte de valeur exerce une restriction sur la cible de la propriété lorsqu'elle s'applique à cette description de classe particulière. Ce type de conversion génère une cible de la propriété qui est une classe anonyme[48] en exerçant des contraintes sur l'extension de classe. La classe contrainte hérite alors d'une classe anonyme définie par la contrainte. Ceci ne convient pas à la représentation d'un attribut EXPRESS dont la source et l'image sont bien précises. Ainsi, pour éviter d'engendrer une classe anonyme comme cible de la propriété, et dans les cas où cela s'avère possible, nous avons décidé de convertir un attribut en EXPRESS dans une contrainte globale en utilisant `owl:ObjectProperty` et `owl:DatatypeProperty` selon que la cible est un concept ou un type de données. Pour garantir que l'attribut EXPRESS est interprété comme une contrainte locale définie dans le cadre de l'entité, nous préfixons l'attribut du nom de l'entité où l'attribut est défini. Un exemple de conversion d'un attribut est présenté dans la ligne 6 du Tableau 7.

[48] C'est-à-dire la classe de tous les individus satisfaisant à la restriction.

Tableau 7 – Correspondances entre les constructeurs des langages EXPRESS, DL et OWL-DL.

ID	EXPRESS	Signification	DL	OWL
1	ENTITY	Un ensemble d'instances	C:concept	owl:Class
2	BOOLEAN, STRING, REAL, DATE, ...	Types basiques de données	B:boolean, S:string, F:float, D:date, ...	xsd:boolean, xsd:string, xsd:float, xsd:dateTime
3	ENTITY XtdA; SUBTYPE OF XtdB END_ENTITY;	Sous-typage : chaque instance de la classe XtdA est aussi une instance de la classe XtdB	XtdA ⊑ XtdB	<owl:Class rdf:about="#XtdB"> <rdfs:subClassOf rdf:resource="# XtdB"/> </owl:Class>
4	TYPE XtdA = SELECT (XtdB1, XtdB2); END_TYPE; Exemple : XtdSimpleValue, dans ISO 12006-3	Chaque instance de la classe XtdA est soit une instance de la classe XtdB1, soit une instance de la classe XtdB2	XtdB1 ⊑ ¬XtdB2 XtdA ⊑ XtdB1 ⊔ XtdB2	<owl:Class rdf:about="#XtdB1"> <owl:disjointWith rdf:resource="#XtdB2"/> </owl:Class> <owl:Class rdf:about="#XtdA"> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#XtdB1"/> <owl:Class rdf:about="#XtdB2"/> </owl:unionOf> </owl:Class>
5	ENTITY XtdA; SUPERTYPE OF (ONEOF(XtdB1, XtdB2)); END_ENTITY; Exemple : XtdCollection, dans ISO 12006-3	Les classes XtdB1 et XtdB2 sont disjointes et héritent de XtdA	XtdB1 ⊑ XtdA XtdB2 ⊑ XtdA XtdB1 ⊑ ¬XtdB2	<owl:Class rdf:ID="XtdA"/> <owl:Class rdf:ID="XtdB1"> <rdfs:subClassOf rdf:resource="#XtdA"/> <owl:disjointWith rdf:resource="#XtdB2"/> </owl:Class> <owl:Class rdf:ID="XtdB2"> <rdfs:subClassOf rdf:resource="#XtdA"/> <owl:disjointWith rdf:resource="#XtdB1"/> </owl:Class>
6	ENTITY XtdA; attribute : XtdB; END_ENTITY; Exemple : XtdRelAssociates	L'attribut attribute est une relation entre deux ensembles d'instances du type entité (XtdA et XtdB). Chaque instance de la	XtdA ⊑ =1 XtdA_attribute.XtdB	<owl:Class rdf:ID="XtdA"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#XtdA_attribute"/> <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>

	, dans ISO 12006-3	classe XtdA doit avoir uniquement une instance de la classe XtdB. Pour cela, la restriction de cardinalité exacte est ajoutée sur l'attribut. Dans EXPRESS, les attributs sont définis localement dans le cadre de l'entité (classe). Ainsi, dans la conversion, le nom de l'attribut est préfixé de celui de l'entité		</owl:Restriction> </rdfs:subClassOf> </owl:Class> <owl:ObjectProperty rdf:ID="XtdA_attribute"> <rdfs:domain rdf:resource="#XtdA"/> <rdfs:range rdf:resource="#XtdB"/> </owl:ObjectProperty> <owl:Class rdf:ID="XtdB"/>
7	ENTITY XtdA; INVERSE Attribute : XtdC FOR XtdC_attribute; END_ENTITY; Exemple : IfcProduct, dans Ifc-Kernel	Une contrainte d'existence (dont la syntaxe fait usage du mot réservé INVERSE) fait dépendre l'existence d'une instance de XtdA de l'existence d'exactement une instance de XtdC qui a un attribut XtdC_attribute. Ceci est traduit en une propriété inverse avec la particularité d'une cardinalité exactement égale à 1 pour l'attribut XtdC_attribute	$XtdC\_attribute \equiv attribute^-$ $XtdA \sqsubseteq =1\ Attribute.XtdC$ $XtdC \sqsubseteq \forall XtdC\_attribute.XtdA$	<owl:ObjectProperty rdf:ID="attribute"> <rdfs:domain rdf:resource="#XtdA"/> <rdfs:range rdf:resource="#XtdC"/> <owl:inverseOf rdf:resource="#XtdC_attribute"/> </owl:ObjectProperty> <owl:Class rdf:ID="XtdA"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#attribute"/> <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality> <owl:valuesFrom rdf:resource="#XtdC"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class> <owl:Class rdf:ID="XtdC"/> <owl:ObjectProperty rdf:ID="XtdC_attribute"> <rdfs:domain rdf:resource="#XtdC"/> <rdfs:range rdf:resource="#XtdA"/>

				<owl:inverseOf rdf:resource="#attribute"/> </owl:ObjectProperty>
8	TYPE XtdText = STRING; END_TYPE;	Un type de données est traduit en utilisant le constructeur owl:DatatypeProperty	xtdText ⊑ String	<owl:DatatypeProperty rdf:ID="xtdText"> <rdfs:range rdf:resource="&xsd;string"/> </owl:DatatypeProperty>
9	ENTITY XtdA; RelatingObject: XtdB; RelatedObjects : SET[1 :?] OF XtdB; WHERE WR1: SIZEOF(QUERY (Result<*Related Objects | RelatingObject:= :Result))=0; END_ENTITY; Exemple : XtdRelAssociates , dans ISO 12006-3	L'attribut RelatedObjects est une relation 1-n entre un ensemble d'instances de l'entité (classe) xtdA et n instances de l'entité xtdB. La contrainte WR1 signifie que l'instance de XtdB reliée à l'attribut RelatingObject n'appartient pas à l'ensemble des instances de XtdB reliées à l'attribut relatedObjects	XtdA ⊑ =1 XtdA_relatingObject.XtdB XtdA ⊑ ≥1 XtdA_relatedObjects.XtdB XtdA ⊑ ≤n XtdA_relatedObjects.XtdB <-contrainte WR1-> XtdA_relatingObject ⊑ relationship XtdA_relatedObjects ⊑ relationship XtdA ⊑ =(n+1) relationship.⊤	<owl:ObjectProperty rdf:ID="relationship"/> <owl:Class rdf:ID="XtdA"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#XtdA_relatedObjects"/> <owl:minCardinality rdf:datatype="&xsd;int">1</owl:minCardinality> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#XtdA_relatedObjects"/> <owl:maxCardinality rdf:datatype="&xsd;int">n</owl:maxCardinality> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#XtdA_relatingObject"/> <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#relationship"/> <owl:cardinality rdf:datatype="&xsd;int">n+1</owl:cardinality>

				<owl:valuesFrom rdf:resource="&owl;Thing"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class> <owl:ObjectProperty rdf:ID="XtdA_relatedObjects"> <rdfs:domain rdf:resource="#XtdA"/> <rdfs:range rdf:resource="#XtdB"/> <rdfs:subPropertyOf rdf:resource="#relationship"/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="XtdA_relatingObject"> <rdfs:domain rdf:resource="#XtdA"/> <rdfs:range rdf:resource="#XtdB"/> <rdfs:subPropertyOf rdf:resource="#relationship"/> </owl:ObjectProperty>
10	ENTITY XtdA; attribute : XtdB; UNIQUE uniquecontrainte : attribute; END_ENTITY;	Les contraintes d'unicité sont traduites à l'aide du constructeur owl:FunctionalProperty. Dans la version DL, un troisième nom de concept XtdC est utilisé pour représenter l'unicité de la valeur de l'instance cible de attribute	XtdA ⊑ attribute.XtdB XtdA ⊑ attribute.XtdC XtdB ≡ XtdC	<owl:ObjectProperty rdf:ID=" XtdA_attribute"> <rdf:type rdf:resource="&owl;FunctionalProperty"/> <rdfs:domain rdf:resource="#XtdA"/> <rdfs:range rdf:resource="#XtdB"/> </owl:ObjectProperty> <owl:Class rdf:ID="XtdA"/> <owl:Class rdf:ID="XtdB"/>

La ligne 9 de ce même tableau montre un exemple de traduction d'une contrainte locale ou règle de domaine EXPRESS. Dans cet exemple, la règle est convertie en créant une super-propriété et en jouant sur la cardinalité des propriétés. Cependant, dans d'autres cas la traduction des règles de domaine peut être plus complexe. C'est le cas des règles qui font appel à des fonctions. Puisque OWL spécifie un langage déclaratif, qui repose sur l'hypothèse d'un monde ouvert, et EXPRESS fait appel à des constructions procédurales, qui reposent sur l'hypothèse d'un monde fermé, une traduction entre eux peut générer de la perte sémantique. Dans ce travail nous nous sommes affranchis de ces problèmes puisque nous avons sélectionné des sous-ensembles de ressources représentées en EXPRESS qui ne spécifient pas de fonctions.

Finalement, la ligne 10 de ce même tableau, propose une solution pour la traduction des contraintes d'unicité, en utilisant des propriétés fonctionnelles en OWL.

3.2.3 Découverte de correspondances sémantiques

Cette section présente le processus de découverte de correspondances sémantiques, *i.e.* la phase 3 de la méthodologie de travail (Ferreira da Silva *et al.* 2006c). Plusieurs éléments sont à prendre en compte avant de présenter le processus de découverte de correspondances sémantiques.

3.2.3.1 Paramètres à prendre en compte

La définition de ressources sémantiques prise en compte pour ce travail est composée de quatre éléments : les concepts, les relations, les instances et les définitions en langage naturel. Les trois premiers éléments sont le fondement de toute base de connaissances. Dans l'ensemble des relations nous incluons les relations non-hiérarchiques qui enrichissent une ressource sémantique par rapport à une simple taxonomie. La prise en compte des individus dans une ressource sémantique permet l'appariement d'instances. Les définitions des concepts en langage naturel sont importantes parce qu'il n'est pas toujours évident pour le concepteur d'ontologies de représenter avec une LD les connaissances du domaine. La LD choisie peut ne pas proposer les constructeurs nécessaires pour la représentation souhaitée. Par exemple, les connaissances procédurales ou règles nécessitent une extension supplémentaire aux LD connues. Sans la possibilité d'attacher ces connaissances "libres", elles peuvent être perdues. Les quatre éléments exposés sont pris en compte dans la définition suivante.

Définition 1 – Ressource sémantique générique. Une ressource sémantique est un quadruplet (C, R, I, A) respectant les critères suivants :

- C est l'ensemble des concepts d'une ressource sémantique. Cet ensemble est partiellement ordonné par la relation d'ordre "est une sorte de" ou relation hiérarchique de subsomption, ce qui définit une condition nécessaire mais non suffisante pour établir l'appartenance d'un individu à un concept. Pour tous $(c_1, c_2) \in C \times C$, $c_1 \sqsubseteq c_2$, signifie que c_1 est plus spécifique que c_2 et c_2 est plus général que c_1. Chaque concept $c \in C$ peut être aussi définit de manière nécessaire et suffisante.
- R est l'ensemble de *relations* r reliant les concepts. R représente l'ensemble de relations hiérarchiques et les relations non hiérarchiques, *i.e.* la subsomption et les propriétés.
- I est l'ensemble des *instances*. Cet ensemble représente les individus du monde réel.
- A représente l'ensemble des définitions en langage naturel. Ainsi, chaque concept c ou relation r peut être associé à une définition en langage naturel a, $a \in$ A.

Deux ressources sémantiques, $RS_1 = (C_1, R_1, I_1, A_1)$ et $RS_2 = (C_2, R_2, I_2, A_2)$ sont prises en compte pour le processus de découverte de correspondances sémantiques. Nous définissons une correspondance sémantique de la manière suivante.

Définition 2 – Correspondance sémantique. Une correspondance sémantique $Corresp_{RS1 \rightarrow RS2}$ entre deux éléments de $RS_1 = (C_1, R_1, I_1, A_1)$ et $RS_2 = (C_2, R_2, I_2, A_2)$ est le triplet (s, g, e) dont

- s est un concept, une relation ou une instance de la ressource source, $s \in RS_1$;
- g est un concept, une relation ou une instance de la ressource cible, $g \in RS_2$;
- e est l'étiquette qui définit le type de la correspondance et $e \in$ {équivalence, subsomption, transitivité ou conjonction}.

Les moteurs d'inférences actuels fondés sur les LD ne permettent pas d'aligner directement deux éléments de nature différente, *i.e.* ils ne permettent pas d'apparier un concept avec une relation. Un moteur d'inférences se base sur la relation de subsomption entre concepts. S'il trouve une relation de subsomption entre un concept d'une ressource et un autre concept d'une autre ressource, l'étiquette de la correspondance est subsomption. Si le moteur d'inférences retrouve une autre subsomption entre ces mêmes deux concepts mais dans le sens inverse au précédent, l'étiquette de la correspondance est équivalence. S'il trouve une troisième subsomption

lié à une des précédentes, l'étiquette de la correspondance est transitivité ou conjonction dépendant de la position de la subsomption, tel qu'il est expliqué ensuite.

Définition 3 – Étiquette de correspondance sémantique. L'étiquette d'une correspondance sémantique $Corresp_{RS1 \rightarrow RS2}$ entre RS_1 et RS_2 est du type équivalence, subsomption, transitivité ou conjonction quand les conditions suivantes se vérifient pour chaque (c, c', e), dont c et c' sont des concepts appartenant à RS_1 et RS_2 respectivement :

- si $c \sqsubseteq c'$ et $c' \sqsubseteq c$ alors e est équivalence. Une équivalence signifie que l'extension, *i.e.* l'ensemble des individus, de c est équivalente à l'extension de c' (Figure 16a).
- si $c \sqsubseteq c'$ ou $c' \sqsubseteq c$ alors e est subsomption. Pour le cas $c \sqsubseteq c'$, une subsomption signifie que l'extension de c est classifiée dans l'extension de c' (Figure 16b).
- Si $(c_1 \sqsubseteq c') \sqcap (c \sqsubseteq c_1) \Rightarrow c \sqsubseteq c'$, dont $c_1 \in SR_1$, alors e est transitivité. Ce type de correspondance est rendu possible par les correspondances de subsomption (Figure 16c).
- Si $(c \sqsubseteq c') \sqcap (c \sqsubseteq c_1) \Rightarrow c \sqsubseteq (c' \sqcap c_1)$, dont $c_1 \in SR_1$, alors e est conjonction. Ce type de correspondance est une conséquence des correspondances de subsomption préalables (Figure 16d).

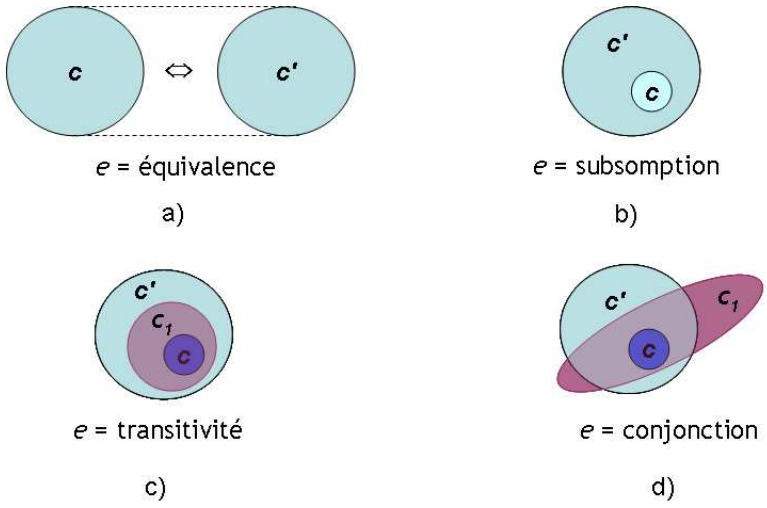

Figure 16 – Types de correspondances : a) Équivalence ; b) Subsomption ; c) Transitivité et d) Conjonction.

3.2.3.2 Processus de découverte de correspondances sémantiques

Pour amorcer le processus de découverte de correspondances, le moteur d'inférences reçoit en entrée deux ressources sémantiques à comparer et un ensemble d'axiomes d'amorçage. Ces axiomes d'amorçage sont entre trois et neuf, selon les ressources sémantiques. Ces axiomes ont été définis d'après les indications des experts du domaine d'application. Chaque axiome d'amorçage met en relation, en général par subsomption, un concept de RS_1 avec un concept de de RS_2. Ces axiomes représentent alors des points d'ancrage reliant initialement les deux ressources sémantiques. Un axiome de points d'ancrage est de la forme *rel c c'*, dont

- *rel* est la relation liant les concepts ;
- *c* est le concept antécédent de l'axiome et $c \in SR_1$;
- *c'* est le concept conséquent de l'axiome et $c' \in SR_2$.

Par exemple, dans le cas d'un axiome de points d'ancrage dont *rel* est égal à $\sqsubseteq$, l'axiome établit que *c* est subsumé par *c'*.

La Figure 17 schématise le processus de découverte de correspondances sémantiques. Ce processus prend en entrée deux ressources sémantiques à comparer, représentées en OWL-DL, et les points d'ancrage. Ces connaissances d'entrée sont chargées dans le moteur d'inférences. Il teste la satisfiabilité et la subsomption entre les concepts des deux ressources sémantiques. Pour cela, le moteur d'inférences que nous avons utilisé applique l'algorithme des tableaux expliqué au chapitre 2 (*cf.* section 2.2.2). Le résultat de ce processus est une liste de correspondances d'équivalence, de subsomption, de transitivité et de conjonction entre concepts des deux ressources sémantiques, quand ces correspondances existent.

Le travail de modélisation du type de correspondance sémantique ainsi que l'implantation du processus de découverte de correspondances sémantiques ont été menés dans le cadre du projet FUNSIEC, en collaboration avec Celson Lima et Chan Le Duc (Lima *et al.* 2005b, Lima, Ferreira da Silva et Le Duc 2005e). Dans le contexte de ce projet, le prototype développé, nommé FUNONDIL (*cf.* section 4.4.1), réutilise le moteur d'inférences d'Ondil (Le Duc 2004). Ondil ajoute des couches supplémentaires de gestion d'ontologies et utilise les services du moteur d'inférences FaCT, en particulier le test de subsomption. La version de FaCT utilisée pour ce travail ne traite pas le langage OWL. Ainsi les ontologies ont dû être préalablement traduites en un langage interne du moteur, fondé sur le langage Lisp.

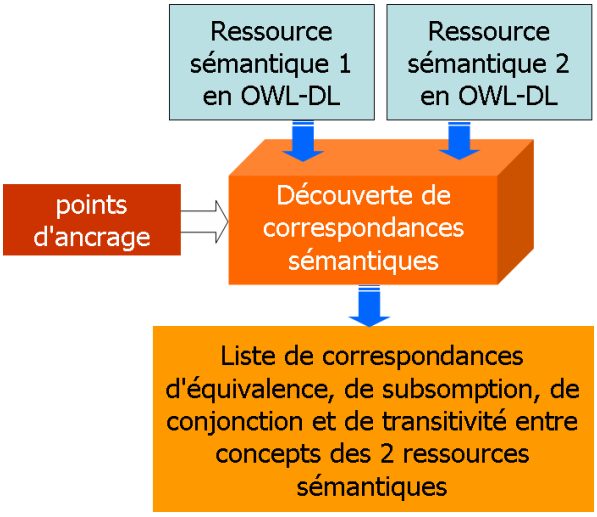

Figure 17 – Processus de l'algorithme de découverte de correspondances sémantiques.

3.2.3.3 Algorithme pour la découverte de correspondances sémantiques

L'algorithme 1 décrit le processus de découverte de correspondances sémantiques, schématisé dans la Figure 17, avec plus de détails. Ainsi, l'entrée de ce processus est deux ressources sémantiques à comparer et un ensemble d'axiomes de points d'ancrage. Les lignes 1 à 5 déclarent des variables. La ligne 3 déclare un tableau `a` à deux dimensions pour contenir les axiomes de points d'ancrage. Chaque axiome est à son tour représenté par un tableau. La ligne 4 déclare la variable `e`, *i.e.* l'étiquette qui typifie la correspondance. La ligne 5 déclare un tableau $\texttt{ListeCorresp}_{RS1 \rightarrow RS2}$ à deux dimensions pour contenir l'ensemble de correspondances découvertes à la fin du processus. La ligne 6 envoie les deux ressources sémantiques à comparer et les axiomes de points d'ancrage au moteur d'inférences. Ce dernier procédera au dépliage et à la classification des ressources sémantiques en accord avec ce qui a été présenté dans la section 2.2.2. La ligne 7 appelle le test de satisfiabilité pour les comparaisons entre concepts des lignes suivantes. La 8 initie la structure de contrôle qui va itérer sur chaque paire de concepts à comparer. Les lignes 9 à 25 servent à tester la subsomption, l'équivalence, la transitivité et la conjonction, par la comparaison des descriptions de concepts en faisant appel au service de test de satisfiabilité du moteur d'inférences. Si une subsomption est retrouvée, l'étiquette de la correspondance sémantique `e` prend la valeur subsomption et la correspondance est stockée dans la structure $\texttt{ListeCorresp}_{RS1 \rightarrow RS2}$. Un processus similaire est appliqué aux autres types de correspondances, en suivant la définition 3. Finalement, la ligne 27 retourne le résultat de ce processus.

Les résultats de l'implantation de cet algorithme sont présentés au chapitre 5. Nous présentons ci-dessous quelques exemples de résultats obtenus.

Algorithme 1 : Découverte de correspondances sémantiques

```
1.  OWLClass c, c1 ∈ RS1;
2.  OWLClass c' ∈ RS2;
3.  OWLClass [][] a;
4.  String e ← ∅;
5.  String[][] ListeCorresp_RS1→RS2 ← ∅;
6.  Charger moteur d'inférences (RS1, RS2, a);
7.  Appel test satisfiabilité;
8.   pour chaque paire (c, c') faire
9.     si c ⊑ c' alors e ← "subsomption";
10.      si c' ⊑ c alors e ← "équivalence";
11.      fin si
12.       ListeCorresp_RS1→RS2 ← ListeCorresp_RS1→RS2 + (c, c', e);
13.    sinon
14.      si c' ⊑ c alors e ← "subsomption";
15.           ListeCorresp_RS1→RS2 ← ListeCorresp_RS1→RS2 + (c', c, e);
16.      fin si
17.    fin si
18.    si (c1 ⊑ c') ⊓ (c ⊑ c1) alors c ⊑ c';
19.           e ← "transitivité";
20.           ListeCorresp_RS1→RS2 ← ListeCorresp_RS1→RS2 + (c, c', e);
21.    fin si
22.    si (c ⊑ c') ⊓ (c ⊑ c1) alors c ⊑ (c' ⊓ c1);
23.           e ← "conjonction";
24.           ListeCorresp_RS1→RS2 ← ListeCorresp_RS1→RS2 +
                                          (c, c' ⊓ c1, e);
25.    fin si
26. fin pour
27. retourne ListeCorresp_RS1→RS2;
```

3.2.3.4 Exemples de résultats de correspondances sémantiques

Considérons deux ressources sémantiques : *bcBuildingSpecifications* et *e-Cognos*. Ces ressources sémantiques sont présentées dans la section 4.2. Le processus de découverte de correspondances suggère la correspondance de subsomption (3) suivante :

$$\texttt{eCognos:Stair\_element} \sqsubseteq \texttt{bc:Stair} \quad \textbf{(3)}$$

Autrement dit, le moteur d'inférences découvre que le concept nommé `Stair` de la ressource bcBuildingSpecifications (identifié par le préfixe `bc`) subsume le concept nommé `Stair_element` de la ressource e-Cognos. La représentation en OWL de cette correspondance est la suivante :

```
<rdf:RDF xmlns="http://www.bcBuildingSpecification-
eCognosMappings.owl#"
     xml:base="http://www.bcBuildingSpecification-eCognosMappings.owl"
     xmlns:bc="http://www.bcXML.org/2002/bcXML#"
     xmlns:eCognos="http://www.cstb.fr/eCognos#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:about="&bc;Stair"/>
    <owl:Class rdf:about="&eCognos;Stair_element">
        <rdfs:subClassOf rdf:resource="&bc;Stair"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Subsumption mapping: eCognos:Stair_element is_subclassOf
bc:Stair</rdfs:comment>
    </owl:Class>
(...)
</rdf:RDF>
```

Un exemple de correspondance de transitivité (6) est proposé par le moteur d'inférences à partir de la correspondance de subsomption (4) préalable et l'information de subsomption (5) contenue dans la ressource e-Cognos. Les deux correspondances (4) et (6) résultent du processus de découverte de correspondances.

$$\texttt{eCognos:Manufactured\_material} \sqsubseteq \texttt{bc:Material} \quad \textbf{(4)}$$

eCognos:Processed_material ⊑ eCognos:Manufactured_material **(5)**

eCognos:Processed_material ⊑ bc:Material **(6)**

Pour un exemple de correspondance de conjonction (14) considérons l'information de subsomption (7) et la définition du concept SeparationConstructionType (8) contenues dans la ressource bcBuildingSpecifications. Considérons aussi, pour la ressource e-Cognos, l'information de subsomption (9) et la définition du concept building_non_structural_system (10).

bc:WallType ⊑ bc:SeparationConstructionType **(7)**

bc:SeparationConstructionType ≡ (bc:SeparationConstruction ⊔
bc:AccessConstructionType ⊔ bc:WallType ⊔
bc:PenetrationConstructionType ⊔
bc:OpeningRestrictingConstructionType ⊔
bc:LightEntranceConstructionType) **(8)**

eCognos:Wall_and_Cladding ⊑ eCognos:building_non_structural_system **(9)**

eCognos:building_non_structural_system ≡ (eCognos:Roof ⊔
eCognos:HVAC_system ⊔ eCognos:Stair_and_ramp ⊔
eCognos:Building_mechanical_system ⊔ eCognos:Wall_and_Cladding ⊔ ...) **(10)**

À partir du processus d'inférences, nous obtenons les correspondances sémantiques (11) et (12). Tenant ainsi compte des axiomes (7) à (12) nous obtenons la correspondance sémantique de conjonction (13) qui correspond à (14).

bc:WallType ⊑ eCognos:building_non_structural_system **(11)**

eCognos:Wall_and_Cladding ⊑ bc:SeparationConstructionType **(12)**

(bc:WallType ⊓ eCognos:building_non_structural_system) ⊑
(eCognos:building_non_structural_system ⊓
bc:SeparationConstructionType) **(13)**

`bc:WallType ⊑ eCognos:Wall_and_Cladding` **(14)**

En général, les correspondances et éventuellement les points d'ancrage ne sont actualisés que lorsqu'il y a des modifications dans les ressources sémantiques ou si les experts du domaine veulent ajouter une nouvelle ressource sémantique à l'infrastructure.

3.2.4 Processus d'unification de propriétés

Cette partie présente une étape supplémentaire de la méthodologie proposée (*cf.* section 3.2), visant à générer de nouvelles correspondances à partir de l'adjonction de connaissances supplémentaires. Le principe est d'établir des équivalences entre certaines propriétés[49] intervenant dans les définitions de concepts des différentes ressources à aligner et de lancer à nouveau le processus de découverte de correspondances. Le choix des propriétés équivalentes est effectué par les experts du domaine. L'intérêt d'établir des équivalences entre propriétés de ressources sémantiques différentes est de pouvoir produire de nouvelles correspondances sémantiques et d'inférer des instances entre concepts lors de l'appel aux services d'un moteur d'inférences.

3.2.4.1 Algorithme d'unification de propriétés

L'algorithme 2 décrit le processus d'unification par l'équivalence entre propriétés. Les lignes 1 et 2 déclarent les variables propriétés p et p' qui appartiennent aux ressources sémantiques RS_1 et RS_2 respectivement. La ligne 3 lit les deux ressources sémantiques d'entrée. La ligne 4 lit les deux propriétés à déclarer équivalentes. Dans la ligne 5, l'axiome qui déclare l'équivalence entre les deux propriétés est construit. La ligne 6 ajoute l'axiome d'équivalence à une des ressources, créant une nouvelle version RS_1'. Notons qu'ajouter le même axiome aux deux ressources serait redondant. Finalement la ligne 7 sauvegarde RS_1'. Une implantation Java est proposée dans `Unification.java` (annexe 3). Cet algorithme a été implanté dans le prototype UNIREL (*cf.* section 4.4.2), qui étend FUNONDIL avec d'autres fonctionnalités.

[49] Une équivalence de propriétés est un axiome qui spécifie un lien d'équivalence d'une propriété à une autre. L'équivalence de propriétés est différente de l'égalité de propriétés. Les propriétés équivalentes ont les mêmes valeurs, *i.e.* la même extension de propriété, mais elles peuvent avoir des significations intensionnelles différentes, *i.e.* représenter des concepts différents.

Le processus de découverte de correspondances est ensuite lancé entre RS_1' et RS_2. L'influence de l'unification de propriétés sur les correspondances produites dépend des définitions de concepts. Nous présentons deux exemples dans la section 3.2.6.

Algorithme 2 : Unification de propriétés

1. *OWLObjectProperty p* ∈ *RS₁*;
2. *OWLObjectProperty p'* ∈ *RS₂*;
3. Lire (*RS₁*, *RS₂*);
4. Lire (*p*, *p'*);
5. créer_axiome_propriété_équivalente (*p*, *p'*);
6. ajouter_axiome_propriété_équivalente (*RS₁'*);
7. sauvegarder (*RS₁'*);

L'établissement de l'équivalence entre propriétés de ressources différentes permet, dans les cas que nous analysons plus loin, d'inférer des instances entres concepts de ressources différentes. La section suivante présente un algorithme d'inférence d'instances.

3.2.5 Algorithme d'inférence d'instances

Le processus d'unification de propriétés présenté dans la sous-section précédente spécifie une équivalence entre deux propriétés de deux ressources différentes. Ensuite, les correspondances entre concepts peuvent engendrer le partage d'instances entre concepts de ressources différentes. Autrement dit, les instances d'un concept C1 dont la définition contient une propriété R1 équivalente à une autre propriété R2 apparaissant dans la définition d'un concept C2 d'une autre ressource, peuvent être inférées comme étant des instances de C2. Notons que ceci dépend des définitions de concepts où les propriétés sont impliquées. Dans ce cadre, il nous a paru intéressant d'étudier l'impact de l'unification des propriétés au niveau des inférences d'instances.

Pour cela, nous proposons l'algorithme 3 qui décrit le processus d'inférence d'instances des concepts de l'ensemble des ressources sémantiques d'entrée. La ligne 1 et 2 déclarent des variables *i* et *c*. La variable *i* sert à contenir l'instance à inférer. La variable *c* représente le concept pour lequel nous voulons déterminer s'il instancie des individus ou d'où des individus sont inférés. La ligne 3 déclare une structure tabulaire pour contenir les instances inférées. Les

ressources sémantiques sont lues dans la ligne 4 et la ligne 5 charge ces ressources dans le moteur d'inférences. La ligne 7 initie la structure de contrôle qui va itérer sur chaque concept des ressources sémantiques d'entrée, pour inférer les instances, si elles existent. Chaque individu *i* qui est inféré comme instance du concept *c* est stocké ensuite dans le tableau `listeInstances` (ligne 8). Finalement, la ligne 10 retourne le résultat de ce processus.

Algorithme 3 : Inférence d'instances

```
1. Instance i;
2. OWLClass c;
3. Instance [] listeInstances ← ∅;
4. Lire (RS1, RS2);
5. Charger moteur d'inférences (RS1, RS2);
6.  pour chaque c ∈ (RS1, RS2) faire
7.   i ← inférer_instances(c);
8.   listeInstances ← listeInstances + i;
9. fin pour
10. retourne listeInstances;
```

Ce processus utilise le service de réalisation fourni par le moteur d'inférences. Une implantation Java de cet algorithme est proposée dans `GetIndividualsOfClass.java` (annexe 3). Cette implantation attend le nom d'un concept en entrée sur lequel seront inférés les instances. Nous avons aussi implanté une autre version qui itère sur tous les concepts des ressources d'entrée pour inférer les instances et déterminer les concepts les plus spécifiques auxquels un individu appartient.

En pratique, tant les correspondances découvertes que les instances inférées dépendent du type de restrictions auxquelles les propriétés participent. Les deux exemples suivants montrent cette situation.

3.2.6 Exemples de résultats pour l'unifications de propriétés

Les exemples qui suivent sont destinés à clarifier le fonctionnement des algorithmes d'unification des propriétés et d'inférences d'instances. Dans le premier exemple les ressources contiennent

des restrictions globales. Le deuxième exemple considère deux ressources qui comportent deux restrictions locales en plus des globales.

Exemple 1 : Considérons les ressources sémantiques génériques RS_1 et RS_2 qui contiennent des restrictions globales sous forme d'axiomes de propriétés. La ressource RS_1 contient l'axiome (15) et le fait (16). La ressource RS_2 contient l'axiome (17). Les descriptions des concepts A_1, B_1, A_2 et B_2 sont effectuées de la manière la plus simple, dans les ressources sémantiques respectives, *i.e.* identifiées au moyen de leurs noms. L'axiome (18) qui représente l'équivalence entre les propriétés $relation_1 \in RS_1$ et $relation_2 \in RS_2$ est ajouté à RS_1. Ceci génère une nouvelle version de la ressource, RS_1'.

$$relation_1(A_1, B_1) \quad \textbf{(15)}$$

$$relation_1(a_1, b_1) \quad \textbf{(16)}$$

$$relation_2(A_2, B_2) \quad \textbf{(17)}$$

$$relation_1 \Leftrightarrow relation_2 \quad \textbf{(18)}$$

Ces connaissances permettent ensuite au service d'inférences du moteur de récupérer des instances supplémentaires. Les instances a_1 et b_1 (16), de RS_1', sont inférées respectivement comme étant des instances des concepts A_2 et de B_2 (17), de RS_2, et ceci indépendamment du fait que RS_2 ait des instances auparavant assignées. Cette configuration de restrictions ne génère pas de correspondances supplémentaires. En revanche, l'exemple 2 engendre des résultats différents.

Exemple 2 : Considérons maintenant dans les ressources sémantiques, en plus des axiomes (15) à (17), les définitions de concepts primitifs A_1 (19) de la ressource RS_1 et A_2 (20) de la ressource RS_2, qui contiennent les restrictions locales suivantes :

$$A_1 \sqsubseteq \forall relation_1.B_1 \sqcap \exists relation_1.B_1 \quad \textbf{(19)}$$

$$A_2 \sqsubseteq \exists relation_2.B_2 \quad \textbf{(20)}$$

Nous ajoutons l'axiome (18), à l'une des ressources. Ensuite le moteur d'inférences découvre deux correspondances du type subsomption, puisqu'il génère $A_1 \sqsubseteq A_2$ et $A_2 \sqsubseteq A_1$. Le moteur génère une troisième correspondance, d'équivalence, qui est une conséquence des deux autres : A_1 équivalent à A_2. Quant à l'inférence d'individus entre concepts, les instances de B_1 sont

inférées comme instances de B_2 et vice-versa. Les instances de A_1 sont inférées comme instances de A_2 et vice-versa.

Une analyse plus approfondie au sujet de l'influence de l'unification de propriétés et des définitions de concepts sur la découverte de correspondances et sur l'inférence d'instances entre concepts est présentée dans les sections 5.3 et 5.4. La section suivante présente une modélisation théorique des connaissances du domaine d'application visant à aider les experts à quantifier les correspondances non équivalentes.

3.2.7 Modélisation pour l'évaluation des correspondances sémantiques

L'évaluation des correspondances sémantiques découvertes est faite par les experts du domaine. En effet, leur expertise permet de vérifier que les liens découverts sont valides par rapport aux connaissances représentées dans les ressources sémantiques.

Cependant, il nous semble important d'apporter des méthodes permettant d'aider les experts dans cette tâche de vérification et de validation. Les méthodes sémantiques, en particulier l'utilisation d'un moteur d'inférences fondé sur les LD pour découvrir des correspondances appuyées sur les définitions formelles des ressources sémantiques, sont des méthodes exactes. Ainsi, les correspondances découvertes apportent un degré absolu de proximité sémantique pour des correspondances d'équivalence seulement. Leurs extensions, *i.e.* l'ensemble des instances, sont 100% équivalentes. Dans les correspondances non équivalentes, un concept c n'est pas 100% équivalent à un concept c'. De plus, pendant le processus de découverte de correspondances nous avons exclusivement tenu compte de l'information représentée en OWL. Cependant, il s'avère important de pouvoir mesurer les correspondances produites, en utilisant des informations supplémentaires si nécessaire, afin d'aider à déterminer la qualité des correspondances sémantiques. Par exemple, en utilisant des définitions des entités (concepts) en langage naturel.

Une partie de la résolution de ce problème dépend de la manière dont nous définissons la qualité des correspondances sémantiques ; autrement dit, trouver la manière qui nous semble la plus adéquate pour définir la qualité d'une correspondance entre insuffisante (*i.e.* une correspondance dont un coefficient de mesure de qualité est égal à 0) et parfaite (*i.e.* une correspondance dont un coefficient de mesure de qualité est égal à 1). Pour ce faire, nous utilisons la théorie de la logique floue (Zadeh 1965), qui fournit une approche qualitative permettant de modéliser cette notion vague de qualité de correspondance sémantique (*cf.* section 2.4).

Nous proposons une méthode de fuzzification adaptée à notre problème, *i.e.* la phase de modélisation de l'information visant à aider l'évaluation de la qualité des correspondances sémantiques en utilisant la logique floue (Ferreira da Silva *et al.* 2006a, Lima, Ferreira da Silva et Pimentão 2006).

Définition 4 – Variable linguistique *correspondances non équivalentes*. La variable linguistique "correspondances non équivalentes" (NEM) est l'ensemble de termes suivant :

$D_{(nem)}$ = {*non-acceptable*, *acceptable*, *good*, *strong*}.

Définition 5 – Fonctions d'appartenance. Pour modéliser les fonctions d'appartenance nous définissons trois variables d'entrée du processus de fuzzification : E_1, E_2, E_3 qui visent à représenter les similarités entre plusieurs caractéristiques des concepts. Ces variables d'entrée sont définies de la manière suivante :

- E_1 est la quantité de propriétés partagées. Autrement dit, cette variable d'entrée représente les propriétés d'objets (`owl:objectproperty`) ou de valeurs de données (`owl:DatatypeProperty`) et leurs valeurs cibles, pour ce qui est des restrictions globales.
- E_2 est la quantité d'entrées lexicales partagées par deux concepts. Les entrées lexicales (Lima *et al.* 2003) sont des termes jugés équivalents à un nom de concept donné et qui sont utilisés pour enrichir les concepts ontologiques, *i.e.* ce sont en quelque sorte des synonymes. Les entrées lexicales sont utilisées pour fournir une liste de termes qui peuvent être utilisés pour faire référence à un concept. Par exemple le concept *Actor* peut être référencé par les entrées lexicales *employee*, *person*, *driver*, *engineer*, etc.
- E_3 représente la similarité entre les annotations de concepts. Une annotation de concept peut inclure des commentaires ou des définitions de concepts en langage naturel. Une définition en langage naturel contient des termes et des expressions, dont une partie peut être segmentée en des expressions représentant un contenu sémantique plus riche que la simple addition des parties. Considérons *n* l'ordre d'une expression, *i.e.* la quantité de termes dans une expression. Un terme est alors une expression d'ordre *n* = 1. Les termes considérés vides de sens, par exemple *the*, *to*, *of*, *for*, etc. sont exclus. Notons que l'implantation de cette variable d'entrée implique une analyse linguistique du texte libre en

langage naturel contenu dans les annotations (par exemple, lemmatisation et comparaison de chaînes de caractères)[50].

Ces variables d'entrée sont fuzzifiées avec les quatre termes linguistiques de la définition 4 à travers la définition de fonctions d'appartenance (Tableau 8). Si la définition de deux concepts c et c' qui ont une correspondance sémantique non équivalente, ne partage aucune propriété, alors la valeur linguistique de la similarité relative aux propriétés (E_1) est "*non-acceptable*". Si c et c' partagent une à deux propriétés dont la cible est la même (même concept ou même type de données) alors la valeur linguistique de la similarité relative aux propriétés relève du sous-ensemble de valeur "*acceptable*". Si c et c' partagent deux à cinq propriétés, alors la valeur linguistique de la similarité relative aux propriétés est "*good*". Enfin, si c et c' partagent plus de quatre propriétés et cibles respectives, alors la valeur de la variable E_1 est "*strong*". Un raisonnement similaire est appliqué aux variables d'entrée E_2 et E_3. Par exemple, si c et c' partagent deux à six entrées lexicales, alors la valeur linguistique de la similarité entre les deux concepts par rapport à leurs entrées lexicales est "*good*". Si les annotations associées aux concepts c et c' partagent plus de six expressions, alors la valeur linguistique de la similarité entre les annotations de c et c' est "*strong*".

Le Tableau 8 résume ainsi l'attribution des termes linguistiques aux variables d'entrée spécifiées, en termes de sous-ensembles de quantités partagées pour les propriétés (E_1), les entrées lexicales (E_2) et les annotations (E_3). Les ensembles de ces valeurs peuvent varier et peuvent être définis et paramétrables par les experts du domaine. Notre choix de sous-ensembles de valeurs dépend donc des caractéristiques des ressources sémantiques et favorise l'importance des connaissances représentées en langage naturel, car celles-ci ne sont pas prises en compte dans la représentation OWL-DL et ni par le raisonnement fondé sur les LD.

[50] Cependant cet aspect d'analyse linguistique n'a pas été implanté puisqu'il sort du sujet de ces travaux.

Tableau 8 – Résumé de l'attribution des plages de valeurs spécifiées pour les termes linguistiques aux variables d'entrée définies (*n* est un entier).

		Linguistic terms			
		Non-acceptable	***Acceptable***	***Good***	***Strong***
Input variables	$\mathbf{E_1}$	*0*	*{1,2}*	*{2,...,5}*	*{4,...,n}*
	$\mathbf{E_2}$	*0*	*{1,2}*	*{2,...,6}*	*{5,...,n}*
	$\mathbf{E_3}$	*0*	*{1,...,4}*	*{3,...,7}*	*{6,...,n}*

Lors de ce processus de fuzzification, différentes fonctions d'appartenance peuvent être définies. Les graphiques normalisés de fonctions d'appartenance possibles pour chaque variable d'entrée (Figure 18, 19 et 20) incluent l'information du Tableau 8. Ces représentations graphiques permettent de visualiser plus facilement l'aspect flou de la modélisation à travers les chevauchements des formes, ainsi que les dégradés associés aux sous-ensembles de valeurs. Par exemple, au fur et à mesure qu'on s'approche de la valeur 1 pour l'une au l'autre des trois variables d'entrée, la variable linguistique *non-acceptable* perd de l'importance au profit de la variable linguistique *acceptable*.

Nous avons privilégié les formes triangulaires et trapézoïdales et ceci de manière à éviter les lacunes ou chevauchements insuffisants, et les chevauchements trop importants. En effet, les lacunes ou chevauchements insuffisants sont des zones mortes qui conduiraient à des espaces non couverts par la modélisation et les chevauchements trop importants conduiraient à un aplatissement des différences sémantiques attribuées aux variables linguistiques.

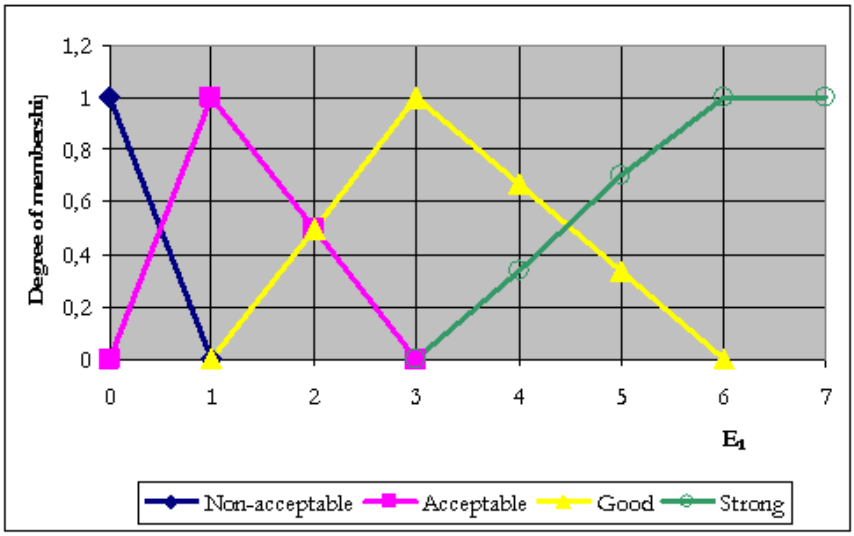

Figure 18 – Fonction d'appartenance pour la variable d'entrée E_1.

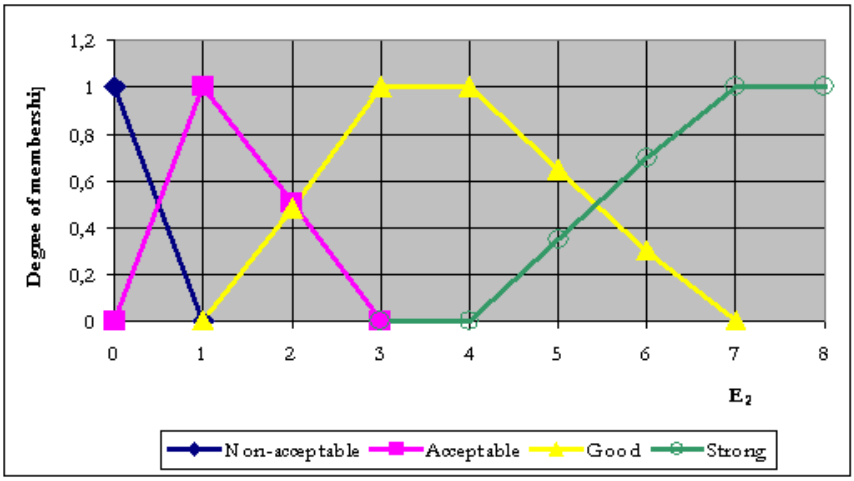

Figure 19 – Fonction d'appartenance pour la variable d'entrée E_2.

Par exemple, pour $E_3 = 3$, *i.e.* les concepts c et c' ont une similarité du contenu des annotations de valeur 3, puisqu'elles partagent 1 expression d'ordre 3 (ou 3 expressions d'ordre 1, ou une expression d'ordre 2 et un terme). Dans ce cas, la similarité par rapport aux annotations (Figure 20) prend la valeur linguistique *acceptable* à un degré d'appartenance de 0,66 (représenté par $\mu_{acceptable}(3) = 0{,}66$) et la valeur linguistique *good* à un degré d'appartenance de 0,33 (représenté par $\mu_{good}(3) = 0{,}33$). Notons que $\sum \mu_{nem}(n) = 1$.

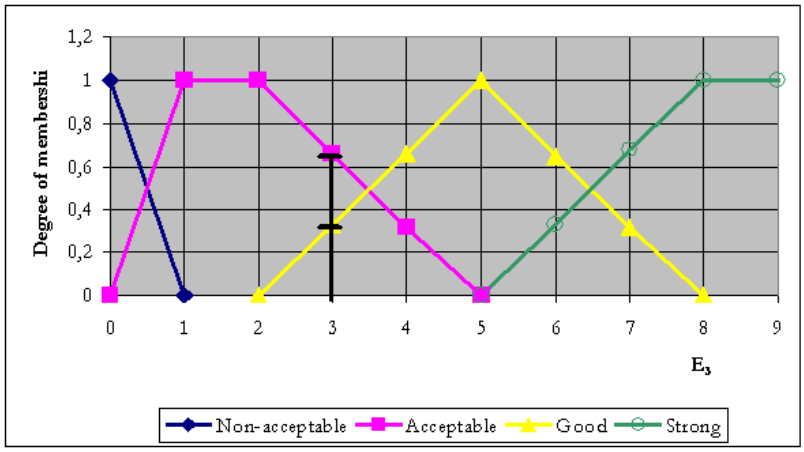

Figure 20 – Fonction d'appartenance pour la variable d'entrée E_3.

Cette modélisation va permettre de créer les règles sur lesquelles un système d'inférences floues (SIF) va raisonner. Pour toutes les combinaisons possibles des entrées, il est nécessaire de définir une règle. Une règle combine alors les variables d'entrée en utilisant les opérateurs ET et OU (min et max) qui constituent la prémisse de la règle. La conclusion de la règle est affectée d'un coefficient de modification des divers sous-ensembles de sortie. Pour deux variables d'entrée, la structure de la règle est la suivante :

SI l'entrée E_x appartient au sous-ensemble correspondant à la variable linguistique *g* avec une fonction d'appartenance $\mu_g(x)$ ET, que l'entrée E_y appartient au sous-ensemble correspondant à la variable linguistique *h* avec une fonction d'appartenance $\mu_h(y)$, ALORS la sortie appartient au sous-ensemble i affecté d'un coefficient de modification α.

En général, pour trouver ce coefficient de modification, les moteurs d'inférences floues effectuent une conjonction entre les fonctions d'appartenance des entrées des sous-ensembles de la forme $\alpha = \min \{\mu_1(x), \mu_2(y), ..., \mu_n(x)\}$. Mais d'autres opérateurs existent. Le moteur peut générer pour des combinaisons de règles différentes des coefficients de modification divers pour un même sous-ensemble de sortie. Donc, une disjonction peut être considéré entre les divers coefficients de modification de la forme $\alpha_s = \max \{\alpha_1, \alpha_2, ..., \alpha_n\}$.

Nous souhaitons que la conclusion des règles prenne en compte les mêmes termes linguistiques des *correspondances non équivalentes* de la définition 4. Une des règles possibles est proposée en tenant compte de l'information des figures 18, 19 et 20 :

SI $E_1 \in$ (*good* (0.63), *strong* (0.37)) ET $E_2 \in$ (*good* (1)) ET $E_3 \in$ (*acceptable* (0.66), *good* (0.33)) ALORS Correspondance $\in$ *good* (α_s), où α_s est le coefficient à calculer entre les coefficients de la prémisse en utilisant, par exemple, l'opérateur min.

La conclusion de la règle est une heuristique qui quantifie la correspondance de manière partielle, *i.e.* en rapport avec les prémisses de cette règle. La combinaison des différentes règles est effectuée par le moteur d'inférences floues, en jouant sur les coefficients de modification tels qu'expliqués ci-dessus. La sortie floue inférée par un SIF dépend bien entendu de la base de règles mais aussi des opérateurs d'agrégation et peut être transformée en une sortie nette par un processus de défuzzification.

3.3 Synthèse

Ce chapitre est le noyau de nos travaux. L'état de l'art nous a montré l'absence d'une approche intégrée, adaptée à l'alignement de ressources hétérogènes. De plus il nous a semblé envisageable d'approfondir l'étude des méthodes sémantiques, en particulier quant aux propriétés, peu traitées jusqu'ici. Pour ces raisons nous proposons notre méthodologie qui est composée des phases suivantes : analyse et sélection des ressources sémantiques, homogénéisation syntaxique, découverte de correspondances sémantiques entre concepts, unification de propriétés et évaluation des correspondances. La phase d'analyse et sélection des ressources sémantiques spécifie un ensemble de critères qui sont utilisés pour sélectionner les ressources sémantiques du domaine d'application que nous voulons comparer. La phase d'homogénéisation syntaxique vise à transformer la représentation des ressources sémantiques de leur format d'origine vers un format commun. Cette phase de conversion de formats rend plus facile la tâche suivante, puisque celle-ci peut alors se focaliser sur les aspects sémantiques. La phase de découverte de correspondances sémantiques est fondée sur les définitions formelles des concepts contenus dans les ressources sémantiques et utilise le raisonnement déductif d'un moteur d'inférences fondé sur les LD pour aboutir à des conclusions, *i.e.* aux correspondances sémantiques. La phase d'unification de propriétés est une phase supplémentaire qui tire parti de la création d'axiomes de propriétés équivalentes pour générer d'autres correspondances sémantiques et inférer des instances entre des concepts de ressources différentes. Pour clarifier nos propos nous présentons des exemples basés sur l'application des algorithmes proposés. Enfin, la phase d'évaluation des correspondances sémantiques vise à aider les experts du domaine dans leur tâche de vérification et de validation des résultats. Nous proposons de modéliser un ensemble d'informations qui

caractérise les entités des ressources sémantiques en utilisant la logique floue de manière à pouvoir qualifier les correspondances sémantiques non équivalentes en utilisant les services d'un moteur d'inférences floues. Le chapitre suivant présente le contexte d'application, les ressources sémantiques sélectionnées, les prototypes FUNONDIL et UNIREL développés et expliquons leur fonctionnement.

Chapitre 4 – Application et mise en œuvre

L'objectif de ce chapitre est de présenter le contexte d'application, les ressources sémantiques utilisées et les prototypes FUNONDIL et UNIREL développés pour mettre en œuvre en partie la méthodologie proposée.

4.1 Contexte de l'application

Le travail de cette thèse CIFRE a été en grande partie développé au CSTB, dans le département de Technologies de l'Information et Diffusion du Savoir. Une partie de ce travail a été élaborée dans le cadre du projet européen FUNSIEC[51]. Le CSTB a été le membre coordinateur de ce projet. L'objectif du projet FUNSIEC a été l'étude de faisabilité de la construction et de la maintenance d'une infrastructure sémantique ouverte pour le secteur européen de la construction, nommée OSIECS (pour *Open Semantic Infrastructure for the European Construction Sector*), et ce aux niveaux technique, organisationnel et métier. Cette infrastructure a recensé et rassemblé un ensemble de ressources sémantiques dédiées au secteur européen de la construction. Un des résultats de ce projet a été la mise en place du site *Funsiec Semantic Experience Centre*[52] (FSEC). Ce site vise à montrer et à promouvoir l'utilisation d'un ensemble de ressources sémantiques disponibles pour le secteur européen de la construction. Pour cela nous avons mis à disposition des explications sur les ressources, un ensemble de scénarios d'utilisation de ces ressources, des démonstrations, des films et des maquettes.

Un autre résultat de ce projet est le système FUNONDIL, de démonstration de conversion de formats des ressources sémantiques et d'utilisation d'un moteur d'inférences pour découvrir des correspondances sémantiques entre des ressources.

Notre travail de thèse dans le cadre du projet FUNSIEC a consisté à participer à l'analyse des ressources sémantiques, à la modélisation d'OSIECS, à l'implantation du prototype FUNONDIL en collaboration avec Chan Le Duc, et aux tests effectués avec les ressources. Forte de l'expérience du projet FUNSIEC, nous avons redéveloppé un prototype permettant d'approfondir certains aspects qui dépassaient les fonctionnalités de FUNONDIL.

---

[51] http://www.funsiec.org/

[52] http://195.83.41.67/SemanticExperienceCentre/index.html. Un tutoriel de FSEC est disponible à http://liris.cnrs.fr/catarina.ferreira.da.silva/FSEC_Tutorial-v1.1.PDF

En effet, après la finalisation de ce projet, nous avons étendu ce travail avec des phases supplémentaires dans la méthodologie proposée (*cf.* Figure 13). Ce sont les phases d'unification de propriétés et d'évaluation des correspondances, que nous avons présenté au chapitre précédent. Ce travail complémentaire est en partie mis en œuvre dans le prototype UNIREL qui étend FUNONDIL. UNIREL implante le travail sur les propriétés et permet également de réaliser la découverte de correspondances sémantiques ainsi que l'inférence d'instances entre concepts des ressources sémantiques.

Pour utiliser ces prototypes nous avons besoin des ressources du domaine d'application. Pour cela les ressources sémantiques sélectionnées sont présentées dans la section suivante.

4.2 Ressources sémantiques sélectionnées

Après l'analyse d'un ensemble de ressources sémantiques européennes effectuée lors du projet FUNSIEC (*cf.* Tableau 19, annexe 2), quatre ressources sémantiques ont été sélectionnées pour tester le prototype. En effet, nous avons sélectionnées les ressources du domaine d'application qui représentent des standards, qui sont exprimées en anglais et qui possèdent des modèle et méta-modèle utilisables. Ce sont quatre ressources dont les modélisations résultent d'un consensus élargi. Parmi ces ressources, deux sont des standards (ISO 12006-3 et ISO/PAS 16739:2005) qui servent de modèles pour plusieurs ressources sémantiques de pays européens. Les quatre ressources sémantiques sélectionnées sont les suivantes :

- L'ontologie *bcBuildingDefinitions* (Tolman *et al.* 2001)
- L'ontologie e-Cognos (Wetherill *et al.* 2002)
- Le standard ISO 12006-3 (ISO 12006-3:2007)
- Le standard *Industry Foundation Classes* (ISO/PAS 16739:2005)

4.2.1 L'ontologie bcBuildingDefinitions

L'ontologie *bcBuildingDefinitions* a été développée dans le cadre du projet eConstruct[53] (*Electronic Business in the Building and Construction Industry: Preparing for the New Internet*). Cette taxonomie a été crée pour les besoins du secteur de la construction. bcBuildingDefinitions peut être instanciée pour créer le contenu de catalogues de produits. Le scénario de démonstration utilisé dans le projet eConstruct montre l'utilisation de bcBuildingDefinitions, qui utilise le format bcXML

[53] http://xml.coverpages.org/econstructSummary.html

(Tolman *et al.* 2001). Une entreprise qui cherche des informations sur une porte, par exemple une porte en bois particulier, peut utiliser le navigateur d'eConstruct pour requêter le serveur de la taxonomie et trouver les informations détaillées sur le produit porte. L'entreprise accède ainsi aux informations de la structure de la porte, précise les propriétés du produit et peut ensuite chercher aussi dans les catalogues de produits dans d'autres langues comme l'allemand, le français ou l'anglais. Si le produit est retrouvé dans les catalogues, l'entreprise reçoit une liste des produits disponibles.

L'ontologie bcBuildingDefinitions est bâtie sur le méta-schéma XTD (*XML Taxonomy Definition*), dont la Figure 21 présente une représentation UML. XTD est utilisé pour définir les concepts et leurs propriétés. L'élément d'entrée du méta-schéma est la classe *Taxonomy*. Dans ce modèle, les concepts du domaine d'application sont des instances de la classe *Object*. Ces concepts sont des produits du domaine de la construction, par exemple des équipements, des matériaux, des composants, des documents, etc. La classe *Relationship* est utilisée pour représenter les relations entre les instances de la classe Object. La classe *Property* permet d'exprimer les caractéristiques ou attributs de chaque classe Object. Des informations complémentaires peuvent être trouvées dans (Tolman *et al.* 2001). Les propriétés peuvent exister seulement quand elles sont assignées à un Object. La classe *ExternalReference* permet à un objet d'instancier ou de réutiliser une propriété définie ailleurs. Les classes Taxonomy, Object, Property, et Relationship peuvent être associées à leur description, à travers la classe *Description*.

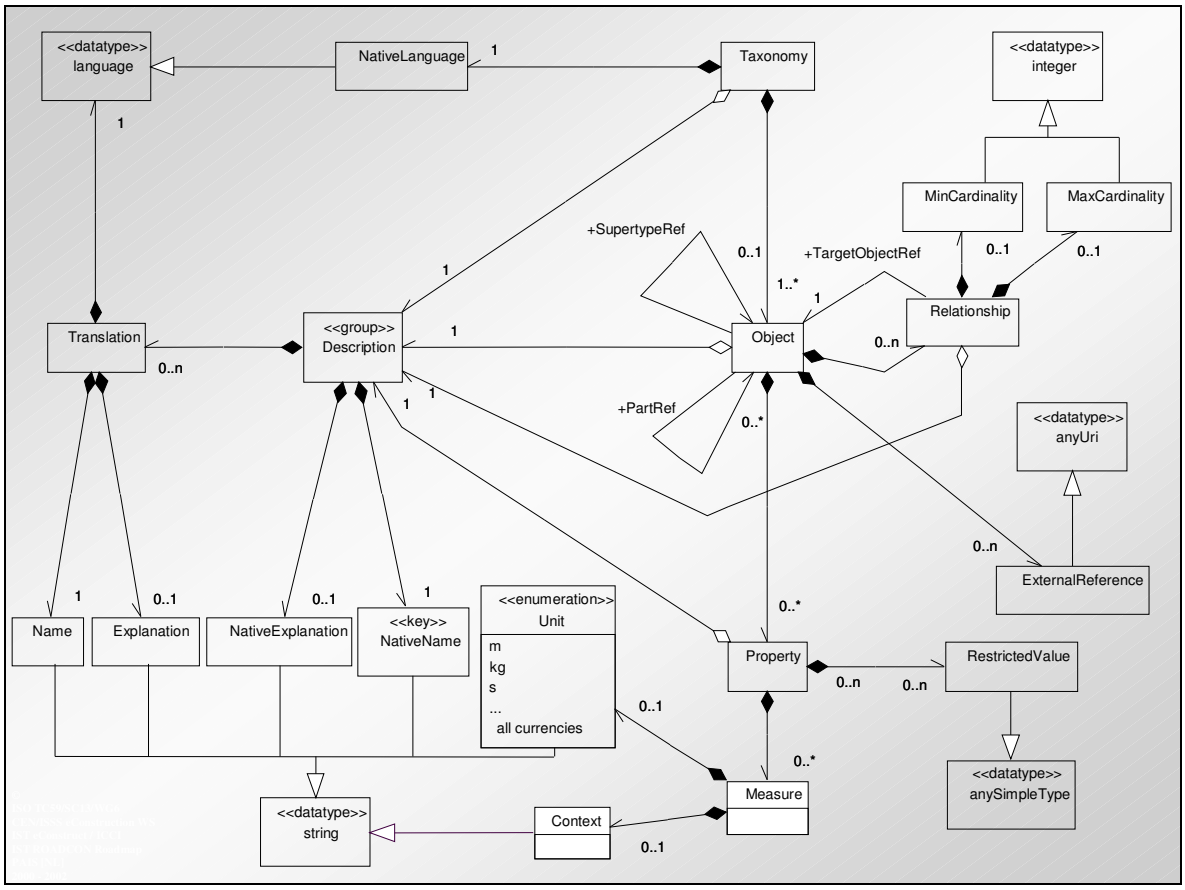

Figure 21 – Représentation UML du méta-schéma XTD, version Lite (Tolman *et al.* 2001).

4.2.2 L'ontologie e-Cognos

L'ontologie e-Cognos (Wetherill *et al.* 2002) a été développée dans le cadre du projet e-Cognos[54] (*Methodology, tools and architectures for* ***e****lectronic* ***CO****nsistent knowled****G****e ma****N****agement across pr****O****jects and between enterpri****S****es in the construction domain*). Cette ontologie décrit une partie du domaine de la construction. Elle est fondamentalement composée d'une taxonomie de concepts (Figure 22a) et d'une taxonomie de relations (Figure 22b). L'ontologie contient plus de 17 000 concepts et relations. La taxonomie de concepts est fondée sur les concepts du standard IFC (ISO/PAS 16739:2005) qui forment les niveaux supérieurs de la taxonomie, comme par exemple les concepts *Actor* et *Process*.

[54] http://e-cognos.cstb.fr/Downloads/Download.htm

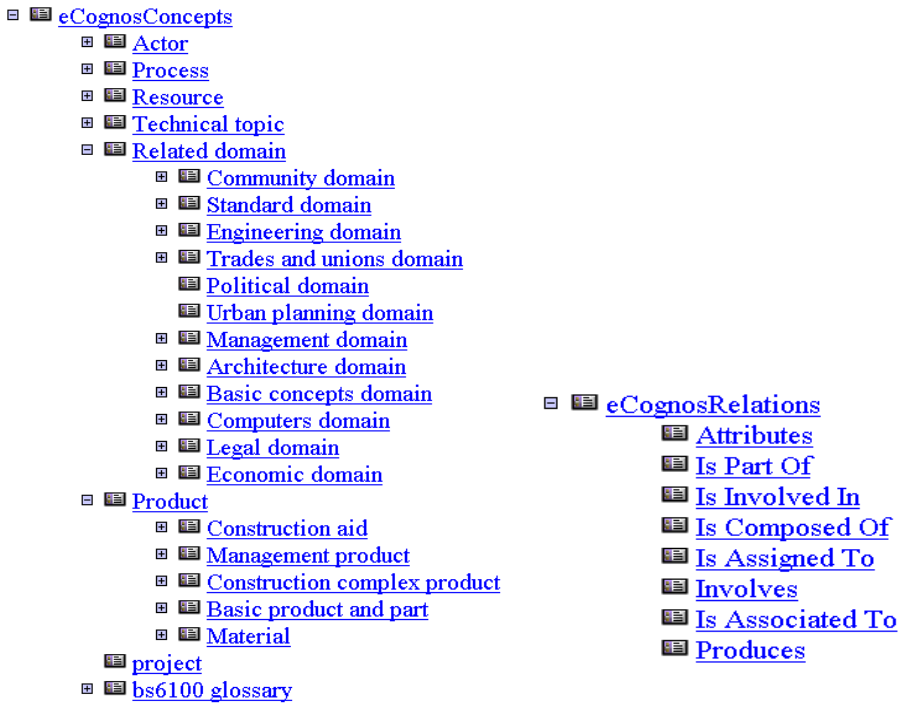

Figure 22 – a) Taxonomie de concepts. b) Taxonomie de relations.

Essentiellement, l'ontologie *e-Cognos* a été mise au point pour aider les pratiques de gestion des connaissances, telles que l'acquisition, l'indexation, la recherche, etc. de documents du domaine de la construction. L'infrastructure e-CKMI (*e-Cognos Knowledge Management Infrastructure*) est la base d'un scénario d'application de l'ontologie e-Cognos qui a été testé auprès d'entreprises du domaine de la construction. À travers cette infrastructure, il est possible d'acquérir des *Knowledge Items* (KI), *i.e.* des documents relatifs aux projets, aux organisations, aux experts, etc. et de créer les *knowledge representations* (KR) respectives. Ces KR sont ensuite indexées en utilisant des mots-clés et les concepts ontologiques. Pendant le processus de recherche, l'ontologie est à nouveau utilisée. Pour préciser sa recherche, l'utilisateur peut naviguer dans l'ontologie pour préparer sa requête. Le méta-schéma de l'ontologie e-Cognos (Figure 23) représente le modèle conceptuel de l'ontologie. Toutes les entités de l'ontologie sont spécialisées à partir de la classe *Object*, *i.e.* les classes *Concept* et *Relation*.

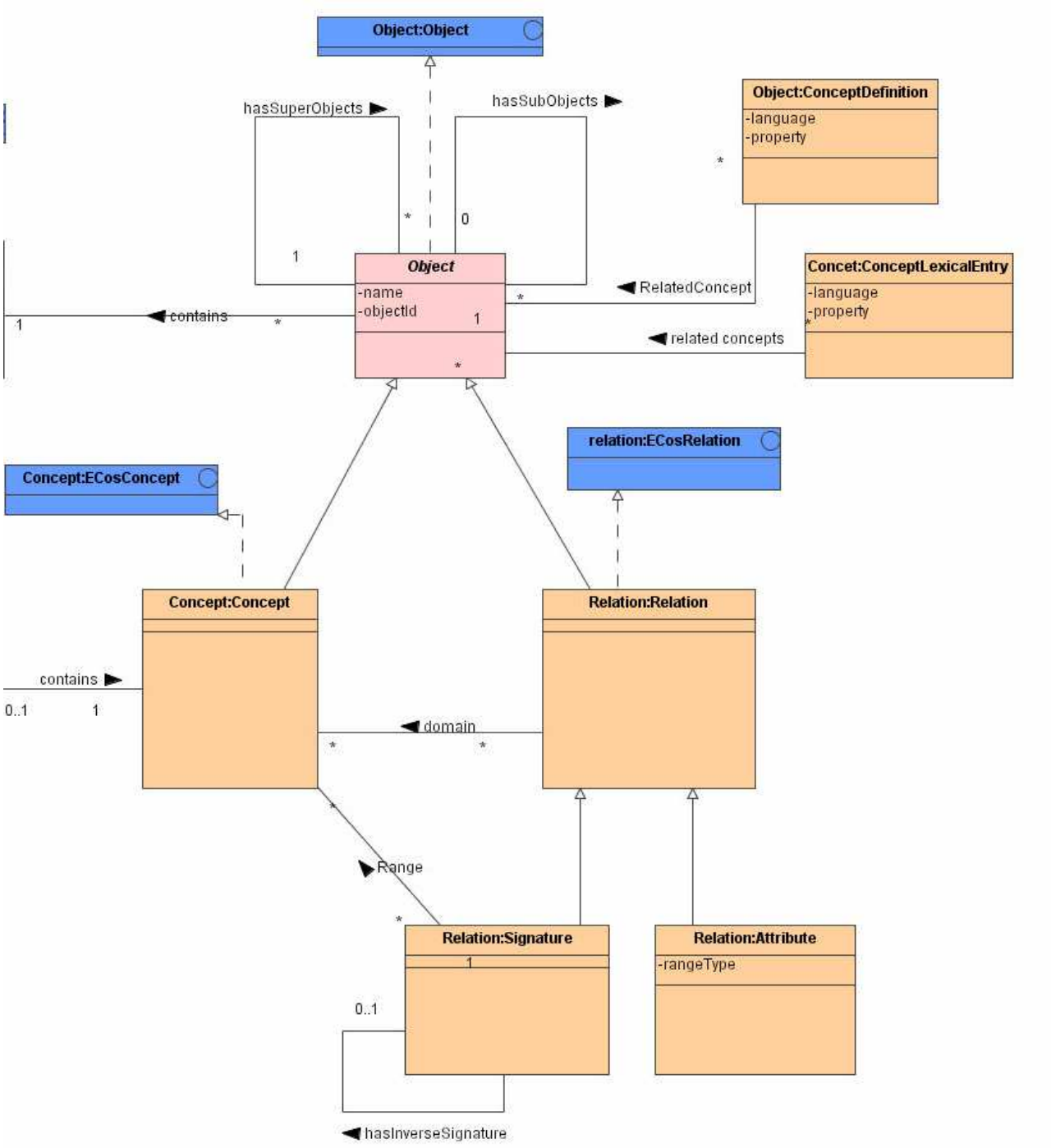

Figure 23 – Représentation graphique UML du méta-schéma de l'ontologie e-Cognos (Wetherill *et al.* 2002).

Pour caractériser un concept, le méta-schéma de l'ontologie définit la classe *Attribute*. L'autre spécialisation de la classe Relation est la classe *Signature* qui permet de lier deux concepts. La classe Concept ou *eCognosConcepts* (Figure 24) est spécialisée dans les classes *Project*, *Technical Topics*, *Related Domain*, *Process*, *Product*, *Resources* et *Actor*. Ces classes de concepts se retrouvent au premier niveau sous la racine de l'ontologie (Figure 22).

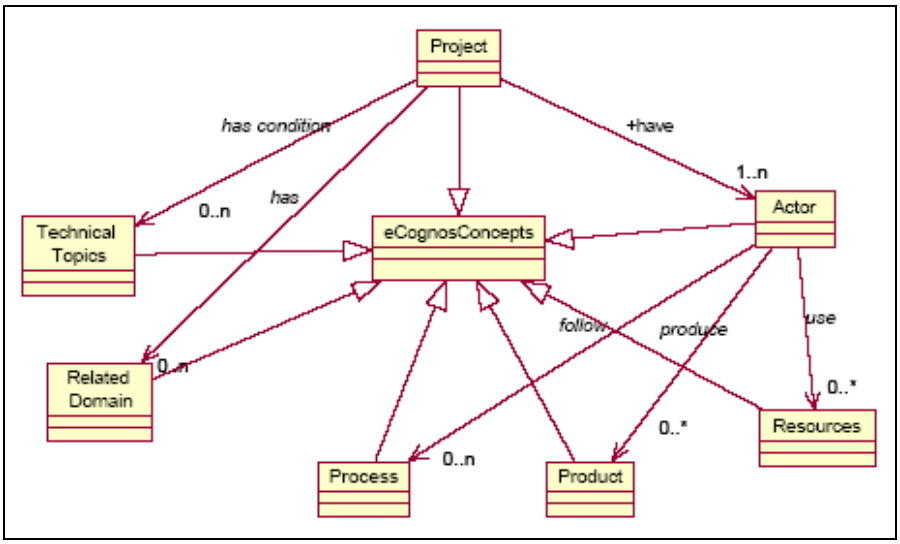

Figure 24 – Ensemble de concepts qui spécialisent la classe eCognosConcept (Wetherill *et al.* 2002).

L'idée principale sous-jacente à cette ontologie est résumée dans phrase suivante, d'où découlent les concepts de base de l'ontologie : dans le contexte d'un projet (*Project*), un groupe d'acteurs (*Actor*) utilise un ensemble de ressources (*Resources*) pour produire un ensemble de produits (*Product*) suivant certains processus (*Process*), dans un environnement de travail (*Related Domain*) et par rapport à des conditions données (*Technical Topics*).

4.2.3 Le standard ISO 12006-3

L'ISO 12006-3 (ISO 12006-3:2007) est un standard spécifique pour le domaine de la construction qui définit un schéma pour des modèles de taxonomies. Il permet de définir des concepts et des propriétés spécifiques. Ce standard permet aussi de grouper des concepts et de définir des relations entre concepts. L'ensemble de propriétés associées à un objet fournit sa définition formelle et son comportement. L'entité *xtdRoot* est la racine du modèle et fournit les caractéristiques minimales nécessaires pour qu'un objet appartienne au modèle. *xtdRoot* est une entité abstraite (ABS) qui est spécialisée dans une des entités *xtdObject*, *xtdRelatonship* ou *xtdCollection*. L'entité *xtdObject* est elle aussi un concept abstrait qui est subdivisé dans les entités *xtdActor*, *xtdSubject*, *xtdActivity*, *xtdUnit*, *xtdProperty* et *xtdMeasureWithUnit*. Par exemple, xtdActivity est utilisé pour décrire les processus. Les autres classes du modèle (Figure 25) servent à décrire les objets du domaine et les relations entre eux. L'entité xtdProperty est utilisée pour décrire ou quantifier un concept et peut contenir différents types de données.

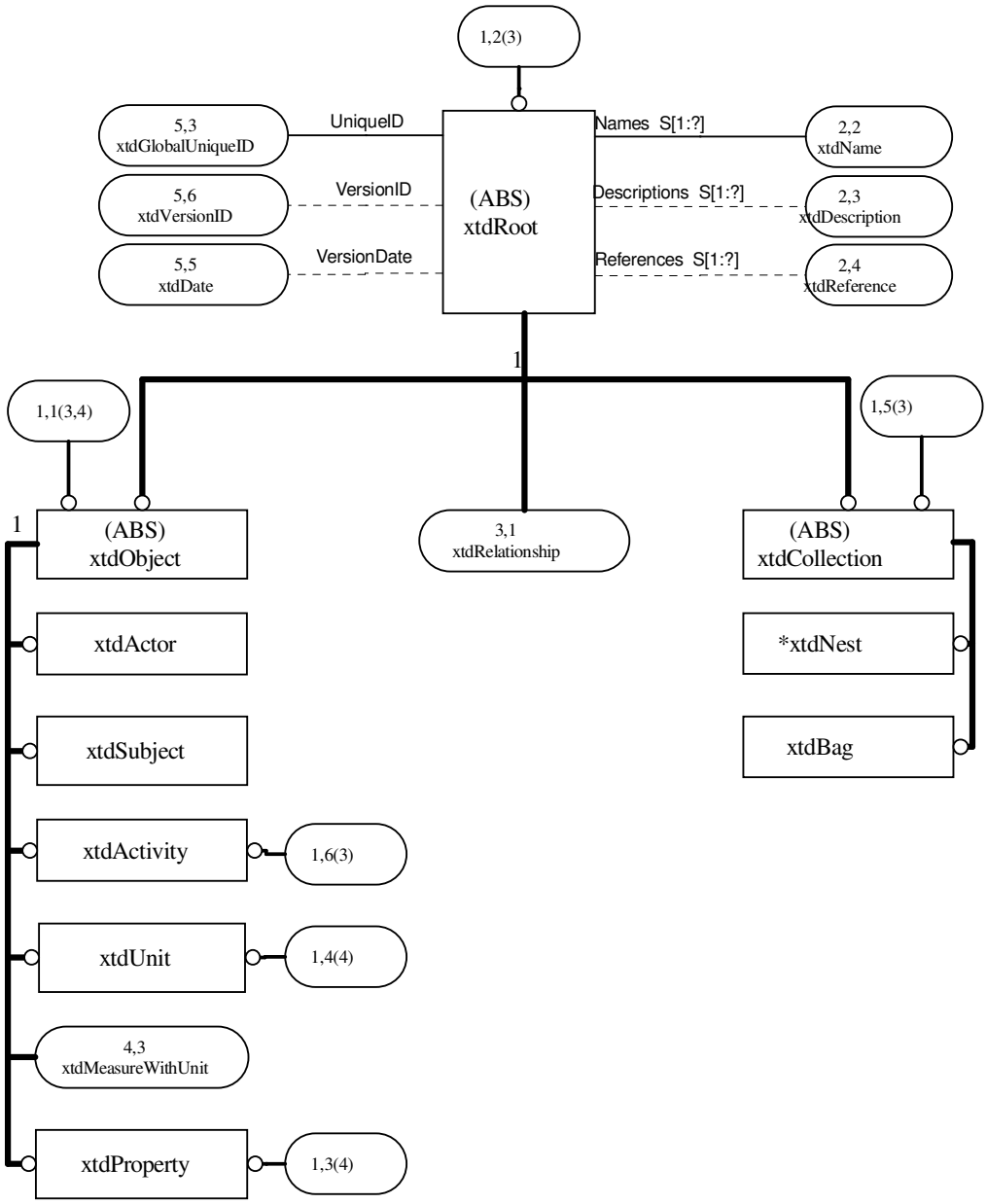

Figure 25 – Entités de base du méta-schéma d'ISO 12006-3, représentées dans le langage graphique EXPRESS-G (ISO/PAS 12006-3 version 2, 2002).

L'entité xtdRelationship fournit un mécanisme de connexion entre les objets. Un ensemble limité de types de relation peut ainsi être défini (Figure 26). L'entité *xtdRelCollects* permet de relier un objet à une collection d'autres objets. Une relation de participation peut être représentée en utilisant l'entité *xtdRelActsUpon*. Notons que les éléments graphiques du langage EXPRESS-G (ISO 10303-11: 2001) ont une correspondance avec les entités du langage EXPRESS présenté au chapitre 2.

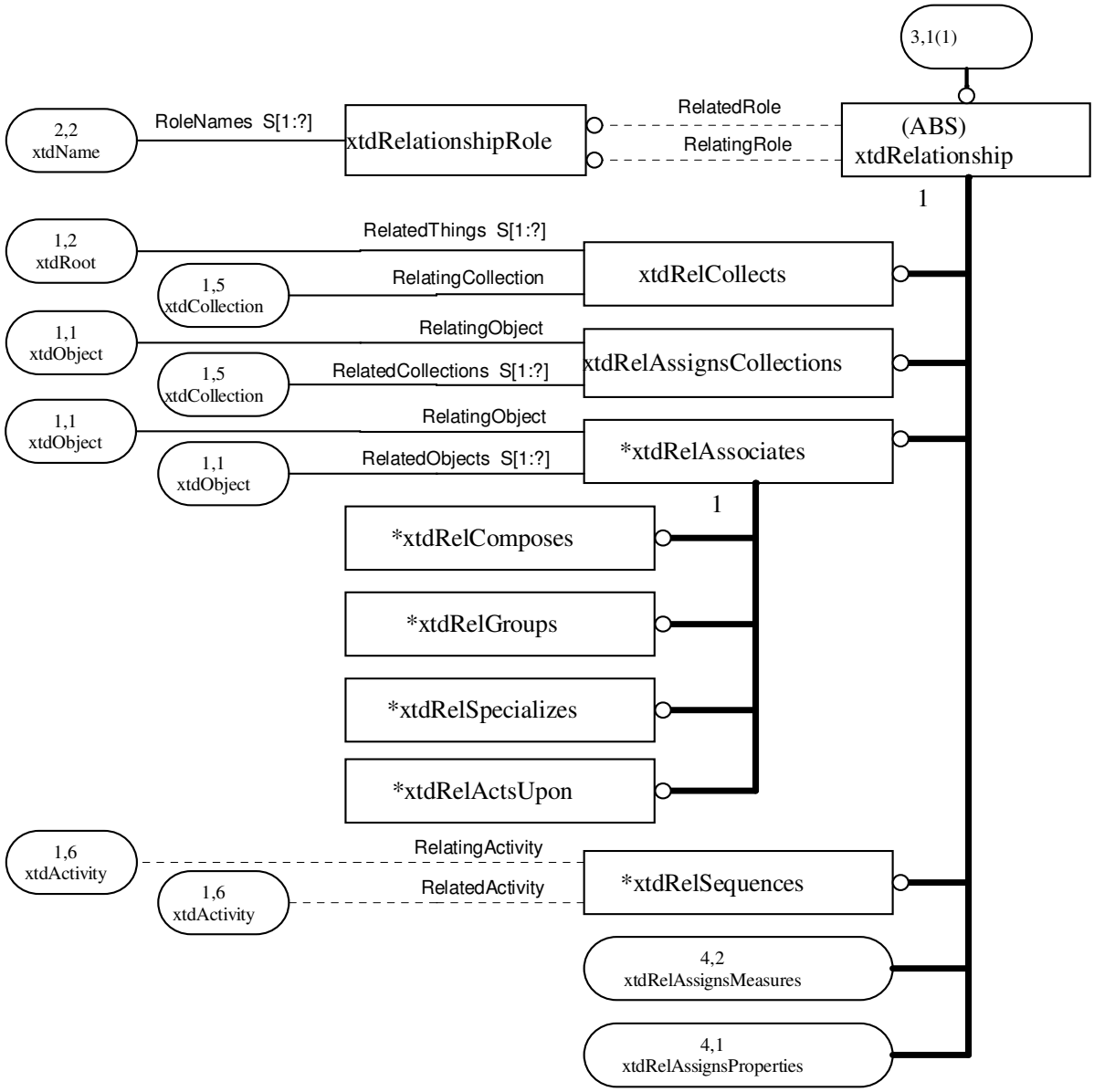

Figure 26 – Détails de l'entité xtdRelationship du méta-schéma d'ISO 12006-3, représenté en EXPRESS-G (ISO/PAS 12006-3 version 2, 2002).

Le standard 12006-3 sert de modèle à plusieurs ressources sémantiques. Par exemple, la taxonomie Lexicon[55] de l'institut STABU et BARBi[56] (*Norwegian Building and Construction Reference Data Library*) sont fondées sur ce standard.

[55] http://www.stabu-lexicon.nl/

[56] http://www.barbi.no/index.jsp

4.2.4 Le standard IFC

Le standard IFC[57] (ISO/PAS 16739:2005) est développé par *l'International Alliance for Interoperability* (IAI). Ce standard a été développé pour permettre l'échange et le partage de modèles d'information de la construction. Ceci pour mieux gérer les opérations de conception, de mise en œuvre et de maintenance dans le cycle de vie des projets de construction à travers des applications logicielles. Le modèle d'architecture d'IFC contient plus de 400 entités distribuées dans quatre couches : *resource*, *core*, *interoperability* et *domain*. Nous ne détaillons qu'une partie de la couche *core* : le niveau *kernel*, parce que celui-ci fournit les concepts de plus haut niveau qui sont communs à tous les autres schémas du standard. Ce standard permet seulement l'héritage simple. Dans le kernel, il y a trois entités fondamentales qui héritent de l'entité *IfcRoot* (Figure 27) :

- *IfcObject* – est l'entité qui généralise tous les objets sémantiques à traiter dans un modèle IFC. IfcObject est une entité abstraite prévue pour, à travers ses sous-entités, représenter les objets du monde de la construction, tels que les murs, poutres, revêtements, projets de constructions, processus, tâches de travail, coûts, ressources, acteurs impliqués dans les projets, etc. Ces concepts sont instanciés à travers les sept entités spécialisant l'entité IfcObject : *IfcProduct*, *IfcProcess*, *IfcControl*, *IfcResource*, *IfcActor*, *IfcGroup*, and *IfcProject*. Le contexte de ces concepts est ensuite défini par les relations dans lesquelles ils sont impliqués.
- *IfcRelationship* – est l'entité qui généralise toutes les relations entre des objets dans un modèle IFC. IfcRelationship permet de représenter les relations entre les concepts. Cette entité est sous-dévisée en cinq types de relations : assignement (*IfcRelAssigns*), définition (*IfcRelDefines*), décomposition (*IfcRelDecomposes*), association (*IfcAssociates*) et connectivité (*IfcRelConnects*).
- *IfcPropertyDefinition* – est l'entité qui généralise toutes les caractéristiques, qui sont aussi des types d'entités ou des ensembles de propriétés, et qui peuvent être attribuées aux objets. IfcPropertyDefinition sert à définir les propriétés des concepts. Ces propriétés peuvent être individuelles ou groupées. Les propriétés représentent des caractéristiques qui peuvent être partagées entres plusieurs instances de concepts.

[57] http://www.iai-international.org/Model/R2x3_final/index.htm

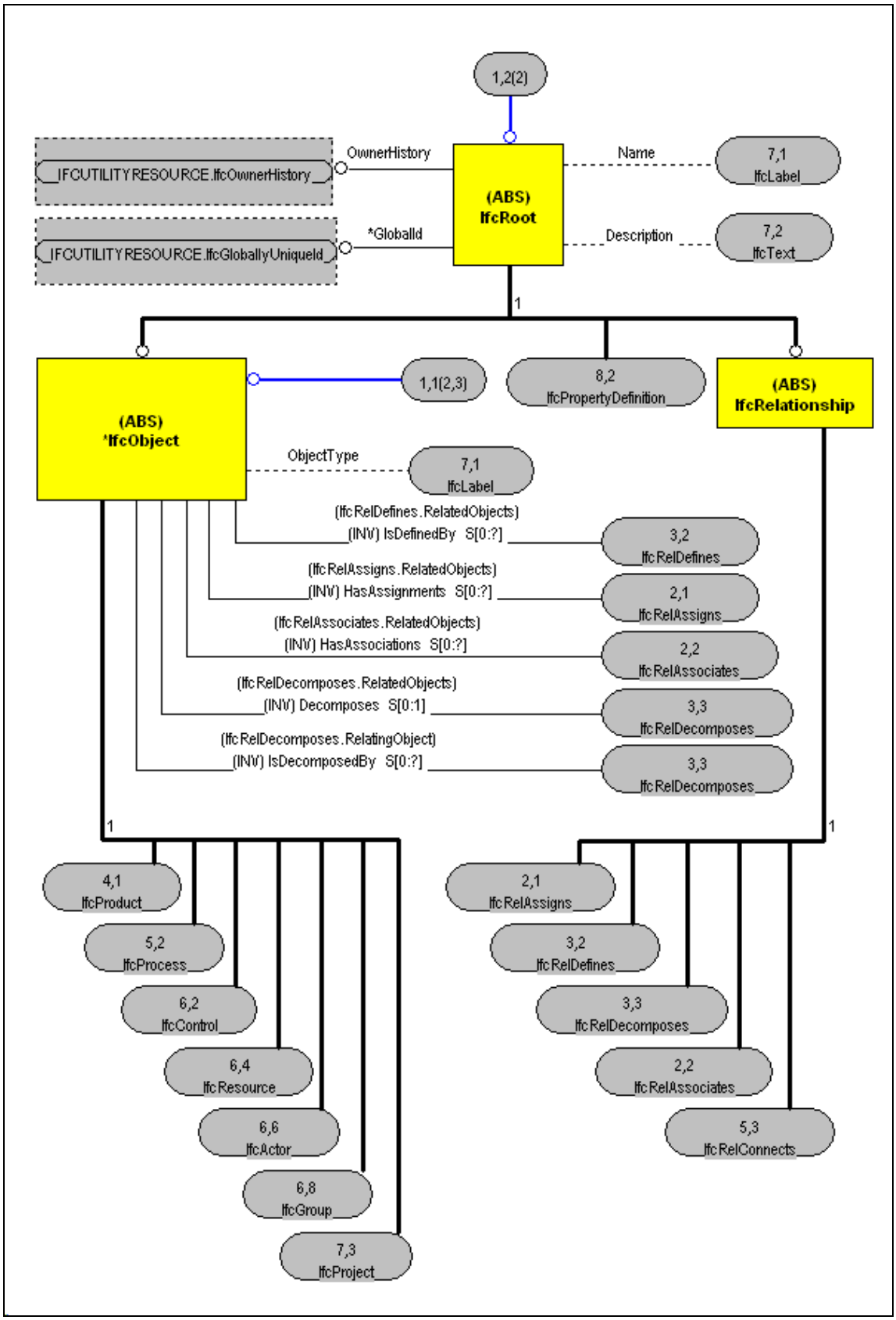

Figure 27 – Représentation en EXPRESS-G du méta-schéma racine de kernel IFC version 2x2 qui a été utilisé pour ce travail. La version la plus récente, IFC2x3, ajoute l'entité abstraite *IfcObjectDefinition*.

4.2.5 Comparaison des quatre ressources sémantiques

Au delà des différences entre les langages de représentation utilisés pour les quatre méta-schémas, ceux-ci présentent des différences entre leurs modèles conceptuels. Le Tableau 9 compare quelques entités, des quatre modèles, qui ont des significations proches. Par exemple, les *items Property* et *Assign properties to objects* dans les schémas d'e-Cognos et de bcBuildingDefinitions n'existent qu'à travers l'assignement à un objet (concept). Elles n'ont pas d'existence par elles-mêmes. Dans les standards ISO 12006-3 et IFC, les valeurs *Property* (ligne 4) xtdProperty et IfcProperty sont des entités qui peuvent avoir une existence autonome et être liées aux autres entités (concepts) lorsque nécessaire pour l'instanciation du schéma.

Tableau 9 – Similarités entre les entités des quatre méta-schémas.

Meta-schema / *Item*	*eCognos*	*bcXML*	*ISO 12006-3*	*IFC*
Semantic Resource identification	Object (abstract)	Taxonomy	XtdRoot (ABS)	IfcRoot (ABS)
Concept	eCognosConcept	Object	XtdObject (ABS)	IfcObject (ABS)
Relation	Relation	Relationship	xtdRelationship	IfcRelationship
Property	Attribute	Property	xtdProperty	IfcProperty
Assign properties to objects	Attribute	Property	xtdRelAssignsProperties	IfcRelDefinedByProperties
Description	ObjectConceptDefinition	Description	xtdDescription	Description (attribute)
Reference to an external resource / entity	-	ExternalReference	xtdReference	IfcExternalReference

Un autre exemple qui montre la différence entre les paradigmes de représentation ainsi que de points de vue est l'*item Description*. Dans le modèle IFC cet item est un attribut nommé *Description*. Donc, cet attribut existera attaché à une entité seulement, *i.e.* sans existence autonome. Alors que, dans le modèle ISO 12006-3, le même *item* est représenté par le concept *xtdDescription* qui a un statut plus indépendant dans son modèle.

4.3 Scénario d'utilisation : l'infrastructure OSIECS

Les ressources présentées ci-dessus sont reprises dans un scénario proposé pendant le projet FUNSIEC afin de présenter une utilisation possible du prototype de découverte de correspondances sémantiques. Dans ce scénario, les acteurs peuvent accéder, en utilisant l'outil ou les logiciel(s) associé(s) aux ressources sémantiques, aux informations des autres ressources

sémantiques en passant par les éléments de liaison qui sont le méta-modèle et le modèle OSIECS (Figure 28). Ces éléments permettent aux clients du système une utilisation transparente des ressources. Par exemple, une requête écrite dans un format spécifique à travers l'outil préféré d'un client va viser plusieurs autres ressources sémantiques en simultané. OSIECS comprend les quatre ressources sémantiques sélectionnées : l'ontologie bcBuildingDefinitions, l'ontologie e-Cognos, le standard ISO 12006-3 (la ressource Lexicon est une instance de ce standard, voir le scénario de la Figure 28) et le standard IFC (niveau kernel).

Le méta-modèle et le modèle OSIECS sont des tables identifiant les correspondances entre concepts des ressources sémantiques. Le méta-modèle OSIECS est l'ensemble des tables de correspondances sémantiques entre les quatre méta-schémas sélectionnés (*cf.* section 4.2).

Le modèle OSIECS est l'ensemble des tables qui établissent des correspondances sémantiques entre les schémas des ressources sémantiques qui composent OSIECS. Il a été obtenu à partir de l'ontologie e-Cognos et de la taxonomie bcBuilingDefinitions.

Le *Kernel* OSIECS est l'ensemble du processus qui permet de convertir les ressources sémantiques à partir de leurs formats initiaux vers le format commun LD et de retrouver les correspondances sémantiques en utilisant un moteur d'inférences fondé sur les LD. Ces fonctionnalités se retrouvent dans le prototype FUNONDIL.

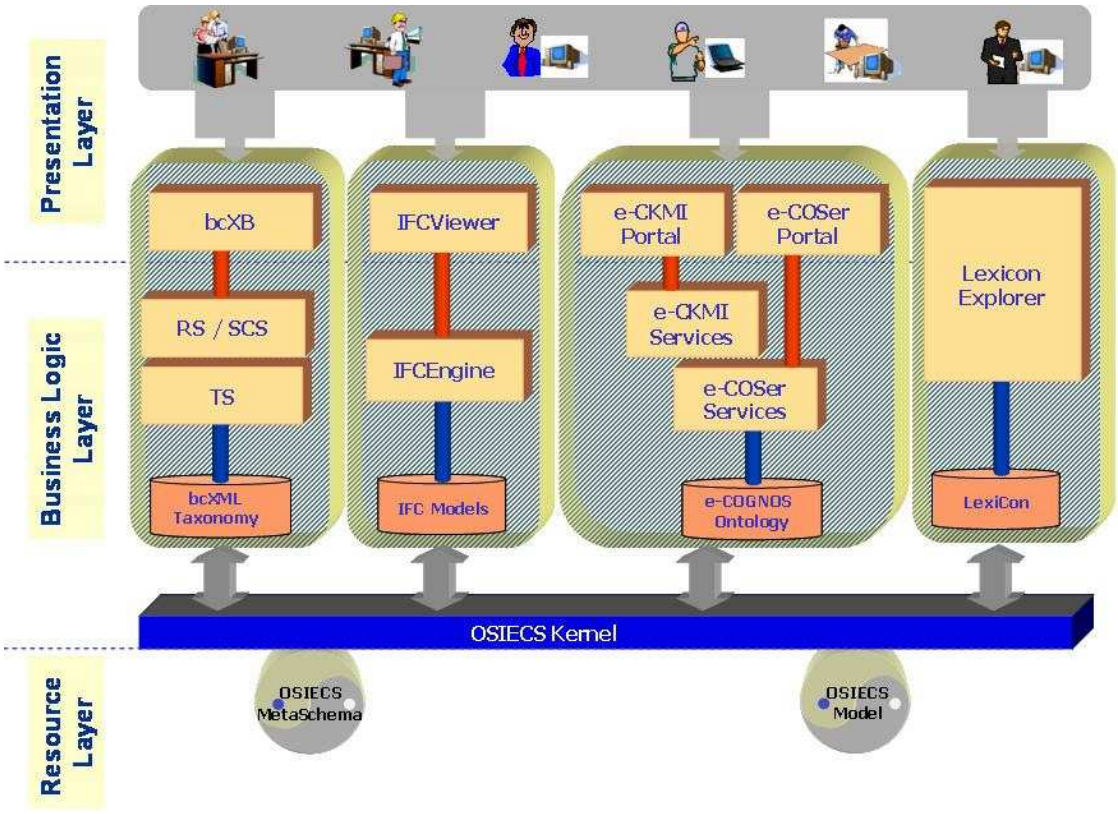

Figure 28 – Scénario d'application dans le projet FUNSIEC (Lima *et al.* 2005a).

Pour se familiariser avec des scénarios d'utilisation de chacune des ressources sémantiques, le site FSEC (*cf.* section 4.1) rend disponibles des informations supplémentaires, notamment des vidéos sur les scénarios. Les prototypes réalisés sont présentés dans la section suivante.

4.4 Prototypes réalisés

Cette partie présente la mise en œuvre opérationnelle des outils méthodologiques présentés au chapitre précédent. Nous montrons les fonctionnalités et le mode opératoire des outils réalisés : les prototypes FUNONDIL et UNIREL.

4.4.1 Le prototype FUNONDIL

Le prototype FUNONDIL a été développé pendant le projet FUNSIEC. L'objectif principal de ce prototype est la découverte de correspondances entre concepts de ressources sémantiques du secteur de la construction. Les correspondances produites sont destinées à alimenter le méta-modèle et le modèle OSIECS. Le calcul des correspondances est effectué avec le prototype ONDIL (Le Duc 2004) qui est intégré dans FUNONDIL. ONDIL incorpore le moteur FaCT (*cf.* 3.2.3.2).

FUNONDIL a été développé en Java et est installé dans un serveur Tomcat. Le prototype traite les ressources sémantiques qui sont représentées dans les langages de modélisation UML (XMI), EXPRESS et schémas XML (XSD). Les connaissances représentées dans les diagrammes UML et EXPRESS, tels que les classes, les relations, les entités, les attributs, les héritages, les associations et contraintes sont traitées et traduites dans le langage OWL-DL.

La Figure 29 montre une copie d'écran du prototype FUNONDIL. Le menu de gauche donne accès aux fonctionnalités suivantes :

- "READ ME FIRST" permet de visualiser une courte explication sur le fonctionnement de l'outil et sur les types de correspondances sémantiques qu'il est possible d'obtenir avec cet outil (menu principal de la Figure 29).
- "Loading Meta-schema" assure le chargement et la compilation des ressources sémantiques à partir des formats d'origine EXPRESS, XMI, XSD et XML vers la syntaxe OWL-DL et la syntaxe classique des LD.

- "Visualising Meta-schema" permet à l'utilisateur de visualiser les ressources sémantiques en des formats différents : syntaxe OWL, syntaxe LD classique et en représentation graphique avec Treebolic[58].
- "Adding Axioms" prépare le processus de production de correspondances. Cette fonctionnalité permet de charger un fichier contenant des axiomes d'amorçage entre concepts des ressources sémantiques. Elle permet aussi de définir les axiomes directement en faisant référence aux identifiants des concepts des ressources chargées au préalable dans l'outil.
- "Producing Mappings" demande à l'utilisateur de choisir, parmi les ressources sémantiques chargées, les ressources à comparer et de lancer le calcul des correspondances.
- "OSIECS Meta-model" présente le méta-modèle OSIECS. Ce méta-modèle contient les correspondances sémantiques auparavant calculées entre les quatre ressources sélectionnées (*cf.* section 4.2).
- "OSIECS model" montre le modèle OSIECS. Cette option permet de visualiser les correspondances sémantiques entre deux ressources sélectionnées.
- "Basic Inferences" fournit l'accès aux services d'inférence du moteur. Cette fonctionnalité permet de tester la satisfiabilité ou d'interroger sur la subsomption et l'équivalence entre deux concepts des ressources sémantiques.

Le prototype FUNONDIL peut être testé en ligne[59]. Nous exposons ensuite les paramètres nécessaires à son utilisation.

4.4.1.1 Chargement des ressources sémantiques

La fonctionnalité de chargement des ressources sémantiques fournit quatre sous-menus qui correspondent aux quatre formats traités par le prototype (Figure 30) : EXPRESS, XMI (format d'échange des diagrammes UML), XSD (schémas XML) et XML avec des étiquettes spécifiques au moteur FaCT. L'utilisateur choisit le format et nomme la ressource (Figure 31). Ce nom est l'identifiant de la ressource tout au long de la session. Ensuite l'utilisateur navigue dans le système de fichiers de la machine pour sélectionner la ressource (Figure 32) et lance le chargement de la

[58] http://treebolic.sourceforge.net/en/home.htm

[59] Ce prototype est installé dans un serveur du CSTB à l'url http://195.83.41.67/ondil/InferenceEngine

ressource dans l'outil. L'ordre de chargement lance par la suite la conversion de la ressource aux formats LD classique et OWL-DL. Notons qu'une ressource sémantique est nommée *Ontology* dans ce prototype.

Le même processus doit suivre pour le chargement d'une deuxième ressource. Ceci fournit au prototype au moins deux ressources au cours d'une même session, qu'il faudra sélectionner plus loin afin de déclencher le calcul des correspondances sémantiques.

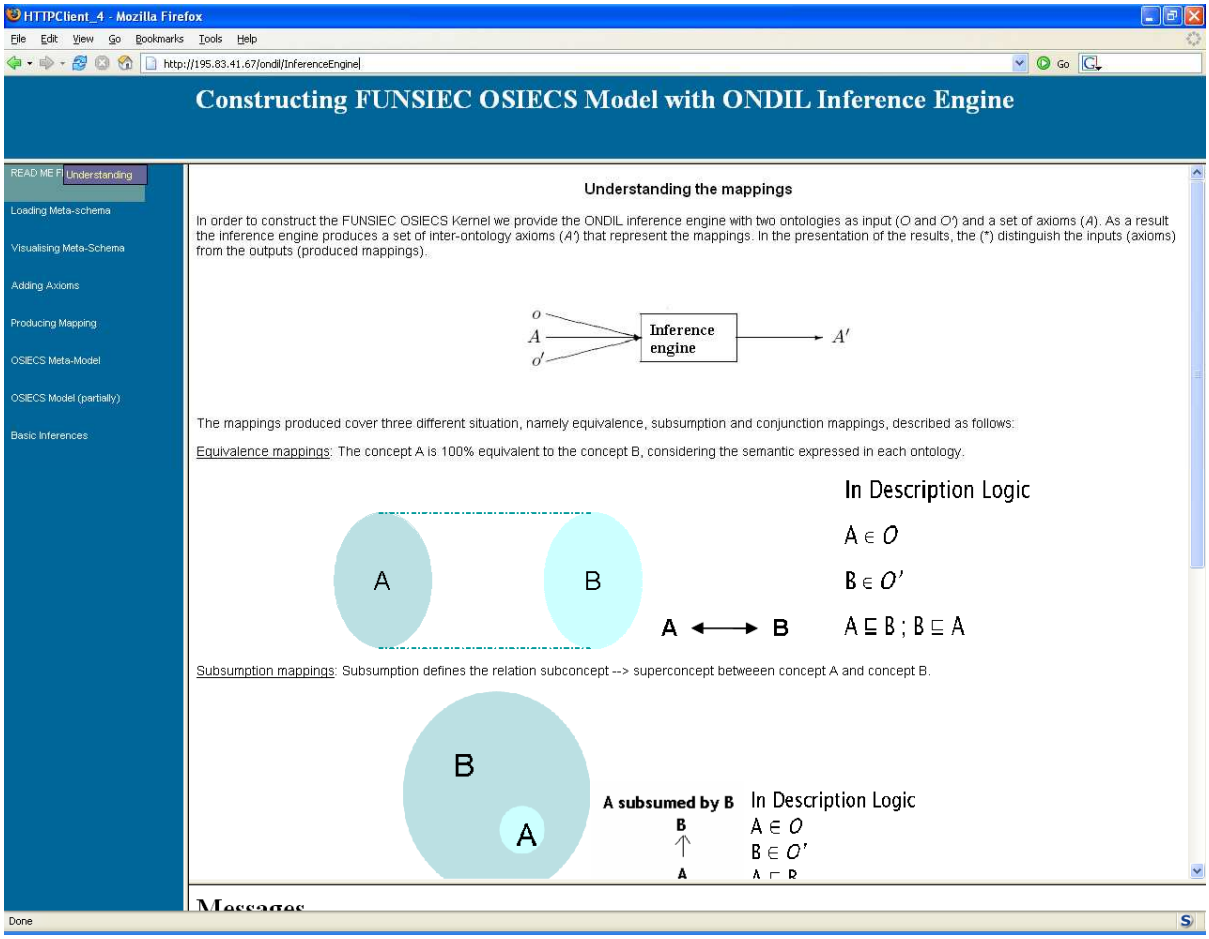

Figure 29 – Copie d'écran de la page web du prototype FUNONDIL. Explication sur les types de correspondances sémantiques.

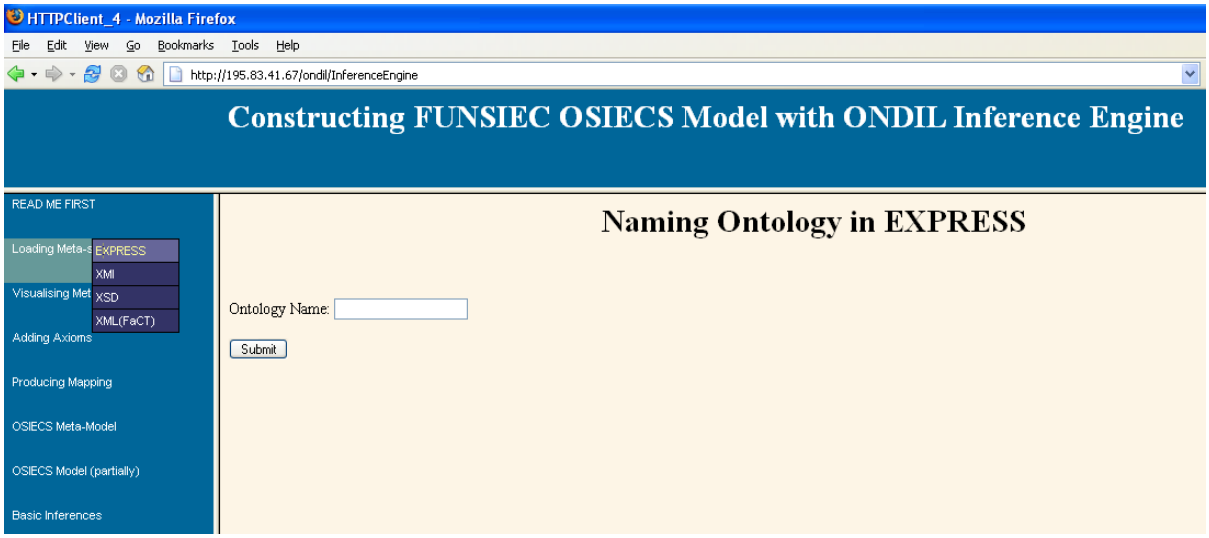

Figure 30 – Sous-menus des formats de représentation permis pour chargement des ressources sémantiques.

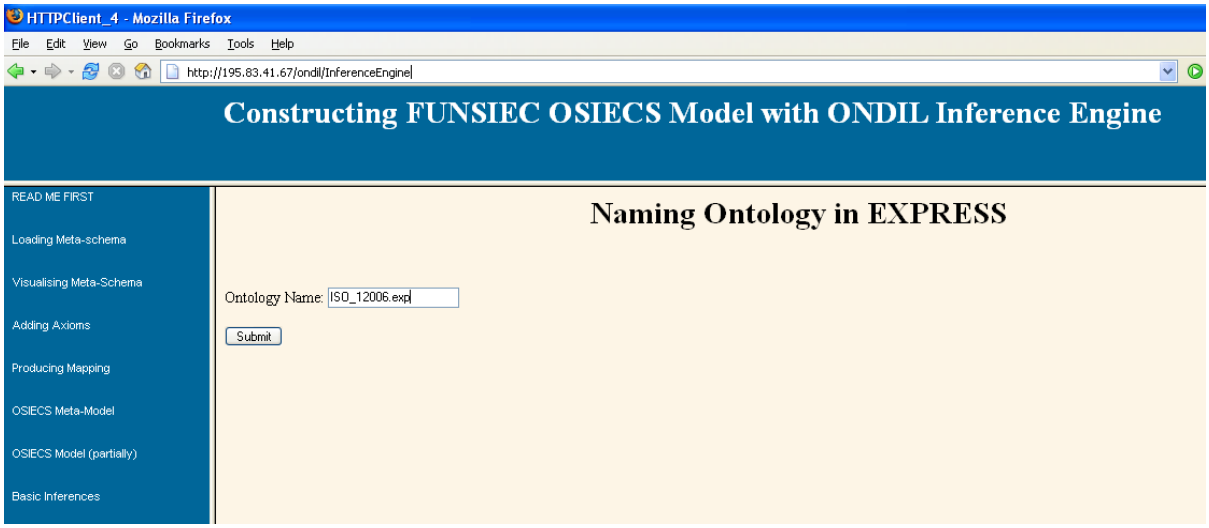

Figure 31 – Étape d'assignement d'identifiant à une ressource sémantique, après le choix du format de représentation et avant son chargement.

4.4.1.2 Visualisation des ressources sémantiques

Le menu de visualisation donne accès aux les trois options de représentation traitées par l'outil (Figure 33) :

- la syntaxe OWL ;
- la représentation graphique à travers l'outil Treebolic intégré dans le prototype ;
- la syntaxe classique en axiomes des logiques de description.

Après le choix de l'option de visualisation, l'utilisateur choisit le nom de la ressource dans la liste des ressources sémantiques qu'il a auparavant chargées. La Figure 34 montre la représentation graphique avec Treebolic d'une partie de la ressource ISO 12006-3.

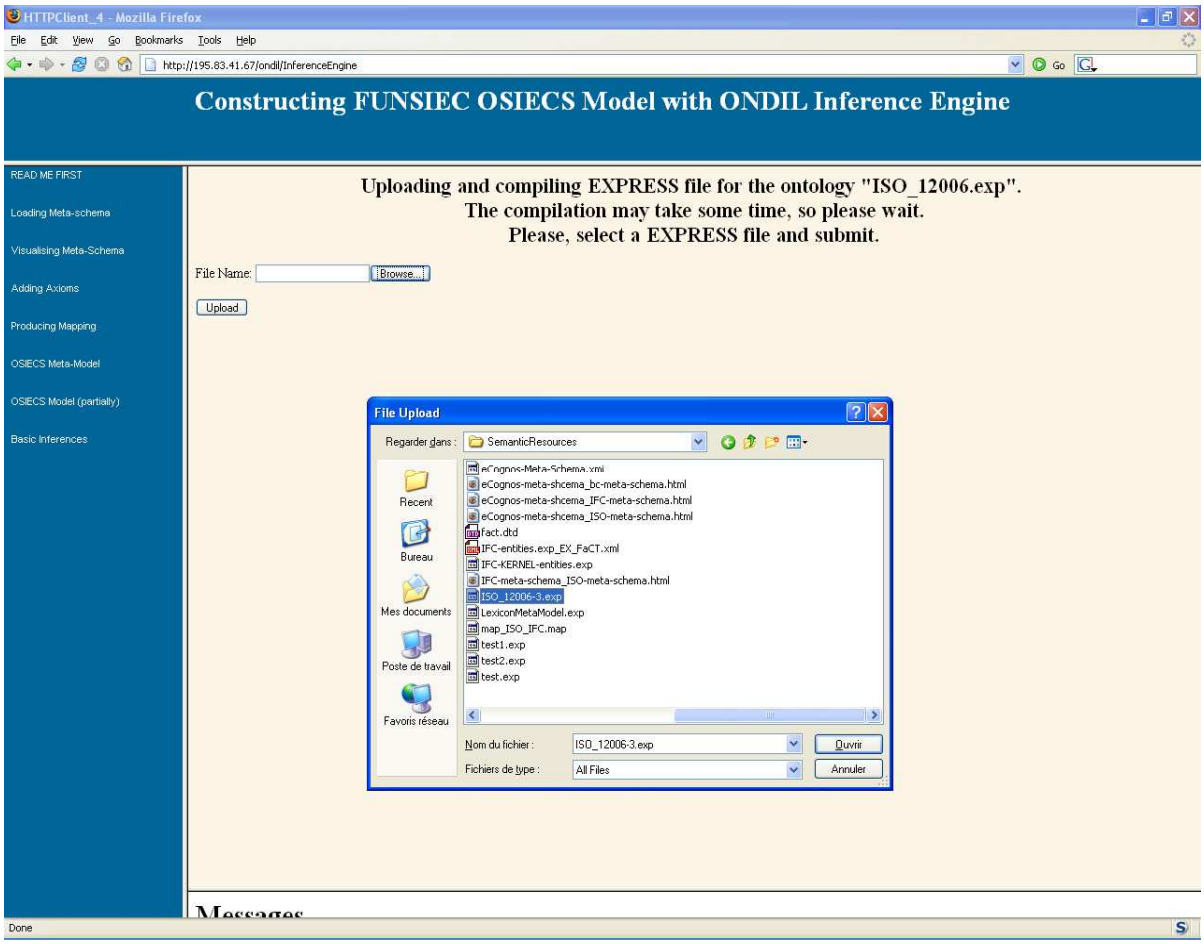

Figure 32 – Navigation dans le système de fichiers de la machine pour sélection du fichier à charger.

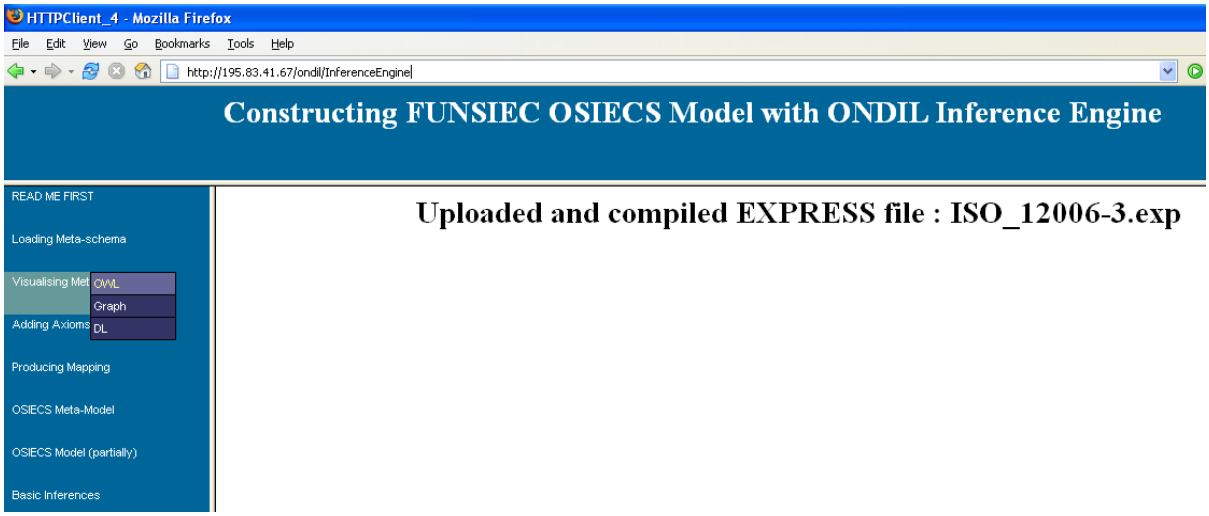

Figure 33 – Sous-menus des options de visualisation.

4.4.1.3 Processus de découverte des correspondances sémantiques

Avant de lancer le processus de calcul des correspondances, l'utilisateur peut charger un fichier contenant les axiomes d'amorçage ou les définir pendant la session de travail. Pour cela, l'utilisateur sélectionne le menu gauche "Adding Axioms" et choisit ensuite de charger un fichier contenant ces axiomes ou de les définir. Le chargement de fichier est une opération similaire à celle de chargement d'une ressource. Un exemple de définition d'axiomes d'amorçage est présenté dans la Figure 35. L'utilisateur choisit les deux ressources sur lesquelles il veut établir les axiomes d'amorçage. Ensuite, pour chaque ressource il choisit les concepts concernés par l'axiome. La sélection d'un concept se fait dans une liste déroulante. La sélection de deux concepts des deux ressources établit un axiome de subsomption entre ces deux concepts.

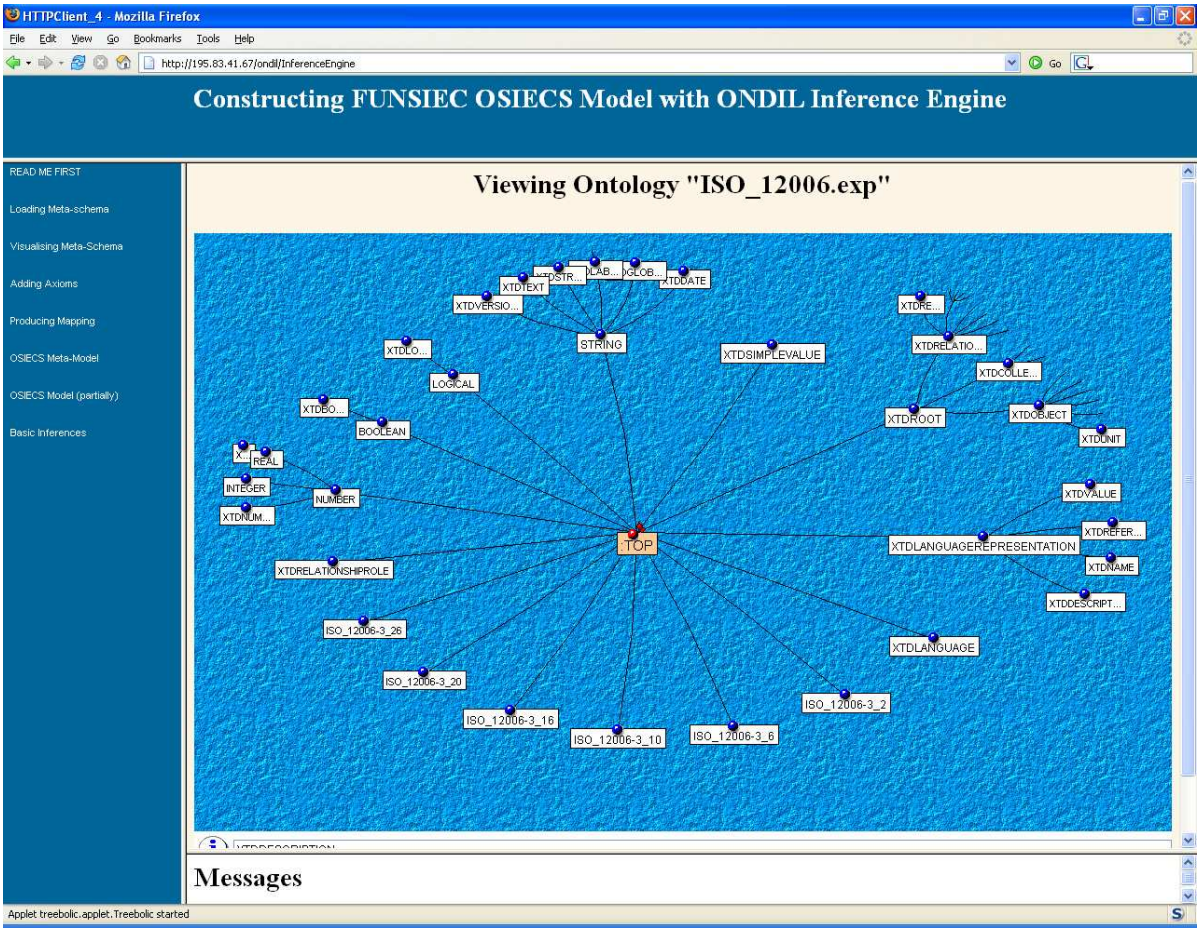

Figure 34 – Visualisation graphique d'une partie de la ressource ISO 12006-3 à travers l'outil Treebolic.

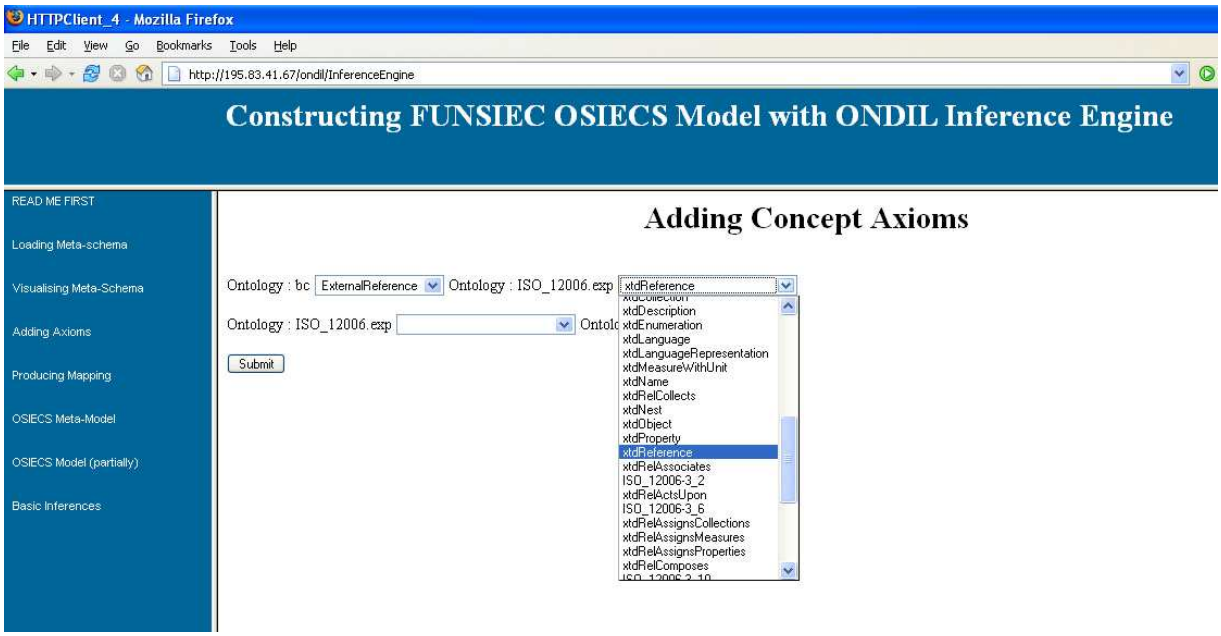

Figure 35 – Choix des noms des concepts des ressources dans la liste des concepts pour établir des axiomes d'amorçage.

La sélection de l'option "Produce Mappings" dans le menu de gauche amène l'utilisateur vers le choix des deux noms identificateurs des ressources à comparer (Figure 36). Il peut ensuite lancer le processus de découverte de correspondances. Le résultat du calcul des correspondances apparaît à l'utilisateur dans une fenêtre similaire à la Figure 37.

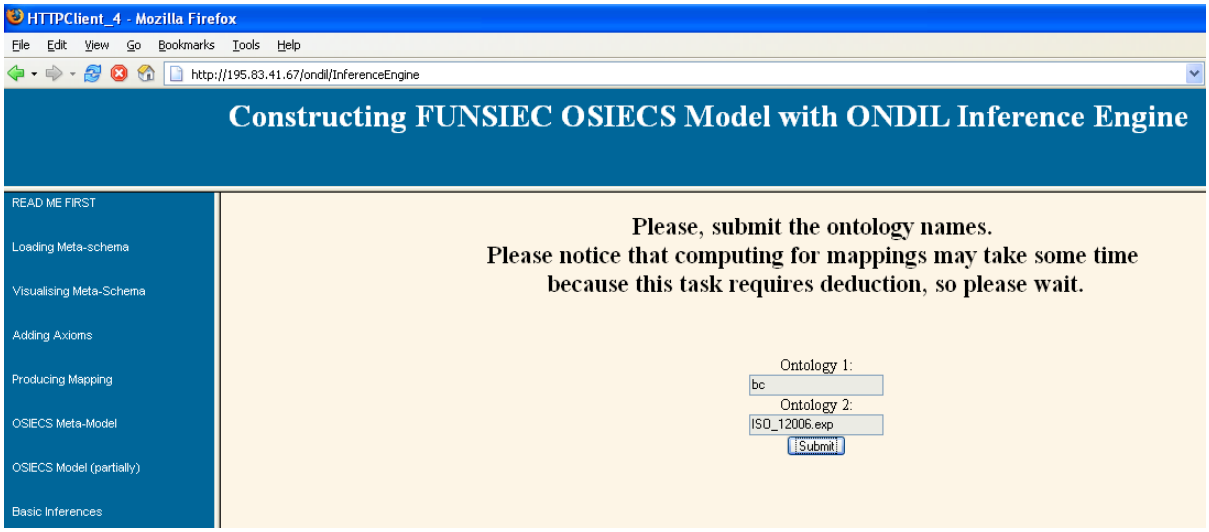

Figure 36 – Choix des noms des ressources pour lancer le processus de découverte de correspondances sémantiques.

4.4.1.4 Visualisation des correspondances sémantiques

L'utilisateur peut visualiser les correspondances sémantiques produites auparavant et qui constituent le méta-modèle et le modèle OSIECS (Figure 37), par le choix de l'option *ad hoc* dans le menu de gauche.

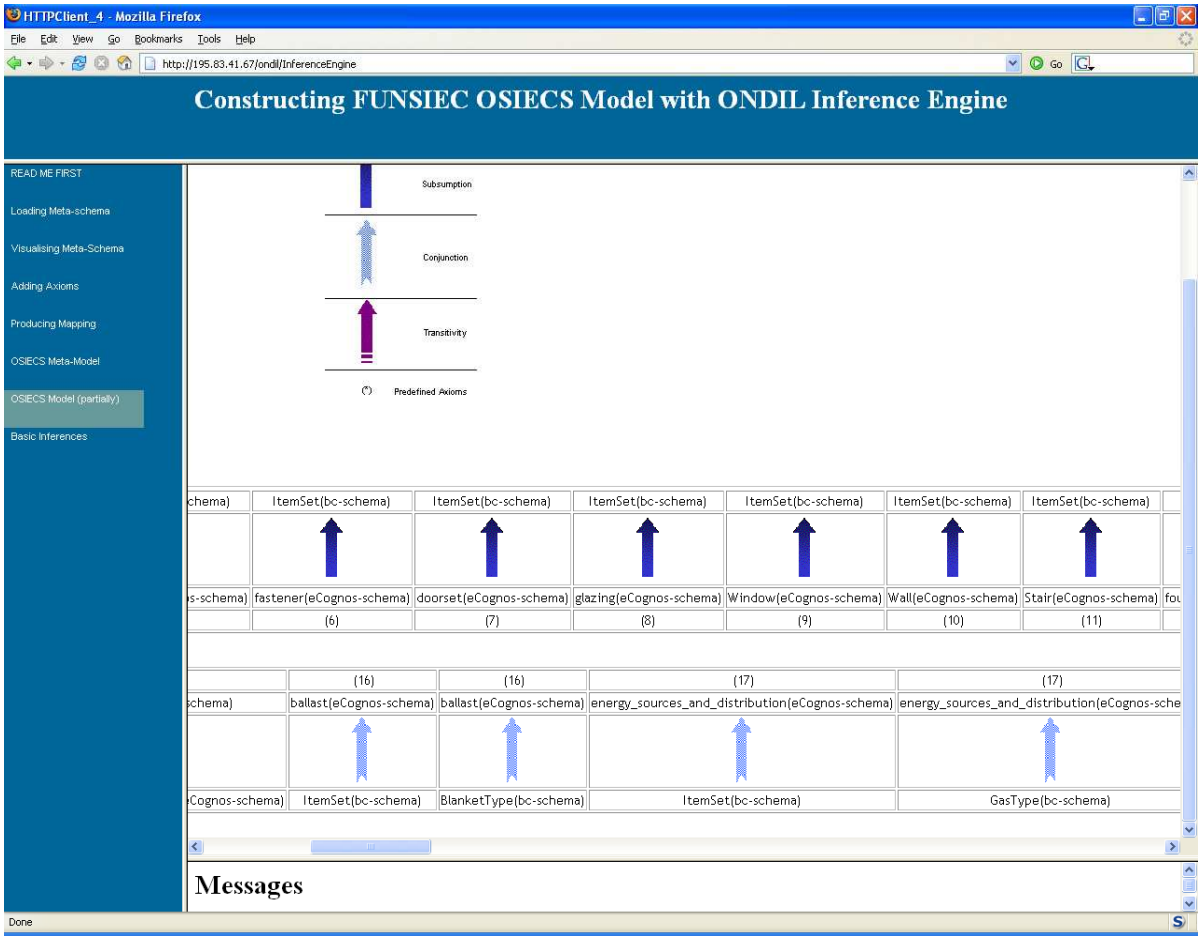

Figure 37 – Visualisation du modèle OSIECS.

4.4.2 Le prototype UNIREL

L'objectif principal du prototype UNIREL est l'implantation de la phase unification de propriétés de la méthodologie proposée, ainsi que de l'inférence d'instances. UNIREL réutilise le gabarit du prototype FUNONDIL et une partie de la technologie d'implantation. Les différences majeures en termes techniques sont l'utilisation de deux éléments externes :

- le moteur d'inférence FaCT est remplacé par Pellet version 1.5.0 (*cf.* section 2.2.3), qui permet de traiter directement des ressources en OWL ; Pellet permet à la fois de raisonner sur les concepts et sur les instances.
- l'API OWL version 2.1.1 (*cf.* section 2.3.2) permet au noyau de l'application de manipuler directement des ressources en OWL-DL, de communiquer avec le moteur d'inférences et d'utiliser un ensemble de fonctionnalités supplémentaires indisponibles dans JENA (*cf.* section 2.3.1) ; c'est également à travers cette API que les correspondances sémantiques sont rendues.

UNIREL prend en entrée et traite les ressources représentées en OWL. Les processus mis à disposition par UNIREL ont été implantés en Java et les fichiers spécifiques à ces processus se trouvent à l'annexe 3. Certaines fonctionnalités d'UNIREL sont communes avec celles de FUNONDIL. Dans cette section nous mettons l'accent sur les fonctionnalités nouvelles. Le menu de gauche donne accès aux fonctionnalités suivantes :

1. "READ ME FIRST" permet de visualiser une courte explication sur le fonctionnement de l'outil et ses fonctionnalités.
2. "Load Semantic Resources" assure le chargement des ressources sémantiques en format OWL. Cette opération est similaire à celle du prototype FUNONDIL.
3. "Visualise Semantic Resources" permet à l'utilisateur de visualiser les ressources sémantiques en syntaxe OWL. La Figure 38 montre la représentation déclarative en OWL-DL de la ressource IFC-Kernel.
4. "Select Relations to Unify" est une fonctionnalité supplémentaire par rapport au prototype FUNONDIL. Elle permet de choisir deux relations à considérer comme équivalentes parmi les ressources chargées auparavant. L'utilisateur sélectionne ces relations dans des listes déroulantes (Figure 39).
5. "Produce Mappings" demande à l'utilisateur de choisir parmi les ressources sémantiques chargées au préalable, les ressources à comparer et permet de lancer ensuite le calcul des correspondances. Le résultat du calcul des correspondances apparaît à l'utilisateur dans une fenêtre similaire à la Figure 51, dépendant des ressources sélectionnées.
6. "Infer instances" fournit le moyen de sélectionner deux ressources sémantiques chargées pour ensuite lancer le processus d'inférence d'individus de ces ressources. L'utilisateur sélectionne les deux ressources à utiliser. Il peut demander toutes les instances des ressources ou sélectionner un concept particulier dont il veut inférer les individus (Figure 40). La présentation des résultats de l'inférence d'instances est similaire à la Figure 54, dépendant des options de sélection de l'utilisateur.

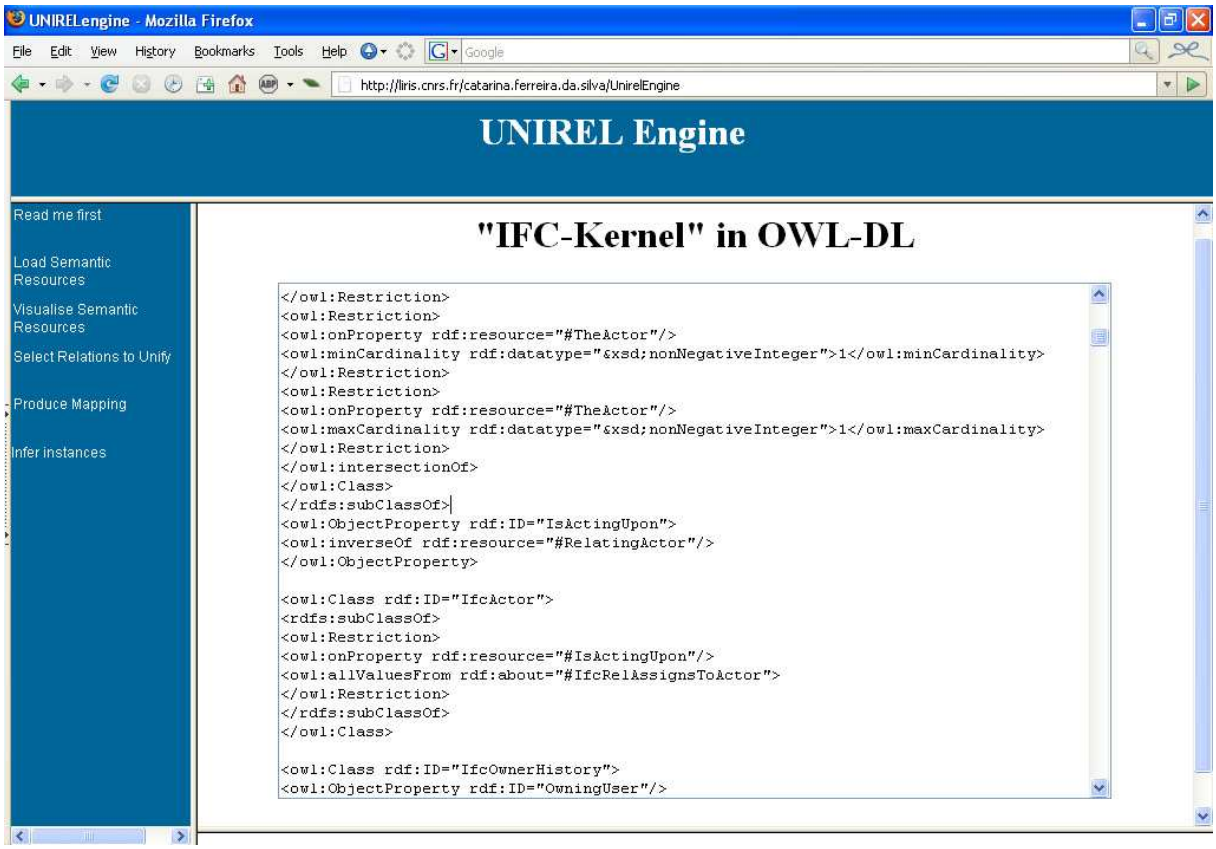

Figure 38 – Copie d'écran du prototype UNIREL : visualisation de la ressource IFC-Kernel en OWL-DL.

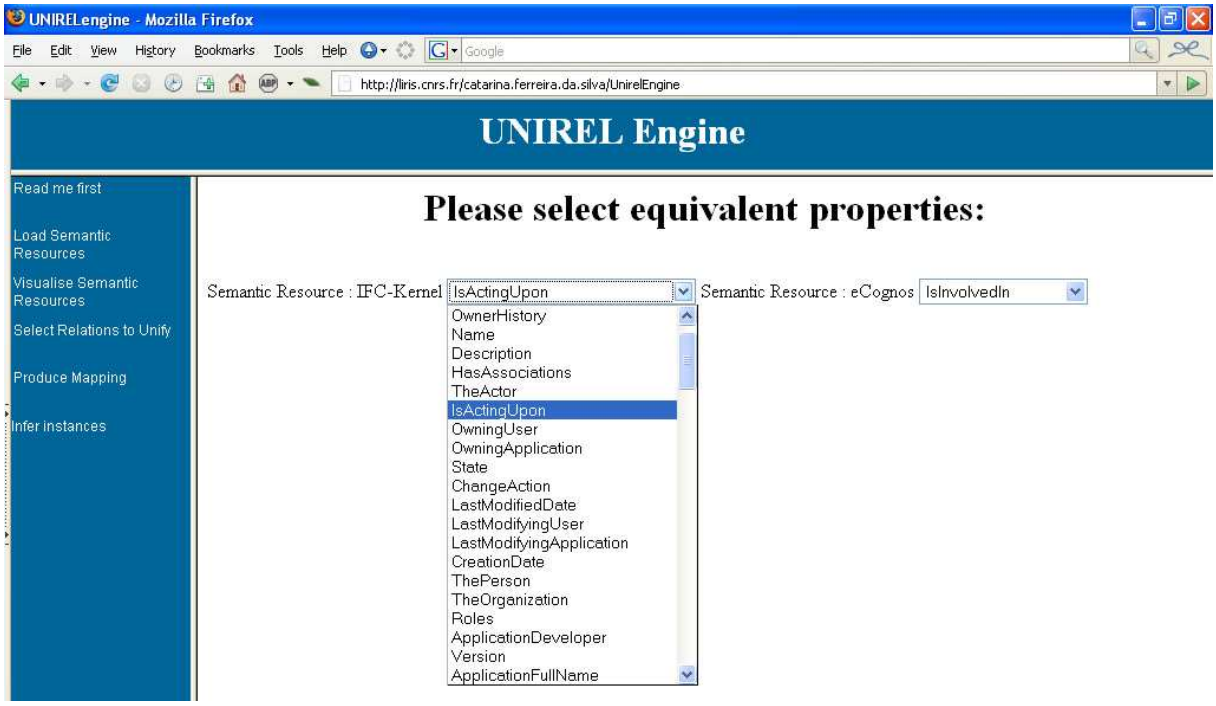

Figure 39 – Sélection des propriétés à rendre équivalentes.

Figure 40 – Sélection d'une classe d'une des deux ressources pour lancer l'inférence d'instances.

4.5 Synthèse

Dans ce chapitre, nous avons présenté le contexte d'application, qui est en partie celui du projet FUNSIEC. Nous avons également présenté un ensemble de ressources sémantiques du secteur européen de la construction. Ces ressources sémantiques servent de base au scénario d'application et sont utilisées pour tester et appliquer la méthodologie proposée pour la découverte de correspondances sémantiques. Deux prototypes ont été développés afin de mettre en œuvre certaines phases de la méthodologie proposée et d'autres fonctionnalités. Le prototype FUNONDIL inclut un noyau prévu pour être utilisé en middleware par d'autres applications et une interface développée pour permettre aux utilisateurs de le tester dans le cadre du projet FUNSIEC. Le prototype UNIREL réutilise la même interface, mais se fonde sur un moteur d'inférences plus récent. Cela a permis d'expérimenter de nouvelles fonctionnalités et d'enrichir les résultats obtenus avec le prototype précédent. Des tests ont été effectués avec les deux prototypes développés. Le chapitre suivant présente et discute les résultats obtenus à l'aide de ces deux prototypes.

Chapitre 5 – Évaluation et discussion

Ce chapitre présente et discute les résultats obtenus à l'aide des deux prototypes décrits au chapitre précédent. L'objectif est d'analyser ces résultats en termes quantitatifs et méthodologiques.

Ce chapitre présente les protocoles expérimentaux, les tests effectués et les résultats correspondant aux phases 3 (découverte de correspondances) et 4 (unification de propriétés) de la méthodologie proposée (Figure 13). Les sections 5.1 et 5.2 présentent et analysent les tests effectués dans le cadre du projet FUNSIEC, avec le moteur FUNONDIL et les 4 ressources sélectionnées présentées au paragraphe 4.2. Ensuite, la section 5.3 détaille les tests correspondant à la phase d'unification des propriétés, effectuées avec le prototype UNIREL. Finalement, la section 5.4 analyse globalement les résultats obtenus dans les différentes séquences de tests et discute les résultats obtenus.

5.1 Découverte de correspondances entre les méta-schémas des ressources sémantiques

Avant de présenter les résultats des différents tests effectués, nous avons jugé utile d'indiquer les conditions expérimentales, et en particulier les ensembles d'axiomes d'amorçage utilisés pour chaque processus de découverte de correspondances. Les sous-sections suivantes se rapportent au processus de découverte de correspondances sémantiques entre les ressources deux à deux. Dans l'annexe 4 se trouve la représentation OWL des correspondances sémantiques découvertes entre les méta-schémas de bcBuildingDefinitions, e-Cognos, IFC-kernel et ISO 12006-3. Ce fichier contient une centaine de correspondances validées par les experts du domaine.

5.1.1 Découverte de correspondances entre les méta-schémas de bcBuildingDefinitions et e-Cognos

Le Tableau 10 présente les axiomes d'amorçage entre les méta-schémas de bcBuildingDefinitions et e-Cognos, qui ont été établis par les experts. Une partie des correspondances sémantiques qui résultent du processus de découverte de correspondances entre les méta-schémas de bcBuildingDefinitions et d'e-Cognos est présentée dans la Figure 41. La légende des flèches se trouve en haut à droite dans cette figure. Ces correspondances sont pour une grande partie entre

des types de données ou entre un concept d'une ressource sémantique et un type de données de l'autre ressource (21). Ces résultats ont été soumis aux experts du domaine afin qu'ils déterminent les correspondances qu'il est pertinent de conserver. Les correspondances entre types de données n'ont par exemple pas été retenues comme valides.

$$\texttt{bcBuildingDefinitions-ms:Explanation} \sqsubseteq \texttt{Cognos-ms:String} \quad (21)$$

Tableau 10 – Axiomes d'amorçage entre les méta-schémas de bcBuildingDefinitions et e-Cognos.

Relationship ⊑ Relation_Relation
Relation_Relation ⊑ Relationship
Description ⊑ Object_ConceptDefinition
Object_ConceptDefinition ⊑ Description
Object ⊑ Concept_Concept

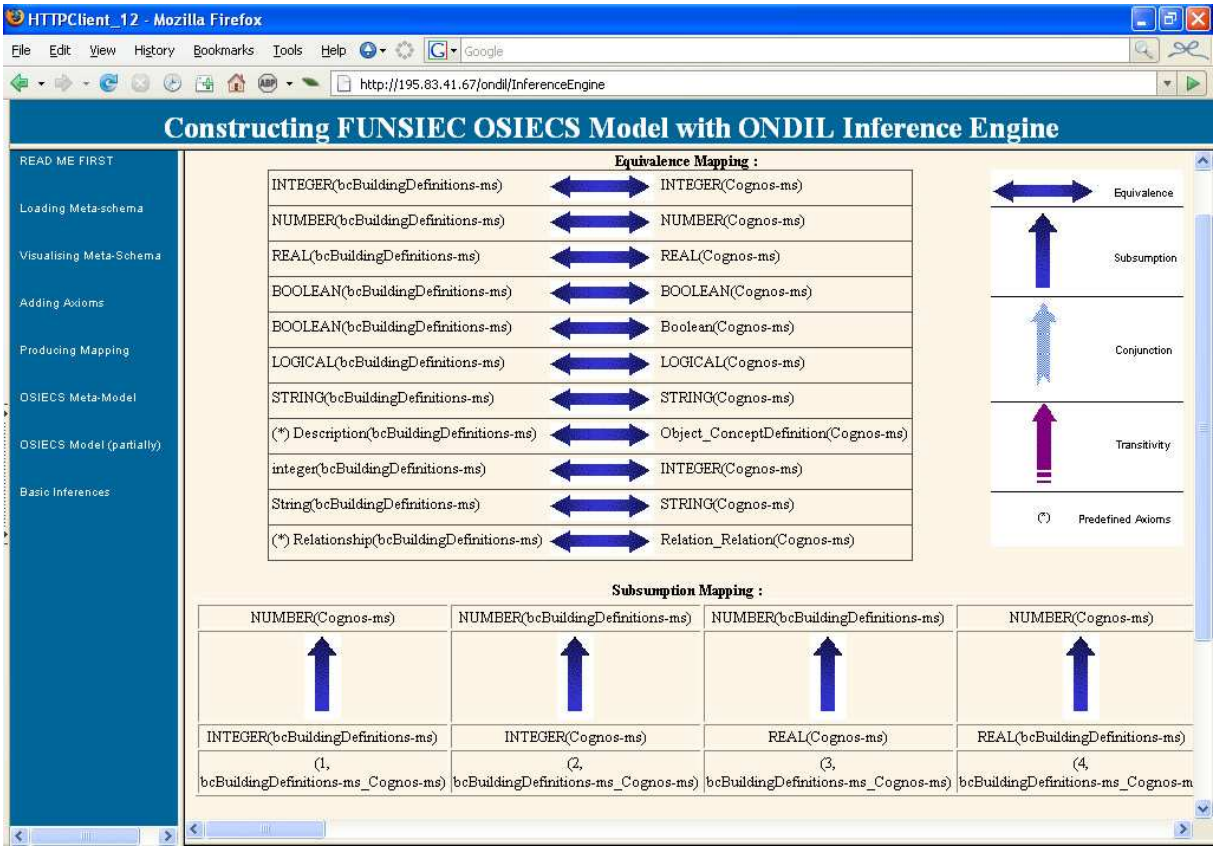

Figure 41 – Correspondances découvertes entre les méta-schémas de bcBuildingDefinitions et e-Cognos par FUNONDIL.

5.1.2 Découverte de correspondances entre les méta-schémas de bcBuildingDefinitions et ISO 12006-3

Le Tableau 11 présente les axiomes d'amorçage entre les méta-schémas de bcBuildingDefinitions et ISO 12006-3. Une partie des correspondances sémantiques qui résultent du processus de découverte de correspondances entre ces méta-schémas est présentée dans la Figure 42.

Tableau 11 – Axiomes d'amorçage entre les méta-schémas de bcBuildingDefinitions et ISO 12006-3.

`Relationship ⊑ xtdRelationship`
`xtdLanguage ⊑ language`
`language ⊑ xtdLanguage`
`Description ⊑ xtdDescription`
`xtdDescription ⊑ Description`
`xtdSubject ⊑ Object`
`ExternalReference ⊑ xtdReference`
`xtdReference ⊑ ExternalReference`

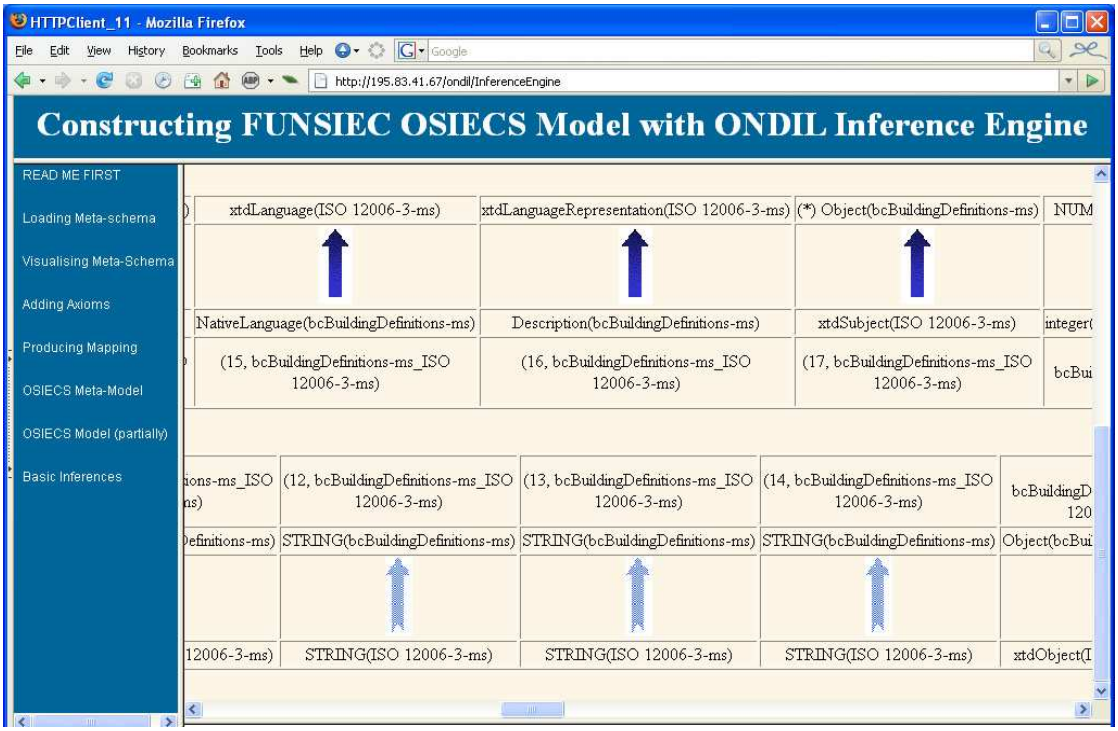

Figure 42 – Correspondances découvertes entre les méta-schémas de bcBuildingDefinitions et ISO 12006-3 par FUNONDIL.

Plus de correspondances de sémantique plus riche sont découvertes entres ces ressources sémantiques par rapport au cas précédent. Par exemple, le concept *NativeLanguage* du méta-

schéma de bcBuildingDefinitions est subsumé par le concept *xtdLanguage* du méta-schéma d'ISO 12006-3. Un autre exemple est le concept *Description* du méta-schéma de bcBuildingDefinitions qui est subsumé par le concept *xtdLanguageRepresentation* du méta-schéma d'ISO 12006-3.

5.1.3 Découverte de correspondances entre les méta-schémas de bcBuildingDefinitions et IFC-kernel

Le Tableau 12 présente les axiomes d'amorçage entre les méta-schémas de bcBuildingDefinitions et IFC-kernel. La Figure 43 montre une partie des correspondances sémantiques qui résultent du processus de découverte de correspondances entre ces méta-schémas.

Tableau 12 – Axiomes d'amorçage entre les méta-schémas bcBuildingDefinitions et IFC-kernel.

`Relationship ⊑ IfcRelationship`
`Object ⊑ IfcObject`
`ExternalReference ⊑ IfcExternalReference`
`IfcExternalReference ⊑ ExternalReference`

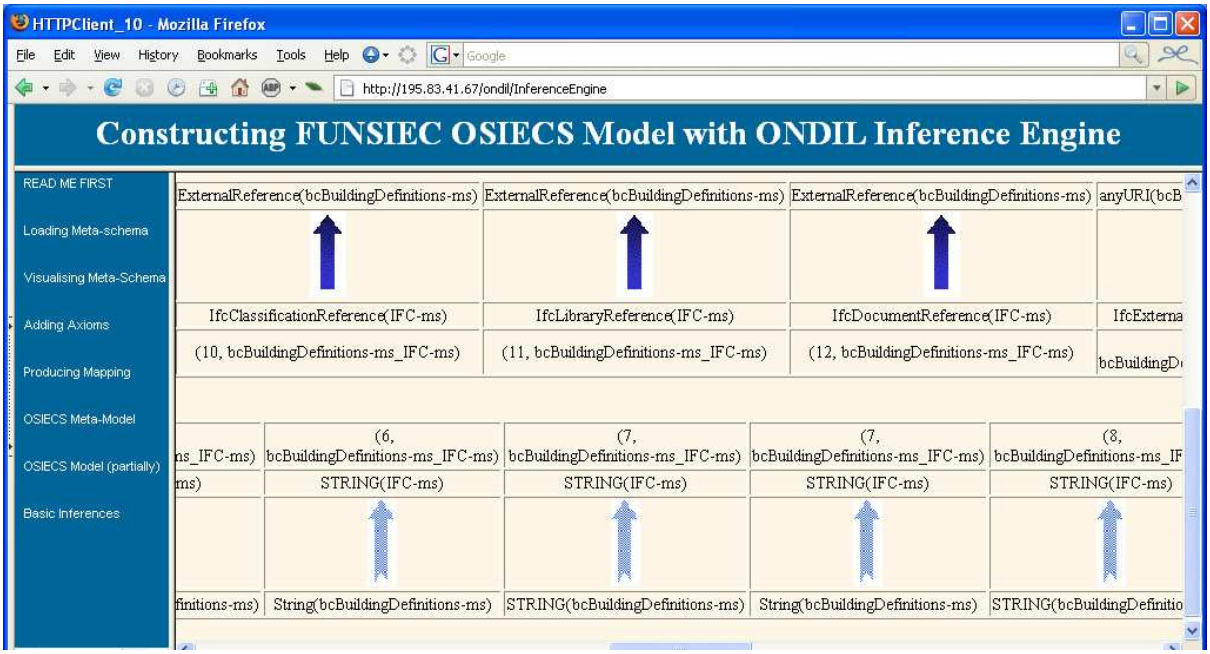

Figure 43 – Correspondances découvertes entre les méta-schémas de bcBuildingDefinitions et IFC-kernel par FUNONDIL.

Nous constatons que le concept *ExternalReference* du méta-schéma de bcBuildingDefinitions subsume les concepts *IfcClassificationReference*, *IfcLibraryReference* et *IfcDocumentReference* du méta-schéma IFC-kernel. Ce résultat est en accord avec la sémantique spécifiée par ces concepts dans leurs méta-schémas respectifs.

5.1.4 Découverte de correspondances entre les méta-schémas e-Cognos et ISO 12006-3

Le Tableau 13 présente les axiomes d'amorçage entre les méta-schémas e-Cognos et ISO 12006-3. La Figure 44 montre une partie des correspondances sémantiques qui résultent du processus de découverte de correspondances entre ces méta-schémas.

Tableau 13 – Axiomes d'amorçage entre les méta-schémas e-Cognos et ISO 12006-3.

```
Relation_Relation ⊑ xtdRelationship

xtdRelationship ⊑ Relation_Relation

Object_ConceptDefinition ⊑ xtdDescription

xtdDescription ⊑ Object_ConceptDefinition

Concept_Concept ⊑ xtdObject
```

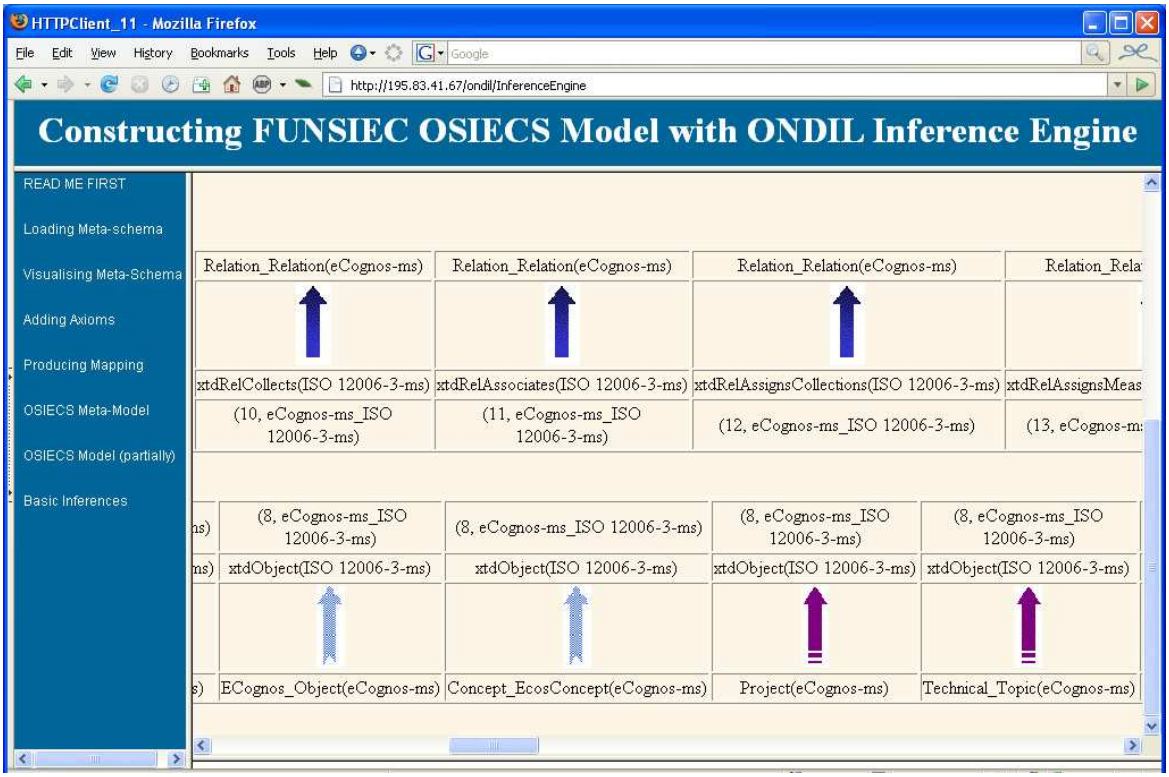

Figure 44 – Correspondances découvertes entre les méta-schémas e-Cognos et ISO 12006-3 par FUNONDIL.

Le processus de découverte de correspondances sémantiques produit plusieurs résultats sémantiquement riches en ce qui concerne les méta-schémas e-Cognos et ISO 12006-3. Par exemple, nous remarquons que les concepts *xtdRelCollects*, *xtdRelAssociates*, *xtdRelAssignsCollections*, du méta-schéma ISO 12006-3, sont subsumés par le concept *Relation_Relation* du méta-schéma e-Cognos. Ceci est un résultat attendu, parce ce que le concept e-Cognos est plus abstrait que les concepts ISO 12006-3 qu'il subsume. Par ailleurs, des correspondances de conjonction sont

découvertes : entre le concept *ECognos_Object* du méta-schéma e-Cognos et le concept *xtdObject* du méta-schéma ISO 12006-3, et aussi entre *Concept_EcosConcept* et *xtdObject*. Des correspondances de transitivité sont retrouvées entre les concepts *Project* et *Technical_Topic* du méta-schéma e-Cognos et le concept *xtdObject* du méta-schéma ISO 12006-3. Les concepts *Project* et *Technical_Topic* sont subsumés par transitivité par le concept *xtdObject*, et non par subsomption directe.

5.1.5 Découverte de correspondances entre les méta-schémas e-Cognos et IFC-kernel

Le Tableau 14 présente les axiomes d'amorçage entre les méta-schémas e-Cognos et IFC-kernel. La Figure 45 montre une partie des correspondances sémantiques qui résultent du processus de découverte de correspondances entre ces méta-schémas.

Tableau 14 – Axiomes d'amorçage entre les méta-schémas e-Cognos et IFC-kernel.

IfcRelationship ⊑ Relation_Relation
Relation_Relation ⊑ IfcRelationship
IfcObject ⊑ Concept_Concept

Les concepts *IfcRelAssigns*, *IfcRelConnects* et *IfcRelDecomposes* du méta-schéma IFC-kernel sont subsumés par le concept *Relation_Relation* du méta-schéma e-Cognos. Nous constatons, à travers la copie d'écran suivante, qu'une correspondance de transitivité est retrouvée entre le concept *IfcActor* du méta-schéma IFC-kernel et *Concept_Concept* du méta-schéma e-Cognos. Des correspondances de conjonction sont également découvertes entre les concepts *Ecognos_Object* et *Relation_EcosRelation*, et le concept *IfcRoot*.

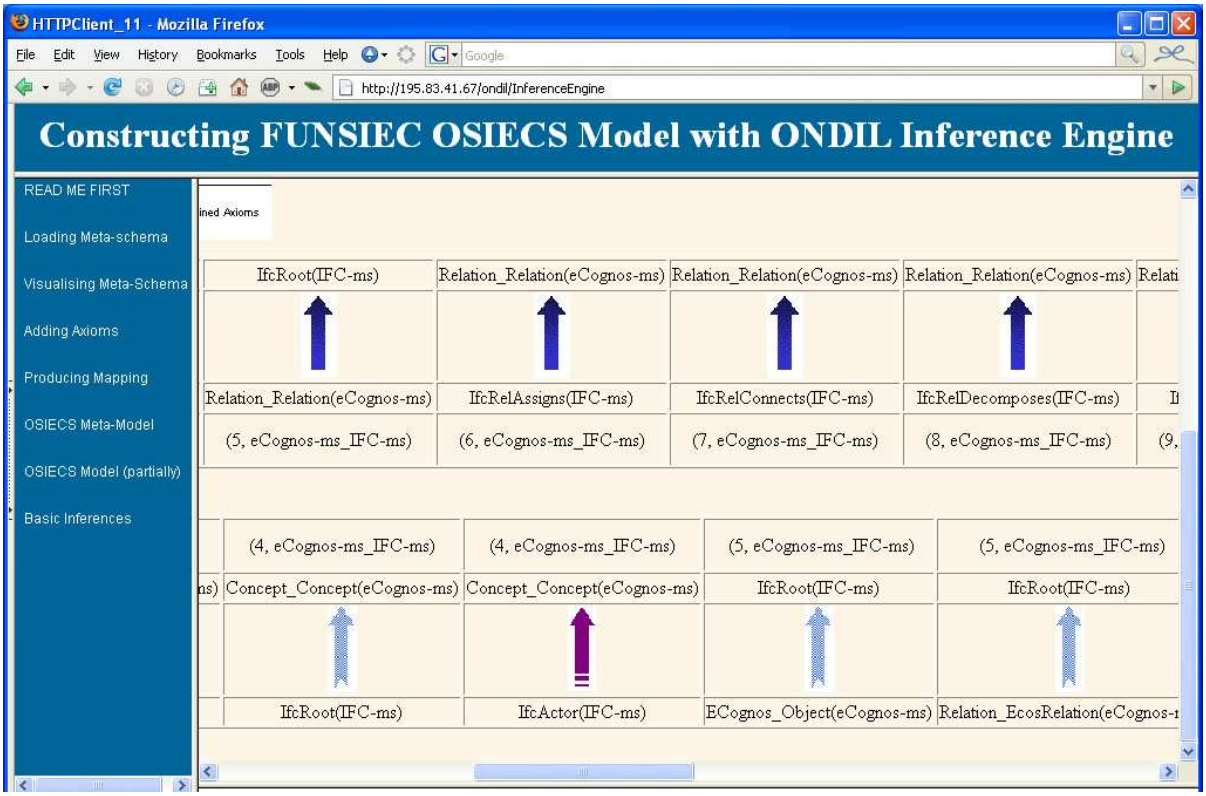

Figure 45 – Correspondances découvertes entre les méta-schémas e-Cognos et IFC-kernel par FUNONDIL.

5.1.6 Découverte de correspondances entre les méta-schémas IFC-kernel et ISO 12006-3

Le Tableau 15 présente les axiomes d'amorçage entre les méta-schémas IFC-kernel et ISO 12006-3. La Figure 46 montre une partie des correspondances sémantiques qui résultent du processus de découverte de correspondances entre ces méta-schémas.

Tableau 15 – Axiomes d'amorçage entre les méta-schémas IFC-kernel et ISO 12006-3

```
xtdRelationship ⊑ IfcRelationship

xtdObject ⊑ IfcObject

IfcObject ⊑ xtdObject

IfcExternalReference ⊑ xtdReference

xtdReference ⊑ IfcExternalReference
```

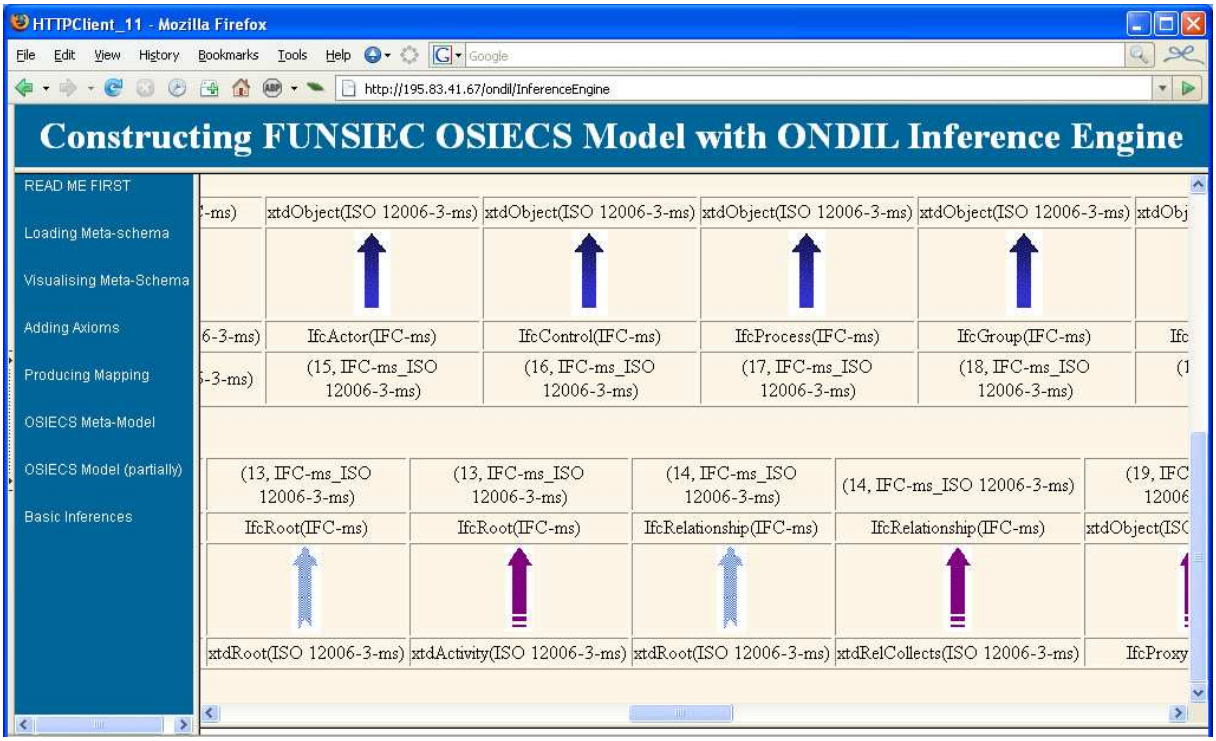

Figure 46 – Correspondances découvertes entre les méta-schémas ISO 12006-3 et IFC-kernel par FUNONDIL.

Le concept *xtdObject* du méta-schéma ISO 12006-3 est le subsumeur des concepts *IfcActor*, *IfcControl*, *IfcProcess* et *IfcGroup* du méta-schéma IFC-kernel. Une correspondance de transitivité est aussi retrouvée entre le concept *xtdActivity* du méta-schéma ISO 12006-3 et le concept *IfcRoot* du méta-schéma IFC-kernel.

5.1.7 Analyse des correspondances produites au niveau des méta-schémas

Pour une moyenne de 5 axiomes d'amorçage par processus de découverte de correspondances entre chaque couple de ressources sémantiques, nous obtenons une moyenne de 17 correspondances sémantiques validées par les experts du domaine.

Les correspondances d'équivalence sont beaucoup moins nombreuses que les autres types de correspondances. Ceci s'explique en raison de la quasi-inexistence de définitions de concepts 100% équivalents entre ces schémas. En effet, il est plus difficile de trouver deux définitions de concepts, de deux ressources sémantiques, équivalentes que deux définitions entre lesquelles existe une subsomption, une transitivité ou une conjonction.

Du point de vue sémantique, nous reconnaissons que les correspondances produites sont assez abstraites, en raison de l'utilisation des méta-schémas des ressources. Par ailleurs, le méta-

schéma e-Cognos est plus abstrait que les autres. Par conséquent, les correspondances découvertes conviennent plus à un usage par des concepteurs d'ontologies qu'à une utilisation opérationnelle.

5.2 Découverte de correspondances entre schémas des ressources sémantiques

5.2.1 Expérimentation menée

Pour la découverte de correspondances sémantiques au niveau des schémas, nous avons sélectionné les ontologies e-Cognos et bcBuildingDefinitions version 20020129. Nous n'avons pas considéré d'axiome d'amorçage. FaCT permet de considérer que les concepts qui ont le même nom sont équivalents. Puisque le domaine d'application est le même pour les deux ressources, il est fort probable, par exemple, que le concept *porte* ait une signification non ambiguë si nous considérons la même langue naturelle. Ainsi nous avons fait l'hypothèse que deux concepts de deux ressources sémantiques identifiés par la même chaîne de caractères sont les mêmes.

5.2.2 Résultats produits avec les schémas

Nous présentons quelques exemples de résultats des correspondances obtenues entre les ontologies e-Cognos et bcBuildingDefinitions (Figure 47 et Figure 48). Dans l'annexe 5 se trouve la représentation OWL des correspondances sémantiques entre les ontologies bcBuildingDefinitions et e-Cognos.

Le concept *ItemSet* de bcBuildingDefinitions est découvert comme subsumeur de plusieurs concepts, tels que *Material*, *blanket*, *head*, *gas* et *Building* d'e-Cognos. Dans bcBuildingDefinitions, le concept *ItemSet* subsume un très grand nombre de concepts, tels que *Supplier*, *Material* et *Gas*.

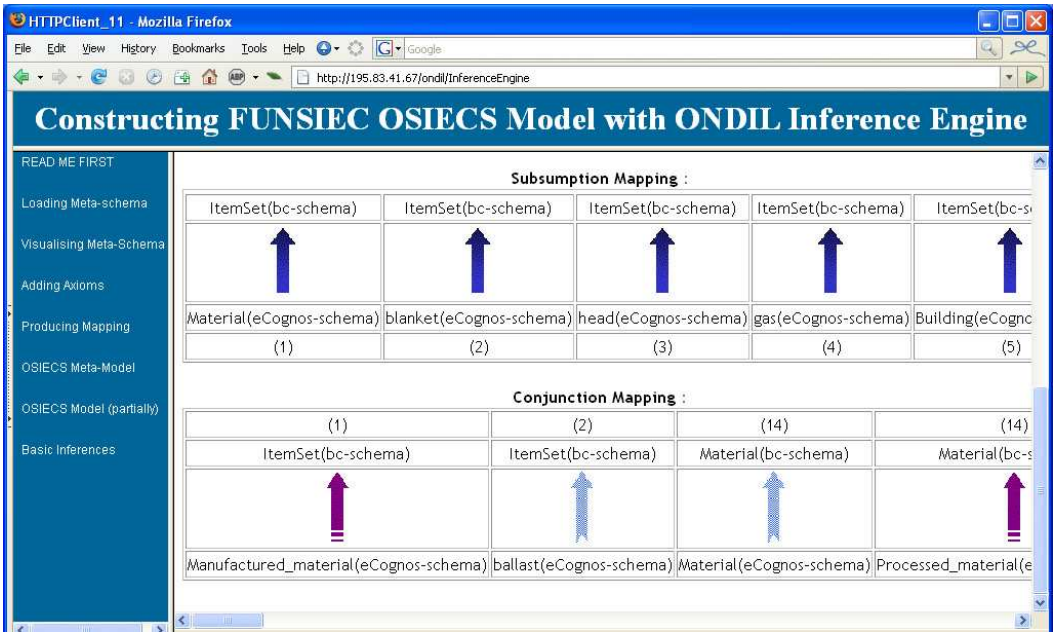

Figure 47 – Correspondances découvertes entre les ontologies e-Cognos et bcBuildingDefinitions par FUNONDIL.

Des correspondances de conjonction sont découvertes entre les concepts *ballast* et *Material* d'e-Cognos et les concepts *ItemSet* et *Material* de bcBuildingDefinitions, respectivement. Le concept *ballast* est subsumé, dans l'ontologie e-Cognos, par le concept *track components* et subsume plusieurs concepts, tels que *heaped shoulder* et *blanket*.

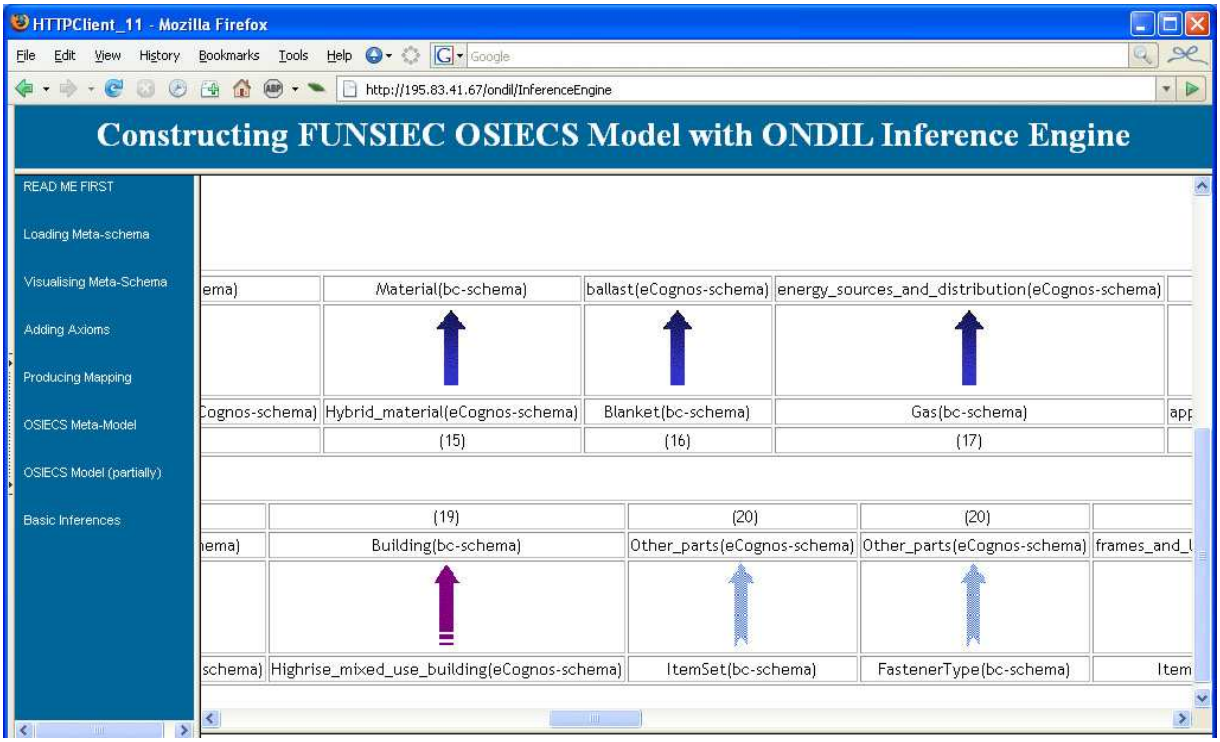

Figure 48 – D'autres correspondances découvertes entre les ontologies e-Cognos et bcBuildingDefinitions par FUNONDIL.

Une correspondance de transitivité est découverte entre le concept *Highrise_mixed_use_building* d'e-Cognos et le concept *Building* de bcBuildingDefinitions (Figure 48). Les correspondances trouvées associent des concepts à différents niveaux hiérarchiques dans les ressources. Par exemple, dans e-Cognos, *Highrise_mixed_use_building* est un concept feuille. Il est subsumé par les concepts *Mixed use building* et *Highrise building*. Dans bcBuildingDefinitions, le concept *Building* est subsumé par le concept *ItemSet*, qui est proche de la racine. Ces correspondances peuvent être utilisées dans un projet opérationnel pour mettre en relation les concepts de ces deux ressources.

5.2.3 Analyse des correspondances produites au niveau des schémas

Entre les deux schémas pris en compte pour ce test, 9 équivalences entre concepts des ontologies e-Cognos et bcBuildingDefinitions ont été établies par FUNONDIL par identification de chaînes de caractères identiques. FUNONDIL a ensuite découvert 37 correspondances de subsomption, 8 de transitivité et 28 de conjonction, soit un total de 73 correspondances non équivalentes. Toutes ces correspondances ont été estimées en accord avec la sémantique représentée dans les schémas utilisés.

Par rapport aux correspondances découvertes en utilisant les méta-schémas, les correspondances générées avec les schémas s'avèrent plus utilisables dans un cadre opérationnel. En effet, les correspondances entre schémas lient des concepts dont la sémantique est plus spécifique.

5.3 Unification de propriétés

Nous avons utilisé le prototype UNIREL afin d'évaluer l'influence de l'unification de propriétés sur la découverte de correspondances. UNIREL implante l'unification de propriétés, la découverte de correspondances et l'inférence d'instances. Nous présentons les tests et résultats effectués d'abord avec les méta-schémas IFC-kernel et ISO 12006-3. Ensuite nous nous focalisons sur des sous-parties des ressources e-Cognos et IFC-kernel *ad hoc* (*cf.* section 5.3.2), pour faciliter l'analyse des implications des restrictions locales et globales sur les résultats. Dans tous les cas testés, nous débutons les expériences par l'ajout d'un axiome de propriété équivalente entre des propriétés des deux ressources d'entrée.

5.3.1 Influence de l'unification des propriétés sur les correspondances découvertes entre les méta-schémas IFC-kernel et ISO 12006-3

Les entrées de ce test sont les méta-schémas IFC-kernel et ISO 12006-3 auxquels ont été ajoutés les axiomes d'amorçage du Tableau 15. En utilisant UNIREL, nous avons ensuite ajouté à un des méta-schémas, l'axiome d'unification de propriétés (22) :

`IFC-kernel: IfcRoot_Name ⇔ ISO 12006-3: xtdName_Name` **(22)**

Notons que la propriété `IfcRoot_Name` est impliquée dans une restriction globale dont la source de la propriété est le concept `IfcRoot` et la cible est le concept `IfcLabel`. Cette propriété intervient dans une restriction de cardinalité minimale dont la valeur est 1. Concernant ISO 12006-3, la propriété `xtdName_Name` intervient aussi dans une restriction de cardinalité minimale dont la valeur est 1, et participe dans une restriction globale dont la source de la propriété est le concept `xtdName` et la cible est le concept `xtdLabel`.

Le processus de découverte de correspondances, qui teste la subsomption et l'équivalence entre concepts (*cf.* `Correspondances.java`, annexe 3), a produit une correspondance additionnelle (23) par rapport à celles présentées dans la section 5.1.6. Cette correspondance additionnelle est en accord avec la sémantique représentée dans les méta-schémas IFC-kernel et ISO 12006-3. Ainsi, l'équivalence entre les propriétés *Name* des entités `IfcRoot` et `xtdName` des deux méta-schémas est une information supplémentaire qui a permis au moteur d'inférences de suggérer que l'entité `xtdName` est subsumée par l'entité `IfcRoot`.

`ISO 12006-3: xtdName ⊑ IFC-kernel: IfcRoot` **(23)**

Autrement dit, et en termes de restrictions globales, un des concepts source de la propriété équivalente est suggéré subsumé par le concept source de l'autre propriété équivalente.

5.3.2 Spécificités des ressources *ad hoc* utilisées

Nous avons décidé d'utiliser des ressources *ad hoc* pour faciliter l'étude de l'influence des restrictions locales et globales sur l'inférence de correspondances et d'instances. Une ressource est une sous-partie de l'ontologie e-Cognos et l'autre est une sous-partie du standard IFC-kernel. Les hiérarchies de concepts de ces ressources sont montrées respectivement dans les Figures 49 et 50. Signalons que dans l'ontologie e-Cognos, un `Actor` peut être une personne, par exemple un ingénieur ou une organisation (El-Diraby, Fiès et Lima 2002). La notion de `Project` dans e-

Cognos englobe les activités d'ingénierie qui mènent à l'aboutissement d'un produit. Notons que des projets, tels que définis par l'ontologie e-Cognos, retrouvent une sémantique similaire dans l'entité `IfcProject` d'IFC-kernel. Celle-ci est une sous-entité de `IfcObject`. L'entité `IfcRelAssignsToActor` permet l'attribution des objets (`IfcObject`), tels que des projets, à un acteur (`IfcActor`).

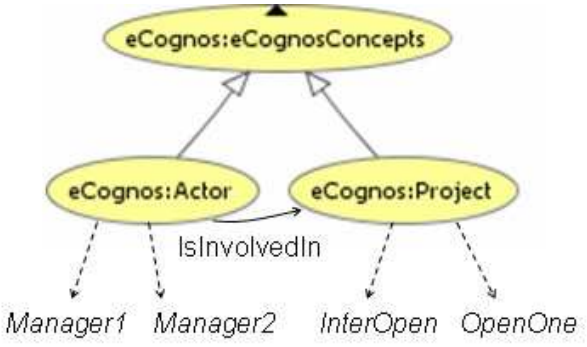

Figure 49 – Hiérarchie de concepts dede la sous-partie e-Cognos.

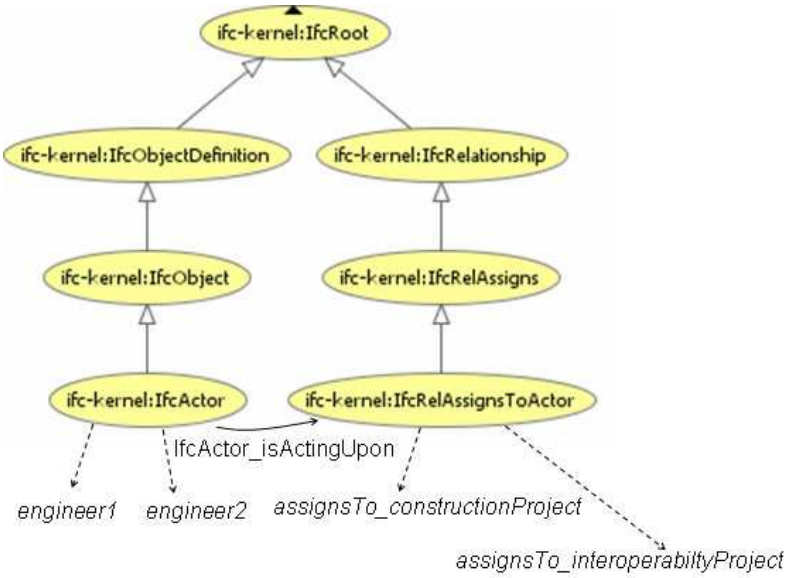

Figure 50 – Hiérarchie de concepts de la sous-partie d'IFC-kernel.

Ces deux ressources ont été adaptées de manière à instancier des individus pour les concepts feuilles. Ainsi, le concept `Actor` d'e-Cognos instancie les individus *Manager1* et *Manager2*. Le concept `Project` instancie les individus *InterOpen* et *OpenOne*. En ce qui concerne la ressource IFC-kernel, le concept `IfcActor` instancie les individus *engineer1* et *engineer2*, et le concept `IfcRelAssignsToActor` instancie les individus *assignsTo_constructionProject* et *assignsTo_interoperabiltyProject*.

En nous appuyant sur les hiérarchies de concepts des sous-parties d'e-Cognos et IFC-kernel, nous avons défini plusieurs combinaisons de restrictions impliquant les propriétés des concepts de ces ressources. Nous détaillons deux de ces cas ci-dessous.

Exemple 1

Considérons la définition du concept primitif `Actor` (24) de l'ontologie e-Cognos qui est composée de deux restrictions locales : universelle et existentielle. La propriété `eCognos:isInvolvedIn` intervient dans ces deux restrictions, dont le concept `eCognos:Project` est la cible. La propriété `eCognos:isInvolvedIn` définit également une restriction globale entre les concepts `eCognos:Actor` et `eCognos:Project` (25).

$$\texttt{eCognos:Actor} \sqsubseteq \forall \texttt{eCognos:isInvolvedIn.eCognos:project} \sqcap \exists \texttt{eCognos:isInvolvedIn.eCognos:Project} \quad \textbf{(24)}$$

$$\texttt{eCognos:isInvolvedIn(eCognos:Actor,eCognos:Project)} \quad \textbf{(25)}$$

Considérons aussi la définition du concept primitif `IfcActor` (26) du standard IFC-kernel qui contient une restriction locale existentielle. La propriété `ifc-kernel:IfcActor_isActingUpon` participe à cette restriction, dont le concept `ifc-kernel:IfcRelAssignsToActor` est la cible. La propriété `ifc-kernel:IfcActor_isActingUpon` définit aussi une restriction globale entre les concepts `ifc-kernel:IfcActor` et `ifc-kernel:IfcRelAssignsToActor` (27).

$$\texttt{ifc-kernel:IfcActor} \sqsubseteq \exists\ \texttt{ifc-kernel:IfcActor\_isActingUpon.ifc-kernel:IfcRelAssignsToActor} \quad \textbf{(26)}$$

$$\texttt{ifc-kernel:IfcActor\_isActingUpon(ifc-kernel:IfcActor, ifc-kernel:IfcRelAssignsToActor)} \quad \textbf{(27)}$$

Exemple 2

Considérons un autre test où nous avons enlevé la restriction globale (25) de la sous-partie de l'ontologie e-Cognos prise en compte pour l'exemple 1. C'est-à-dire qu'une des ressources ne contient plus la propriété `isInvolvedIn` avec les *domain* et *range* respectifs. Cependant, cette propriété continue à participer aux restrictions locales (24). L'autre ressource d'entrée, le sous-ensemble de l'IFC-kernel, n'a pas été modifié.

Ces exemples sont deux des cas de test qui ont été expérimentés à l'aide du prototype UNIREL. La section suivante présente l'étape d'unification de propriétés appliquée à ces deux cas.

5.3.3 Adjonction d'axiome pour l'unification de propriétés

Pour unifier les propriétés à travers UNIREL, nous utilisons les deux sous-parties des ressources, qui sont les entrées du processus d'unification de relations (*cf.* `Unification.java`, annexe 3). Nous ajoutons l'axiome (28) à une des deux ressources, dans les deux cas ci-dessus. L'équivalence est ainsi attribuée aux propriétés (29) et (30), de la sous-partie de l'ontologie e-Cognos et de celle d'IFC-kernel respectivement. Les sous-parties d'e-Cognos et d'IFC-kernel relatives à l'exemple 1 qui résultent de ce processus sont respectivement dans les annexes 6 et 7.

`eCognos:isInvolvedIn` ⇔ `ifc-kernel:IfcActor_isActingUpon` (**28**)

http://www.cstb.fr/eCognos#isInvolvedIn (**29**)

http://www.iai-international.org/Model/R2x3_final/index.htm#IfcActor_isActingUpon (**30**)

5.3.4 Résultats de la découverte de correspondances après l'unification de propriétés

À partir des ressources sémantiques et de l'équivalence de propriétés présentées plus haut, nous avons testé plusieurs types de configurations, en termes de restrictions apparaissant dans les ressources. Le Tableau 16 montre les neuf cas de combinaisons de restrictions que nous avons étudié sur les hiérarchies de concepts des Figures 49 et 50. Le Tableau 17 montre les correspondances découvertes pour chaque cas du Tableau 16.

Tableau 16 – Les neuf cas de combinaisons de restrictions étudiés.

		e-Cognos		
		Restrictions locales ∀ et ∃	Restrictions locales ∀ et ∃ et globale	Restriction globale
IFC-kernel	Restriction locale ∃	Cas 1	Cas 2	Cas 3
	Restriction locale ∃ et globale	Cas 4	Cas 5	Cas 6
	Restriction globale	Cas 7	Cas 8	Cas 9

Tableau 17 – Types de restrictions et correspondances découvertes. La colonne intitulée Ressources sémantiques représente les sous-parties des ressources mentionnées.

Cas	Ressources sémantiques	Type de restrictions	Correspondances découvertes
1	e-Cognos	Restrictions locales ∀ et ∃	Non
	IFC-kernel	Restriction locale ∃	
2	e-Cognos	Restrictions locales ∀ et ∃ et restriction globale	IfcActor ⊑ eCognosConcepts IfcActor ⊑ Actor
	IFC-kernel	Restriction locale ∃	
3	e-Cognos	Restriction globale	IfcActor ⊑ eCognosConcepts IfcActor ⊑ Actor
	IFC-kernel	Restriction locale ∃	
4	e-Cognos	Restrictions locales ∀ et ∃	Actor ⊑ IfcObjectDefinition Actor ⊑ IfcRoot Actor ⊑ IfcObject Actor ⊑ IfcActor
	IFC-kernel	Restriction locale ∃ et restriction globale	
5	e-Cognos	Restrictions locales ∀ et ∃ et restriction globale	Actor ⇔ IfcActor IfcActor ⊑ eCognosConcepts Actor ⊑ IfcObjectDefinition Actor ⊑ IfcRoot Actor ⊑ IfcObject Actor ⊑ IfcActor IfcActor ⊑Actor
	IFC-kernel	Restriction locale ∃ et restriction globale	
6	e-Cognos	Restriction globale	IfcActor ⊑ eCognosConcepts IfcActor ⊑ Actor
	IFC-kernel	Restriction locale ∃ et restriction globale	
7	e-Cognos	Restrictions locales ∀ et ∃	Actor ⊑ IfcObjectDefinition Actor ⊑ IfcRoot Actor ⊑ IfcObject Actor ⊑ IfcActor
	IFC-kernel	Restriction globale	
8	e-Cognos	Restrictions locales ∀ et ∃ et restriction globale	Actor ⊑ IfcObjectDefinition Actor ⊑ IfcRoot

	IFC-kernel	Restriction globale	Actor ⊑ IfcObject Actor ⊑ IfcActor
9	e-Cognos	Restriction globale	Non
	IFC-kernel	Restriction globale	

Dans l'exemple 1, le prototype UNIREL produit 1 équivalence et 6 correspondances de subsomption, comme indiqué dans le cas 5 du Tableau 17 et illustré en Figure 51.

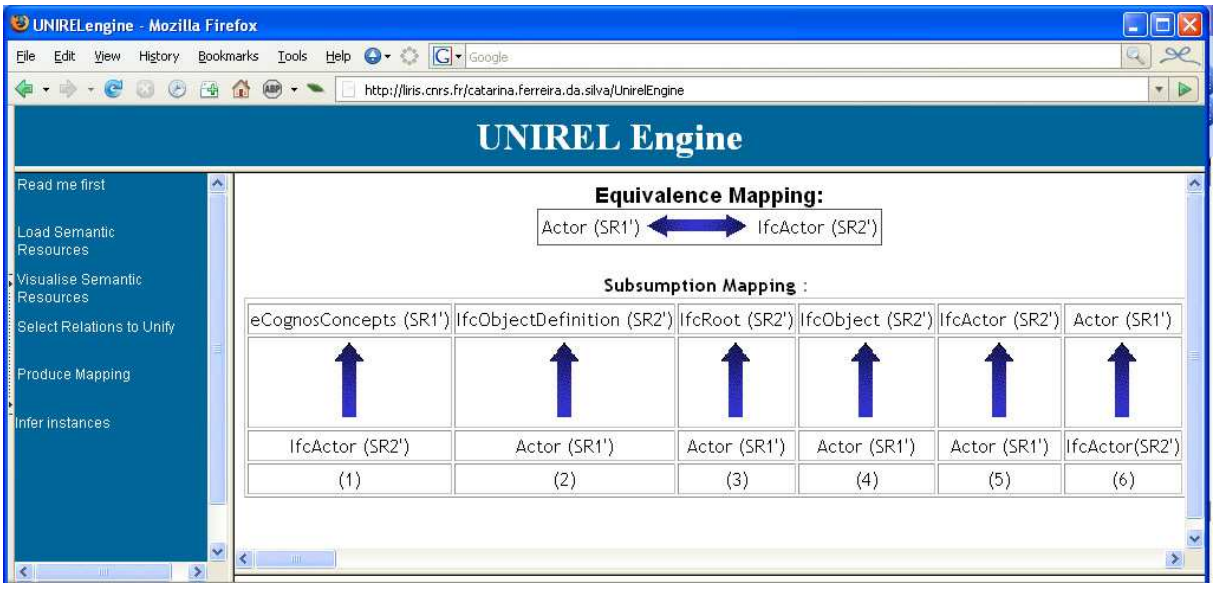

Figure 51 – Correspondances entre les sous-parties d'eCognos et d'IFC-kernel (cas 5, tableau 17) après l'unification de relations, produites par le prototype UNIREL.

L'exemple 2 (*cf.* 5.3.2) correspond au cas 4 de ce tableau et est illustré en Figure 52. Dans ce cas, UNIREL produit quatre correspondances de subsomption.

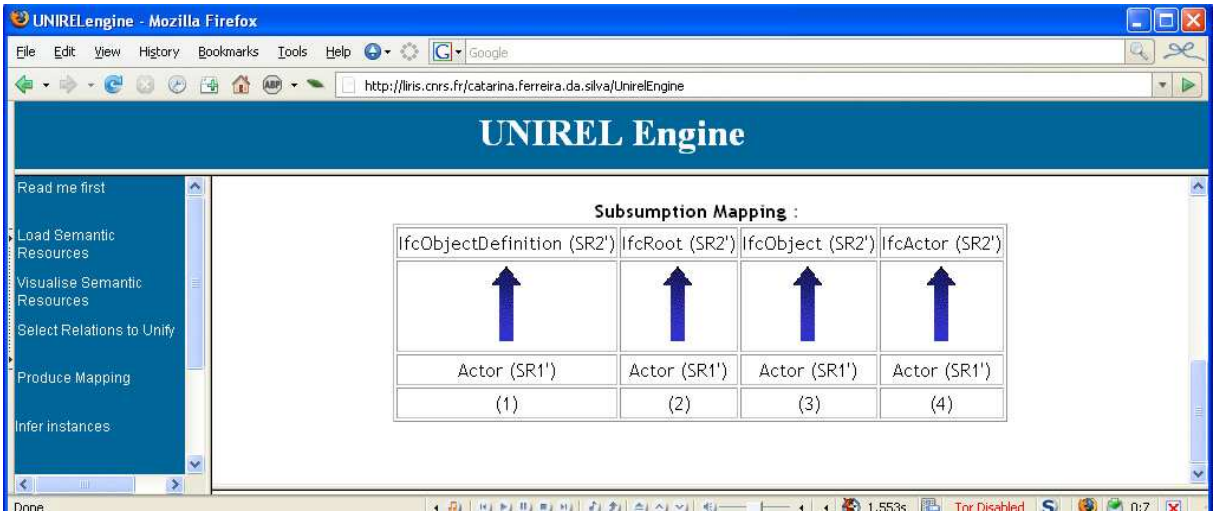

Figure 52 – Correspondances entre les sous-parties d'eCognos et d'IFC (cas 4, tableau 17) après l'unification de propriétés, produites par le prototype UNIREL. Dans ce cas, la première ressource ne contient pas la restriction globale sur la propriété `e-Cognos:isInvolvedIn`.

5.3.5 Analyse des résultats de la découverte de correspondances après l'unification de propriétés

Dans les résultats ci-dessus, le cas 4 montre qu'en enlevant la restriction globale (25) nous ne pouvons plus obtenir la correspondance d'équivalence (31) ni de correspondance de subsomption (32) et (33).

$$\texttt{Actor} \Leftrightarrow \texttt{IfcActor} \quad \textbf{(31)}$$

$$\texttt{IfcActor} \sqsubseteq \texttt{eCognosConcepts} \quad \textbf{(32)}$$

$$\texttt{IfcActor} \sqsubseteq \texttt{Actor} \quad \textbf{(33)}$$

Nous constatons que l'existence des restrictions locales seules (cas 1, Tableau 17) dans les ressources utilisées, ne permet pas de générer de correspondances, après l'unification des propriétés. La même situation se vérifie quand il n'existe que des restrictions globales (cas 9).

Notons aussi que l'existence de restrictions locales, universelle et existentielle, dans une des ressources et d'une restriction globale dans l'autre (cas 2, Tableau 17) produit les deux même correspondances de subsomption que l'existence d'une restriction globale dans une des ressources et d'une restriction locale existentielle dans l'autre (cas 3).

5.4 *Inférence d'instances*

Les tests sur l'inférence d'instances ont été effectués avec les sous-parties d'e-Cognos et d'IFC-kernel, après l'unification de propriétés. Pour chaque individu inféré, nous récupérons aussi ses types, *i.e.* les concepts dont l'individu est instance. Pour l'exemple 1 (*cf.* section 5.3.2), les individus *Manager1* et *Manager2*, instances du concept `Actor` de la sous-partie d'e-Cognos, sont inférés comme instances du concept `IfcActor` de la sous-partie d'IFC-kernel, comme illustré en Figure 53. D'autre part, les individus *InterOpen* et *OpenOne*, instances du concept `Project`, sont inférés comme instances du concept `IfcRelAssignsToActor`. Les instances des concepts feuille du sous-ensemble IFC sont aussi inférées comme instances des concepts du sous-ensemble eCognos. Ceci est un résultat de l'apport d'un axiome d'équivalence entre les propriétés (28) des deux ressources.

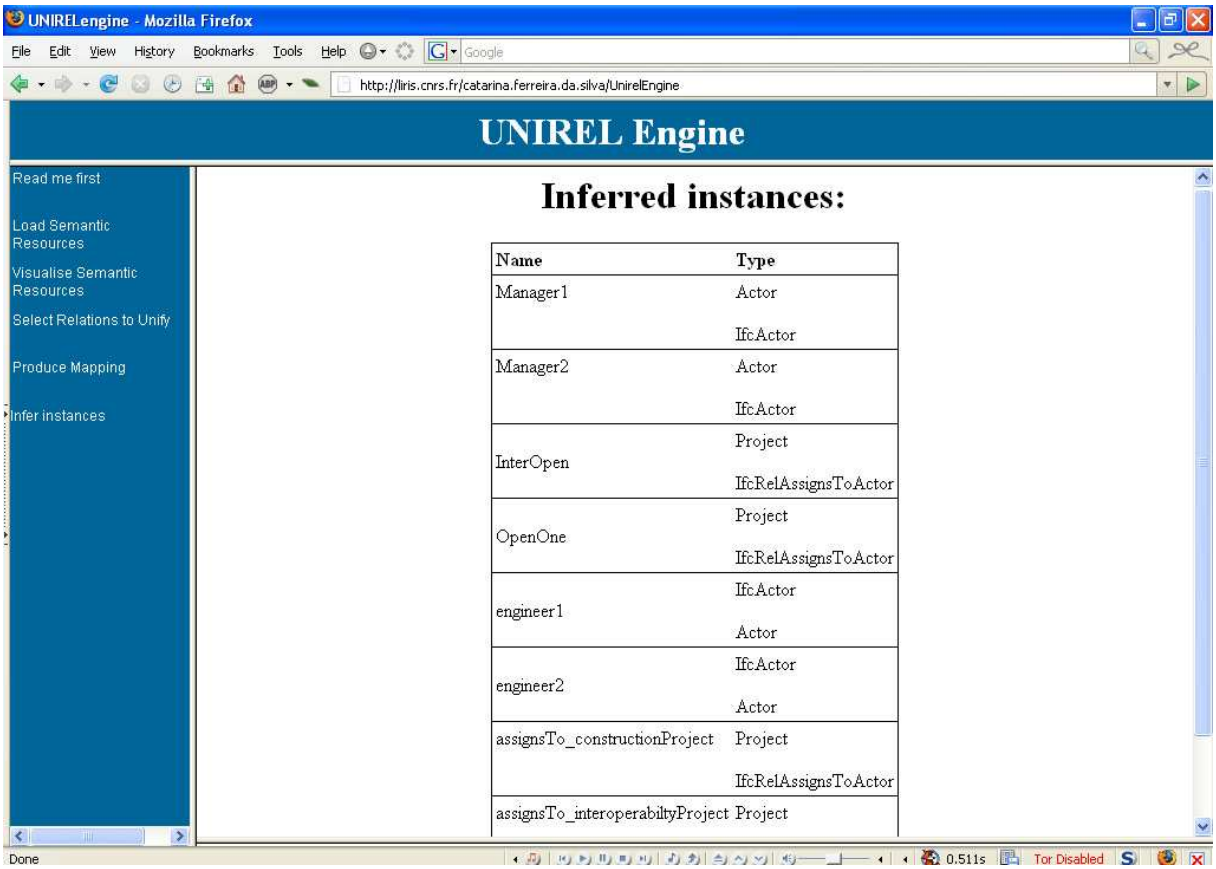

Figure 53 – Inférence d'instances des concepts de la sous-partie d'IFC-kernel de l'exemple 1 (cas 5, tableau 18), produites par le prototype UNIREL.

Comme pour la découverte de correspondances, le Tableau 18 liste plusieurs combinaisons de restrictions que nous avons étudiées sur les hiérarchies de concepts des Figure 49 et 50 et l'inférence d'instances pour chaque cas.

Tableau 18 – Types de restrictions et instances inférées.

Cas	Ressources sémantiques	Type de restrictions	Individus inférés
1	e-Cognos	Restrictions locales ∀ et ∃	Instances respectives seulement
	IFC-kernel	Restriction locale ∃	
2	e-Cognos	Restrictions locales ∀ et ∃ et restriction globale	Manager1 (Types : Actor) Manager2 (Types : Actor) InterOpen (Types : Project) OpenOne (Types : Project) engineer1 (Types : IfcActor, Actor) engineer2 (Types : IfcActor, Actor) assignsTo_constructionProject (Types : Project, IfcRelAssignsToActor) assignsTo_interoperabilityProject (Types : Project, IfcRelAssignsToActor)
	IFC-kernel	Restriction locale ∃	
3	e-Cognos	Restriction globale	Manager1 (Types : Actor) Manager2 (Types : Actor) InterOpen (Types : Project) OpenOne (Types : Project) engineer1 (Types : Actor, IfcActor) engineer2 (Types : Actor, IfcActor) assignsTo_constructionProject (Types : Project, IfcRelAssignsToActor) assignsTo_interoperabiltyProject (Types : Project, IfcRelAssignsToActor)
	IFC-kernel	Restriction locale ∃	
4	e-Cognos	Restrictions locales ∀ et ∃	Manager1 (Types : Actor, IfcActor) Manager2 (Types : Actor, IfcActor)

	IFC-kernel	Restriction locale ∃ et restriction globale	InterOpen (Types : Project, IfcRelAssignsToActor) OpenOne (Types : Project, IfcRelAssignsToActor) engineer1 (Types : IfcActor) engineer2 (Types : IfcActor) assignsTo_constructionProject (Types : IfcRelAssignsToActor) assignsTo_interoperabiltyProject (Types : IfcRelAssignsToActor)
5	e-Cognos	Restrictions locales ∀ et ∃ et restriction globale	Manager1 (Types : Actor, IfcActor) Manager2 (Types : Actor, IfcActor) InterOpen (Types : Project, IfcRelAssignsToActor) OpenOne (Types : Project, IfcRelAssignsToActor) engineer1 (Types : Actor, IfcActor) engineer2 (Types : Actor, IfcActor) assignsTo_constructionProject (Types : Project, IfcRelAssignsToActor) assignsTo_interoperabiltyProject (Types : Project, IfcRelAssignsToActor)
	IFC-kernel	Restriction locale ∃ et restriction globale	
6	e-Cognos	Restriction globale	Manager1 (Types : Actor, IfcActor) Manager2 (Types : Actor, IfcActor) InterOpen (Types : Project, IfcRelAssignsToActor) OpenOne (Types : Project, IfcRelAssignsToActor) engineer1 (Types : Actor, IfcActor) engineer2 (Types : Actor, IfcActor) assignsTo_constructionProject (Types : Project, IfcRelAssignsToActor) assignsTo_interoperabiltyProject (Types : Project, IfcRelAssignsToActor)
	IFC-kernel	Restriction locale ∃ et restriction globale	
7	e-Cognos	Restrictions locales ∀ et ∃	Manager1 (Types : Actor, IfcActor) Manager2 (Types : Actor, IfcActor)

	IFC-kernel	Restriction globale	InterOpen (Types : Project, IfcRelAssignsToActor) OpenOne (Types : Project, IfcRelAssignsToActor) engineer1 (Types : IfcActor) engineer2 (Types : IfcActor) assignsTo_constructionProject (Types : IfcRelAssignsToActor) assignsTo_interoperabiltyProject (Types : IfcRelAssignsToActor)
8	e-Cognos	Restrictions locales ∀ et ∃ et restriction globale	Manager1 (Types : Actor, IfcActor) Manager2 (Types : Actor, IfcActor) InterOpen (Types : Project, IfcRelAssignsToActor) OpenOne (Types : Project, IfcRelAssignsToActor) engineer1 (Types : Actor, IfcActor) engineer2 (Types : Actor, IfcActor) assignsTo_constructionProject (Types : Project, IfcRelAssignsToActor) assignsTo_interoperabiltyProject (Types : Project, IfcRelAssignsToActor)
	IFC-kernel	Restriction globale	
9	e-Cognos	Restriction globale	Manager1 (Types : Actor, IfcActor) Manager2 (Types : Actor, IfcActor) InterOpen (Types : Project, IfcRelAssignsToActor) OpenOne (Types : Project, IfcRelAssignsToActor) engineer1 (Types : Actor, IfcActor) engineer2 (Types : Actor, IfcActor) assignsTo_constructionProject (Types : Project, IfcRelAssignsToActor) assignsTo_interoperabiltyProject (Types : Project, IfcRelAssignsToActor)
	IFC-kernel	Restriction globale	

Le cas 4 de ce tableau correspond à l'exemple 2 (*cf.* section 5.3.2), où la restriction globale dans la sous-partie e-Cognos à été retirée, par rapport au cas 5. Nous vérifions que les individus *engineer1* et *engineer2* assignés au concept `IfcActor` de la sous-partie IFC-kernel ne sont plus inférés comme instances du concept `Actor` de la sous-partie eCognos, comme illustré en Figure 54. En revanche, les individus *InterOpen* et *OpenOne*, instances du concept `Project` d'e-Cognos,

continuent d'être inférés comme instances du concept `IfcRelAssignsToActor`, comme dans le cas 5.

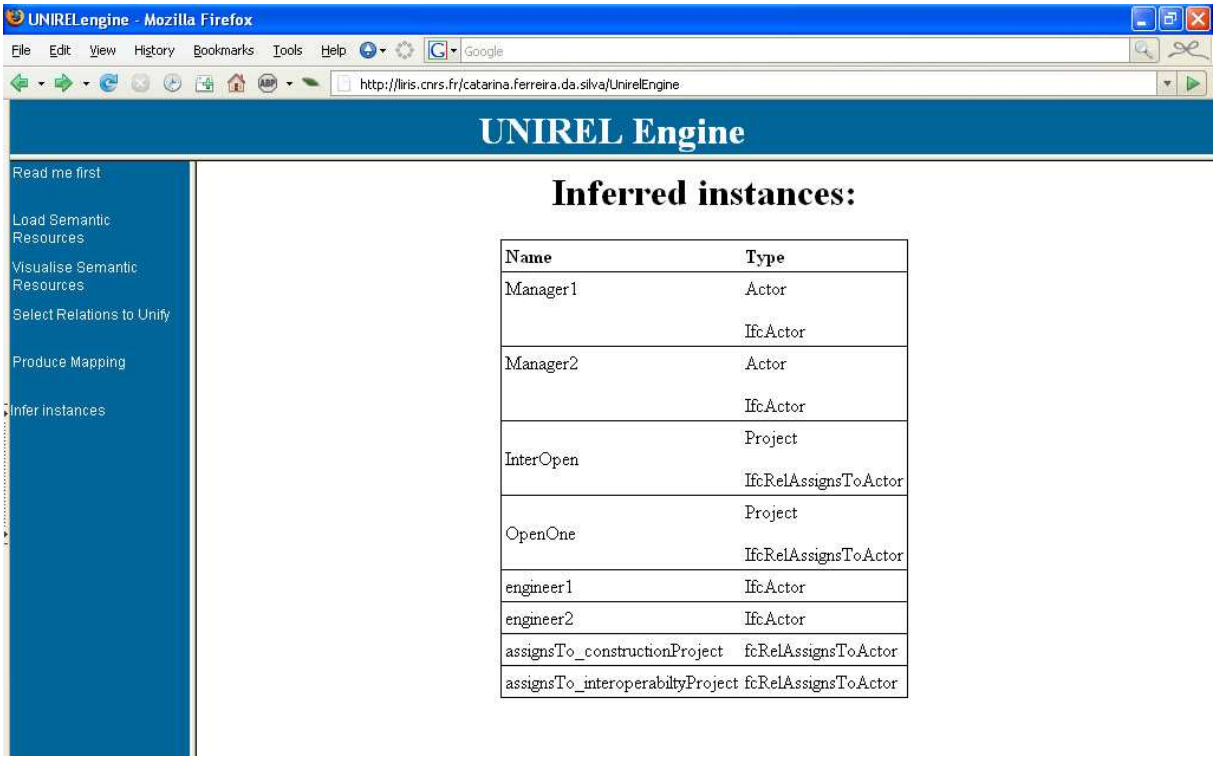

Figure 54 – Inférence d'instances pour l'exemple 2 (cas 4, tableau 18), produites par UNIREL.

Nous constatons que l'existence de restrictions globales uniquement (cas 9) dans les ressources utilisées pour ces tests, permet d'inférer toutes les instances du concept `IfcActor` comme étant des instances du concept `Actor` de la sous-partie d'e-Cognos et vice-versa. De même, les instances du concept `Project`, sont inférés comme instances du concept `IfcRelAssignsToActor` et vice-versa. Le cas 9 n'a pas généré de correspondances, donc l'inférence d'instances entre concepts des deux ressources est le résultat de la création de l'axiome de propriétés équivalentes entre les restrictions globales.

Nous remarquons que l'existence de restrictions locales uniquement (cas 1) ne permet pas d'inférer d'instances, au delà des individus assignées dans les ressources aux concepts qui les instancient. La section suivante présente une analyse globale de notre travail de thèse, et en particulier des résultats présentés dans ce chapitre.

5.5 Analyse globale des résultats

D'une manière globale, en termes d'alignement de ressources sémantiques, notre travail de thèse pourra être comparé à celui de l'outil CtxMatch-1.2 (Bouquet, Serafini et Zanobini 2006). Ce dernier cherche les liens initiaux, ce que nous appelons axiomes d'amorçage, lors d'un appel externe à WordNet pour enrichir les définitions des termes. CtxMatch-1.2 semble, sur ce point, être plus automatisé. Néanmoins, nous pensons qu'il est important d'impliquer les experts du domaine d'application dans la comparaison des ressources sémantiques pour dégager les axiomes d'amorçage entre concepts ou propriétés. Ceci pour assurer autant que possible la validité des résultats par rapport aux connaissances du domaine. Par ailleurs, nous pensons avoir approfondi le travail sur les méthodes sémantiques pour l'alignement de ressources par rapport à la proposition de CtxMatch-1.2, en raison de l'analyse que nous avons faite sur les restrictions des propriétés d'objets. En effet, à notre connaissance, aucune autre étude n'a été menée sur les implications de l'unification et des restrictions de propriétés sur les correspondances produites.

Les tests initiaux du processus de découverte de correspondances sémantiques, ont été réalisés avec le prototype FUNONDIL. La deuxième série de tests à été réalisé avec le prototype UNIREL. Les moteurs d'inférences intégrées, FaCT pour le premier prototype et Pellet pour le deuxième, utilisant un algorithme décidable et complet, nous pouvons considérer qu'ils produisent toutes les correspondances inférables à partir des ressources sémantiques et des axiomes donnés.

5.5.1 Tests avec le prototype FUNONDIL

Au niveau des méta-schémas, pour une moyenne de 5 axiomes d'amorçage par chaque processus de découverte de correspondances entre chaque paire de ressources sémantiques, nous obtenons une moyenne de 17 correspondances sémantiques validées par les experts du domaine. Ceci totalise une centaine de correspondances pour l'ensemble des quatre méta-schémas pris en compte. Des 205 correspondances obtenues, 50 % ont été validées. Les autres 50% ont été écartés parce qu'ils concernent des correspondances redondantes ou entre types de données. Ces dernières ont été exclues, en raison de leur sémantique moins pertinente que les correspondances entre concepts qui dénotent des objets du domaine d'application représentés dans les méta-schémas. Ce taux de correspondances peut facilement être amélioré si nous ne considérons pas les types de données comme des concepts.

Quantitativement, au niveau des schémas, 82 correspondances ont été découvertes par FUNONDIL entre les ontologies e-Cognos et bcBuildingDefinitions, dont 9 équivalences entre concepts. Cependant, ces équivalences n'ayant pas été obtenues à l'aide de la méthodologie proposée, elles ne sont pas comptabilisées parmi les résultats de notre travail. En effet, FaCT permet de considérer que les chaînes de caractères identiques correspondent à des concepts équivalents. Les autres 73 correspondances sont constituées de 37 correspondances de subsomption, 8 de transitivité et 28 de conjonction. Toutes ces correspondances ont été estimées en accord avec la sémantique représentée dans les schémas utilisés par les experts du domaine.

D'un point de vue qualitatif, nous constatons que les correspondances découvertes entre les méta-schémas conviennent plus à un usage par des concepteurs d'ontologies qu'à une utilisation opérationnelle, alors que les correspondances générées avec les schémas s'avèrent plus utilisables dans le cadre d'un projet opérationnel. En effet, les correspondances entre schémas lient des concepts dont la sémantique est plus spécifique.

Enfin, en termes de performance, pour traiter des ressources sémantiques volumineuses, comme l'ontologie e-Cognos qui contient plus de 17 000 concept et relations, le temps de calcul des correspondances, par le moteur FaCT du prototype FUNONDIL, peut atteindre une dizaine de minutes, sur une machine Pentium IV 2.0 GHz et 256MB de ram. Cependant, ce temps est diminué avec l'utilisation de Pellet, plus optimisé que FaCT.

5.5.2 Tests avec le prototype UNIREL

Le test effectué avec une paire de méta-schémas, IFC-kernel et ISO 12006-3, et les axiomes d'amorçage utilisés auparavant avec le prototype FUNONDIL, a généré une correspondance supplémentaire de subsomption entre deux concepts des méta-schémas, en raison de l'unification préalable de deux de leurs propriétés.

En ce qui concerne les sous-parties de l'ontologie e-Cognos et du standard IFC-kernel, les résultats obtenus avec le prototype UNIREL montrent que nous obtenons le plus de correspondances en présence de restrictions locales et globales simultanément. En termes d'inférence d'instances, les combinaisons de restrictions qui possèdent des restrictions locales et globales dans une ressource et que globales dans l'autre ressource (cas 6 et 8 du Tableau 18) permettent d'inférer les instances des concepts d'une ressource comme instances des concepts de l'autre ressource dans les deux sens. Le même résultat se vérifie quand toutes les deux ressources

possèdent des restrictions locales et globales (cas 5) ou quand les ressources ne possèdent que des restrictions globales (cas 9). Ce dernier cas ne produit pas de correspondances, mais il a permis d'inférer des instances entre les concepts des ressources liés par l'unification d'une paire de leurs propriétés impliqués dans des restrictions globales.

L'analyse globale de ces résultats, concernant les correspondances produites et la réalisation (d'instances), nous mènent à conclure que les résultats dépendent pour beaucoup de la richesse des ressources sémantiques, en particulier de la présence de restrictions locales et globales dans les définitions des concepts. Dans les tests effectués, nous tirons partie de la complexité des ontologies, en particulier par les diverses combinaisons de restrictions locales et globales. Cette richesse de résultats serait impossible avec des taxonomies où seule la relation hiérarchique existe.

5.6 Synthèse

Ce chapitre présente les tests effectués et les résultats obtenus avec les prototypes FUNONDIL et UNIREL. Nous avons analysé les correspondances produites entre les méta-schémas et les schémas des ressources sémantiques avec le prototype FUNONDIL lors du projet FUNSIEC. La deuxième suite de tests est consacrée à l'analyse des implications de l'unification de propriétés sur les correspondances découvertes et sur les instances inférées. Ces tests ont été exécutés avec le prototype UNIREL. Finalement, nous avons analysé globalement les résultats du travail que nous avons effectué par rapport à ses limites et en comparaison avec l'outil CtxMatch-1.2 qui utilise aussi une méthode sémantique fondée sur les LD. Les conclusions et perspectives clôturent ce mémoire de thèse.

Conclusions et perspectives

Les pratiques de travail collaboratif conduisent les experts à utiliser des ressources sémantiques externes à leurs organisations, et ceci à travers un ensemble d'outils différents et de manière transparente. Parce que ces ressources sont hétérogènes, plusieurs problèmes en termes d'interopérabilité se posent, tant au niveau syntaxique, structurel que sémantique. Ce dernier niveau englobant les deux précédents, il existe un besoin d'améliorer l'interopérabilité sémantique des ressources pour favoriser l'automatisation des pratiques collaboratives.

Ce travail de thèse adresse ce problème en proposant d'aligner ces ressources en utilisant des méthodes fondées sur la sémantique de leurs contenus. Pour cela, nous avons étudié les problèmes d'hétérogénéité et d'interopérabilité au chapitre 1. L'alignement de ressources sémantiques est proposé par la communauté de recherche comme pouvant y apporter des solutions. Différentes méthodes d'alignement ont été étudiées dans le premier chapitre. Nous y avons également comparé plusieurs systèmes d'alignement d'ontologies et de taxonomies. Le chapitre 2 est consacré aux méthodes et outils utilisés dans ce travail de thèse. Il aborde les langages de représentation, les techniques d'inférence, les moteurs de raisonnement et les interfaces de programmation pour les langages du Web Sémantique. Le chapitre 3 présente la méthodologie que nous proposons pour améliorer l'interopérabilité sémantique entre ressources hétérogènes. Nous détaillons chacune des phases de la méthodologie et proposons des algorithmes pour la découverte de correspondances, l'unification de propriétés et l'inférence d'instances. Le contexte de l'application, les ressources sémantiques du domaine, le scénario d'utilisation de ces ressources, et les prototypes FUNONDIL et UNIREL développés pour mettre en œuvre ce scénario sont présentés au chapitre 4. Le chapitre 5 est dédié à la présentation des deux séries de tests que nous avons effectués avec les prototypes FUNONDIL et UNIREL, ainsi qu'à l'analyse des résultats obtenus.

Nos contributions

D'un point de vue scientifique, nous avons proposé les innovations suivantes :

- Une méthodologie de découverte de correspondances sémantiques, depuis la sélection des ressources jusqu'à l'évaluation des correspondances. Cette méthodologie s'articule en cinq phases : analyse et choix des ressources sémantiques utilisées (section 4.2) ; homogénéisation syntaxique (section 3.2.2) ; découverte de correspondances sémantiques

à l'aide de moteurs d'inférences fondés sur les logiques de description (sections 5.1, 5.2 et 5.3.4) ; unification de propriétés et inférences d'instances (section 5.4) ; modélisation des connaissances du domaine pour quantifier les correspondances sémantiques non équivalentes produites (*cf.* section 3.2.7).

- Un traitement des relations entre concepts. Nous proposons d'unifier des propriétés équivalentes de ressources différentes et étudions les conséquences de cette proposition sur la découverte de correspondances sémantiques et sur l'inférence d'instances. En particulier, nous analysons les implications des restrictions locales et globales dans la découverte de correspondances sémantiques et dans l'inférence d'instances.
- Une modélisation des connaissances du domaine d'application issues des ressources sémantiques, pour aider à la quantification des correspondances sémantiques en utilisant la logique floue.

L'application de ce travail s'appuie sur un ensemble de ressources sémantiques du domaine de la construction, qui ont été traduites dans un format commun (OWL-DL). Nous avons participé au développement du prototype FUNONDIL, afin de tester une partie de la méthodologie proposée. FUNONDIL a été implanté lors du projet FUNSIEC et permet de valider la conversion de formats et la découverte de correspondances. Forte de l'expérience du projet FUNSIEC, nous avons développé UNIREL pour mettre en œuvre l'unification de propriétés, la découverte de correspondances avec un autre moteur d'inférences et l'inférence d'instances. Le prototype UNIREL nous a notamment permis d'approfondir plusieurs aspects qui dépassent les fonctionnalités de FUNONDIL.

L'application de la méthodologie proposée ainsi que les tests effectués montrent qu'il est possible de découvrir des correspondances entre concepts de ressources sémantiques hétérogènes dans un domaine d'application donné. De plus, nous avons approfondi l'étude des relations, afin de découvrir de nouvelles correspondances et d'inférer des instances entre concepts des ressources sémantiques. Le ratio entre les nombres d'axiomes d'amorçage et de correspondances produites et validées par les experts nous encourage à penser que les techniques et outils développés dans cette thèse permettent effectivement d'assister les experts dans le processus de découverte de correspondances sémantiques, et donc de favoriser l'interopérabilité des ressources dans les situations de travail collaboratif.

Perspectives

En termes de réalisation informatique, le travail que nous avons mené pourra être parachevé par l'implantation de l'accès aux correspondances sémantiques découvertes en tant que *middleware* pour les acteurs du domaine d'application, par exemple, par le couplage avec les outils de travail collaboratif qu'ils utilisent actuellement ou à venir. Ce sera d'ailleurs vraisemblablement le cas dans le travail de thèse en cours de Patrick Hoffmann. Ce travail vise à réutiliser les correspondances que nous avons produites afin de quantifier les proximités sémantiques entre paires de concepts.

Les améliorations à apporter à moyen terme sont les suivantes :

- Trouver, de manière automatique, des propriétés candidates à participer aux axiomes d'équivalence pour le processus d'unification de propriétés. Ceci pourrait être mené en fonction par exemple des caractéristiques des propriétés et des concepts qui entourent la ou les propriété(s) candidate(s) à une équivalence.
- Étudier l'influence d'autres types de restrictions, telles que les restrictions de cardinalité et les restrictions de valeurs, sur la découverte de correspondances et sur l'inférence d'instances.
- Appliquer le processus d'unification de propriétés à d'autres types de propriétés, comme les propriétés symétriques, transitives ou fonctionnelles.
- Exploiter les définitions ou les annotations de concepts en langage naturel que certaines ressources possèdent dans leurs formats d'origine et qui ne sont pas prises en compte par la méthode sémantique d'alignement que nous avons employée. L'utilisation d'autres types de méthodes d'alignement, et notamment terminologiques, peut tirer partie des définitions en langage naturel pour générer des correspondances. Ceci pourrait apporter des correspondances complémentaires du travail que nous avons mené, voire aider à la découverte d'axiomes d'amorçage pour le processus d'inférences.

Les résultats de ces travaux permettent également d'envisager à plus long terme de :

- Tester la modélisation des connaissances du domaine réalisée en logique floue (*cf.* section 3.2.5) pour l'évaluation des correspondances sémantiques dans le cadre d'un système d'inférence floue.

- Réduire le temps de calcul des correspondances sémantiques obtenues par la méthode sémantique employée dans ce travail de thèse, par l'utilisation de techniques de raisonnement distribué et la fragmentation des ressources sémantiques.

Références bibliographiques

Abels, S., Haak, L. et Hahn, A. (2005). Identifying ontology integration methods and their applicability in the context of product classification and knowledge integration tasks. Rapport technique n° WI-OL-TR-01-2005, ISSN 1861-3748.

Amsili, P. (2007). Cours sémantique computationnelle. Disponible à http://www.linguist.jussieu.fr/~amsili/Ens/LI334.php (consulté le 8 janvier 2007).

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (Eds) (2003). The Description Logic Handbook - Theory, Implementation and Applications. Cambridge University Press. ISNB 0-521-78176-0

Baader, F. et Nutt, W. (2003). Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, et P. Patel-Schneider (Eds.) The Description Logic Handbook: Theory, Implementation and Applications, chapitre 2, pp. 48-101, Cambridge University Press, ISNB 0-521-78176-0

Baader, F. et Sattler, U. (2000). Tableau Algorithms for Description Logics. In R. Dyckhoff (Ed.) Proceedings of the International Conference on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000), St Andrews, Scotland, UK, Lecture Notes in Artificial Intelligence, Springer-Verlag, vol. 1847, pp. 1-18.

Bach, T. L. (2006). Construction d'un Web Sémantique multi-points de vue. Thèse de doctorat. École de Mines de Paris à Sophia-Antipolis.

Barresi, S., Rezgui, Y., Meziane, F., Lima, C., Fiès, B., Ferreira da Silva, C., Pimentão, J.P., Sousa, P., et Acevedo, J. (2004). D1.1 - State of the Art Analysis. Feasibility study for a UNified Semantic Infrastructure in the European Construction sector (FUNSIEC) project. eContent European Programme 42059Y3C3FPAL2.

Batini, C., Lenzerini, M. et Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. ACM Press, New York, USA, vol. 18 (4), pp. 323-364.

Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. et Stein, L. A. (2004a). OWL Web Ontology Language Reference. W3C Recommendation 10 February 2004. Disponible à http://www.w3.org/TR/owl-ref/ (consulté le 17 mai 2004).

Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. et Stein, L. A. (2004b). Appendix D. Changes from DAML+OIL, In OWL Web Ontology Language Reference. W3C Recommendation 10 February 2004. Disponible à http://www.w3.org/TR/2004/REC-owl-ref-20040210/#appD (consulté le 17 mai 2004).

Bechhofer, S., Lord, P. et Volz, R. (2003). Cooking the Semantic Web with the OWL API. 2nd International Semantic Web Conference (ISWC), Sanibel Island, Florida, USA.

Bechhofer, S. (2003). The DIG Description Logic Interface: DIG/1.1, Proceedings of DL 2003 Workshop.

Beckett, D. (Ed.) (2004). RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February 2004. Disponible à http://www.w3.org/TR/rdf-syntax-grammar/ (consulté le 15 février 2004).

Berners-Lee, T., Hendler, J. et Lassila, O. (2001). The Semantic Web, Scientific American.

Bohring, H., Auer, S. (2005). Mapping XML to OWL Ontologies. Proceedings of 13th Leipziger Informatik-Tage (LIT 2005), Lecture Notes in Informatics (LNI).

Bouquet, P., Serafini, L. et Zanobini, S. (2003). Semantic Coordination: a new approach and an application. International Semantic Web Conference 2003, Lecture Notes in Computer Science, vol. 2870/2003, Springer Berlin / Heidelberg, pp. 130-145, ISBN 978-3-540-20362-9.

Bouquet, P., Ehrig, M., Euzenat, J., Franconi, E., Hitzler, P., Krötzsch, M., Serafini, L., Stamou, G., Sure, Y. et Tessaris, S. (2005). D2.2.1 Specification of a common framework for characterizing alignment KWEB/2004/D2.2.1/v2.0. Projet Knowledge web – realizing the semantic web, KWEB EU-IST-2004-507482- work package 2.2.

Bouquet, P., Serafini, L. et Zanobini, S. (2006). Bootstrapping semantics on the web: meaning elicitation from schemas. In Proceedings of the World Wide Web Conference (WWW), pp. 505-512.

Brachman, R. et Levesque, H. (2004). Knowledge representation and reasoning. Morgan Kaufmann Publishers, Elsevier, San Francisco, USA. ISNB 1-55860-932-6

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. et Yergeau, F. (Eds.) (2006a). Extensible Markup Language (XML) 1.0 (Fourth Edition) W3C Recommendation 16 August 2006. Disponible à http://www.w3.org/TR/xml/ (consulté le 9 septembre 2006).

Bray, T., Hollander, D., Layman, A. et Tobin, R. (Eds.) (2006b). Namespaces in XML 1.0 (Second Edition) W3C Recommendation 16 August 2006. Disponible à http://www.w3.org/TR/xml-names/ (consulté le 9 septembre 2006).

Brickley, D. et Guha, R. V. (2004). RDF Vocabulary Description Language 1.0: RDF Schema W3C Recommendation 10 February 2004. . Disponible à http://www.w3.org/TR/rdf-schema/ (consulté le 17 mai 2004).

Cardoso, J. et Sheth, A. (2006). The semantic web and its applications. In Cardoso, J. et Sheth, A. (Eds.), Semantic Web Services, Processes and Applications, Series Semantic Web and Beyond, vol. 3, pp. 1-33. ISBN-10: 0-387-30239-5, ISBN-13: 978-0-387-30239-3

Choi, N., Song, Il-Y. et Han, H. (2006). A survey on ontology mapping. ACM SIGMOD Record, vol. 35 (3), ACM Press, New York, USA, pp. 34-41, ISSN:0163-5808

Cohen, W., Ravikumar, P., et Fienberg, S. (2003). A comparison of string metrics for matching names and records. Proc. KDD-2003 Workshop on Data Cleaning and Object Consolidation.

Connolly, D., Van Harmelen, F., McGuinness, D. L., Patel-Schneider, P. F. et Stein, L. A. (2001). DAML+OIL (March 2001) Reference Description, W3C Note 18 December 2001. Disponible à http://www.w3.org/TR/daml+oil-reference (consulté le 15 novembre 2003).

Costa, C. et Cunha, P. R. (2006). Workflows without engines: Modeling for today's heterogeneous information systems, Journal of Issues in Informing Science and Information Technology, Vol. 3, pp. 175-187.

Degoulet, P., Fieschi, M. et Attali, C. (1997). Les enjeux de l'interopérabilité sémantique dans les systèmes d'information de santé. In Kohler, F., Brémond, M. et Mayeux, D. (Eds.), *Informatique et Gestion Médicalisée*, vol. 9, Paris, Springer-Verlag France.

Djioua, B. (2000). Modélisation Informatique d'une base de connaissances lexicales (DiSSC). Thèse de doctorat. Institut des Sciences Humaines Appliquées, Université de Paris IV – Sorbonne. Section 'Sémantique cognitive'. Disponible à http://www.lalic.paris4.sorbonne.fr/djioua/these/finale.html (consulté lé 30 janvier 2007).

Djurić, D. (2004). MDA-based Ontology. Infrastructure International Journal on Computer Science and Information Systems, vol. 1, N°. 1, pp. 91-116.

Doan, A., Madhavan, J., Domingos, P. et Halevy, A. (2002). Learning to Map between Ontologies on the Semantic Web. In Proceedings International WWW Conference, Honolulu, Hawaii, USA. Disponible à http://wwwconf.ecs.soton.ac.uk/archive/00000320/ (consulté le 3 juillet 2004).

Ehrig, M. et Staab, S. (2004). QOM – Quick Ontology Mapping. In van Harmelen, F., McIlraith, S. et Plexousakis, D. (Eds.), Proc. of the Third International Semantic Web Conference, Hiroshima, Japan. The Semantic Web – ISWC 2004, vol. 3298/2004, LNCS, Springer Berlin / Heidelberg, pp. 683 - 697, ISBN978-3-540-23798-3.

Ehrig, M. et Sure, Y. (2005). FOAM – Framework for Ontology Alignment and Mapping Results of the Ontology Alignment Evaluation Initiative. In Ashpole, B., Ehrig, M., Euzenat, J. et Stuckenschmidt, H. (Eds.), Proc. of the Workshop on Integrating Ontologies, vol. 156, pp. 72--76. CEUR-WS.org.

Ehrig, M., Staab, S. et Sure, Y. (2005). Supervised learning of an ontology alignment process. In Althoff K-D., Dengel A., Bergmann R., Nick M. et Roth-Berghofer T. (Eds.) Professional Knowledge Management: Third Biennial Conference (WM 2005), vol. 3782 of LNAI, Springer, pp. 508 - 517, Kaiserslautern, Germany.

El-Diraby, T., Fiès, B. et Lima, C. (2002). D3.6: The e-COGNOS Ontology V1.1.0 - WP3. E-Cognos project IST-2000-28671.

Euzenat, J. et Stuckenschmidt, H. (2002). The 'family of languages' approach to semantic interoperability. In Omelayenko, B. et Klein, M. (Eds.), Proc. ECAI 2002 workshop on Knowledge Transformation for the Semantic Web, pp. 92-99, Lyon, France.

Euzenat, J. (2004). An API for Ontology Alignment. In van Harmelen, F., McIlraith, S. et Plexousakis, D. (Eds.), Proc. of the Third International Semantic Web Conference, Hiroshima, Japan. The Semantic Web – ISWC 2004, vol. 3298/2004, LNCS, Springer Berlin / Heidelberg, pp. 698-712, ISBN978-3-540-23798-3.

Euzenat, J., Le Bach, T., Barrasa, J., Bouquet, P., De Bo, J., Dieng, R., Ehrig, M., Hauswirth, M., Jarrar, M., Lara, R., Maynard, D., Napoli, A., Stamou, G., Stuckenschmidt, H., Shvaiko, P., Tessaris, S.,

Van Acker, S. et Zaihrayeu, I. (2004). D2.2.3: State of the art on ontology alignment. Projet Knowledge web – realizing the semantic web, KWEB EU-IST-2004-507482- work package 2.2.

Fellbaum, C. (1998). Towards a representation of idioms in WordNet. In: Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems, Montreal, Canada, 1998.

Feferman, S. (2006). Tarski's influence on computer science. Logical Methods in Computer Science, vol. 2 issue 3, paper 6. Disponible à http://dx.doi.org/10.2168/LMCS-2(3:6)2006 (consulté le 27 février 2007).

Ferrario, R. (2006). Who Cares about Axiomatization? Representation, Invariance, and Formal Ontologies. Epistemologia, Special Issue on the Philosophy of Patrick Suppes, vol. 2.

Ferreira da Silva, C., Lima, C., Médini, L., Ghodous, P. et Zarli, A. (2006a). Enhancing Semantic Interoperability among Semantic Resources for Construction. In Rivard, H., Melhem, H. and Miresco, E. (Eds.), Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montreal, Canada. pp. 2487-2496. ISBN 2-921145-58-8.

Ferreira da Silva, C., Médini, L., Abdul Ghafour, S., Hoffmann, P., Ghodous, P. et Lima, C. (2006b). Semantic Interoperability of Heterogeneous Semantic Resources. Electronic Notes in Theoretical Computer Science 150(2):71-85, Elsevier, http://dx.doi.org/10.1016/j.entcs.2005.11.035

Ferreira da Silva, C., Médini, L., Lima, C., Ghodous, P. (2006c). Improving Mappings Discovery for Semantic Interoperability. 13th ISPE International Conference on Concurrent Engineering: Research and Applications, Leading the Web in Concurrent Engineering, Antibes, France. pp. 227-234. Frontiers in Artificial Intelligence and Applications 143. IOS Press Nieuwe Hemweg 6B 1013 BG Amsterdam Nethe. ISBN 1-58603-651-3, ISSN 0922-6389.

Ferreira da Silva, C., Médini, L., Abdul Ghafour, S., Hoffmann, P., Ghodous, P. (2005). Interoperability of heterogeneous semantic resources. In Thiran, P., Risch, T. and Benslimane D. (Eds), International Workshop on Database Interoperability (InterDB 2005) in connection with the 7th International Conference on Coordination Models and Languages, Namur, Belgique. pp. 74-89.

Fournier-Viger, P. (2005). Un modèle de représentation des connaissances à trois niveaux de sémantique pour les systèmes tutoriels intelligents. Mémoire de master (M.Sc.), Université de Sherbrooke, Sherbrooke, Canada, chapitre 4.

Genesereth, M. R. et Nilsson, N. J. (1987). Logical Foundations of Artificial Intelligence. Morgan-Kaufman, Los Altos, California, USA. Guarino, N. (1998). Formal Ontology and Information Systems. In Guarino, N., (Ed.), *Formal Ontology in Information Systems*, Series Frontiers in Artificial Intelligence and Applications, vol. 46, pp. 3-15. IOS Press, Amsterdam, Netherlands, ISBN 90 5199 399 4.

Gil, R., García, R., et Delgado, J. (2005). An interoperable framework for IPR using web ontologies. Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2005), IAAIL Workshop Series, pp. 135-148, Wolf Legal Publishers, ISBN 90-5850-504-9

Giunchiglia, F., Shvaiko, P. et Yatskevich M. (2004). S-Match: an algorithm and an implementation of semantic matching. In Proc. of the European Semantic Web Symposium (ESWS), LNCS 3053, pp. 61–75.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, vol. 43, issue 5-6, pp. 907-928, Academic Press, Duluth, MN, USA, ISSN:1071-5819.

Guillaume, S. (2005). Représentation des connaissances et systèmes d'inférence floue. Habilitation à Diriger des Recherches. Université Paul Sabatier, Toulouse III, 36 pages.

Guzelian, G. et Cauvet, C. (2005). Modèles Sémantiques de Composants pour l'Ingénierie des Systèmes d'Information. 16es journées francophones d'ingénierie des connaissances (IC 2005), Plate-forme AFIA, pp. 145-156, Nice, 1-3 juin 2005.

Heflin, J. et Hendler, J. (2000). Semantic Interoperability on the Web. Proceedings of Extreme Markup Languages 2000. Graphic Communications Association, Alexandria, VA, pp. 111-120. Disponible à http://www.cs.umd.edu/projects/plus/SHOE/pubs/extreme2000.pdf (consulté le 14 janvier 2004).

Hofweber, T. (2005). Section 3. Ontology. In Logic and Ontology. In Zalta, E. N., (Ed.), The Stanford Encyclopedia of Philosophy (Winter 2005 Edition). Disponible à http://plato.stanford.edu/archives/win2005/entries/logic-ontology/ (consulté le 28 décembre 2005).

Horridge, M., Bechhofer, S. et Noppens, O. (2007). Igniting the OWL 1.1 Touch Paper: The OWL API. OWLED 2007, 3rd OWL Experienced and Directions Workshop, Innsbruck, Austria, June 2007.

Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R. et Wang, H.H. (2006). The Manchester OWL Syntax. In: Proc. of the OWL Experiences and Directions Workshop (OWLED'06) at the ISWC'06.

Horrocks, I., Patel-Schneider, P. F. et Van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics, 1(1), pp. 7-26.

Horrocks, I. et Sattler, U. (2007). A Tableaux Decision Procedure for SHOIQ. Journal of Automated Reasoning, Springer Verlag.

Horrocks, I. (2003). Implementation and Optimisation Techniques. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, et P. Patel-Schneider (Eds.) The Description Logic Handbook: Theory, Implementation and Applications, chapitre 9, pp. 312-354, Cambridge University Press, ISNB 0-521-78176-0

Horrocks, I. (1998). The FaCT system. In de Swart, H. (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98, number 1397 in Lecture Notes in Articial Intelligence, pp. 307-312, Springer-Verlag.

Hughes, G. (2004). A Guide to Interoperability for Regional Initiatives. Brussels, Belgium: eris@ European Regional Information Society Association.

IAI, International Alliance for Interoperability, http://www.iai-international.org/

IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. Institute of Electrical and Electronics Engineers. New York, NY: 1990.

IFC2x3, Industry Foundation Classes version 2x, édition 3, International Alliance for Interoperability, http://www.iai-international.org/Model/R2x3_final/index.htm

ISO/PAS 16739:2005, Industry Foundation Classes, Release 2x, Platform Specification (IFC2x Platform), http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=38056&scopelist=PROGRAMME

ISO 10303-11:1994 (1994). Systèmes d'automatisation industrielle et intégration – Représentation et échange de données de produits – Partie 11: Méthodes de description: Manuel de référence du langage EXPRESS.

ISO 2788:1986 (1986). Documentation - Guidelines for the establishment and development of monolingual thesauri. 2nd ed. (32 p.)

ISO 10303-11 (2001). EXPRESS-G - annex A. USA: National Institute of Standards and Technology.

ISO 12006-3:2007 (2007). Building construction - Organization of information about construction works - Part 3: Framework for object-oriented information. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=38706&COMMID=&scopelist=PROGRAMME

ISO/PAS 12006-3 version 2 (2002). Building construction - Organization of information about construction works - Part 3: Framework for object-oriented information exchange - revised version 2.

Kalfoglou, Y. et Schorlemmer, M. (2003). Ontology Mapping: The State of the Art. The Knowledge Engineering Review Journal, vol. 18:1, pp. 1–31, Cambridge University Press, UK.

Kashyap, V. et Sheth, A. (1996). Semantic heterogeneity in global information systems: The role of métadata, context and ontologies. In Papazoglou, M. P. and Schlageter, G., (Eds.), *Cooperative Information Systems*: *Current Trends and Directions*, pp. 139-178. Academic Press, San Diego.

Klein, M. (2001). Combining and relating ontologies: an analysis of problems and solutions. In Gomez-Perez, A., Gruninger, M., Stuckenschmidt, H., et Uschold, M., (Eds), Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, WA.

Laskey, K., Laskey, K. et Costa, P. (2006). A Proposal for a W3C XG on Uncertainty Reasoning for the World Wide Web. In Costa, P. Laskey, K., Laskey, K., Fung, F. et Pool, M., (Eds), Proceedings of the Second ISWC Workshop on Uncertainty Reasoning for the Semantic Web, Athens, Georgia, USA. Disponible à http://www.iet.com/iswc/2006/ursw/files/papers/URSW06_P5_LaskeyCostaLaskey.pdf (consulté le 17 décembre 2006).

Le Duc, C. (2004). Transformation d'ontologies basées sur la logique de description, application dans le commerce électronique. Thèse de doctorat, Université de Nice - Sophia Antipolis.

Li, J. (2004). LOM: A Lexicon-based Ontology Mapping Tool. Proceedings of the Performance Metrics for Intelligent Systems (PerMIS'04).

Lima, C., Ferreira da Silva, C., et Pimentão, J.P. (2006). Assessing the quality of mappings between Semantic Resources in Construction. Ian Smith, F.C. (Ed.), Intelligent Computing in Engineering and Architecture - 13th EG-ICE Workshop 2006, Ascona, Switzerland. pp. 416-427. LNCS, Lecture Notes in Artificial Intelligence, vol. 4200/2006. Springer Berlin / Heidelberg, DOI 10.1007/11888598, ISBN 978-3-540-46246-0.

Lima, C., Ferreira da Silva, C., Barresi, S., Rezgui, Y., Meziane, F., Pimentão, J.P., Sousa, P., et Acevedo, J. (2005a). D2.1 – The OSIECS Specification. Feasibility study for a UNified Semantic Infrastructure in the European Construction sector (FUNSIEC) project. eContent European Programme 42059Y3C3FPAL2.

Lima, C., Ferreira da Silva, C., Le Duc, C., Barresi, S., Pimentão, J.P., Sousa, P., et Acevedo, J. (2005b). D2.2 – The OSIECS Infrastructure – Assessment and Validation. Feasibility study for a UNified Semantic Infrastructure in the European Construction sector (FUNSIEC) project. eContent European Programme 42059Y3C3FPAL2.

Lima, C., Ferreira da Silva, C., Sousa, P. et Pimentão, J.P. (2005c). Interoperability among Semantic Resources in Construction: Is it Feasible?. In Scherer, R.J., Katranuschkov, P. and Schapke, S.-E. (Eds.) 22nd CIB-W78 Conference Information Technology in Construction, Dresde, Allemagne. pp. 285-292. CIB-w78 304. CIB publications Institute for Construction Informatics, Technische Universität Dresden, Germany. ISBN 3-86005-478-3.

Lima, C., Storer, G., Zarli, A. et Ferreira da Silva, C. (2005d). Towards a framework for managing standards-base semantic e-Resources in the European Construction Industry. In Tommelein, I. D. (Ed) Construction Research Congress 2005 BROADENING PERSPECTIVES (CRC 2005), San Diego, California, EUA. pp. 1-10. American Society of Civil Engineers, ASCE International Headquarters 1801 Alexander Bell Drive Reston, VA 20191-4400 USA. ISBN 0-7844-0754-1.

Lima, C., Ferreira da Silva, C. et Le Duc, C. (2005e). A Framework to Support Interoperability among Semantic Resources. In Konstantas, D., Bourrières, J.-P., Léonard, M. et Boudjlida, N. (Eds.) First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'2005), Genève, Suisse. pp. 87-98. Springer-Verlag London. ISBN 1-84628-151-2.

Lima, C., Fiès, B., Ferreira da Silva, C. et Barresi, S. (2004). Setting up the Open Semantic Infrastructure for the European Construction Sector - the FUNSIEC Project. European Conferences on Product and Process Modeling in the Building Industry (Ecppm), Istanbul, Turquie.

Lima, C. P., Fiès, B., Lefrancois, G. and Diraby, T. E. (2003). The challenge of using a domain Ontology in KM solutions: the e-COGNOS experience. In: 10TH ISPE 2003, Funchal, Portugal. International Conference on Concurrent Engineering: Research and Applications, pp. 771-778.

Liu, K. (2000). Semiotics in Information Systems Engineering, pp. 13-16. Cambridge University Press, United Knigdom, ISBN 0 521 59335 2.

Madhavan, J., Bernstein, P. et Rahm, E. (2001). Generic Schema Matching with Cupid. Proceedings of the 27th International Conference on Very Large Data Bases, pp 49–58. Morgan Kaufmann Publishers Inc.

Maedche, A., Motik, B., Silva, N., et Volz, R. (2002). MAFRA - A Mapping Framework for Distributed Ontologies. Proc. of the 13th European Conference on Knowledge Engineering and Knowledge Management (EKAW-2002),, Sigüenza, Spain, Springer-Verlag.

McBride, B. (2001). Jena: Implementing the RDF Model and Syntax Specification. Semantic Web Workshop, WWW2001. Disponible à http://www.hpl.hp.com/personal/bwm/papers/20001221-paper/

McGuinness, D. L. et Van Harmelen, F. (Eds) (2004). OWL Web Ontology Language Overview, W3C Recommendation 10 February 2004. Disponible à http://www.w3.org/TR/owl-features/ (consulté le 15 février 2004).

Mitra, P. et Wiederhold, G. (2002). Resolving Terminological Heterogeneity in Ontologies. Proc. of Workshop on Ontologies and Semantic Interoperability, 15th European Conference on Artificial Intelligence (ECAI), Lyon, France.

Missikoff, M. (2002). Harmonise: An Ontology-Based Approach for Semantic Interoperability. In European Research Consortium for Informatics and Mathematics, Number 51.

Möller, R. et Haarslev. V. (2003). Description logic systems. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, et P. Patel-Schneider (Eds.) The Description Logic Handbook: Theory, Implementation and Applications, chapitre 8, pp. 282–305, Cambridge University Press, ISNB 0-521-78176-0

Motik, B., Patel-Schneider, P. F. et Horrocks, I. (2006). OWL 1.1 Web Ontology Language, Structural Specification and Functional-Style Syntax. W3C Member Submission 19 December 2006. Disponible à http://www.w3.org/Submission/2006/SUBM-owl11-owl_specification-20061219/

Naiman, E. C. et Ouksel, A. M. (1995). A Classification of Semantic Conflicts in Heterogeneous Database Systems. Journal of Organizational Computing, 5(2), pp. 167-193.

NF Z47-100 1981 (1981). Documentation - Règles d'établissement des thésaurus monolingues. Association Française de Normalisation, pp. 461-480.

Noy, N. F. et Musen, M. A. (1999). An algorithm for merging and aligning ontologies: automation and tool support. AAAI Workshop and ontology management.

Noy, N. F. et Klein, M. (2002). Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems, 6(4), pp. 428-440.

Noy, N.F. et Musen, M. A. (2001). Anchor-PROMPT: Using Non-Local Context for Semantic Matching. Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001), Seattle, WA, USA.

Patel-Schneider, P. F., Hayes, P. et Horrocks, I. (Eds) (2004). OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation 10 February 2004. Disponible à http://www.w3.org/TR/owl-semantics/

Parsia, B., Sirin, E. et Kalyanpur, A. (2005). Debugging owl ontologies. In Proceedings of the 14th International World Wide Web Conference (WWW-2005), Chiba, Japan.

Pinto, S. H. et Martins, J. P. (2001). A methodology for ontology integration. International Conference On Knowledge Capture. ACM Special Interest Group on Artificial Intelligence (SIGART), ACM Press, New York, USA, pp. 131-138, ISBN 1-58113-380-4.

Porter, M.F. (1980). An Algorithm for Suffix Stripping, Program, 14(3): 130-137.

Rahm, E. et Bernstein, P.A. (2001). A survey of approaches to automatic schema matching. VLDB Journal: Very Large Data Bases, 10:4, pp. 334-350. Disponible à http://citeseer.ist.psu.edu/rahm01survey.html (consulté le 27 mai 2004).

Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence, 11(11), pp. 95–130.

Reynaud, C. et Safar, B. (2007). Utilisation de connaissances supplémentaires pour la découverte de mappings dans le système TaxoMap. Atelier DECOR – passage à l'échelle des techniques de découverte de correspondances. Workshop EGC 2007, 23 janvier 2007, Namur, Belgique.

Schenck, D. et Wilson, P. (1994). Information modeling: the EXPRESS way. Oxford University Press, New York. ISBN 0-19-508714-3.

Sheth, A et Kashyap, V. (1992). So Far (Schematically) yet So Near (Semantically). Proceedings of the MT DS-5 Conference on Semantics of Interoperable Database Systems, Lorne, Australia.

Sheth, A., Ramakrishnan C. et Thomas C. (2005). Semantics for the Semantic Web: The Implicit, the Formal and the Powerful. In Sheth, A., (Ed.), *International Journal on Semantic Web & Information Systems*, 1(1), pp. 1-18. Disponible à http://lsdis.cs.uga.edu/~cartic/publications/SRT05-IJ-SW-IS (consulté le 17 novembre 2006).

Shvaiko, P. (2006). Iterative schema-based semantic matching. Thèse de doctorat, Università degli Studi di Trento, novembre 2006.

Silva, N. (2004). Multi-dimensional Service-oriented Ontology Mapping, 3.2 Ontology Vs. Database schema, page 32. Thèse de doctorat. Universidade de Trás-os-Montes e Alto Douro, Portugal.

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A. et Katz, Y. (2007). Pellet: A practical OWL-DL reasoner, Journal of Web Semantics: Science, Services and Agents on the World Wide Web. In E. Wallace (Ed.) Software Engineering and the Semantic Web, 5(2), pp. 51-53.

Solar, G. V. et Doucet, A. (2002). Médiation de données : solutions et problèmes ouverts. Actes des deuxièmes assises nationales du GdR I3. Nancy, décembre 2002. Disponible à http://www.irit.fr/GDR-I3/fichiers/assises2002/papers/12-MediationDeDonnees.pdf (consulté le 14 mars 2005).

Sowa, J. (2000). Knowledge Representation - logical, philosophical, and computational foundations, Brooks/Cole, USA. ISBN 0 534-94965-7.

Straccia, U. et Troncy, R. (2005). oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies. In 6th International Conference on Web Information Systems Engineering (WISE'05), vol. LNCS 3806, pp. 133-147, New York City, New York, USA, November 20-22.

Stuckenschmidt, H. (2003). Ontology-Based Information Sharing in Weakly-Structure Environments Ph.D. Thesis,Faculty of Sciences, Vrije Universiteit Amsterdam.

Stumme, G. et Maedche, A. (2001). FCA-Merge: Bottom-Up Merging of Ontologies. In Proc. 17th Intl. Conf. on Artificial Intelligence (IJCAI '01), Seattle, WA, USA, pp. 225-230.

Suwanmanee, S., Benslimane, D., Champin, P-A. et Thiran, P. (2005). Wrapping And Integrating Heterogeneous Databases With OWL. 7th International Conference on Enterprise Information Systems, (ICIES 2005), Chin-Sheng Chen, Joaquim Filipe, Isabel Seruca, José Cordeiro (Eds), Miami, Florida, USA. pp. 11-18, ISBN 972-8865-19-8.

Thomasson, A. (2004). Categories. In Zalta, E. N., (Ed.), *The Stanford Encyclopedia of Philosophy (Fall 2004 Edition)*. Disponible à http://plato.stanford.edu/archives/fall2004/entries/categories/ (consulté le 7 octobre 2004).

Tolman, F., Böhms, M., Lima, C., Van Rees, R., Fleuren, J., et Stephens, J. (2001). eConstruct: expectations, solutions and results. In Rezgui, Y. et Zarli, A. (Eds.), ITcon Vol. 6, Special Issue Information and Communication Technology Advances in the European Construction Industry, pp. 175-197. Disponible à http://www.itcon.org/2001/13 (consulté le 8 janvier 2004)

Tsarkov, D. et Horrocks, I. (2004). Efficient reasoning with range and domain constraints. In Proc. of the 2004 Description Logic Workshop (DL 2004), pp. 41-50.

UML, Unified Modeling Language. Disponible à http://www.uml.org/

Vaillant, P. (1999). Sémiotique des langages d'icônes. Glossaire de sémiotique. Éditions Honoré Champion, Paris, ISBN 2-7453-0242-6.

Van Rees, R., Tolman, F. et Beheshti (2002). How BcXML handles construction semantics. CIB W78 conference proc. distributing knowledge in building. Disponible à http://www.cib-w78-2002.dk/papers/papers/cib02-12.pdf

Visser, P. R. S., Jones, D. M., Bench-Capon, T. J. M. et Shave, M. J. R. (1998). Assessing Heterogeneity by Classifying Ontology Mismatches. In Guarino, N., (Ed.), *Formal Ontology in Information Systems*, Series Frontiers in Artificial Intelligence and Applications, vol. 46, pp. 148-162. IOS Press, Amsterdam, Netherlands, ISBN 90 5199 399 4.

Vossen, P. (2002). "EuroWordNet" In: *EuroWordNet Project-report*. Disponible à http://www.vossen.info/docs/2002/EWNGeneral.pdf (consulté le 17 octobre 2006).

Wetherill, M., Yacine, R., Lima, C. et Zarli, A. (2002). Knowledge management for the construction industry: the e-cognos project, ITcon Vol. 7, Special Issue ICT for Knowledge Management in Construction, pp. 183-196. Disponible à http://www.itcon.org/2002/12 (consulté le 14 janvier 2004).

Wilkinson, K., Sayers, C., Kuno, H. et Reynolds, D. (2003). Efficient RDF Storage and Retrieval in Jena2. First International Workshop on Semantic Web and Databases, 7 September 2003, Berlin, Germany.

Willerval, B., Boisseau, Y., Serres-Cousiné, H. (Eds) (1989). Le petit Larousse illustré 1990. Librairie Larousse. ISBN 2-03-301190-9.

Wilson, P. 1998. STEP and EXPRESS. Additional contribution to an Invitational NSF Workshop on Distributed Information, Computation and Process Management for Scientific and Engineering Environments. Disponible à http://deslab.mit.edu/DesignLab/dicpm/step.html

Zacklad, M. (2005). Introduction aux ontologies sémiotiques dans le Web Socio Sémantique. 16es journées francophones d'ingénierie des connaissances (IC 2005). Nice, 1-3 juin 2005, pp. 241- 252. Presses Universitaires de Grenoble. ISBN 2 7061 1284 0.

Zadeh, L.A. (1965). "Fuzzy Sets." *Information and Control*, (8), pp. 338-353.

Zhang, J. (1992). Classifying approaches to semantic heterogeneity in multidatabase systems. IBM Centre for Advanced Studies Conference, Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research. Vol. 2, pp. 153-173. Disponible à http://portal.acm.org/citation.cfm?id=962271 (consulté le 3 mars 2004)

Annexes

Annexe 1 – Langages de représentation XML, RDF et RDFS

Extensible Markup Language (XML)

Un méta-langage à balises combine le texte et les informations supplémentaires, *i.e.* des méta-données, sur le texte. L'auteur d'un document XML est libre de définir la structure du document ainsi que les balises qu'il préfère. Mais le langage XML impose quelques règles syntaxiques afin qu'un document XML puisse être traité par n'importe quelle machine qui adopte ce standard *de facto*. Une des règles de base est, par exemple : chaque élément XML qui a une balise ouvrante doit posséder aussi une balise fermante. Sinon, le document est mal formé. Une autre des règles imposées à un document XML est l'existence d'une seule racine.

Un document XML est valide s'il satisfait un ensemble de règles de structuration. Ces règles sont spécifiées dans un *Document Type Définition* ou dans un *Schéma XML*.

L'espace de noms (*namespace*) en XML est une chaîne de caractères qui identifie de manière unique tout élément dans un document XML. Un espace de noms est une référence URI (*Uniform Resource Identifier* - identifiant uniforme de ressource). L'URI est utilisé pour identifier et localiser une ressource, soit concrète et localisable par l'Internet (par exemple, une page web), soit abstraite (par exemple, le nom d'un concept). L'espace de noms en XML est aussi une recommandation du W3C, dont la deuxième édition date du 16 août 2006 (Bray *et al.* 2006b). Un élément défini dans un espace de nom correspond à un objet unique. Ainsi, deux objets différents ont deux noms locaux différents, *i.e.* ils peuvent partager le même espace de nom mais leur identification est complétée par un nom local suffixe de l'espace de nom.

Exemple d'une partie du schéma XML de la taxonomie bcXML[60] - *Building Construction Extensible Markup Language* (Tolman *et al.* 2001) (*cf.* section 4.2) :

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema  xmlns:xs=http://www.w3.org/2001/XMLSchema
xmlns="http://www.bcXML.org/2003/bcXML" (...)>
<!--(...)-->
<xs:element name="ExternalReference" type="ExternalReferenceType"
minOccurs="0" maxOccurs="unbounded"/>
<!--(...)-->
        <xs:complexType name="ExternalReferenceType" abstract="false">
```

[60] Voir aussi http://bcxml.sourceforge.net/formats/taxonomy/xtd-explanation.html

```
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="type" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
        </xs:schema>
```

Les espaces de nom déclarés dans le schéma ci-dessus indiquent que les éléments et types de données, pour `xmlns:xs`, suivent l'URI `http://www.w3.org/2001/XMLSchema` et doivent dans le présent schéma être préfixés par `xs`. Le deuxième espace de nom indique l'espace de nom par défaut du présent schéma. Ensuite, dans ce fragment, l'élément nommé `ExternalReference` est déclaré et il est de type complexe.

Morceau d'un document XML conforme au schéma XML précédent et qui instancie l'élément `ExternalReference` deux fois :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Taxonomy
     xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns="http://www.bcXML.org/2002/bcXML"
xsi:schemaLocation="http://www.bcXML.org/2002/bcXML
http://www.bcXML.org/2002/XTD-20011210.xsd">
<!--(...)-->
<ExternalReference type="documentation">
  http://www.m-w.com/cgi-bin/dictionary?book=Dictionary&va=elevator
</ExternalReference>
<ExternalReference type="image">
  http://www.bcxml.org/images/Door.jpg
</ExternalReference>
</Taxonomy>
```

Dans le document XML ci-dessous le préfixe xsi correspond à l'espace de nom d'instances de schéma XML, ce qui permet ensuite de déclarer l'attribut schemaLocation. Ce dernier a deux valeurs : la première est l'espace de noms à utiliser dans le contexte de la taxonomie bcXML et la deuxième est l'emplacement du schéma XML à utiliser pour cette taxonomie.

Resource Description Framework (RDF)

La Figure 11 montre une instanciation, représentée dans le formalisme RDF et en forme de graphe, inspiré d'une partie de l'ontologie e-Cognos (Wetherill *et al.* 2002) (*cf.* section 4.2.2), et qui représente deux déclarations RDF entre un individu instance de la classe *Actor*, une instance de la classe *Project* et l'attribut nom de ce dernier. Une représentation déclarative en RDF peut se faire de plusieurs manières. La représentation du graphe de la Figure 11 en forme de triplets est la suivante :

```
<http://www.construction.fr/const.rdfs#Project_manager_1>
<const:IsAssociatedTo> <http://www.construction.fr/const.rdfs#OpenOne>
<http://www.construction.fr/const.rdfs#OpenOne> <const:name>
          "Open standard for construction interoperability in Europe"/>
```

Une représentation du même graphe au format RDF/XML est la suivante :

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:const="http://www.construction.fr/const.rdfs#">
  <const:Actor rdf:ID="Project_manager_1">
        <const:IsAssociatedTo>
          <const:Project rdf:ID="OpenOne">
<const:name xml:lang="en">Open standard for construction
interoperability in Europe</const:name>
          </const:Project>
       </const:IsAssociatedTo>
  </const:Actor>
</rdf:RDF>
```

Resource Description Framework – Schema (RDF-S)

Le schéma RDF suivant correspond à celui de l'exemple instancié ci-dessus :

```
<!DOCTYPE rdf:RDF [
      <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
      <!ENTITY const 'http://www.construction.fr/const.rdfs#'>
      <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema'>
```

```
]>
<rdf:RDF xmlns:rdf="&rdf;"
      xmlns:base ="&const;"
      xmlns:rdfs="&rdfs;#">
<rdfs:Class rdf:about="&const;Actor">
      <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdf:Property rdf:about="&const;IsAssociatedTo">
     <rdfs:domain rdf:resource="&const;Actor"/>
     <rdfs:range rdf:resource="&const;Project"/>
</rdf:Property>
<rdf:Property rdf:about="&const;name">
     <rdfs:domain rdf:resource="&const;Project"/>
     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Property>
<rdfs:Class rdf:about="&const;Project">
     <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
</rdf:RDF>
```

Le schéma RDF ainsi que l'instanciation correspondante ont été tous les deux validés par rapport à la spécification syntaxique par le service de validation RDF du W3C (http://www.w3.org/RDF/Validator/).

Annexe 2 – Comparaison d'un ensemble de ressources sémantiques européennes

Tableau 19 – Comparaison d'un ensemble de ressources sémantiques européennes, données issues du projet FUNSIEC (Barresi *et al.* 2004).

	Uniclass	**BS6100**	**e-Cognos**	**LexiCon**	**BcBuilding Definitions**	**BARBi**	**SDC**	**Edibatec**
SR Type	Classification system	Glossary	Ontology	Vocabulary of terms	Taxonomy	Data Library	Dictionary	Dictionary
General Issues								
Availability	Free browsing of the content or purchasable on floppies	Online access to standard is available to subscribers	Free	Partially free	GNU Public Licence	Free browsing of the content after free registration	Free	HTML format is freely available on-line
Location	http://thesaurus.foraec.com/data/uniclass or Riba Bookshop	http://bsonline.techindex.co.uk	http://195.83.41.67/eCOSer/Login.jsp	http://www.stabu-lexicon.com/	http://www.bcxml.org	http://www.barbi.no/index.jsp	http://www.sdc.biz	http://www.edibatec.org/Accueil/Default4.htm
Commissioned by	Construction Project Information Committee (CPIC)	British Standards Institution (BSI)	European Commission	STABU	European Commission	Norwegian building industry	Gencod EAN France, MédiaConstruct, Edibatec, and Edifrance	Edibatec
Developed by	National Building Specification Services	BSI	e-Cognos project partners	STABU Foundation Netherlands	IST e-Construct project	Norwegian Building Research Institute + Norwegian Council for Building Standardisation	Gencod EAN France, MédiaConstruct, Edibatec, and Edifrance	Edibatec Réalisation
Publishing year	1997	Various - depending on Parts	2003	1997	2002	January 2003 (latest version)	2002	2000
Application Domain								
Domain	Building & Construction	Building & Construction	Building & Construction	Building & Construction	Building & Construction	Building & Construction	Building & Construction	Electrical, sanitary and thermal products

Scope	Architectural & Civil Engineering	Building & Construction	Management of Building & Construction Knowledge	Buildings and utilities, and technical installation sectors	Building and construction	Building and construction	Building and construction	Supply chain management in Buildign & Construction
Application	Product Libraries	Terms definition	Ontology management	Project specifications, product catalogues	E-procurement	E- commerce	E-commerce	E-catalogues
Development and Architecture								
Supporting documentation	National Building Specification Services Ltd (NBS). Uniclass-unified classification for the construction industry. London. 1996	http://bsonline.techindex.co.uk	http://www.e-cognos.org/	http://www.stabu-lexicon.com/	http://www.bcxml.org	http://edmserver.epmtech.jotne.com/barbi/document.jsp?menid=3	http://www.sdc.biz	Edibatec Réalisation, http://www.edibatec.org/Consult/Default.htm
Building methodology	N/A	N/A	Information not available	Information not available	Information not available	Information not available	Information not available	Information not available
Underlying language	N/A	N/A	DAML+OIL	Proprietary	XML	Proprietary	HTML	HTML
Building tool	N/A	N/A	e-COSer Ontology Server	LexiCon Explorer	LexiCon Explorer	BARBi Library tool	Information not available	Information not available
Structure/Architecture	Faceted & Hierarchical	Faceted & Hierarchical	Hierarchical	Hierarchical (based on ISO 12006-3)	Hierarchical	Hierarchical (based on ISO 12006-3)	Hierarchical	Faceted & Hierarchical
Extensibility	Information not available	Information not available	Yes	Information not available	Yes	Information not available	Information not available	Information not available

Scalability	Information not available	Yes	Yes	Information not available	Yes	Information not available	Information not available	Contains information concerning 60 000 products
Interoperability								
Multilingual	No	No	Yes (French, English, Finish and German)	Yes (Dutch, English, French, German, Greek and Norwegian)	English, French, Dutch, German, Norwegian and Greeklish	Norwegian and English	No	No

Annexe 3 – Implantation Java

```
/*
 * Unification.java
*/

package unifirelation;

import java.net.URI;
import java.util.HashSet;
import java.util.Set;

import org.mindswap.pellet.owlapi.Reasoner;

import org.semanticweb.owl.apibinding.OWLManager;
import org.semanticweb.owl.io.DefaultOntologyFormat;
import org.semanticweb.owl.io.OWLFunctionalSyntaxOntologyFormat;
import org.semanticweb.owl.io.OWLXMLOntologyFormat;
import org.semanticweb.owl.model.AddAxiom;
import org.semanticweb.owl.model.OWLAxiom;
import org.semanticweb.owl.model.OWLDataFactory;
import org.semanticweb.owl.model.OWLEquivalentObjectPropertiesAxiom;
import org.semanticweb.owl.model.OWLException;
import org.semanticweb.owl.model.OWLObjectProperty;
import org.semanticweb.owl.model.OWLObjectPropertyExpression;
import org.semanticweb.owl.model.OWLOntology;
import org.semanticweb.owl.model.OWLOntologyCreationException;
import org.semanticweb.owl.model.OWLOntologyManager;

/**
 * @author Catarina
 */
public class Unification {
```

```
    /** Creates a new instance of Unification */
    public Unification () {
    }

    private static Set<OWLObjectProperty> processedProperties;

    public static void main(String[] args) throws OWLOntologyCreationException {
     try {

    OWLOntologyManager manager = OWLManager.createOWLOntologyManager();

    URI RS1URI = URI.create(args[0]);

/*Premier argument pour un teste est un fichier par HTTP à travers l'URL, par exemple
http://liris.cnrs.fr/catarina.ferreira.da.silva/test/eCognos-isInvolvedIn-SEP2.owl
ou un fichier en local
*/
    URI RS2URI = URI.create(args[1]);

/*Deuxième argument pour un teste est aussi un URL
http://liris.cnrs.fr/catarina.ferreira.da.silva/test/ifc-kernel-isActingUpon-v5.3.owl
ou un fichier en local
*/
    //Création des références pour les ressources sémantiques, en utilisant l'URI

    OWLOntology RS1 = manager.loadOntology( RS1URI );
    OWLOntology RS2 = manager.loadOntology( RS2URI );

    /*Une OWLDataFactory est nécessaire pour créer de nouveaux objets des ressources sémantiques
    Chaque ressources sémantiques a une référence vers le factory
    */
    OWLDataFactory factory = manager.getOWLDataFactory();

    //Utiliser le factory pour créer une référence de ces propriétés d'objets
```

```
    OWLObjectProperty argsp1 = factory.getOWLObjectProperty( URI.create(args[2]) );
    OWLObjectProperty argsp2 = factory.getOWLObjectProperty( URI.create(args[3]) );

    /*Exemple : args[2] est http://www.cstb.fr/eCognos#isInvolvedIn
    et args[3] est http://www.iai- international.org/Model/R2x3_final/index.htm#IfcActor_isActingUpon
    */
        for ( OWLObjectProperty p1: RS1.getReferencedObjectProperties()){
            for ( OWLObjectProperty p2: RS2.getReferencedObjectProperties()){

             if ((p1 == argsp1) & (p2 == argsp2))
               processedProperties = new HashSet <OWLObjectProperty> ();

  if (!processedProperties.contains(argsp1) & (!processedProperties.contains(argsp2))) {

              Set <OWLObjectPropertyExpression> exp1 = new HashSet <OWLObjectPropertyExpression> ();
              exp1.add(argsp1);
              exp1.add(argsp2);

        OWLEquivalentObjectPropertiesAxiom propEqui = factory.getOWLEquivalentObjectPropertiesAxiom(exp1);

        manager.applyChange(new AddAxiom(RS1, propEqui));
      /*manager.applyChange(new AddAxiom(RS2, propEqui)); ajouter l'axiome d'équivalence dans une seule
ressource est suffisant

      Sauvegarder les ressources sémantiques modifiées
     */
URI physicalURI1 = URI.create("file:/NetBeans/RS1.V2-DefaultOntologyFormat.owl");
URI physicalURI2 = URI.create("file:/NetBeans/RS2.V2-DefaultOntologyFormat.owl");

manager.saveOntology(RS1, new DefaultOntologyFormat(), physicalURI1);
manager.saveOntology(RS2, new DefaultOntologyFormat(), physicalURI2);

     processedProperties.add(argsp1);
     processedProperties.add(argsp2);
     }
```

```
    }
  //Enlever les ressources sémantiques du manager
  manager.removeOntology(RS1.getURI());
  manager.removeOntology(RS2.getURI());
  }
 }
 catch (OWLException e) {
          e.printStackTrace();
        }
    }
}
-------------------------------------------------------------------------------------------------------
----------
/*
 * Correspondances.java
 */

package unifirelation;

import java.net.URI;
import java.util.HashSet;
import java.util.Set;

import org.mindswap.pellet.owlapi.Reasoner;

import org.semanticweb.owl.apibinding.OWLManager;
import org.semanticweb.owl.model.OWLClass;
import org.semanticweb.owl.model.OWLDataFactory;
import org.semanticweb.owl.model.OWLOntology;
import org.semanticweb.owl.model.OWLOntologyCreationException;
import org.semanticweb.owl.model.OWLOntologyManager;
import org.semanticweb.owl.model.OWLException;
/**
 * @author Catarina
 */
```

```
public class Correspondances {

    /** Creates a new instance of Correspondances */
    public Correspondances() {
    }
    public static void main(String[] args)throws OWLOntologyCreationException {
     try {

        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();

        URI physicalURI1 = URI.create(args[0]);
        URI physicalURI2 = URI.create(args[1]);
        /*Les deux premiers arguments sont les ressources sémantiques en local ou l'URI en protocole HTTP
        Par exemple, en local "file:/C://NetBeans//RS1.V2-DefaultOntologyFormat.owl"
        "file:/C://NetBeans//RS2.V2-DefaultOntologyFormat.owl"
        */

        OWLOntology RS1 = manager.loadOntology( physicalURI1 );
        OWLOntology RS2 = manager.loadOntology( physicalURI2 );

  // Créer une instance du moteur d'inférences Pellet
       Reasoner reasoner = new Reasoner( manager );

        Set <OWLOntology> setOfOntologies = new HashSet <OWLOntology> ();
        setOfOntologies.add(RS1);
        setOfOntologies.add(RS2);

        // Charger les ressources sémantiques dans le moteur
        reasoner.loadOntologies( setOfOntologies );

         for ( OWLClass classe1: RS1.getReferencedClasses() ) {
            for ( OWLClass classe2: RS2.getReferencedClasses() ){
            if ( reasoner.isSubClassOf(classe1,classe2) ) {
                System.out.println( classe1.getURI().getFragment() + " is subsumed by " +
classe2.getURI().getFragment() );
```

```
            }
            /*else{
                System.out.println( classe1.getURI().getFragment() + " is not subsumed by " +
classe2.getURI().getFragment() );
            }
             */
            if ( reasoner.isSubClassOf(classe2,classe1) ) {
                System.out.println( classe2.getURI().getFragment() + " is subsumed by " +
classe1.getURI().getFragment() );
            }
            /*else{
                System.out.println( classe2.getURI().getFragment() + " is not subsumed by " +
classe1.getURI().getFragment() );
            }
             */
            if ( reasoner.isEquivalentClass(classe1,classe2)){
              System.out.println( classe1.getURI().getFragment() + " is equivalent to " +
classe2.getURI().getFragment() );
            }
           }
        }
        reasoner.clearOntologies();
        manager.removeOntology( RS1.getURI() );
        manager.removeOntology( RS2.getURI() );
      }
      catch (OWLException e) {
            e.printStackTrace();
     }
 }
}
-------------------------------------------------------------------------------------------------------
----------

/*
 * NewGetIndividualsInSetOfOntologies.java
```

```
 */

package unifirelation;

import java.net.URI;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

import org.mindswap.pellet.owlapi.Reasoner;

import org.semanticweb.owl.apibinding.OWLManager;
import org.semanticweb.owl.model.OWLClass;
import org.semanticweb.owl.model.OWLDataFactory;
import org.semanticweb.owl.model.OWLDescription;
import org.semanticweb.owl.model.OWLException;
import org.semanticweb.owl.model.OWLIndividual;
import org.semanticweb.owl.model.OWLOntology;
import org.semanticweb.owl.model.OWLOntologyCreationException;
import org.semanticweb.owl.model.OWLOntologyManager;

/**
 * @author Catarina
 */
public class NewGetIndividualsInSetOfOntologies {

    /** Creates a new instance of NewGetIndividualsInSetOfOntologies */
    public NewGetIndividualsInSetOfOntologies() {
    }

    public static void main(String[] args) throws OWLOntologyCreationException {
       try {

        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
```

```
   URI physicalURI1 = URI.create(args[0]);
   URI physicalURI2 = URI.create(args[1]);

   OWLOntology ontology1 = manager.loadOntology( physicalURI1 );
   OWLOntology ontology2 = manager.loadOntology( physicalURI2 );

   OWLDataFactory factory = manager.getOWLDataFactory();

   // Charger l'ontologie dans le moteur
Reasoner reasoner = new Reasoner( manager );
Set <OWLOntology> setOfOntologies = new HashSet <OWLOntology> ();
   setOfOntologies.add(ontology1);
   setOfOntologies.add(ontology2);

  reasoner.loadOntologies( setOfOntologies );

   Set<OWLClass> classes = reasoner.getClasses();

reasoner.realise();

reasoner.getKB().printClassTree();

{
       for(OWLClass cls : classes){
           Set <OWLIndividual> individuals = reasoner.getIndividuals(cls, true);
           // l'argument true assure que seulement les instances directes sont retournées

     for(Iterator i = individuals.iterator(); i.hasNext(); ) {
              OWLIndividual ind = (OWLIndividual) i.next();
                  System.out.println("Nom de l'instance : " + ind);

                  Set<Set<OWLClass>> types = reasoner.getTypes(ind, true);
                  // l'argument true assure que seulement les types directes sont retournées

                  if (!types.isEmpty()){
```

```
                            System.out.println("L'instance est inférée à partir des classes : ");
                             for (Set<OWLClass> type : types){
                                for(OWLClass ty : type)
                                System.out.println("Nom de la classe : " + ty);
                            }
                                  System.out.println();
                        }
                    }
                }
            }
            reasoner.clearOntologies();
            manager.removeOntology( ontology1.getURI() );
            manager.removeOntology( ontology2.getURI() );
            }
                catch (OWLException e) {
                e.printStackTrace();
      }
    }
}
```

Annexe 4 – Représentation OWL-DL des correspondances sémantiques entre les méta-schémas de bcBuildingDefinitions, d'e-Cognos, d'IFC-kernel et d'ISO 12006-3

Ce fichier s'intitulé "mapping-meta-schemas4SRs-V1.6.owl".

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:iso-12006-3="http://www.iso12006-3.com#"
   xmlns:ifc="http://www.iai-international.org/Model/R2x3_final/index.htm#"
   xmlns="http://www.owl-ontologies.com/mappings-4iniSRs.owl#"
   xmlns:bcXML-ms="http://www.bcXML-metaschema.xmi#"
   xmlns:eCognos-ms="http://www.cstb.fr/eCognos-ms#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
  xml:base="http://www.owl-ontologies.com/mappings-4iniSRs.owl">
  <owl:Ontology rdf:about="">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >This file represents mappings among four selected semantic resources of the building and
construction domain. These semantic resources are the bcXML metaschema, the eCognos metaschema,
the ifc-kernel and the ISO 12006-3. The following types of mapping are represented: equivalence,
subsumption and transitivity. No procedures, fonctions or rules that exist in some input semantic
resources were taken into account.</rdfs:comment>
  </owl:Ontology>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcActor">
   <rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcActor is subsumed by the
concept iso-12006-3:xtdObject</rdfs:comment>
   <rdfs:subClassOf>
    <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing">
     <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     ></rdfs:comment>
    </rdf:Description>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.iso12006-3.com#xtdRoot"/>
   </rdfs:subClassOf>
   <rdfs:comment xml:lang="en">Transitivity mapping: ifc:IfcActor is_subsumed_by (is subclassOf) iso-
12006-3:xtdRoot by transitivity</rdfs:comment>
   <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.iso12006-3.com#xtdObject"/>
   </rdfs:subClassOf>
   <rdfs:comment xml:lang="en">Transitivity mapping: ifc:IfcActor is_subsumed_by (is subclassOf)
eCognos-ms:Concept_Concept by transitivity</rdfs:comment>
   <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Concept_Concept"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Concept_Concept">
   <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.iso12006-3.com#xtdObject"/>
   </rdfs:subClassOf>
```

```
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">The subsumption input axiom (eCognos-ms:Concept_Concept
is_subsumed_by iso-12006-3:xtdObject) was defined by human experts of the application
domain.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcExternalReference">
    <rdfs:comment xml:lang="en">The concept ifc:IfcExternalReference is involved in a subsumption
mapping: the concept ifc:IfcExternalReference is subsumed by the concept iso-12006-
3:xtdLanguageRepresentation</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdReference"/>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">The equivalence (iso-12006-3:xtdReference =
ifc:IfcExternalReference) is an input axiom defined by human experts of the application
domain.</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdLanguageRepresentation"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#anyURI"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#LOGICAL">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#LOGICAL"/>
          <owl:Class rdf:about="http://www.iso12006-3.com#LOGICAL"/>
          <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#LOGICAL"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelActsUpon">
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRelActsUpon is_subsumed_by (is subclassOf)
eCognos-ms:Relation_Relation by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Context">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
```

```
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#BOOLEAN">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.iso12006-3.com#BOOLEAN"/>
          <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#BOOLEAN"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelCollects">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRelCollects is_subsumed_by (is subclassOf) eCognos-
ms:Relation_EcosRelation by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#ECognos_Object"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRelCollects is_subsumed_by (is subclassOf)
ifc:IfcRelationship by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_EcosRelation"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRelCollects is_subsumed_by (is subclassOf) eCognos-
ms:ECognos_Object by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelDecomposes">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelAssignsCollections">
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRelAssignsCollections is_subsumed_by (is subclassOf)
ifc:IfcRelationship by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#ExternalReference">
```

<rdfs:comment xml:lang="en">The equivalence (bcXML-ms:ExternalReference=iso-12006-3:xtdReference and bcXML-ms:ExternalReference=ifc:IfcExternalReference) input axioms were defined by human experts of the application domain.</rdfs:comment>
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="http://www.iso12006-3.com#xtdReference"/>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcExternalReference"/>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdLanguageRepresentation"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelAssigns">
<rdfs:comment xml:lang="en">Transitivity mapping: ifc:IfcRelAssigns is_subsumed_by (is subclassOf) eCognos-ms:Relation_EcosRelation by transitivity</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#ECognos_Object"/>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
</rdfs:subClassOf>
<rdfs:comment xml:lang="en">Transitivity mapping: ifc:IfcRelAssigns is_subsumed_by (is subclassOf) eCognos-ms:ECognos_Object by transitivity</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_EcosRelation"/>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="http://www.iso12006-3.com#REAL">
<owl:equivalentClass>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#REAL"/>
</owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcDocumentReference">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcDocumentReference is subsumed by the concept iso-12006-3:xtdReference</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdReference"/>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#ExternalReference"/>
</owl:Class>
<owl:Class rdf:about="http://www.bcXML-metaschema.xmi#NativeLanguage">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdLanguage"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Technical_Topic">
<rdfs:subClassOf>

```
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdObject"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">eCognos-ms:Technical_Topic is_subsumed_by (is subclassOf) iso-12006-3:xtdObject by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdInteger">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#INTEGER"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#INTEGER"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcGroup">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcGroup is subsumed by the concept iso-12006-3:xtdObject</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdObject"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Description">
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdDescription"/>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">The equivalent (bcXML-ms:Description =
      eCognos-ms:Object_ConceptDefinition and bcXML-ms:Description=iso-12006-3:xtdDescription) input axioms were defined by human experts of the application domain.</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdLanguageRepresentation"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelAssociates">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRelAssociatesis_subsumed_by (is subclassOf) ifc:IfcRelationship by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#LOGICAL">
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#LOGICAL"/>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdGlobalUniqueID">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdGlobalUniqueID is subsumed by the concept ifc:STRING</rdfs:comment>
```

```
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#STRING"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#BOOLEAN">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#BOOLEAN"/>
          <owl:Class rdf:about="http://www.iso12006-3.com#BOOLEAN"/>
          <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#BOOLEAN"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Resource">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdObject"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">eCognos-ms:Resource is_subsumed_by (is subclassOf) iso-12006-
3:xtdObject by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdObject">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    </rdfs:subClassOf>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcObject"/>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">The equivalence (iso-12006-3:xtdObject = ifc:ifcObject) is an input
axiom defined by human experts of the application domain.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Conjunction mapping: iso-12006-3:xtdObject is_subsumed_by (is
subclassOf) bcXML-ms:Object by conjunction</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Object"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">iso-12006-3:xtdObject is involved in a subsumption mapping. It is
subsumed by the concept ifc:IfcRoot</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdActivity">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdActivity is
subsumed by the concept ifc:IfcObject</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
```

```
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >iso-12006-3:xtdActivity is_subsumed_by (is subclassOf) ifc:IfcRoot by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcObject"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdLabel">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdLabel is subsumed
by the concept ifc:STRING</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#STRING"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdText">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdText is subsumed
by the concept ifc:STRING</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Product">
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">eCognos-ms:Product is_subsumed_by (is subclassOf) iso-12006-
3:xtdObject by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdDescription">
    <rdfs:comment xml:lang="en">This equivalent (eCognos-ms:Object_ConceptDefinition =
      iso-12006-3:xtdDescription) input axiom was defined by human experts of the application
domain.</rdfs:comment>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Object_ConceptDefinition"/>
    </owl:equivalentClass>
    <owl:equivalentClass rdf:resource="http://www.bcXML-metaschema.xmi#Description"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#NUMBER">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
```

```
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#NUMBER"/>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelSequences">
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRelSequences is_subsumed_by (is subclassOf)
ifc:IfcRelationship by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#INTEGER">
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#INTEGER"/>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelDefines">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcResource">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcResource is subsumed by the
concept iso-12006-3:xtdObject</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Signature">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdRoot"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-ms:Relation_Signature is_subsumed_by
(is subclassOf) ifc:IfcRoot by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-ms:Relation_Signature is_subsumed_by
(is subclassOf) iso-12006-3:xtdRoot by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelationship"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Relationship"/>
```

```
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcProxy">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:comment xml:lang="en">ifc:IfcProxy is_subsumed_by (is subclassOf) iso-12006-3:xtdObject by
transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdProperty">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdProperty is
subsumed by the concept ifc:IfcObject</rdfs:comment>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >iso-12006-3:xtdProperty is_subsumed_by (is subclassOf) ifc:IfcRoot by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcObject"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelAssignsMeasures">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRelAssignsMeasures is_subsumed_by (is subclassOf)
ifc:IfcRelationship by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdRoot">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#ECognos_Object"/>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRoot is_subsumed_by (is subclassOf) eCognos-
ms:ECognos_Object by conjunction. The combination of this mapping with a previous one causes these
both concepts to be presented as nec&amp;amp;suffic equivalent</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">iso-12006-3:xtdRoot is_subsumed_by (is subclassOf)
ifc:IfcRelationship by transitivity</rdfs:comment>
    <rdfs:comment xml:lang="en">conjunction mapping: iso-12006-3:xtdRoot is_subsumed_by
ifc:IfcRoot was obtained by conjunction of other classes of concepts</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#BOOLEAN">
    <owl:equivalentClass>
```

```
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#BOOLEAN"/>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#MinCardinality">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#NUMBER"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#NUMBER"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">bcXML-ms:MinCardinality is_subsumed_by (is subclassOf) eCognos-
ms:NUMBER by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#NUMBER"/>
    <rdfs:comment xml:lang="en">bcXML-ms:MinCardinality is_subsumed_by (is subclassOf) iso-12006-
3:NUMBER by transitivity</rdfs:comment>
    <rdfs:comment xml:lang="en">bcXML-ms:MinCardinality is_subsumed_by (is subclassOf)
ifc:NUMBER by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#INTEGER"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdActor">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcObject"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdActor is subsumed
by the concept ifc:IfcObject</rdfs:comment>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >iso-12006-3:xtdActor is_subsumed_by (is subclassOf) ifc:IfcRoot by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdVersionID">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdVersionID is
subsumed by the concept ifc:STRING</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Process">
    <rdfs:comment xml:lang="en">eCognos-ms:Process is_subsumed_by (is subclassOf) iso-12006-
3:xtdObject by transitivity</rdfs:comment>
```

```
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdLanguage">
    <rdfs:comment xml:lang="en">This equivalent (bcXML-ms:language = iso-12006-3:xtdLanguage)
input axiom was defined by human experts of the application domain.</rdfs:comment>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#language"/>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdNumber">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdNumber is
subsumed by the concept ifc:NUMBER</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#NUMBER"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#NUMBER"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#NUMBER"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#REAL">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#REAL"/>
          <owl:Class rdf:about="http://www.iso12006-3.com#REAL"/>
          <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#REAL"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdUnit">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcObject"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >iso-12006-3:xtdUnit is_subsumed_by (is subclassOf) ifc:IfcRoot by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#STRING">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#STRING"/>
          <owl:Class rdf:about="http://www.iso12006-3.com#STRING"/>
```

<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#STRING"/>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdRelAssignsProperties">
<rdfs:comment xml:lang="en">iso-12006-3:xtdRelAssignsProperties is_subsumed_by (is subclassOf) ifc:IfcRelationship by transitivity</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#LOGICAL">
<owl:equivalentClass rdf:resource="http://www.iso12006-3.com#LOGICAL"/>
</owl:Class>
<owl:Class rdf:about="http://www.bcXML-metaschema.xmi#NativeExplanation">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iso12006-3.com#STRING"/>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#STRING"/>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#STRING"/>
</owl:Class>
<owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Relation">
<owl:equivalentClass>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
</owl:equivalentClass>
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Relationship"/>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdRelationship"/>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:comment xml:lang="en">The equivalences (eCognos-ms:Relation_Relation = bcXML-ms:Relationship and
eCognos-ms:Relation_Relation = iso-12006-3:xtdRelationship and
eCognos-ms:Relation_Relation=ifc:IfcRelationship) are input axioms defined
by human experts of the application domain.</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdRoot"/>

```
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdReal">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#NUMBER"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#REAL"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#REAL"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdReal is subsumed
by the concept ifc:REAL</rdfs:comment>
    <rdfs:comment xml:lang="en">Transitivity mapping: iso-12006-3:xtdReal is_subsumed_by (is
subclassOf) bcXML-ms:NUMBER by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#REAL"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#INTEGER">
    <owl:equivalentClass rdf:resource="http://www.bcXML-metaschema.xmi#INTEGER"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Object">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">The subsumption input axiom (bcXML-ms:Object is_subsumed_by
ifc:IfcObject) was defined by human experts of the application domain.</rdfs:comment>
    <rdfs:comment xml:lang="en">The subsumption input axiom (bcXML-ms:Object is_subsumed_by
eCognos-ms:Concept_Concept) was defined by human experts of the application
domain.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#Concept_Concept"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcObject"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_EcosRelation">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdRoot"/>
    <rdfs:comment xml:lang="en">Conjunction mapping: eCognos-ms:Relation_EcosRelation
is_subsumed_by (is subclassOf) ifc:IfcRoot by conjunction</rdfs:comment>
    <rdfs:comment xml:lang="en">Conjunction mapping: eCognos-ms:Relation_EcosRelation
is_subsumed_by (is subclassOf) iso-12006-3:xtdRoot by conjunction</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdBoolean">
    <rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#BOOLEAN"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#BOOLEAN"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Subsumption mapping: the concept iso-12006-3:xtdBoolean is subsumed by the concept
ifc:BOOLEAN</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#BOOLEAN"/>
```

```
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#NUMBER">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#NUMBER"/>
          <owl:Class rdf:about="http://www.iso12006-3.com#NUMBER"/>
          <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#NUMBER"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Relationship">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelationship"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">The subsumption input axiom (bcXML-ms:Relationship
is_subsumed_by iso-12006-3:xtdRelationship) was defined by human experts of the application
domain.</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#ECognos_Object"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">The subsumption input axiom (bcXML-ms:Relationship
is_subsumed_by ifc:IfcRelationship) was defined by human experts of the application
domain.</rdfs:comment>
    <rdfs:comment xml:lang="en">The equivalence (eCognos-ms:Relation_Relation = bcXML-
ms:Relationship) is an input axiom defined by human experts of the application
domain.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#Relation_EcosRelation"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Explanation">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#STRING"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#STRING">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.iso12006-3.com#STRING"/>
          <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
        </owl:intersectionOf>
      </owl:Class>
```

```
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#RelatedDomain">
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">eCognos-ms:RelatedDomain is_subsumed_by (is subclassOf) iso-
12006-3:xtdObject by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcProduct">
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcProduct is subsumed by the
concept iso-12006-3:xtdObject</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdDate">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdDate is subsumed
by the concept ifc:STRING</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#STRING"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#STRING"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot">
    <owl:equivalentClass rdf:resource="http://www.iso12006-3.com#xtdRoot"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">ifc:IfcRoot is_subsumed_by (is subclassOf) 12006-3:xtdRoot by
conjunction. As a previous mapping states 12006-3:xtdRoot is_subsumed_by ifc:IfcRoot, the current
mapping implies ifc:IfcRoot is necessary and sufficiently defined by 12006-3:xtdRoot.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdLogical">
    <rdfs:subClassOf rdf:resource="http://www.iai-
international.org/Model/R2x3_final/index.htm#LOGICAL"/>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#LOGICAL"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#LOGICAL"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Subsumption mapping: the concept iso-12006-3:xtdLogical is subsumed by the concept
ifc:LOGICAL</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Unit">
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#STRING"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Relation_Attribute">
```

```
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">Transitivity conjunction: eCognos-ms:Relation_Attribute
is_subsumed_by (is subclassOf) ifc:IfcRoot by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRoot"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#xtdRelationship"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#Relationship"/>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdRoot"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Transitivity conjunction: eCognos-ms:Relation_Attribute
is_subsumed_by (is subclassOf) iso-12006-3:xtdRoot by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#NUMBER">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.iso12006-3.com#NUMBER"/>
          <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#NUMBER"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelAssignsToControl">
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Transitivity mapping: ifc:IfcRelAssignsToControl is_subsumed_by (is
subclassOf) eCognos-ms:Relation_Relation by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelationship">
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#Relation_EcosRelation"/>
    <owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#ECognos_Object"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">The equivalence (ifc:IfcRelationship=eCognos-ms:Relation_Relation)
is an
    input axiom defined by human experts of the application domain.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdLanguageRepresentation">
    <rdfs:comment xml:lang="en">This concept is involved in a subsumption mapping: it is the subsumer
of the concept ifc:IfcExternalReference</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iso12006-3.com#xtdSubject">
    <rdfs:comment xml:lang="en">The subsumption input axiom (iso-12006-3:xtdSubject
is_subsumed_by bcXML-ms:Object) was defined by human experts of the application
domain.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
```

<rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdSubject is subsumed by the concept ifc:IfcObject.</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#Object"/>
<rdfs:subClassOf rdf:resource="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcObject"/>
</rdfs:subClassOf>
<rdfs:comment xml:lang="en">iso-12006-3:xtdSubject is_subsumed_by (is subclassOf) ifc:IfcRoot by transitivity</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdMeasureWithUnit">
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcObject"/>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>iso-12006-3:xtdMeasureWithUnit is_subsumed_by (is subclassOf) ifc:IfcRoot by transitivity</rdfs:comment>
<rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdMeasureWithUnit is subsumed by the concept ifc:IfcObject</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="http://www.bcXML-metaschema.xmi#LOGICAL">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="http://www.iso12006-3.com#LOGICAL"/>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#LOGICAL"/>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Concept_EcosConcept">
<rdfs:comment xml:lang="en">eCognos-ms:Concept_EcosConcept is_subsumed_by (is subclassOf) iso-12006-3:xtdObject by conjunction</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcClassificationReference">
<rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#ExternalReference"/>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdReference"/>
</rdfs:subClassOf>
<rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcClassificationReference is subsumed by the concept iso-12006-3:xtdReference</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">Transitivity mapping: ifc:IfcClassificationReference is_subsumed_by bcXML-ms:anyURI by transitivity</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.bcXML-metaschema.xmi#anyURI"/>

```
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">Transitivity mapping: ifc:IfcClassificationReference is_subsumed_by
iso-12006-3:xtdLanguageRepresentation by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdLanguageRepresentation"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcObject">
    <owl:equivalentClass rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcObject is subsumed by the
concept iso-12006-3:xtdRoot</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">The equivalence (iso-12006-3:xtdObject = ifc:IfcObject) is an input
axiom defined by human experts of the application domain.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#Concept_Concept"/>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdRoot"/>
    <rdfs:comment xml:lang="en">The subsumption input axiom (ifc:IfcObject is_subsumed_by
eCognos-ms:Concept_Concept) was defined by human experts of the application
domain.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Project">
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">eCognos-ms:Project is_subsumed_by (is subclassOf) iso-12006-
3:xtdObject by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#NUMBER">
    <owl:equivalentClass rdf:resource="http://www.iso12006-3.com#NUMBER"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Name">
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#STRING"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.iso12006-3.com#STRING"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Object_ConceptDefinition">
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdLanguageRepresentation"/>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#Description"/>
          <owl:Class rdf:about="http://www.iso12006-3.com#xtdDescription"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:equivalentClass rdf:resource="http://www.iso12006-3.com#xtdDescription"/>
    <rdfs:comment xml:lang="en">The equivalent (bcXML-ms:Description = eCognos-
ms:Object_ConceptDefinition
    and eCognos-ms:Object_ConceptDefinition = iso-12006-3:xtdDescription)
    input axioms are defined by human experts of the application domain.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#ECognos_Object">
```

<rdfs:comment xml:lang="en">Conjunction mapping: eCognos-ms:ECognos_Object is_subsumed_by (is subclassOf) iso-12006-3:xtdObject by conjunction</rdfs:comment>
<rdfs:comment xml:lang="en">Conjunction mapping: eCognos-ms:ECognos_Object is_subsumed_by (is subclassOf) ifc:IfcRoot by conjunction</rdfs:comment>
<owl:equivalentClass rdf:resource="http://www.iso12006-3.com#xtdRoot"/>
<rdfs:comment xml:lang="en">Conjunction mapping: eCognos-ms:ECognos_Object is_subsumed_by (is subclassOf) iso-12006-3:xtdRoot by conjunction</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf rdf:resource="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot"/>
<rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
</owl:Class>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdReference">
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.bcXML-metaschema.xmi#anyURI"/>
</rdfs:subClassOf>
<rdfs:comment xml:lang="en">The equivalence (iso-12006-3:xtdReference = ifc:IfcExternalReference and iso-12006-3:xtdReference=bcXML-ms:ExternalReference) input axioms were defined by human experts of the application domain.</rdfs:comment>
<owl:equivalentClass rdf:resource="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcExternalReference"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdString">
<rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#STRING"/>
<rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#STRING"/>
<rdfs:comment xml:lang="en">Subsumption mapping: the concept iso-12006-3:xtdString is subsumed by the concept ifc:STRING</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#STRING"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcControl">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Subsumption mapping: the concept ifc:IfcControl is subsumed by the concept iso-12006-3:xtdObject</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#REAL">
<owl:equivalentClass rdf:resource="http://www.iso12006-3.com#REAL"/>
</owl:Class>
<owl:Class rdf:about="http://www.bcXML-metaschema.xmi#language">
<rdfs:comment xml:lang="en">This equivalent (bcXML-ms:language = iso-12006-3:xtdLanguage) input axiom was defined by human experts of the application domain.</rdfs:comment>
<owl:equivalentClass rdf:resource="http://www.iso12006-3.com#xtdLanguage"/>
</owl:Class>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdRelationship">
<rdfs:comment xml:lang="en">Input axiom: the concept iso-12006-3:xtdRelationship is subsumed by the concept ifc:IfcRelationship.</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#Relation_EcosRelation"/>
<rdfs:comment xml:lang="en">iso-12006-3:xtdRelationship is_subsumed_by (is subclassOf) eCognos-ms:Relation_Relation by conjunction. A previous mapping may state eCognos-ms:Relation_Relation

is_subsumed_by iso-12006-3:xtdRelationship and this implies both definitions are nec&suffic equivalent</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#ECognos_Object"/>
<rdfs:comment xml:lang="en">The equivalence (eCognos-ms:Relation_Relation = iso-12006-3:xtdRelationship) is an input axiom defined by human experts of the application domain.</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf rdf:resource="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelationship"/>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#STRING">
<owl:equivalentClass>
<owl:Class rdf:about="http://www.iso12006-3.com#STRING"/>
</owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcProcess">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcProcess is subsumed by the concept iso-12006-3:xtdObject</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
</owl:Class>
<owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcLibraryReference">
<rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#ExternalReference"/>
<rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdReference"/>
<rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcLibraryReference is subsumed by the concept iso-12006-3:xtdReference</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="http://www.bcXML-metaschema.xmi#MaxCardinality">
<rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#INTEGER"/>
<rdfs:comment xml:lang="en">bcXML-ms:MaxCardinality is_subsumed_by (is subclassOf) iso-12006-3:NUMBER by transitivity</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#NUMBER"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">bcXML-ms:MaxCardinality is_subsumed_by (is subclassOf) ifc:NUMBER by transitivity</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.iai-international.org/Model/R2x3_final/index.htm#NUMBER"/>
</owl:Class>
<owl:Class rdf:about="http://www.iso12006-3.com#xtdEnumeration">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">transitivity mapping: iso-12006-3:xtdEnumeration (is subclassOf) is_subsumed_by ifc:STRING by transitivity</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#STRING"/>
<rdfs:comment xml:lang="en">transitivity mapping: iso-12006-3:xtdEnumeration is_subsumed_by (is subclassOf) bcXML-ms:STRING by transitivity</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.bcXML-metaschema.xmi#STRING"/>
<rdfs:subClassOf rdf:resource="http://www.iai-international.org/Model/R2x3_final/index.htm#STRING"/>
<rdfs:comment xml:lang="en">transitivity mapping: iso-12006-3:xtdEnumeration is_subsumed_by (is subclassOf) eCognos-ms:STRING by transitivity</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.iso12006-3.com#STRING">
<owl:equivalentClass rdf:resource="http://www.iai-international.org/Model/R2x3_final/index.htm#STRING"/>

```
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#IfcProject">
    <rdfs:comment xml:lang="en">Subsumption mapping: the concept ifc:IfcProject is subsumed by the
concept iso-12006-3:xtdObject</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#BOOLEAN">
    <owl:equivalentClass rdf:resource="http://www.iso12006-3.com#BOOLEAN"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.iai-
international.org/Model/R2x3_final/index.htm#IfcRelConnects">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#Relation_Relation"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#anyURI">
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdLanguageRepresentation"/>
    <rdfs:comment xml:lang="en">Conjunction mapping: bcXML-ms:anyURI is_subsumed_by (is
subclassOf) iso-12006-3:xtdLanguageRepresentation by conjunction</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#NativeName">
    <rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognos-ms#STRING"/>
    <rdfs:subClassOf rdf:resource="http://www.iai-
international.org/Model/R2x3_final/index.htm#STRING"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#STRING"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognos-ms#Actor">
    <rdfs:subClassOf rdf:resource="http://www.iso12006-3.com#xtdObject"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">eCognos-ms:Actor is_subsumed_by (is subclassOf) iso-12006-
3:xtdObject by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcXML-metaschema.xmi#REAL">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.iso12006-3.com#REAL"/>
          <owl:Class rdf:about="http://www.iai-international.org/Model/R2x3_final/index.htm#REAL"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
</rdf:RDF>
```

Annexe 5 – Représentation OWL-DL des correspondances sémantiques entre le schéma de bcBuildingDefinitions et celui d'eCognos

Ce fichier s'intitulé "mappings-bcBuildingSpecifications-20020129-eCognosOnto-V1.1.owl".

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:eCognos-onto="http://www.cstb.fr/eCognosOntology.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://www.owl-ontologies.com/mappings-bcBuildingSpecifications-20020129-
eCognosOnto.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:bcXml-taxo="http://www.bcBuildingSpecifications-20020129.xsd#"
    xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
  xml:base="http://www.owl-ontologies.com/mappings-bcBuildingSpecifications-20020129-
eCognosOnto.owl">
  <owl:Ontology rdf:about="">
    <rdfs:comment xml:lang="en">This file contains the mappings between the bcBuildingSpecifications-
20020129 taxonomy and the eCognos ontology. The following types of mapping are represented:
equivalence, subsumption and transitivity.</rdfs:comment>
  </owl:Ontology>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#gas">
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:gas is_subclassOf bcXml-
taxo:GasType</rdfs:comment>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:gas is_subclassOf bcXml-
taxo:ItemSet</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#GasType"/>
    </rdfs:subClassOf>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Gas"/>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Volume"/>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#appliance_governor">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:appliance_governor
is_subsumed_by (is subclassOf) bcXml-taxo:Gas by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Gas"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#wall">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#WallType"/>
    </rdfs:subClassOf>
    <owl:equivalentClass>
```

```
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Wall"/>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:wall is_subclassOf bcXml-
taxo:WallType</rdfs:comment>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:wall is_subclassOf bcXml-
taxo:ItemSet</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#FoundationType"/>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#DoorSet">
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#doorset"/>
    </owl:equivalentClass>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#frames_and_linings"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">Subsumption mapping: bcXml-taxo:DoorSet is_subclassOf eCognos-
onto:frames_and_linings</rdfs:comment>
    <rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:DoorSet is equivalent to eCognos-
onto:doorset</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Fastener">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Other_parts"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#fastener"/>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">Subsumption mapping: bcXml-taxo:Fastener is_subclassOf eCognos-
onto:Other_parts</rdfs:comment>
    <rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Fastener is equivalent to eCognos-
onto:fastener</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#GlazingType"/>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#BlanketType"/>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#ballast">
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:ballast is_subsumed_by (is
subclassOf) bcXml-taxo:ItemSet by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#blanket">
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Blanket"/>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#BlanketType"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:blanket is_subclassOf bcXml-
taxo:BlanketType</rdfs:comment>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:blanket is_subclassOf bcXml-
taxo:ItemSet</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#appliances">
    <rdfs:subClassOf>
```

```
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Gas"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:appliances is_subclassOf
bcXml-taxo:Gas</rdfs:comment>
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:appliances is_subsumed_by (is
subclassOf) bcXml-taxo:GasType by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#GasType"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#HeadType"/>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#BuildingType"/>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#fastener">
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:fastener is_subclassOf bcXml-
taxo:ItemSet</rdfs:comment>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:fastener is_subclassOf bcXml-
taxo:FastenerType</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#FastenerType"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:equivalentClass rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#Fastener"/>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#foundation">
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:foundation is_subclassOf
bcXml-taxo:FoundationType</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-
20020129.xsd#FoundationType"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Foundation"/>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:foundation is_subclassOf
bcXml-taxo:ItemSet</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Highrise_building">
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-
20020129.xsd#BuildingType"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:Highrise_building is_subclassOf
bcXml-taxo:Building</rdfs:comment>
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:Highrise_building
is_subsumed_by (is subclassOf) bcXml-taxo:BuildingType by transitivity</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Building"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#window">
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Window"/>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:window is_subclassOf bcXml-
taxo:WindowType</rdfs:comment>
```

<rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:window is_subclassOf bcXml-taxo:ItemSet</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#WindowType"/>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#head">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#HeadType"/>
<owl:equivalentClass>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Head"/>
</owl:equivalentClass>
<rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:head is_subclassOf bcXml-taxo:ItemSet</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
<rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:head is_subclassOf bcXml-taxo:HeadType</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#energy_sources_and_distribution"/>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#stair">
<owl:equivalentClass>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Stair"/>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#StairType"/>
</rdfs:subClassOf>
<rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:stair is_subclassOf bcXml-taxo:StairType</rdfs:comment>
<rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:stair is_subclassOf bcXml-taxo:ItemSet</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Gas">
<rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognosOntology.owl#energy_sources_and_distribution"/>
<rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Gas is equivalent to eCognos-onto:gas</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognosOntology.owl#gas"/>
<rdfs:comment xml:lang="en">Subsumption mapping: bcXml-taxo:Gas is_subclassOf eCognos-onto:energy_sources_and_distribution</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Window">
<rdfs:comment xml:lang="en">Subsumption mapping: bcXml-taxo:Window is_subclassOf eCognos-onto:Door_and_window</rdfs:comment>
<owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognosOntology.owl#window"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Window is equivalent to eCognos-onto:window</rdfs:comment>
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Door_and_window"/>
</rdfs:subClassOf>
</owl:Class>

```
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#building">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:building is_subclassOf bcXml-
taxo:ItemSet</rdfs:comment>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Building"/>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:building is_subclassOf bcXml-
taxo:BuildingType</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-
20020129.xsd#BuildingType"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#doorset">
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:doorset is_subclassOf bcXml-
taxo:ItemSet</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    <owl:equivalentClass rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#DoorSet"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:doorset is_subclassOf bcXml-
taxo:DoorSetType</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#DoorSetType"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Stair">
    <rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Stair is equivalent to eCognos-
onto:stair</rdfs:comment>
    <owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognosOntology.owl#stair"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Building">
    <rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Building is equivalent to eCognos-
onto:building</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognosOntology.owl#building"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#MaterialType"/>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Manufactured_material">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:Manufactured_material
is_subsumed_by (is subclassOf) bcXml-taxo:MaterialType by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-
20020129.xsd#MaterialType"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Material"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:Manufactured_material
is_subclassOf bcXml-taxo:Material</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:Manufactured_material
is_subsumed_by (is subclassOf) bcXml-taxo:ItemSet by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Wall">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Wall_and_Cladding"/>
    </rdfs:subClassOf>
```

<rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Wall is equivalent to eCognos-onto:wall</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognosOntology.owl#wall"/>
<rdfs:comment xml:lang="en">Subsumption mapping: bcXml-taxo:Wall is_subclassOf eCognos-onto:Wall_and_Cladding</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Blanket">
<rdfs:subClassOf rdf:resource="http://www.cstb.fr/eCognosOntology.owl#ballast"/>
<owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognosOntology.owl#blanket"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">Subsumption mapping: bcXml-taxo:Blanket is_subclassOf eCognos-onto:ballast</rdfs:comment>
<rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Blanket is equivalent to eCognos-onto:blanket</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Foundation">
<owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognosOntology.owl#foundation"/>
<rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Foundation is equivalent to eCognos-onto:foundation</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Hybrid_material">
<rdfs:subClassOf>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Material"/>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:Hybrid_material is_subclassOf bcXml-taxo:Material</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Stair_element">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:Stair_element is_subsumed_by (is subclassOf) bcXml-taxo:StairType by transitivity</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#Stair"/>
<rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#StairType"/>
<rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:Stair_element is_subclassOf bcXml-taxo:Stair</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Stair_feature_and_finish">
<rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:Stair_feature_and_finish is_subclassOf bcXml-taxo:Stair</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#Stair"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Material">
<rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Material is equivalent to eCognos-onto:material</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<owl:equivalentClass>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#material"/>
</owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Glazing">
<owl:equivalentClass>
<owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#glazing"/>
</owl:equivalentClass>

```
    <rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Glazing is equivalent to eCognos-
onto:glazing</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#material">
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-
20020129.xsd#MaterialType"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:material is_subclassOf bcXml-
taxo:ItemSet</rdfs:comment>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:material is_subclassOf bcXml-
taxo:MaterialType</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    <owl:equivalentClass rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#Material"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Highrise_mixed_use_building">
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:Highrise_mixed_use_building
is_subsumed_by (is subclassOf) bcXml-taxo:Building by transitivity</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#Building"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#Processed_material">
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#Material"/>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment xml:lang="en">Transitivity mapping: eCognos-onto:Processed_material
is_subsumed_by (is subclassOf) bcXml-taxo:Material by transitivity</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.bcBuildingSpecifications-20020129.xsd#Head">
    <owl:equivalentClass rdf:resource="http://www.cstb.fr/eCognosOntology.owl#head"/>
    <rdfs:comment xml:lang="en">Equivalent mapping: bcXml-taxo:Head is equivalent to eCognos-
onto:head</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#glazing">
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:glazing is_subclassOf bcXml-
taxo:ItemSet</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#ItemSet"/>
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#GlazingType"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:glazing is_subclassOf bcXml-
taxo:GlazingType</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:equivalentClass rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#Glazing"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.cstb.fr/eCognosOntology.owl#board_foot">
    <rdfs:subClassOf rdf:resource="http://www.bcBuildingSpecifications-20020129.xsd#Volume"/>
    <rdfs:comment xml:lang="en">Subsumption mapping: eCognos-onto:board_foot is_subclassOf
bcXml-taxo:Volume</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
</rdf:RDF>
```

Annexe 6 – Sous-partie d'e-Cognos après l'unification de propriétés

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY eCognos "http://www.cstb.fr/eCognos#" >
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY owl11 "http://www.w3.org/2006/12/owl11#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl11xml "http://www.w3.org/2006/12/owl11-xml#" >
   <!ENTITY daml "http://www.daml.org/2001/03/daml+oil#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY index "http://www.iai-international.org/Model/R2x3_final/index.htm#" >
   <!ENTITY eCognos-isInvolvedIn-SEP2 "http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/eCognos-
isInvolvedIn-SEP2.owl#" >
]>

<rdf:RDF xmlns="http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/eCognos-isInvolvedIn-SEP2.owl#"
    xml:base="http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/eCognos-isInvolvedIn-SEP2.owl"
    xmlns:eCognos="http://www.cstb.fr/eCognos#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:eCognos-isInvolvedIn-SEP2="http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/eCognos-
isInvolvedIn-SEP2.owl#"
    xmlns:index="http://www.iai-international.org/Model/R2x3_final/index.htm#"
    xmlns:owl11="http://www.w3.org/2006/12/owl11#"
    xmlns:owl11xml="http://www.w3.org/2006/12/owl11-xml#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#">
   <owl:Ontology rdf:about="">
      <rdfs:comment xml:lang="fr"
         >eCognos fragment Actor Project.</rdfs:comment>
   </owl:Ontology>

   <!--

////////////////////////////////////////////////////////////////////////////////
/////
   //
   // Object Properties
   //

////////////////////////////////////////////////////////////////////////////////
/////
    -->

   <!-- http://www.cstb.fr/eCognos#eCognosRelations -->

   <owl:ObjectProperty rdf:about="&eCognos;eCognosRelations"/>

   <!-- http://www.cstb.fr/eCognos#isInvolvedIn -->
```

```
    <owl:ObjectProperty rdf:about="&eCognos;isInvolvedIn">
        <owl:equivalentProperty rdf:resource="&index;IfcActor_isActingUpon"/>
        <rdfs:range rdf:resource="&eCognos;Project"/>
        <rdfs:domain rdf:resource="&eCognos;Actor"/>
        <rdfs:subPropertyOf rdf:resource="&eCognos;eCognosRelations"/>
    </owl:ObjectProperty>

    <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcActor_isActingUpon -->

    <owl:ObjectProperty rdf:about="&index;IfcActor_isActingUpon"/>

    <!--

///////////////////////////////////////////////////////////////////////////////
/////
    //
    // Classes
    //

///////////////////////////////////////////////////////////////////////////////
/////
     -->

    <!-- http://www.cstb.fr/eCognos#Actor -->

    <owl:Class rdf:about="&eCognos;Actor">
        <rdfs:subClassOf rdf:resource="&eCognos;eCognosConcepts"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&eCognos;isInvolvedIn"/>
                <owl:someValuesFrom rdf:resource="&eCognos;Project"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&eCognos;isInvolvedIn"/>
                <owl:allValuesFrom rdf:resource="&eCognos;Project"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <owl:disjointWith rdf:resource="&eCognos;Project"/>
    </owl:Class>

    <!-- http://www.cstb.fr/eCognos#Project -->

    <owl:Class rdf:about="&eCognos;Project">
        <rdfs:subClassOf rdf:resource="&eCognos;eCognosConcepts"/>
    </owl:Class>

    <!-- http://www.cstb.fr/eCognos#eCognosConcepts -->

    <owl:Class rdf:about="&eCognos;eCognosConcepts"/>

    <!--

///////////////////////////////////////////////////////////////////////////////
/////
```

```
    //
    // Individuals
    //

///////////////////////////////////////////////////////////////////////////////////
/////
     -->

    <!-- http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/eCognos-isInvolvedIn-SEP2.owl#InterOpen -->

    <eCognos:Project rdf:about="#InterOpen"/>

    <!-- http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/eCognos-isInvolvedIn-SEP2.owl#Manager1 -->

    <eCognos:Actor rdf:about="#Manager1">
        <eCognos:isInvolvedIn rdf:resource="#InterOpen"/>
    </eCognos:Actor>

    <!-- http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/eCognos-isInvolvedIn-SEP2.owl#Manager2 -->

    <eCognos:Actor rdf:about="#Manager2">
        <eCognos:isInvolvedIn rdf:resource="#OpenOne"/>
    </eCognos:Actor>

    <!-- http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/eCognos-isInvolvedIn-SEP2.owl#OpenOne -->

    <eCognos:Project rdf:about="#OpenOne"/>
</rdf:RDF>
```

Annexe 7 – Sous-parties d'IFC-kernel après l'unification de propriétés

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY eCognos "http://www.cstb.fr/eCognos#" >
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY owl11 "http://www.w3.org/2006/12/owl11#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl11xml "http://www.w3.org/2006/12/owl11-xml#" >
   <!ENTITY daml "http://www.daml.org/2001/03/daml+oil#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY ifc-kernel "http://www.iai-international.org/Model/R2x3_final/index.htm#" >
   <!ENTITY ifc-kernel-isActingUpon-v5 "http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/ifc-kernel-
isActingUpon-v5.3.owl#" >
]>

<rdf:RDF xmlns="http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/ifc-kernel-isActingUpon-v5.3.owl#"
    xml:base="http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/ifc-kernel-isActingUpon-v5.3.owl"
    xmlns:eCognos="http://www.cstb.fr/eCognos#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl11="http://www.w3.org/2006/12/owl11#"
    xmlns:owl11xml="http://www.w3.org/2006/12/owl11-xml#"
    xmlns:ifc-kernel-isActingUpon-v5="http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/ifc-kernel-
isActingUpon-v5.3.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:ifc-kernel="http://www.iai-international.org/Model/R2x3_final/index.htm#">
   <owl:Ontology rdf:about="">
      <rdfs:comment xml:lang="fr"
         >IFC-Kernel fragment IsActingUpon.</rdfs:comment>
   </owl:Ontology>

   <!--

///////////////////////////////////////////////////////////////////////////////
/////
   //
   // Object Properties
   //

///////////////////////////////////////////////////////////////////////////////
/////
    -->

   <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcActor_isActingUpon -->

   <owl:ObjectProperty rdf:about="&ifc-kernel;IfcActor_isActingUpon">
      <rdfs:range rdf:resource="&ifc-kernel;IfcRelAssignsToActor"/>
      <rdfs:domain rdf:resource="&ifc-kernel;IfcActor"/>
   </owl:ObjectProperty>
```

```
    <!--

///////////////////////////////////////////////////////////////////////////////
/////
    //
    // Classes
    //

///////////////////////////////////////////////////////////////////////////////
/////
     -->

     <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcActor -->

    <owl:Class rdf:about="&ifc-kernel;IfcActor">
       <rdfs:subClassOf>
          <owl:Restriction>
             <owl:onProperty rdf:resource="&ifc-kernel;IfcActor_isActingUpon"/>
             <owl:someValuesFrom rdf:resource="&ifc-kernel;IfcRelAssignsToActor"/>
          </owl:Restriction>
       </rdfs:subClassOf>
       <rdfs:subClassOf rdf:resource="&ifc-kernel;IfcObject"/>
    </owl:Class>

    <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcObject -->

    <owl:Class rdf:about="&ifc-kernel;IfcObject">
       <rdfs:subClassOf rdf:resource="&ifc-kernel;IfcObjectDefinition"/>
    </owl:Class>

    <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcObjectDefinition -->

    <owl:Class rdf:about="&ifc-kernel;IfcObjectDefinition">
       <rdfs:subClassOf rdf:resource="&ifc-kernel;IfcRoot"/>
       <owl:disjointWith rdf:resource="&ifc-kernel;IfcRelationship"/>
    </owl:Class>

    <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelAssigns -->

    <owl:Class rdf:about="&ifc-kernel;IfcRelAssigns">
       <rdfs:subClassOf rdf:resource="&ifc-kernel;IfcRelationship"/>
    </owl:Class>

    <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelAssignsToActor -->

    <owl:Class rdf:about="&ifc-kernel;IfcRelAssignsToActor">
       <rdfs:subClassOf rdf:resource="&ifc-kernel;IfcRelAssigns"/>
    </owl:Class>

    <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRelationship -->

    <owl:Class rdf:about="&ifc-kernel;IfcRelationship">
       <rdfs:subClassOf rdf:resource="&ifc-kernel;IfcRoot"/>
    </owl:Class>
```

```
    <!-- http://www.iai-international.org/Model/R2x3_final/index.htm#IfcRoot -->

    <owl:Class rdf:about="&ifc-kernel;IfcRoot"/>

    <!--

///////////////////////////////////////////////////////////////////////////////
/////
    //
    // Individuals
    //

///////////////////////////////////////////////////////////////////////////////
/////
     -->

    <!-- http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/ifc-kernel-isActingUpon-
v5.3.owl#constructionProject -->

    <ifc-kernel:IfcRelAssignsToActor rdf:about="#constructionProject"/>

    <!-- http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/ifc-kernel-isActingUpon-v5.3.owl#engineer1 -->

    <ifc-kernel:IfcActor rdf:about="#engineer1">
        <ifc-kernel:IfcActor_isActingUpon rdf:resource="#constructionProject"/>
    </ifc-kernel:IfcActor>

    <!-- http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/ifc-kernel-isActingUpon-v5.3.owl#engineer2 -->

    <ifc-kernel:IfcActor rdf:about="#engineer2">
        <ifc-kernel:IfcActor_isActingUpon rdf:resource="#interoperabiltyProject"/>
    </ifc-kernel:IfcActor>

    <!-- http://liris.cnrs.fr/catarina.ferreira.da.silva/SRs/ifc-kernel-isActingUpon-
v5.3.owl#interoperabiltyProject -->

    <ifc-kernel:IfcRelAssignsToActor rdf:about="#interoperabiltyProject"/>
</rdf:RDF>
```

RESUME en français

Dans le travail coopératif les experts sont amenés à utiliser des ressources sémantiques. Ces ressources sont en général représentées à l'aide de concepts et de relations et peuvent être modélisées sous forme d'ontologies. Chaque expert dans un domaine d'application utilise fréquemment une ou plusieurs ressources sémantiques et peut également avoir accès à d'autres ressources sémantiques mises à disposition par ses partenaires. Ces ressources sont hétérogènes entre elles à différents niveaux ce qui rend difficile l'interopérabilité syntaxique, structurelle et sémantique. Ce travail de thèse met l'accent sur l'interopérabilité sémantique des ressources sémantiques.
Nous proposons une méthodologie semi-automatique pour aider à la découverte de correspondances sémantiques entre concepts de ressources sémantiques différentes. Pour réaliser ce travail, nous transformons ces ressources sémantiques en OWL-DL, afin d'utiliser les services de moteurs d'inférences fondés sur les logiques de description. Nous proposons aussi un procédé pour créer des équivalences entre relations de ressources sémantiques différentes, appelé unification de relations, et qui vise à découvrir d'autres correspondances sémantiques. Les résultats de ce travail ont été testés à l'aide d'un prototype. L'application a été réalisée avec des ressources sémantiques issues du secteur européen de la construction. Pour aider à l'évaluation des correspondances sémantiques, nous proposons aussi une modélisation des connaissances du domaine à l'aide de la logique floue.

TITRE en anglais : Discovery of semantic mappings between semantic resources in a cooperative environment

RESUME en anglais

In collaborative work experts are encouraged to use semantic resources. These resources are generally represented with the aid of concepts and relationships and can be modelled in the form of ontologies. Each expert in a field of application often uses one or more semantic resources, and may also have access to other semantic resources made available by her/his partners. However, these resources are heterogeneous at different levels making difficult syntactic, structural and semantic interoperability. In this work, we emphasize semantic interoperability. We propose a methodology to help semiautomatic discovery of semantic mappings between concepts of different semantic resources. To accomplish this work, we transform these semantic resources in OWL-DL to use the services of inference engine based on description logics. We also propose a process for creating equivalence for relations between different semantic resources, called unification of relations; it aims to discover other semantic mapping. The results of this work were tested by a prototype. The implementation was carried out with resources from within the industry European construction sector. To assist in the evaluation of semantic mappings, we also propose a domain knowledge modelling based on fuzzy logic.

DISCIPLINE : Informatique

MOTS-CLES : interopérabilité sémantique, correspondance sémantique, ressources hétérogènes, inférences, alignement d'ontologies, logiques de description, OWL, EXPRESS

KEY WORDS : semantic interoperability, semantic mapping, heterogeneous resources, inferences, ontology alignment, description logics, OWL, EXPRESS

INTITULE ET ADRESSE DE L'U.F.R. OU DU LABORATOIRE : Laboratoire d'InfoRmatique en Image et Systèmes d'information, Bâtiment Nautibus (ex 710), 43 Boulevard du 11 novembre 1918, 69622 Villeurbanne cedex, France